百度智能云

BCC 文档

【版权声明】

版权所有©百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司。 未经本公司书面许可，任何单位和个人不得擅自摘抄、复制、传播本文档内容，否则本公司有权依法追究法律责任。

【商标声明】

和其他百度系商标，均为百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司的商标。 本文档涉及的第三方商标，依法由相关权利人所有。未经商标权利人书面许可，不得擅自对其商标进行使用、复制、修改、传播等行为。

【免责声明】

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导。 如您购买本文档介绍的产品、服务，您的权利与义务将依据百度智能云产品服务合同条款予以具体约定。本文档内容不作任何明示或暗示的保证。

目录

目录 2
功能发布记录 15
产品描述 17
什么是云服务器BCC 17
产品优势 18
与传统服务器对比 18
应用场景 19
使用限制 20
实例规格 21
已停售实例规格 56
型号及所在区域 72
原机型与新机型名称对应关系 73
产品定价 74
计费概述 74
退款说明 76
计费方式 77
包年包月（预付费） 77
按量付费（后付费） 77
预留实例券 77
快速入门 78
快速配置Linux云服务器 78
快速配置Windows云服务器 82
云服务器安全说明 85
操作指南 90
实例 90
实例概述 90
创建实例 91
批量连续命名 98
登录实例 100
前提条件 101
操作步骤 101
前提条件 101
使用puttygen创建私钥文件 102
使用putty配置密钥登录 103
前提条件 109
操作步骤 109
浏览器要求 111
操作步骤 111
浏览器要求 118
操作步骤 118

查看实例 119
启动实例 122
停止实例 123
重启实例 128
释放实例 129
实例续费 133
重置密码 134
后付费实例关机不计费 136
抢占实例 138
回收实例 141
跨AZ迁移 144
变更可用区 148
修改实例名称 149
变更实例配置 150
变更计费方式 155
设置IAM角色 159
镜像 164
镜像概述 164
创建自定义镜像 166
删除自定义镜像 167
共享自定义镜像 167
跨地域复制镜像 169
导入自定义镜像 170
重装操作系统 195
基于本地系统创建linux镜像 198
镜像快速指南 200
镜像列表 204
windows镜像制作 206
如何设置网卡多队列属性 211
百度智能云镜像服务协议 216
数据与个人信息保护声明 219
公共镜像发布记录 221
Linux公共镜像 250
CentOS 250
Ubuntu 252
Rocky Linux 253
GPU专用镜像 254
CentOS 254
操作系统官方停止维护计划 256
存储 258
概述 258

系统盘扩容 259
快照 259
快照概述 260
创建磁盘快照 261
回滚快照 264
删除快照 265
自动快照 266
网络 266
弹性网卡 266
配置ipv6地址 279
Windows云服务器配置双网卡外网访问 282
弹性RDMA网卡 284
变更VPC 286
变更内网IP 287
变更子网 288
绑定和解绑弹性公网IP 289
安全 290
安全组 290
使用限制 291
安全组规则 291
组成部分 291
背景信息 292
密钥对 298
多用户访问控制 301
启动模板 306
部署集 310
部署集概述 310
创建部署集 312
在部署集内创建BCC实例 312
修改部署集信息 313
删除部署集 313
为已有实例调整部署集 314
部署集内移出BCC实例 315
服务器迁移 316
服务器迁移概述 316
服务器迁移操作流程 319
迁移操作步骤 319
标签管理 326
标签管理概述 326
配置标签 327

基于标签查看资源账单 329
预留实例券 329
预留实例券概述 329
预留实例券创建 331
预留实例券续费 333
预留实例券跨账号转移 334
预留实例券变更可用区 336
预留实例券变更EHC集群 336
运维与监控 337
监控与报警 337
云助手 339
自助诊断工具 351
维修平台 355
实例健康检查 361
入口一：实例健康检查页 361
入口二：实例列表页 362
疏散故障实例 363
使用救援模式 364
弹性高性能计算集群 367
弹性高性能计算集群概述 367
创建弹性高性能计算集群 368
资源调研 370
视频专区 372
产品简介 372
操作指南 373
基础功能 373
网络相关 374
密码功能 375
环境搭建 376
运维参考 378
故障处理 379
典型实践 379
标签管理实践 379
环境搭建 381
AccessVPN搭建典型实践 381
Linux系统搭建FTP服务 385
windows搭建IIS和FTP教程 396
Windows系统部署SQL Server 2008 R2 Express 416
CentOS 7 yum安装MySQL 8.0 421
Linux系统搭建宝塔 421
Windows系统搭建宝塔 422

Linux幻兽帕鲁服务器一键搭建 422
Windows幻兽帕鲁服务器一键搭建 428
建站教程 434
Centos-7.2部署LNMP环境 434
Centos-7.2部署LAMP环境 437
云服务器部署SSL证书-Nginx 440
云服务器部署SSL证书-Apache 443
云服务器部署SSL证书-tomcat 446
云服务器部署SSL证书-IIS 447
windows安装配置Nginx环境 452
windows安装配置Apache环境 455
Windows2008与Windows2012下使用IIS服务搭建网站 460
API参考 467
概述 467
通用说明 468
服务域名 470
公共请求头与响应头 471
错误返回 471
实例相关接口 474
创建实例（通过指定实例套餐规格） 474
查询实例列表 481
查询指定实例详情 484
启动实例 487
启动实例（批量启动） 488
停止实例 489
停止实例（批量停止） 490
重启实例 492
修改实例密码 493
修改实例属性 494
重装实例 495
重装实例（批量重装） 497
将实例加入安全组 499
将实例移出安全组 500
查询实例VNC地址 501
实例续费 502
释放后付费实例(POST请求的释放) 504
实例变更子网 505
修改实例描述 506
计费变更-转包年包月 507
实例解绑标签 508
实例绑定标签 510

获取实例角色列表 511
绑定角色 512
解绑角色 513
修改实例主机名 515
根据实例id查询实例列表 516
释放ipv6 517
添加ipv6 519
查询多个实例详情 520
删除实例与部署集之间的关系 523
设置实例删除保护 523
批量转后付费 524
批量转包年包月 524
查询可变配规格 526
批量释放实例 528
批量变配实例 529
释放回收站实例 531
创建自动续费规则 532
删除自动续费规则 533
后付费设置自动释放 534
获取实例网卡信息 535
实例变更VPC 536
回收站内BCC恢复计费 538
查询回收站实例列表 539
预付费实例提前释放 541
查询可关机不计费的实例列表 543
批量退订预付费实例 545
设置抢占实例eip自动释放 547
批量增加主网卡辅助IP 548
批量删除主网卡辅助IP 549
实例变配（通过指定实例套餐规格） 550
查询实例自定义数据 552
进入救援模式 553
退出救援模式 554
磁盘相关接口 554
创建CDS磁盘 555
查询磁盘列表 558
查询磁盘详情 559
挂载CDS磁盘 561
卸载CDS磁盘 563
释放CDS磁盘（新） 564
磁盘扩容及类型变更 566

回滚磁盘 567
磁盘续费 568
磁盘重命名 569
修改磁盘属性 570
磁盘计费变更 572
开通自动续费 573
取消自动续费 574
磁盘绑定标签 575
磁盘解绑标签 577
获取可购买磁盘信息 578
查询CDS价格 579
镜像相关接口 581
创建自定义镜像 581
查询镜像列表 583
查询镜像详情 585
删除自定义镜像 586
共享自定义镜像 587
取消共享自定义镜像 588
查询镜像已共享的用户列表 590
根据实例ID批量查询OS信息 591
导入镜像 592
镜像解绑标签 594
镜像绑定标签 595
根据实例规格查询可用公共镜像 596
取消跨地域复制自定义镜像 597
跨地域复制自定义镜像 598
修改自定义镜像名称 599
快照相关接口 600
创建快照 600
查询快照列表 602
查询快照详情 603
删除快照 604
查询快照链列表 605
快照链解绑标签 607
快照链绑定标签 608
跨地域复制快照 609
自动快照策略相关接口 611
创建自动快照策略 611
绑定自动快照策略 612
解绑自动快照策略 613

删除自动快照策略 614
查询自动快照策略列表 615
查询自动快照策略详情 617
自动快照策略变更 618
安全组相关接口 620
创建安全组 620
查询安全组列表 622
授权安全组规则 624
撤销安全组规则 626
删除安全组 628
删除安全组规则 629
更新安全组规则 630
实例关联安全组 632
实例解绑安全组 632
实例替换安全组 633
抢占实例相关接口 634
创建抢占实例 634
查询抢占实例套餐 641
查询抢占实例市场价 642
取消抢占实例订单 645
密钥对相关接口 646
创建密钥对 646
导入密钥对 648
绑定密钥对 649
查询密钥对列表 650
查询密钥对详情 651
更改密钥对描述 652
重命名密钥对 653
解绑密钥对 654
删除密钥对 656
部署集相关接口 657
创建部署集 657
查询部署集列表 658
修改部署集 659
删除部署集 660
部署集移出实例 661
调整部署集 661
获取部署集详情 662
专属集群相关接口 663
创建专属集群 663
查询专属集群列表 665

查询专属集群详情 667
变配专属集群 669
专属集群续费 670
开通专属集群自动续费 671
取消专属集群自动续费 672
专属集群解绑标签 673
专属集群绑定标签 674
预留实例券相关接口 675
创建预留实例券 675
查询预留实例券 678
调整预留实例券 680
续费预留实例券 682
预留实例券转出列表 684
拒绝预留实例券转移 685
发起预留实例券转移 686
接受预留实例券转移 687
预留实例券转入列表 688
撤销预留实例券转移 690
预留实例券绑定标签 691
预留实例券解绑标签 692
预留实例券关闭自动续费规则 694
预留实例券创建自动续费规则 695
维修平台相关接口 696
维修任务相关接口 697
预授权相关接口 704
维修记录相关接口 710
地域及可用区相关接口 712
获取域名列表接口 712
查询可用区列表 713
其他接口 714
查询实例套餐价格 714
查询实例套餐规格(新) 717
通过套餐规格查询可用区 720
附录 721
DeployStrategy 739
集群管理相关接口 749
EHC集群修改 749
EHC集群删除 750
EHC集群列表 750
EHC集群创建 752

Java-SDK 753
概述 753
安装SDK工具包 753
快速入门 754
BccClient 754
BCC实例 757
磁盘 799
镜像 815
快照 825
自动快照策略 829
安全组 836
部署集 840
密钥对 844
预留实例券 853
地域 857
可用区 858
日志 859
异常处理 860
版本变更记录 861
Python-SDK 861
概述 861
安装SDK工具包 861
初始化 862
BccClient 863
实例 864
磁盘 880
镜像 887
快照 889
安全组 891
地域 893
预留实例券 894
版本变更记录 895
Python3-SDK 895
概述 895
安装SDK工具包 895
初始化 896
BccClient 896
实例 897
创建实例在专属服务器（支持对明文密码加密） 900
定时释放 (限定后付费实例) 911
查询关联部署集的实例详情 911

磁盘 928
镜像 939
快照 947
自动快照策略 951
安全组 955
密钥对 958
磁盘专属集群 963
部署集 965
预留实例券 967
地域 969
Go-SDK 970
概述 970
初始化 970
可用区 974
实例 975
磁盘 1018
镜像 1035
快照 1044
自动快照策略 1050
安全组 1055
密钥对 1059
部署集 1064
磁盘专属集群 1067
预留实例券 1069
地域 1074
弹性高性能计算集群 1075
云助手API参考 1077
服务域名 1077
批量获取实例bsm-agent状态 1077
命令接口 1079
历史记录 1088
日志接口 1093
错误返回 1094
公共请求与响应头 1095
通用说明 1095
云助手SDK参考 1096
Python SDK 1096
安装SDK 1096
初始化 1097
CaClient 1098
批量获取实例内bsm-agent状态 1099

命令接口 1100
历史记录 1108
日志接口 1114

运维参考 1115
Windows运维参考 1115
系统激活 1115
Windows自动更新相关配置 1122
蓝屏宕机常规处理方法 1125
性能资源 1132
系统黑屏处理方法 1136
windowsServer关闭IE增强机制教程 1138
使用xshell连接windows服务器 1142
关闭windows防火墙教程 1152
windows设置只允许本地&固定IP访问远程端口 1154
Windows修改默认连接远程端口 1160
Windows快速登录 1164
Windows允许单个用户连接多个会话的方法 1166
配置多用户访问Windows服务器 1169
Windows CPU内存跑满排查步骤 1178
本地播放Windows服务器中的声音 1179
远程连接Windows后无法使用剪切板问题处理 1183
桌面及应用颜色变黑处理方法 1184
服务器重启后无法显示数据盘 1186
Windows释放为硬件保留内存 1190
Windows实例磁盘空间不足清理教程 1193
WindowsBCC实例卡顿问题 1199
WindowsBCC实例入域后实现SID唯一 1201
Windows虚机自动登录失败处理办法 1202
Linux运维参考 1204
Linux服务器CPU内存跑满问题排查 1204
Linux修改默认远程连接端口 1205
Centos7安装docker 1207
Linux系统安装图形化界面并远程连接 1208
进入单用户模式 1239
性能测试 1250
fstab配置导致系统启动异常如何处理 1256
Linux镜像如何开启或关闭intelCPU漏洞补丁 1258
Ubunut16.04订阅扩展安全维护（ESM） 1259
云服务器删除或者修改文件时提示文件系统只读 1260
问题描述 1260
LinuxBCC实例卡顿问题 1261

无法通过SSH方式登录LinuxBCC实例怎么办 1263
配置Linux 分析工具atop和kdump 1265
Linux格式化与挂载实例挂载本地磁盘 1268
LinuxBCC实例提示空间不足问题解决 1271
服务器配置 1272
Debian 9软件源配置 1272
线上带ofed驱动的实例更换内核方案 1272
linux pip配置百度源 1288
Windows pip配置百度源 1289
常见问题 1291
实例类 1291
网络类 1292
存储类 1295
镜像类 1295
快照类 1296
计费类 1296
性能类 1296
安全类 1297
虚拟机运维类 1302
预留实例券类 1315
故障处理 1315
Windows服务器ping不通 1315
无法远程连接Windows实例 1317
无法远程连接Linux实例 1326
Linux云服务器重置密码后无法登录实例 1333
Linux云服务器进入单用户重置密码报错：Authentication token lock busy 1341
Linux云服务器进入单用户重置密码报错：Authentication token manipulation error 1341
GPU实例宕机故障排查 1342
Linux云服务器执行 yum update 报错存在依赖冲突 1344

功能发布记录

发布时间	功能描述
2025-05	• 资源调研支持根据源端配置推荐百度智能云对应的配置信息，当您从IDC或者其他云厂商迁移到BCC时，可以使用该功能了解匹配机型及价格参考 • 当Linux实例出现系统故障，无法进入单用户模式修复时，可以使用救援模式进行系统修复
2025-04	• 价格计算器支持查看裸金属EBC实例机型的价格，前往价格计算器查看
2025-03	• Linux实例导入镜像/创建自定义镜像支持镜像检测
2025-02	• openAPI查询实例列表支持通过实例名称模糊搜索
2025-01	• 新一代裸金属EBC机型（Intel第七代以上、AMD第三代以上）支持NUMA、NPS以及CPU线程数（超线程）能力开关的配置
2024-12	• openAPI/SDK支持查询实例自定义数据，参数请见查询实例自定义数据
2024-11	• 跨AZ迁移支持用户同子网下的部分实例不变私网IP迁移和分批迁移，降低对用户业务影响
2024-10	• 按量付费BCC实例转包年包月时支持同步开启自动续费 • 已开启自动续费的包年包月BCC实例支持直接操作转按量付费
2024-08	• 价格计算器迭代升级，支持选配获取价格或者查看规格的价格列表
2024-07	• EBC实例支持使用快照回滚系统盘 • EBC实例支持加入购物车，与其他产品共同购买结算，提升购买效率 • BCC新规格族g7、c7、m7、ga3、ca3、ma3发布，公测中
2024-06	• openAPI/SDK支持查询BCC所支持地域及域名 • 规格族为ica2、ca2、ga2、ma2的BCC实例支持关机不计费 • 裸金属EBC实例新规格族g7、ca3、l7、la3正式发布
2024-05	• BCC抢占实例创建时支持设置EIP关联自动释放 • EBC实例支持关机 • EBC实例制作的自定义镜像支持跨地域复制功能
2024-04	• 维修平台功能上线，可自动检测硬件故障并发起故障维修，最大限度保障用户业务连续性 • 预留实例券支持跨账号转移

	• 预留实例券支持编辑标签
2024-03	• 包年包月BCC实例支持立即转按量付费，更灵活地按需使用BCC实例，详见包年包月转按量付费
2024-01	• 自助诊断工具功能上线，支持对BCC实例进行应用中断问题、应用性能问题诊断并提供优化建议
2023-11	• 上线新远程连接终端-SmartTerm，支持远程登录Linux实例，搭载AI助手，助力开发与运维
2023-10	• 预付费BCC支持自助退订
2023-03	• 支持预留实例券功能
2022-08	• 支持在线调整部署集 • BCC支持购物车购买 • 支持跨AZ跨子网迁移BCC实例
2022-07	BCC变更VPC时可自定义安全组
2022-06	BCC后付费转预付费时支持开启自动续费
2022-05	• 支持启动模版 • 支持批量计费变更
2022-04	• BCC创建时支持挂载CFS • 支持批量变配
2022-01	• BCC状态变更通知事件 • BCC抢占实例释放前通知事件 • 新增BCC内存监控项
2021-10	支持预付费实例退款后进入回收站
2021-09	新增获取部署集详情接口、查询多个实例详情接口、删除实例与部署集之间的关系接口、设置实例删除保护接口
2021-08	新增跨zone迁移功能。针对cq02可用区中有资源的用户开放，以便进行资源跨zone迁移。
2020-12	云服务器BCC支持在手机上进行运维管理了！支持开启/关闭/重启云服务器，查看监控信息等功能。下载百度智能云APP即可体验。 扫码下载体验：
2020-07	服务器迁移中心全新上线
2020-06	白名单支持预付费资源降配
2020-05	• 支持批量重装操作系统 • 支持批量修改内网IP

	• 支持批量修改内网IP
2020-04	公共镜像新增支持centos8、debian10
2019-07	• 为便于用户基于自身的业务应用，快速选中合适的机型，我们对机型的名称进行了修改。机型名称修改，对机型性能等无任何影响，仅是按照cpu和内存的比值的不同，对套餐名称进行重新划分。详见机型名称变更。 • 支持抢占实例。抢占实例是云服务器BCC实例的一种新运作模式，旨在降低部分场景下使用BCC的成本。一种后付费实例，用户在指定范围内出价，当指定的实例规格的市场价格低于用户的出价且资源库存充足时，客户可以成功创建抢占实例，并按照当前市场价计费
2019-06	新增按需计费实例关机不计费，您可以提交工单进行申请，申请通过后即可让按需计费的实例在关机后不再计费
2019-05	支持镜像跨地域复制功能，镜像跨地域复制适用于跨地域部署实例、跨地域备份数据。您可以复制某一地域中的自定义镜像到其他百度智能云地域，并使用该镜像在相应地域创建BCC实例，实现在不同的地域部署环境相同的BCC实例
2019-04	BCC目前支持部分操作系统的公共镜像使用ipv6网络

产品描述

什么是云服务器BCC

云服务器BCC（Baidu Cloud Compute）是处理能力可弹性伸缩的计算服务。管理方式比物理服务器更简单高效，可根据您的业务需要创建、释放任意多台云服务器实例，提升运维效率。为您快速部署应用构建稳定可靠的基础，降低网络规模计算的难度，使您更专注于核心业务创新，且无需花费时间和金钱来购买及维护托管虚拟机的硬件，有效降低IT成本。

为什么选择云服务器BCC

- 升级灵活，随时拓展或缩减云服务器数量，快速响应业务变化。
- 成本低，按需使用，支持包年包月（预付费）、按量付费（后付费）、抢占实例三种计费模式。
- 高可管理性，可通过客户端、SmartTerm或VNC（Virtual Network Computing）远程登录BCC实例，提供多维度监控预警，帮助您提前发现问题，减少损失。
- 高可靠性，可几秒内将数据快速迁移恢复。

更多选择的理由，请参见产品优势。

相关概念

BCC实例

BCC实例可以理解为一台虚拟的服务器，包含CPU、内存等最基础的计算组件，是云服务器呈献给您的实际操作实体。BCC实例是云服务器最为核心的概念，支持IP绑定，镜像和快照等功能，诸如云磁盘CDS（Cloud Disk Service）、简单缓存服务SCS（Simple Cache Service）只有挂载在BCC实例后才可使用。

EBC实例

EBC实例是第三代百度太行·弹性裸金属服务器。搭配最新型智能网卡，实现存储卸载，分钟级创建实例。用户可以独享1台物理机，物理隔离，安全高效。

镜像

镜像为BCC实例提供操作系统和预装的软件。百度智能云提供了多种公共镜像供您选择。您还可以基于某一时刻的BCC系统盘创建您专属的自定义镜像，该镜像只能您自己看到，您也可以共享镜像给其他账户。

磁盘

磁盘为云服务器提供存储服务，可以用作实例的系统盘或可扩展数据盘使用，是低时延、持久性、高可靠和高弹性的块存储。

地域

地域（Region）指的是BCC实例所在的物理位置。百度智能云目前开放多地域支持，您可以根据您的客户群体分布的不同选择不同地域的云服务器服务，请参考地域选择说明。

> **注意：** 地域内的BCC实例内网间是可以互通的，不同地域之间的BCC实例内网不互通。因为获取物理资源的成本不一，所以每个地域的价格都略有差异。

安全组

安全组可以理解为是一种虚拟防火墙，用于实例的网络访问控制，具有数据包过滤和状态检测功能。BCC实例可以选择默认安全组或者自定义安全组。

- 默认安全组允许所有入站和出站访问通过。
- 自定义安全组是您自己制定的入站与出站规则。

快照

快照是对您指定数据组，在指定时间的备份。

- 快照的作用主要是数据备份与恢复，当设备发生应用故障或者文件损坏时可以快速恢复数据，将数据恢复到某个时间点的状态。
- 快照的另一个作用是为您提供了另外一个数据访问通道，当原数据在处理应用时，用户可以访问快照数据，还可以利用快照进行测试等工作。

产品定价

云服务器BCC支持包年包月（预付费）、按量付费（后付费）、抢占实例三种计费模式。关于计费方式的说明及计费项信息，请参见 计费概述。

云服务器BCC按照不同的规格类型分为不同的规格族，关于规格族信息，请参见 实例规格。关于实例规格定价信息，请参见 包年包月（预付费）定价、按量付费（后付费）定价。

云服务器BCC不同的区域提供的实例型号也不同，您可通过 型号及所在区域 查看您选定区域支持的实例型号。

产品优势

简单高效： 一键开通计算资源，并可按需使用，无需考虑计划、采购和维护硬件的成本和复杂性；秒级资源调度响应，同时创建多台服务器，可在几十秒内完成，显著提高工作效率。

安全可靠： 完善的多层次DDoS攻击防护解决方案，帮助云服务器稳定可靠地对外提供服务，云服务器防入侵解决方案有效抵御黑客攻击，支持分钟级攻击发现和告警，百Gbps级流量攻击清洗。基于百度大规模分布式集群下的智能调度，节点故障无感秒级迁移，故障恢复快，自动快照备份数据，数据持久性高于99.999%。

弹性： 分为横向和纵向两种，纵向指主机层级的低配置向高配置的资源扩展，横向指主机数量的扩展，由一台到多台的资源扩容。BCC支持弹性伸缩，秒级实现多台服务器的创建和释放，在线升级CPU、内存、带宽等资源。

低成本： 按需使用，支持包年包月（预付费）、按量付费（后付费）和抢占实例三种计费模式，支持分钟级计费，不同计费模式云服务器与弹性资源自由组合，搭配镜像、快照、云安全等增值服务，提供业界超高效费比的高性能云服务器。

与传统服务器对比

	云服务器	传统服务器
性能	• 针对不同场景，提供持续稳定的高性能、高性价比的计算能力 • 极高运算要求，结合最前沿处理器技术（Intel Xeon Platinum或AMD EPYC），完美支撑工作负载，如游戏行业 • 无特殊要求，灵活配置，提供最具性价比的计算服务	• 服务器折旧损耗，无法保证性能的稳定性 • 业务升级，受制于成本，无法及时更换更高性能服务器 • 高性能服务器的成本指数增加，往往无法承担
弹性	• 自由配置 CPU、内存、带宽，符合业务需求 • 随时拓展或缩减云服务器数量，快速响应业务变化 • 调整配置，数据不丢失	• 服务器规模固定，无法应对突发的业务规模增长，或造成长期闲置，带来成本的大幅提高
可靠性	• 自动排除故障，自动备份数据，数据持久性高于99.9999999% • 采用领先的智能调度技术，节点发生故障，可几秒内将数据快速迁移恢复，应对高可靠性要求的场景，如金融、游戏	• 往往缺少对机器的实时全面监控。一旦发生问题，很难快速锁定排除故障，可能导致数据的永久丢失 • 需花费较多运维成本保障机器可靠性
可管理性	• 控制台功能强大，支持对上千数量级实例的多重实时管理 • 多维度监控预警，帮助您提前发现问题，减少损失	• 需运维持续投入人力，不断人工重复部署和维护工作
安全	• 免费提供多层次DDoS攻击防护，百Gbps级流量防攻击 • 分钟级响应速度，及时发现并有效抵御黑客攻击 • 主、子账号权限细分管理，保障数据安全	• 需额外购买安全防护服务 • 对突发的恶意攻击事件，应对能力不足
成本	• 支持包年包月（预付费）、按量付费（后付费）和抢占实例三种计费方式，按需购买，大大节约成本 • 支持结合预留实例券优惠方式使用，以优化资源使用成本 • 无需维护硬件和网络设施，提高运维效率 • 云服务器推出了包年优惠，低至3年4折，后付费按分钟计费	• 租用费用高，只能包年包月 • 需持续投入人力维护，运维成本高

应用场景

场景类型	实例类型推荐	说明
个人网站/企业网站/电商	通用型实例	适用于各种类型和规模的企业级应用、中小型数据库系统、缓存、搜索集群。
关系型数据库/分布式缓存	内存型实例	适用于高性能数据库、内存数据库、数据分析与挖掘、分布式内存缓存。
NoSQL 数据库	本地SSD型实例	适用于对磁盘读写和时延要求高的 I/O 密集型应用场景，如Cassandra、云数据库MongoDB等。 • 您可以使用较高配置的BCC，同时采用云磁盘CDS，可实现高I/O并发响应和更高的数据可靠性。 • 您也可以使用多台中等偏下配置的，搭配负载均衡BLB，建设高可用底层架构。 • 更多详情，请参见云磁盘CDS和负载均衡BLB。
大型多人在线游戏	计算型实例	适用于需要高计算资源消耗的应用场景，如Web前端服务器、大型多人在线游戏（MMO）前端、数据分析、批量计算、视频编码、高性能科学和工程应用等。
直播	• 通用型ga2 • 计算型ca2 • 内存型ma2	适用于高网络包收发场景，如视频弹幕、电信业务转发等。
深度学习/图像处理	GPU型实例	能够大幅提高机器学习及科学计算等大规模计算框架的运行速度，为搭建人工智能及高性能计算平台提供基础架构支持。
Hadoop/Spark/ElasticSearch	• 内存型实例 • 大数据型实例	适用于Hadoop、Spark群集以及其他企业大内存需求应用。

使用限制

配额限制

产品配额默认如下，若需更多配额，您可提交工单申请。

BCC实例

- 单地域最多可同时拥有**50个按量付费（后付费）**实例。
- 回收站内最多同时容纳**50个**实例。

> 说明：
>
> - 释放的BCC实例不占配额，回收站内未释放的按量付费BCC实例和因欠费或到期未续费而没有释放的BCC实例占用配额。
> - 以上配额都是针对单个区域的。每个区域独立计算配额。

云磁盘

- 单地域最多可以拥有**n*5**个CDS数据盘，n为BCC数量。
- 每块磁盘最多可同时拥有**512**份快照，包括**256**份手动快照和**256**份自动快照。
- 单实例可挂载的磁盘和实例机型有关。Intel4、5、6系和AMD2系的规格族32C及以上的规格最多支持16个CDS数据盘，小于32C的规格和AMD1系规格族最多支持5个CDS数据盘，Intel7系和AMD3系及以后随实例规格变化，详情可见实例规格。

镜像

- 单地域最多可同时拥有**20**个镜像。
- 单地域单镜像可最多共享给**50**个用户。
- 单地域最多能接受**20**个共享镜像。

密钥对

- 单地域最多可以拥有**500**个密钥对。

部署集

- 单地域可创建**10**个部署集。
- 每个可用区下，每个部署集中最多可有10台BCC实例。一个百度智能云地域下单个部署集中的BCC实例最大数量 = 10 * 当前地域下的可用区数量。

弹性网卡

- 单实例可挂载的弹性网卡数量与实例规格相关，6系及以前的机型实例最多可挂载的弹性网卡数量为min{cpu核数，8}，7系及以后随实例规格变化，详情可见实例规格。

实例规格

根据不同的业务场景，云服务器BCC分为不同的实例规格族，这些实例规格族由 CPU、内存、存储、异构硬件和网络带宽组成不同的组合。在同一个实例规格族中，根据CPU和内存的比值不同分为不同的实例规格。每个实例规格提供不同的计算、内存和存储功能。您可基于需要部署运行的应用规模，选择一种适当的实例规格。

> **注意：**部分实例规格已停售，详情请参见已停售实例规格。

实例规格族

实例类型	实例规格族
经济型实例族	经济型e1
	通用型g7（公测中）
	通用型ga3（公测中）
	RDMA网络增强型g5re
	通用型g6

通用型实例族	通用型g5
	通用型gr1
	通用型g4
	通用型ga2
	通用型ga1
计算型实例族	计算型c7（公测中）
	计算型ca3（公测中）
	计算型c6
	计算型c5
	密集计算型ic5
	计算型cr1
	计算型c4
	密集计算型ic4
	计算型ca2
	计算型ca1
	密集计算型ica2
	密集计算型ica1
内存型实例族	内存型m7（公测中）
	内存型ma3（公测中）
	内存型m6
	内存型m5
	内存型m4

	内存型ma2
	内存型ma1
大数据型实例族	大数据型d1s
	大数据型d1
本地SSD型实例族	本地SSD型I6（公测中）
	本地SSD型I5d
	本地SSD型I5
	本地SSD型I3d
	本地SSD型Ia2
GPU型实例族	计算型gn5
	计算型gn3
弹性裸金属实例族	弹性裸金属本地SSD型I8
	弹性裸金属本地SSD型Ia4
	弹性裸金属本地SSD型hcI7
	弹性裸金属本地SSD型I7
	弹性裸金属通用型g7
	弹性裸金属本地SSD型Ia3
	弹性裸金属本地SSD型I5
	弹性裸金属通用型g5
	弹性裸金属本地SSD型Ia2ne
	弹性裸金属服务器GPU型gn7s
弹性高性能计算集群	弹性高性能计算集群GPU型GN6T

	弹性高性能计算集群GPU型GN5

经济型实例族

经济型e1

规格族特点

- 处理器：Intel Xeon系列
- 处理器与内存配比有1:2、1:4
- 费用较主流实例最高降低**50%**以上
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 中小型网站建设
  - 开发测试
  - 轻量级应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.e1.c2m2	2 核	2 GiB	1.5 Gbps	40万	2
bcc.e1.c2m4	2 核	4 GiB	1.5 Gbps	40万	2
bcc.e1.c2m8	2 核	8 GiB	1.5 Gbps	40万	2
bcc.e1.c4m8	4 核	8 GiB	1.5 Gbps	80万	4
bcc.e1.c4m16	4 核	16 GiB	1.5 Gbps	80万	4
bcc.e1.c8m16	8 核	16 GiB	3 Gbps	150万	4
bcc.e1.c8m32	8 核	32 GiB	3 Gbps	150万	4

通用型实例族

通用型g7（公测中）

规格族特点

- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 支持AMX（Advanced Matrix Extensions），可提高CPU的深度学习训练和推理性能
- 最高可支持160Gbps内网带宽，小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 在线推理、机器学习

- 高网络收发包场景

> 仅支持变配至g7、c7、m7实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

实例规格

实例规格	vCPU	内存	内网带宽基础/突发	网络收发包PPS	连接数	弹性网卡数	队列数	单网卡私有IPv4地址数	单网卡私有IPv6地址数	云磁盘个数（不包含系统盘）	云磁盘存储基础IOPS	云磁盘存储基础带宽
bcc.g7.c2m8	2 核	8 GiB	2.5/最高15 Gbps	100 万	最高30万	1	2	2	2	1	1 万	1 Gbps
bcc.g7.c4m16	4 核	16 GiB	4/最高15 Gbps	120 万	最高30万	3	4	8	8	1	1.5 万	1.5 Gbps
bcc.g7.c8m32	8 核	32 GiB	6/最高15 Gbps	160 万	最高30万	3	8	16	16	4	2 万	2 Gbps
bcc.g7.c16m64	16 核	64 GiB	12/最高25 Gbps	300 万	35万	7	16	16	16	14	2.5 万	3 Gbps
bcc.g7.c32m128	32 核	128 GiB	20/最高25 Gbps	500 万	60万	7	最高32	16	16	16	3 万	3.5 Gbps
bcc.g7.c48m192	48 核	192 GiB	30 Gbps	800 万	75万	7	最高32	30	30	16	4 万	4 Gbps
bcc.g7.c64m256	64 核	256 GiB	50 Gbps	1100 万	100万	7	最高32	30	30	16	5 万	5 Gbps
bcc.g7.c96m384	96 核	384 GiB	70 Gbps	1600 万	150万	15	最高32	40	40	16	14 万	14.5 Gbps

通用型ga3（公测中）

规格族特点

- 处理器：AMD EPYC 9W24，主频 2.6GHz，睿频 3.7GHz
- 处理器与内存配比为1:4
- 最高可支持80Gbps内网带宽，小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 高网络收发包场景，如游戏、直播等

> 仅支持变配至ga3、ca3、ma3实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

产品规格

实例规格	vCPU	内存（GiB）	网络带宽基础/突发（Gbit/s）	网络收发包PPS	连接数	队列数	弹性网卡数	单网卡私网IPV4数	单网卡私网IPV6数	磁盘总数（不包含系统盘）	云盘IOPS基础（万）	云盘带宽基础（Gbit/s）
bcc.ga3.c2m8	2	8	1.5/最高12.5	80万	最高25万	2	1	2	2	1	1	1
bcc.ga3.c4m16	4	16	2.5/最高12.5	100万	最高25万	4	3	8	8	1	1.5	1.5
bcc.ga3.c8m32	8	32	4/最高12.5	160万	最高25万	8	3	16	16	4	2	2
bcc.ga3.c16m64	16	64	10/最高12.5	200万	30万	16	3	16	16	14	2.5	3
bcc.ga3.c32m128	32	128	15/最高25	300万	60万	最高32	3	16	16	16	3	3.5
bcc.ga3.c48m192	48	192	20/最高25	450万	75万	最高32	7	30	30	16	4	4
bcc.ga3.c64m256	64	256	25	600万	100万	最高32	7	30	30	16	5	5
bcc.ga3.c96m384	96	384	40/无	900万	150万	最高32	14	40	40	16	8	8
bcc.ga3.c128m512	128	512	50/无	1200万	200万	最高32	14	40	40	16	10	10
bcc.ga3.c192m768	192	768	80/无	1800万	300万	最高32	14	40	40	16	15	16

RDMA网络增强型g5re

规格族特点

该实例目前白名单开放中，如有需求请提交工单

RDMA网络增强型实例可通过挂载弹性RDMA网卡ERI，在私有网络VPC中以RoCE v2的协议通信，具有低时延、高带宽、高性价比的特点，助力业务在云上规模部署实例。详细ERI的使用可参考弹性RDMA网卡。

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 最高可支持200Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 支持挂载弹性RDMA网卡
- 适用场景
    - 数据库系统、缓存、搜索集群
    - 高性能计算应用，例如流体力学计算、仿真计算

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数	可挂载ERI数量
bcc.g5re.c112m476	112 核	476 GiB	200 Gbps	5000 万	32	1

通用型g6

规格族特点

- 处理器：Intel Xeon Platinum 8463B，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 最高可支持90Gbps内网带宽
- 性能相比于上一代云服务器提升70%
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 在线推理、机器学习

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数	连接数
bcc.g6.c2m8	2 核	8 GiB	2 Gbps	80万	2	最高30万
bcc.g6.c4m16	4 核	16 GiB	4 Gbps	100万	4	最高30万
bcc.g6.c8m32	8 核	32 GiB	6 Gbps	140万	8	最高30万
bcc.g6.c12m48	12 核	48 GiB	10 Gbps	180万	12	40万
bcc.g6.c16m64	16 核	64 GiB	12 Gbps	200万	16	50万
bcc.g6.c24m96	24 核	96 GiB	15 Gbps	250万	24	80万
bcc.g6.c32m128	32 核	128 GiB	20 Gbps	300万	32	100万
bcc.g6.c48m192	48 核	192 GiB	25 Gbps	450万	48	150万
bcc.g6.c64m256	64 核	256 GiB	32 Gbps	800万	64	200万
bcc.g6.c96m384	96 核	384 GiB	50 Gbps	1200万	64	300万
bcc.g6.c128m512	128 核	512 GiB	70 Gbps	1300万	64	350万
bcc.g6.c172m966	172 核	966 GiB	90 Gbps	1500万	64	400万

通用型g5

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 各种类型和规模的企业级应用

- 数据分析和计算
- 中小型数据库系统、缓存、搜索集群
- 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.g5.c2m8	2 核	8 GiB	1.5 Gbps	40 万	2
bcc.g5.c4m16	4 核	16 GiB	1.5 Gbps	80 万	4
bcc.g5.c8m32	8 核	32 GiB	3 Gbps	150 万	8
bcc.g5.c16m64	16 核	64 GiB	6 Gbps	280 万	8
bcc.g5.c24m96	24 核	96 GiB	8 Gbps	400 万	16
bcc.g5.c32m128	32 核	128 GiB	11 Gbps	500 万	16
bcc.g5.c48m192	48 核	192 GiB	16 Gbps	600 万	32
bcc.g5.c56m238	56 核	238 GiB	22 Gbps	700 万	32
bcc.g5.c64m256	64 核	256 GiB	22 Gbps	700 万	32
bcc.g5.c112m476	112 核	476 GiB	35 Gbps	1200 万	32

通用型gr1

规格族特点

- 处理器：Ampere Altra Q80-30，主频3.0GHz
- 处理器与内存配比为1:4
- 最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 云终端仿真、云手机，云游戏
  - Web应用服务
  - 容器、微服务
  - AI推理（数据治理）、数据分析和计算
  - DevOps

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gr1.c1m4	1 核	4 GiB	1 Gbps	75 万	1
bcc.gr1.c2m8	2 核	8 GiB	1 Gbps	90 万	2
bcc.gr1.c4m16	4 核	16 GiB	2 Gbps	100 万	4
bcc.gr1.c8m32	8 核	32 GiB	3 Gbps	150 万	8
bcc.gr1.c16m64	16 核	64 GiB	6 Gbps	180 万	8
bcc.gr1.c24m96	24 核	96 GiB	8 Gbps	225 万	12
bcc.gr1.c32m128	32 核	128 GiB	10 Gbps	275 万	16
bcc.gr1.c48m192	48 核	192 GiB	16 Gbps	375 万	16
bcc.gr1.c64m238	64 核	238 GiB	25 Gbps	500 万	32

通用型g4

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 数据分析和计算
  - 各种类型和规模的企业级应用
  - 中小型数据库系统、缓存、搜索集群
  - 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.g4.c2m8	2 核	8 GiB	1.5 Gbps	40 万	2
bcc.g4.c4m16	4 核	16 GiB	1.5 Gbps	80 万	4
bcc.g4.c8m32	8 核	32 GiB	3 Gbps	150 万	8
bcc.g4.c16m64	16 核	64 GiB	6 Gbps	280 万	8
bcc.g4.c24m96	24 核	96 GiB	8 Gbps	400 万	12
bcc.g4.c32m128	32 核	128 GiB	11 Gbps	500 万	16
bcc.g4.c48m192	48 核	192 GiB	15 Gbps	600 万	16
bcc.g4.c64m256	64 核	256 GiB	22 Gbps	700 万	16
bcc.g4.c84m352	84 核	352 GiB	25 Gbps	800 万	16

通用型ga2

规格族特点

- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:4

- 支持增强型云磁盘
- 网络收发包最高可达1200w
- 适用场景
  - 容器
  - 离线计算类框架
  - HPC高性能计算
  - Web前端业务、分布式缓存

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.ga2.c2m8	2 核	8 GiB	1.5 Gbps	10 万	2
bcc.ga2.c4m16	4 核	16 GiB	3 Gbps	50 万	4
bcc.ga2.c8m32	8 核	32 GiB	6 Gbps	80 万	8
bcc.ga2.c16m64	16 核	64 GiB	12 Gbps	150 万	8
bcc.ga2.c32m128	32 核	128 GiB	15 Gbps	200 万	16
bcc.ga2.c48m192	48 核	192 GiB	20 Gbps	300 万	32
bcc.ga2.c64m256	64 核	256 GiB	30 Gbps	400 万	32
bcc.ga2.c80m320	80 核	320 GiB	35 Gbps	500 万	32
bcc.ga2.c116m488	116 核	488 GiB	45 Gbps	600 万	32
bcc.ga2.c232m978	232 核	978 GiB	90 Gbps	1200 万	32

通用型ga1

规格族特点

- 处理器：AMD EPYC 7502，主频 2.5GHz，睿频 3.35GHz
- 处理器与内存配比为1:4
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 数据分析和计算
  - 各种类型和规模的企业级应用
  - 中小型数据库系统、缓存、搜索集群
  - 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS
bcc.ga1.c2m8	2 核	8 GiB	1.5 Gbps	15万
bcc.ga1.c4m16	4 核	16 GiB	1.5 Gbps	30万
bcc.ga1.c8m32	8 核	32 GiB	3 Gbps	60万
bcc.ga1.c16m64	16 核	64 GiB	6 Gbps	120万
bcc.ga1.c24m96	24 核	96 GiB	8 Gbps	180万
bcc.ga1.c32m128	32 核	128 GiB	12 Gbps	250万
bcc.ga1.c48m192	48 核	192 GiB	14 Gbps	300万
bcc.ga1.c64m256	64 核	256 GiB	18 Gbps	450万

计算型实例族

计算型c7（公测中）

规格族特点

- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:2
- 支持AMX（Advanced Matrix Extensions），可提高CPU的深度学习训练和推理性能
- 最高可支持160Gbps内网带宽，小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 在线推理、机器学习
  - 高网络收发包场景

> 仅支持变配至g7、c7、m7实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

实例规格

实例规格	vCPU	内存	内网带宽基础/突发	网络收发包PPS	连接数	弹性网卡数	队列数	单网卡私有IPv4地址数	单网卡私有IPv6地址数	云磁盘个数（不包含系统盘）	云磁盘存储基础IOPS	云磁盘存储基础带宽
bcc.c7.c2m4	2 核	4 GiB	2.5/最高15 Gbps	100 万	最高30万	1	2	2	2	1	1 万	1 Gbps
bcc.c7.c4m8	4 核	8 GiB	4/最高15 Gbps	120 万	最高30万	3	4	8	8	1	1.5 万	1.5 Gbps
bcc.c7.c8m16	8 核	16 GiB	6/最高15 Gbps	160 万	最高30万	3	8	16	16	4	2 万	2 Gbps
bcc.c7.c16m32	16 核	32GiB	12/最高25 Gbps	300 万	35万	7	16	16	16	14	2.5 万	3 Gbps
bcc.c7.c32m64	32 核	64 GiB	20/最高25 Gbps	500 万	60万	7	最高32	16	16	16	3 万	3.5 Gbps
bcc.c7.c48m96	48 核	96 GiB	30 Gbps	800 万	75万	7	最高32	30	30	16	4 万	4 Gbps
bcc.c7.c64m128	64 核	128 GiB	50 Gbps	1100 万	100万	7	最高32	30	30	16	5 万	5 Gbps
bcc.c7.c96m192	96 核	192 GiB	70 Gbps	1600 万	150万	15	最高32	40	40	16	14 万	14.5 Gbps

计算型ca3（公测中）

规格族特点

- 处理器：AMD EPYC 9W24，主频 2.6GHz，睿频 3.7GHz
- 处理器与内存配比为1:2
- 最高可支持80Gbps内网带宽，小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 高网络收发包场景，如游戏、直播等

> 仅支持变配至ga3、ca3、ma3实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

产品规格

实例规格	vCPU	内存（GiB）	网络带宽基础/突发（Gbit/s）	网络收发包PPS	连接数	队列数	弹性网卡数	单网卡私网IPV4数	单网卡私网IPV6数	磁盘总数（不包含系统盘）	云盘IOPS基础（万）	云盘带宽基础（Gbit/s）
bcc.ca3.c2m4	2	4	1.5/最高12.5	80万	最高25万	2	1	2	2	1	1	1
bcc.ca3.c4m8	4	8	2.5/最高12.5	100万	最高25万	4	3	8	8	1	1.5	1.5
bcc.ca3.c8m16	8	16	4/最高12.5	160万	最高25万	8	3	16	16	4	2	2
bcc.ca3.c16m32	16	32	10/最高12.5	200万	30万	16	3	16	16	14	2.5	3
bcc.ca3.c32m64	32	64	15/最高25	300万	60万	最高32	3	16	16	16	3	3.5
bcc.ca3.c48m96	48	96	20/最高25	450万	75万	最高32	7	30	30	16	4	4
bcc.ca3.c64m128	64	128	25	600万	100万	最高32	7	30	30	16	5	5
bcc.ca3.c96m192	96	192	40/无	900万	150万	最高32	14	40	40	16	8	8
bcc.ca3.c128m256	128	256	50/无	1200万	200万	最高32	14	40	40	16	10	10

计算型c6

规格族特点

- 处理器：Intel Xeon Platinum 8463B，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数	连接数
bcc.c6.c2m4	2 核	4 GiB	2 Gbps	80万	2	最高30万
bcc.c6.c4m8	4 核	8 GiB	4 Gbps	100万	4	最高30万
bcc.c6.c8m16	8 核	16 GiB	6 Gbps	140万	8	最高30万
bcc.c6.c12m24	12 核	24 GiB	10 Gbps	180万	12	40万
bcc.c6.c16m32	16 核	32 GiB	12 Gbps	200万	16	50万
bcc.c6.c24m48	24 核	48 GiB	15 Gbps	250万	24	80万
bcc.c6.c32m64	32 核	64 GiB	20 Gbps	300万	32	100万
bcc.c6.c48m96	48 核	96 GiB	25 Gbps	450万	48	150万
bcc.c6.c64m128	64 核	128 GiB	32 Gbps	800万	64	200万
bcc.c6.c96m128	96 核	192 GiB	50 Gbps	1200万	64	300万
bcc.c6.c128m256	128 核	256 GiB	70 Gbps	1300万	64	350万

计算型c5

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.c5.c2m4	2 核	4 GiB	1.5 Gbps	40 万	2
bcc.c5.c4m8	4 核	8 GiB	1.5 Gbps	80 万	4
bcc.c5.c8m16	8 核	16 GiB	3 Gbps	150 万	8
bcc.c5.c16m32	16 核	32 GiB	6 Gbps	280 万	8
bcc.c5.c24m48	24 核	48 GiB	8 Gbps	400 万	16
bcc.c5.c32m64	32 核	64 GiB	11 Gbps	500 万	16
bcc.c5.c48m96	48 核	96 GiB	16 Gbps	600 万	32
bcc.c5.c64m128	64 核	128 GiB	22 Gbps	700 万	32

密集计算型ic5

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:1

- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.ic5.c2m2	2 核	2 GiB	1.5 Gbps	40 万	2
bcc.ic5.c4m4	4 核	4 GiB	1.5 Gbps	80 万	4
bcc.ic5.c8m8	8 核	8 GiB	3 Gbps	150 万	8
bcc.ic5.c16m16	16 核	16 GiB	6 Gbps	280 万	8
bcc.ic5.c24m24	24 核	24 GiB	8 Gbps	400 万	16
bcc.ic5.c32m32	32 核	32 GiB	11 Gbps	500 万	16

计算型cr1

规格族特点

- 处理器：Ampere Altra Q80-30，主频3.0GHz
- 处理器与内存配比为1:2
- 最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 云终端仿真、云手机，云游戏
  - Web应用服务
  - 容器、微服务
  - DevOps

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.cr1.c1m2	1 核	2 GiB	1 Gbps	75 万	1
bcc.cr1.c2m4	2 核	4 GiB	1 Gbps	90 万	2
bcc.cr1.c4m8	4 核	8 GiB	2 Gbps	100 万	4
bcc.cr1.c8m16	8 核	16 GiB	3 Gbps	150 万	8
bcc.cr1.c16m32	16 核	32 GiB	6 Gbps	180 万	8
bcc.cr1.c24m48	24 核	48 GiB	8 Gbps	225 万	12
bcc.cr1.c32m64	32 核	64 GiB	10 Gbps	275 万	16

计算型c4

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.c4.c2m4	2 核	4 GiB	1.5 Gbps	40 万	2
bcc.c4.c4m8	4 核	8 GiB	1.5 Gbps	80 万	4
bcc.c4.c8m16	8 核	16 GiB	3 Gbps	150 万	8
bcc.c4.c16m32	16 核	32 GiB	6 Gbps	280 万	8
bcc.c4.c24m48	24 核	48 GiB	8 Gbps	400 万	12
bcc.c4.c32m64	32 核	64 GiB	11 Gbps	500 万	16
bcc.c4.c48m96	48 核	96 GiB	15 Gbps	600 万	16
bcc.c4.c64m128	64 核	128 GiB	22 Gbps	700 万	16

密集计算型ic4

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.ic4.c2m2	2 核	2 GiB	1.5 Gbps	40 万	2
bcc.ic4.c4m4	4 核	4 GiB	1.5 Gbps	80 万	4
bcc.ic4.c8m8	8 核	8 GiB	3 Gbps	150 万	8
bcc.ic4.c16m16	16 核	16 GiB	6 Gbps	280 万	8

计算型ca2

规格族特点

- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:2
- 支持增强型云磁盘
- 超高网络PPS收发包能力
- 适用场景
  - 容器
  - 离线计算类框架
  - HPC高性能计算
  - Web前端业务、分布式缓存

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.ca2.c2m4	2 核	4 GiB	1.5 Gbps	10 万	2
bcc.ca2.c4m8	4 核	8 GiB	3 Gbps	50 万	4
bcc.ca2.c8m16	8 核	16 GiB	6 Gbps	80 万	8
bcc.ca2.c16m32	16 核	32 GiB	12 Gbps	150 万	8
bcc.ca2.c32m64	32 核	64 GiB	15 Gbps	200 万	16
bcc.ca2.c64m128	64 核	128 GiB	30 Gbps	400 万	32
bcc.ca2.c80m160	80 核	160 GiB	35 Gbps	500 万	32

计算型ca1

规格族特点

- 处理器：AMD EPYC 7502，主频 2.5GHz，睿频 3.35GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS
bcc.ca1.c2m4	2 核	4 GiB	1.5 Gbps	15 万
bcc.ca1.c4m8	4 核	8 GiB	1.5 Gbps	30 万
bcc.ca1.c8m16	8 核	16 GiB	3 Gbps	60 万
bcc.ca1.c16m32	16 核	32 GiB	6 Gbps	120 万
bcc.ca1.c24m48	24 核	48 GiB	8 Gbps	180 万
bcc.ca1.c32m64	32 核	64 GiB	12 Gbps	250 万
bcc.ca1.c48m96	48 核	96 GiB	14 Gbps	300 万
bcc.ca1.c64m128	64 核	128 GiB	18 Gbps	450 万

密集计算型ica2

规格族特点

- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS
bcc.ica2.c2m2	2 核	2 GiB	1.5 Gbps	10 万
bcc.ica2.c4m4	4 核	4 GiB	3 Gbps	50 万
bcc.ica2.c8m8	8 核	8 GiB	6 Gbps	80 万
bcc.ica2.c16m16	16 核	16 GiB	12 Gbps	150 万
bcc.ica2.c32m32	32 核	32 GiB	15 Gbps	200 万
bcc.ica2.c48m48	48 核	48 GiB	20 Gbps	300 万
bcc.ica2.c64m64	64 核	64 GiB	30 Gbps	400 万

密集计算型ica1

规格族特点

- 处理器：AMD EPYC 7502，主频 2.5GHz，睿频 3.35GHz
- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端

- 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网基础带宽
bcc.ica1.c2m2	2 核	2 GiB	1.5 Gbps
bcc.ica1.c4m4	4 核	4 GiB	1.5 Gbps
bcc.ica1.c8m8	8 核	8 GiB	2 Gbps
bcc.ica1.c16m16	16 核	16 GiB	3 Gbps

内存型实例族

内存型m7（公测中）

规格族特点

- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:8
- 支持AMX（Advanced Matrix Extensions），可提高CPU的深度学习训练和推理性能
- 最高可支持160Gbps内网带宽，小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 在线推理、机器学习
  - 高网络收发包场景

> 仅支持变配至g7、c7、m7实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

实例规格

实例规格	vCPU	内存	内网带宽基础/突发	网络收发包PPS	连接数	弹性网卡数	队列数	单网卡私有IPv4地址数	单网卡私有IPv6地址数	云磁盘个数（不包含系统盘）	云磁盘存储基础IOPS	云磁盘存储基础带宽
bcc.m7.c2m16	2 核	16 GiB	2.5/最高15 Gbps	100 万	最高30万	1	2	2	2	1	1 万	1 Gbps
bcc.m7.c4m32	4 核	32GiB	4/最高15 Gbps	120 万	最高30万	3	4	8	8	1	1.5 万	1.5 Gbps
bcc.m7.c8m64	8 核	64 GiB	6/最高15 Gbps	160 万	最高30万	3	8	16	16	4	2 万	2 Gbps
bcc.m7.c16m128	16 核	128GiB	12/最高25 Gbps	300 万	35万	7	16	16	16	14	2.5 万	3 Gbps
bcc.m7.c32m256	32 核	256 GiB	20/最高25 Gbps	500 万	60万	7	最高32	16	16	16	3 万	3.5 Gbps
bcc.m7.c48m384	48 核	384GiB	30 Gbps	800 万	75万	7	最高32	30	30	16	4 万	4 Gbps

内存型ma3（公测中）

规格族特点

- 处理器：AMD EPYC 9W24，主频 2.6GHz，睿频 3.7GHz
- 处理器与内存配比为1:8
- 小规格实例的网络性能具备突发能力。网络性能与计算规格相关，规格越高，性能越强
- 适用场景
  - 各种类型和规模的企业级应用
  - 数据分析和计算
  - 中小型数据库系统、缓存、搜索集群
  - 高网络收发包场景，如游戏、直播等

> 仅支持变配至ga3、ca3、ma3实例规格
>
> 暂不支持Windows镜像
>
> 不支持低版本的Linux镜像，比如CentOS6及以下版本、Ubuntu14及以下、Debian8及以下等

产品规格

实例规格	vCPU	内存（GiB）	网络带宽基础/突发（Gbit/s）	网络收发包PPS	连接数	队列数	弹性网卡数	单网卡私网IPV4数	单网卡私网IPV6数	磁盘总数（不包含系统盘）	云盘IOPS基础（万）	云盘带宽基础（Gbit/s）
bcc.ma3.c2m16	2	16	1.5/最高12.5	80万	最高25万	2	1	2	2	1	1	1
bcc.ma3.c4m32	4	32	2.5/最高12.5	100万	最高25万	4	3	8	8	1	1.5	1.5
bcc.ma3.c8m64	8	64	4/最高12.5	160万	最高25万	8	3	16	16	4	2	2
bcc.ma3.c16m128	16	128	10/最高12.5	200万	30万	16	3	16	16	14	2.5	3
bcc.ma3.c32m256	32	256	15/最高25	300万	60万	最高32	3	16	16	16	3	3.5
bcc.ma3.c48m384	48	384	20/最高25	450万	75万	最高32	7	30	30	16	4	4
bcc.ma3.c64m512	64	512	25	600万	100万	最高32	7	30	30	16	5	5

内存型m6

规格族特点

- 处理器：Intel Xeon Platinum 8463B，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数	连接数
bcc.m6.c2m16	2 核	16 GiB	2 Gbps	80万	2	最高30万
bcc.m6.c4m32	4 核	32 GiB	4 Gbps	100万	4	最高30万
bcc.m6.c8m64	8 核	64 GiB	6 Gbps	140万	8	最高30万
bcc.m6.c12m96	12 核	96 GiB	10 Gbps	180万	12	40万
bcc.m6.c16m128	16 核	128 GiB	12 Gbps	200万	16	50万
bcc.m6.c24m192	24 核	192 GiB	15 Gbps	250万	24	80万
bcc.m6.c32m256	32 核	256 GiB	20 Gbps	300万	32	100万
bcc.m6.c64m512	64 核	512 GiB	32 Gbps	800万	64	200万

内存型m5

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.m5.c2m16	2 核	16 GiB	1.5 Gbps	40 万	2
bcc.m5.c4m32	4 核	32 GiB	1.5 Gbps	80 万	4
bcc.m5.c8m64	8 核	64 GiB	3 Gbps	150 万	8
bcc.m5.c16m128	16 核	128 GiB	6 Gbps	280 万	8
bcc.m5.c24m192	24 核	192 GiB	8 Gbps	400 万	16
bcc.m5.c32m256	32 核	256 GiB	11 Gbps	500 万	16
bcc.m5.c48m384	48 核	384 GiB	16 Gbps	600 万	32
bcc.m5.c64m512	64 核	512 GiB	22 Gbps	700 万	32

内存型m4

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库

- 数据分析与挖掘、分布式内存缓存
- Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.m4.c2m16	2 核	16 GiB	1.5 Gbps	40 万	2
bcc.m4.c4m32	4 核	32 GiB	1.5 Gbps	80 万	4
bcc.m4.c8m64	8 核	64 GiB	3 Gbps	150 万	8
bcc.m4.c16m128	16 核	128 GiB	6 Gbps	280 万	8
bcc.m4.c24m192	24 核	192 GiB	8 Gbps	400 万	12
bcc.m4.c32m256	32 核	256 GiB	11 Gbps	500 万	16
bcc.m4.c48m384	48 核	384 GiB	15 Gbps	600 万	16
bcc.m4.c64m512	64 核	512 GiB	22 Gbps	700 万	16

内存型ma2

规格族特点

- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:8
- 支持增强型云磁盘
- 超高网络PPS收发包能力
- 适用场景
  - 容器
  - 离线计算类框架
  - HPC高性能计算
  - Web前端业务、分布式缓存

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.ma2.c2m16	2 核	16 GiB	1.5 Gbps	10 万	2
bcc.ma2.c4m32	4 核	32 GiB	3 Gbps	50 万	4
bcc.ma2.c8m64	8 核	64 GiB	6 Gbps	80 万	8
bcc.ma2.c16m128	16 核	128 GiB	12 Gbps	150 万	8
bcc.ma2.c32m256	32 核	256 GiB	15 Gbps	200 万	16

内存型ma1

规格族特点

- 处理器：AMD EPYC 7502，主频 2.5GHz，睿频 3.35GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高

- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS
bcc.ma1.c2m16	2 核	16 GiB	1.5 Gbps	15万
bcc.ma1.c4m32	4 核	32 GiB	1.5 Gbps	30万
bcc.ma1.c8m64	8 核	64 GiB	3 Gbps	60万
bcc.ma1.c16m128	16 核	128 GiB	6 Gbps	120万
bcc.ma1.c24m192	24 核	192 GiB	8 Gbps	180万
bcc.ma1.c32m256	32 核	256 GiB	12 Gbps	250万

大数据型实例族

大数据型d1s

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 实例配备大容量高吞吐的SATA HDD本地盘
- 最高可支持25Gbps内网带宽。实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Hadoop MapReduce、HDFS、Hive、HBase等
  - Spark内存计算、MLlib等
  - ElasticSearch、日志等

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
bcc.d1s.c8m32	8 核	32 GiB	1 块*11174 GiB	6 Gbps	150 万	8
bcc.d1s.c16m64	16 核	64 GiB	2 块*11174 GiB	12 Gbps	280 万	8
bcc.d1s.c24m96	24 核	96 GiB	4 块*11174 GiB	16 Gbps	400 万	12
bcc.d1s.c32m128	32 核	128 GiB	6 块*11174 GiB	20 Gbps	500 万	16
bcc.d1s.c64m256	64 核	256 GiB	10 块*11174 GiB	25 Gbps	700 万	16
bcc.d1s.c80m320	80 核	320 GiB	12 块*11174 GiB	25 Gbps	800 万	16

大数据型d1

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4

- 实例配备高吞吐的SATA HDD本地盘，最高可支持11Gbps内网带宽
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Hadoop MapReduce、HDFS、Hive、HBase等
  - Spark内存计算、MLlib等
  - 互联网行业、金融行业等有大数据计算与存储分析需求的行业客户，进行海量数据存储和计算的业务场景
  - ElasticSearch、日志等

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS
bcc.d1.c8m32	8 核	32 GiB	1 块*7190 GiB	6 Gbps	150 万
bcc.d1.c12m48	12 核	48 GiB	2 块*7190 GiB	6 Gbps	160 万
bcc.d1.c16m64	16 核	64 GiB	4 块*7190 GiB	11 Gbps	250 万

本地SSD型实例族

本地SSD型I6（公测中）

如需申请测试，请提交工单

规格族特点

- 处理器：Intel Xeon Platinum 8463B，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 支持增强型云磁盘
- 单盘随机读性能高达 105w IOPS
- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
bcc.l6.c16m64.1d	16 核	64 GiB	1块*3570 GiB	18 Gbps	330万	8
bcc.l6.c32m128.2d	32 核	128 GiB	2块*3570 GiB	35 Gbps	650万	16
bcc.l6.c64m256.4d	64 核	256 GiB	4块*3570 GiB	70 Gbps	1300万	32
bcc.l6.c90m482.4d	90 核	482 GiB	4块*3570 GiB	90 Gbps	1600万	32
bcc.l6.c180m966.8d	180 核	966 GiB	8块*3570 GiB	170 Gbps	3000万	32

本地SSD型I5d（公测中）

如需申请测试，请提交工单

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 相比上一代，相同的核数可支持更小的本地盘容量
- 单实例随机读性能高达 140w IOPS，单位存储的 IOPS 有1倍的提升
- 单实例最大吞吐达 7GB/s，单位存储的吞吐提升17%
- 支持增强型云磁盘
- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数	最大读IOPS	最大读吞吐量	最大写吞吐量
bcc.l5d.c8m32.1d	8 核	32 GiB	1块*894GiB	3 Gbps	140 万	8	35万	1750MB/s	1200MB/s
bcc.l5d.c16m64.1d	16 核	64 GiB	1块*1788GiB	6 Gbps	280 万	8	70万	3500MB/s	2400MB/s
bcc.l5d.c24m96.2d	24 核	96 GiB	2块*1788GiB	9 Gbps	420 万	12	140万	7000MB/s	4800MB/s
bcc.l5d.c32m128.2d	32 核	128 GiB	2块*1788GiB	11 Gbps	500 万	16	140万	7000MB/s	4800MB/s

本地SSD型l5

规格族特点

- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 单本地盘随机读性能高达 70w IOPS
- 支持增强型云磁盘
- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地（块数*GiB）盘	内网基础带宽	网络收发包PPS	队列数	最大读IOPS	最大读吞吐量	最大写吞吐量
bcc.l5.c16m64.1d	16 核	64 GiB	1 块*3570 GiB	6 Gbps	280 万	8	70万	6GB/s	3GB/s
bcc.l5.c32m128.1d	32 核	128 GiB	1 块*3570 GiB	11 Gbps	500 万	16	70万	6GB/s	3GB/s
bcc.l5.c56m238.2d	56 核	238 GiB	2 块*3570 GiB	20 Gbps	750 万	32	140万	12GB/s	6GB/s
bcc.l5.c112m476.4d	112 核	476 GiB	4 块*3570 GiB	35 Gbps	1200 万	32	280万	24GB/s	12GB/s

本地SSD型l3d

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 数据盘：NVMe SSD本地盘
- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
bcc.l3d.c16m64.1d	16 核	64 GiB	1 块*3570 GiB	6 Gbps	280 万	8
bcc.l3d.c32m64.1d	32 核	64 GiB	1 块*3570 GiB	11 Gbps	500 万	16
bcc.l3d.c32m128.2d	32 核	128 GiB	2 块*3570 GiB	11 Gbps	500 万	16
bcc.l3d.c40m174.2d	40 核	174GiB	2 块*3570 GiB	11 Gbps	500 万	16
bcc.l3d.c64m256.4d	64 核	256 GiB	4 块*3570 GiB	22 Gbps	700 万	16
bcc.l3d.c80m348.4d	80 核	348GiB	4 块*3570 GiB	22 Gbps	700 万	16

本地SSD型la2

规格族特点

- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:4
- 支持增强型云磁盘
- 单盘随机读性能高达65w IOPS
- 网络收发包最高可达1200w
- 适用场景
  - OLTP、高性能关系型数据库

- NoSQL数据库（如Cassandra、MongoDB等）
- Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地盘	内网基础带宽	网络收发包PPS	队列数
bcc.la2.c16m64.1d	16 核	64 GiB	1 块*3570 GiB	14 Gbps	400 万	8
bcc.la2.c32m128.2d	32 核	128 GiB	2 块*3570 GiB	30 Gbps	600 万	16
bcc.la2.c64m256.4d	64 核	256 GiB	4 块*3570 GiB	60 Gbps	1200 万	32

GPU型实例族

GPU实例详情可访问GPU云服务器官网页面

计算型gn5

搭载Intel Xeon Icelake以及英伟达Ampere架构系列GPU

Ampere A100-80G系列

- 适用场景：
  - 人工智能大规模训练：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能计算：生命科学、工业制造仿真、气象预测、计算物理等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
- 规格特点：
  - 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla A100，FP16算力达到312TFLOPS，单GPU显存80GB HBM2，支持双向带宽600GB的Nvlink互联
  - 内存：最高可提供960 GiB内存
  - 存储：支持CDS云磁盘
  - 网络：最高可支持200Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求

实例规格

实例	A100显卡数量	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gn5.c14m120.1A100-80g	1张	14核	120 GiB	25 Gbps	625万	8
bcc.gn5.c28m240.2A100-80g	2张	28核	240 GiB	50 Gbps	1250万	16
bcc.gn5.c56m480.4A100-80g	4张	56核	480 GiB	100 Gbps	2500万	32
bcc.gn5.c112m960.8A100-80g	8张	112核	960 GiB	200 Gbps	5000万	32

Ampere A10系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景

- 图形渲染：3D图形渲染、图形设计、云游戏等场景

- 规格特点：
  - 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla A10，支持RTX光线追踪算法加速， FP16算力达到125TFLOPS，单GPU显存24GB
  - 内存：最高可提供476 GiB内存，处理器与内存配比为1:4
  - 存储：支持CDS云磁盘
  - 网络：最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求

实例规格

实例	A10显卡数量	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gn5.c16m64.1a10	1张	16核	64 GiB	6 Gbps	120万	8
bcc.gn5.c32m128.2a10	2张	32核	128 GiB	12 Gbps	250万	16
bcc.gn5.c64m256.4a10	4张	64核	256 GiB	18 Gbps	450万	32
bcc.gn5.c28m112.1a10	1张	28核	112 GiB	8 Gbps	180万	8
bcc.gn5.c56m224.2a10	2张	56核	224 GiB	14 Gbps	300 万	16
bcc.gn5.c112m476.4a10	4张	112核	476 GiB	25 Gbps	800万	32

Ampere A100-40G系列

- 适用场景：
  - 人工智能大规模训练：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能计算：生命科学、工业制造仿真、气象预测、计算物理等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景

- 规格特点：
  - 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla A100， FP16算力达到312TFLOPS，单GPU显存40GB HBM2，支持双向带宽600GB的Nvlink互联
  - 内存：最高可提供896 GiB内存，处理器与内存配比为1:8
  - 存储：支持CDS云磁盘
  - 网络：最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求

实例规格

实例	A100显卡数量	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gn5.c112m896.8A100-40g	8张	112核	896 GiB	25 Gbps	700万	16

计算型gn3

搭载Intel Xeon Cascade系列以及英伟达主流数据中心级GPU

Volta V100系列

- 适用场景：
  - 人工智能训练及大模型推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能HPC计算：生命科学、CAE、气象预测、碰撞仿真等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
  - 编解码：视频流、图片流编解码等场景
- 规格特点：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla V100 SXM2，FP16算力达到125TFLOPS，单GPU显存32GB HBM2，GPU互联支持带宽300GB/s NVLink
  - 内存：最高可提供640 GiB内存
  - 存储：支持CDS云磁盘
  - 网络：最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求

实例规格

实例	V100-32GB显卡数量	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gn3.c10m80.1v100-32g	1张	10核	80 GiB	4 Gbps	150万	8
bcc.gn3.c20m160.2v100-32g	2张	20核	160 GiB	7 Gbps	280万	8
bcc.gn3.c40m320.4v100-32g	4张	40核	320 GiB	13 Gbps	500万	16
bcc.gn3.c80m640.8v100-32g	8张	80核	640 GiB	25 Gbps	700万	16

Turing T4系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
  - 编解码：视频流、图片流编解码等场景
- 规格特点：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla T4，INT8算力达到130TFLOPS，单GPU显存16GB
  - 内存：最高可提供320 GiB内存
  - 存储：支持CDS云磁盘
  - 网络：超高网络收发包能力，满足极高的内网传输需求

实例规格

实例	T4显卡数量	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
bcc.gn3.c20m80.1t4	1张	20核	80 GiB	6 Gbps	280 万	8
bcc.gn3.c40m160.2t4	2张	40核	160 GiB	11 Gbps	500万	16
bcc.gn3.c80m320.4t4	4张	80核	320 GiB	22 Gbps	700 万	16

弹性裸金属实例规格族

弹性裸金属本地SSD型I8

规格族特点

- 处理器：使用Intel® Xeon® 6 Granite Rapids-AP处理器，主频 3.3GHz，全核睿频 3.7GHz，单核性能强劲
- 处理器与内存配比为1:4
- IO能力综合提高，本地盘性能相较于前代产品实现大幅增长
- 支持eRDMA能力
- 适用场景
  - 游戏服务器、高频交互场景（如直播视频弹幕）
  - 数据分析、批量计算
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	本地存储	内网基础带宽	网络收发包PPS	主网卡队列数
ebc.I8.c192m768.2d	192 核	768 GiB	2* 3570GiB	160 Gbps	3600 万	64

弹性裸金属本地SSD型Ia4

规格族特点

- 处理器：使用AMD EPYC™ Turin Classic处理器，主频 2.7GHz，最大睿频 4.1GHz
- 处理器与内存配比为1:4
- IO能力综合提高，本地盘性能相较于前代产品实现大幅增长
- 支持eRDMA能力
- 适用场景
  - Web应用类后端服务器
  - 数据分析、批量计算
  - 工程应用、仿真处理
  - 视频编解码

实例规格

实例规格	vCPU	内存	本地存储	内网基础带宽	网络收发包PPS	主网卡队列数	云盘带宽
ebc.Ia4.c256m1152.2d	256 核	1152 GiB	2* 3570GiB	160 Gbps	3600 万	64	32 Gbps
ebc.Ia4.c256m1152.16d	256 核	1152 GiB	16* 3570GiB	160 Gbps	3600 万	64	32 Gbps

弹性裸金属本地SSD型hcl7

规格族特点

- 处理器：使用Intel® 5th Xeon® Emerald Rapids处理器，主频 3.3GHz，全核睿频 3.4GHz
- 处理器与内存配比为1:8
- 提供整体性能最优的实例参数配置，支持自定义超线程、NUMA，最大化满足多个高主频类实例场景
- 适用场景
  - 游戏服务器、高频交互场景（如直播视频弹幕）
  - 数据分析、批量计算
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	本地存储（块数*容量）	内网基础带宽	网络收发包PPS	主网卡队列数
ebc.hcl7.c128m1024.4d	128 核	1024 GiB	4 *3570 GiB	160 Gbps	3000 万	104

弹性裸金属本地SSD型l7

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4.92
- 本地NVMe SSD存储，数据物理隔离，高效安全。
- 适用场景
  - 数据库业务
  - 在线计算场景下的分布式缓存等业务

实例规格

实例规格	vCPU	内存	本地存储（块数*容量）	内网基础带宽	网络收发包PPS	队列数	云盘带宽
ebc.l7.c208m1024.2d	208 核	1024 GiB	2 *3570 GiB	160 Gbps	3000 万	64	32 Gbps
ebc.l7.c208m1024.4d	208 核	1024 GiB	4 *3570 GiB	160 Gbps	3000 万	64	32 Gbps

弹性裸金属服务器通用型g7

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4.9
- 适用场景
  - 网站和应用服务器
  - 分布式缓存系统

- 搜索集群
- 常规数据分析和计算任务

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
ebc.g7.c208m1024	208 核	1024 GiB	160 Gbps	3000 万	64

弹性裸金属服务器本地SSD型la3

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：AMD EPYC 9W24，主频 2.6GHz，睿频 3.7GHz
- 处理器与内存配比为1:4
- 本地NVMe SSD存储，数据物理隔离，高效安全。
- 超高网络性能，带来数据交互极致体验。
- 适用场景
  - 数据库业务
  - 在线计算场景下的分布式缓存等业务

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
ebc.la3.c384m1536.2d	384 核	1536 GiB	2块*3570 GiB	160 Gbps	3000 万	64

弹性裸金属服务器本地SSD型l5

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4或1:8
- 本地NVMe SSD存储，数据物理隔离，高效安全。
- 适用场景
  - 数据库业务
  - 在线计算场景下的分布式缓存等业务

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
ebc.l5.c128m512.1d	128 核	512 GiB	1 块*3570 GiB	40 Gbps	1000 万	32
ebc.l5.c128m512.4d	128 核	512 GiB	4 块*3570 GiB	40 Gbps	1000 万	32
ebc.l5.c128m1024.1d	128 核	1024 GiB	1 块*3570 GiB	40 Gbps	1000 万	32

弹性裸金属服务器通用型g5

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- 处理器与内存配比为1:4
- 适用场景
  - 网站和应用服务器
  - 分布式缓存系统
  - 搜索集群
  - 常规数据分析和计算任务

实例规格

实例规格	vCPU	内存	内网基础带宽	网络收发包PPS	队列数
ebc.g5.c128m512	128 核	512 GiB	40 Gbps	1000 万	32

弹性裸金属服务器本地SSD型la2ne

规格族特点

- 使用百度太行架构，提供云环境中独享的高性能服务器，实现资源物理隔离。
- 处理器：AMD EPYC 7W83，主频 2.55GHz，睿频 3.5GHz
- 处理器与内存配比为1:4
- 本地NVMe SSD存储，数据物理隔离，高效安全。
- 超高网络性能，带来数据交互极致体验。
- 支持eRDMA
- 适用场景
  - 数据库业务
  - 在线计算场景下的分布式缓存等业务

实例规格

实例规格	vCPU	内存	本地盘（块数*GiB）	内网基础带宽	网络收发包PPS	队列数
ebc.la2ne.c256m1024.2d	256 核	1024 GiB	2块*3570 GiB	180 Gbps	3000 万	64
ebc.la2ne.c256m1024.12d	256 核	1024 GiB	12块*3570 GiB	180 Gbps	3000 万	64

弹性裸金属服务器GPU型gn7s

> 此规格族暂时白名单开放，如有需求，请提交工单

适用场景：

- 人工智能推理
- 图形、视频渲染

规格特点：

- 具备百度太行·弹性裸金属服务器的特性
- 支持在创建或重装实例时选择公共镜像以及兼容的GPU驱动版本
- 处理器：Intel Xeon Platinum 8563C，主频 2.6GHz，睿频 3.1GHz
- GPU：NVIDIA Tesla L20，单GPU FP16算力达到120TFLOPS，单GPU显存40GB GDDR6
- 网络：支持160G VPC网络带宽

实例规格

实例规格	vCPU	内存	GPU数量	GPU总显存	VPC带宽	网络收发包PPS	队列数
ebc.gn7s.c208m1024.8l20	208 核	1024 GiB	8 张	384G	160 Gbps	3000 万	64

弹性高性能计算集群

弹性高性能计算集群GPU型GN6T

> 此规格族暂时白名单开放，如有需求，请提交工单

适用场景：

- 人工智能超大规模训练：例如无人驾驶、NLP大模型、广告推荐等场景
- 高性能计算：生命科学、计算物理等场景

规格特点：

- 具备百度太行·弹性裸金属服务器的特性
- GPU镜像包含GPU驱动、CUDA、OFED网卡驱动等必备环境。
- 处理器：Intel Xeon Platinum 8463B，主频 2.6GHz
- GPU：NVIDIA Tesla H800 SXM，FP16算力达到989TFLOPS，单GPU显存80GB HBM，支持双向带宽400GB的Nvlink互联
- 存储：可支持16TB容量全闪存本地SSD存储，支持GPU Direct Storage
- 网络：支持180G VPC网络带宽及3200G的RDMA专有网络，并支持GPU Direct RDMA特性

实例规格

实例规格	vCPU	内存	GPU数量	GPU总显存	VPC带宽	本地盘（块数*GiB）	网络收发包PPS	队列数
ehc.lgn6t.c192m3072.8h800.8re.4d	192 核	3072 GiB	8 张	640 G	180 Gbps	4块*3570GiB	3000 万	64

弹性高性能计算集群GPU型GN5

> 此规格族暂时白名单开放，如有需求，请提交工单

适用场景：

- 人工智能超大规模训练：例如无人驾驶、NLP大模型、广告推荐等场景

- 高性能计算：生命科学、计算物理等场景

规格族特点：

- 具备百度太行·弹性裸金属服务器的特性
- GPU镜像包含GPU驱动、CUDA、OFED网卡驱动等必备环境。
- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- GPU：NVIDIA Tesla A800 SXM，FP16算力达到312TFLOPS，单GPU显存80GB HBM2，支持双向带宽400GB的Nvlink互联
- 存储：可支持16TB容量全闪存本地SSD存储，支持GPU Direct Storage
- 网络：支持180G VPC网络带宽及800G的RDMA专有网络，并支持GPU Direct RDMA特性

实例规格

实例规格	vCPU	内存	GPU数量	GPU总显存	VPC带宽	本地盘（块数*GiB）	网络收发包PPS	队列数
ehc.lgn5.c128m1024.8a800.8re.4d	128 核	1024 GiB	8 张	640 G	180 Gbps	4块*3570GiB	3000 万	64

在公测中/邀测阶段的规格族，产品或功能已准备好接受客户测试。公测中规格族通常会公开发布，邀测规格族白名单开放，可通过提交工单的方式申请创建。该类实例功能不一定完整，不提供服务等级协议 (SLA)，如遇到问题，可以提交工单反馈。

公测中/邀测阶段平均持续约1-2个月。

已停售实例规格

本文所列实例规格均已停售，为保证您业务资源的高可拓展性，推荐您使用BCC在售的实例规格。详情请参考：实例规格

变配说明

如果您持有已停售规格的实例，建议变配至其他在售规格。关于实例规格之间支持变配的情况，请参见支持变配的实例规格。

实例规格族

实例规格族	实例类型
通用型实例族	通用网络增强型g3ne
	通用型g3
	通用型g2
	通用型g1
计算型实例族	密集计算型ic3
	计算网络增强型c3ne
	计算型c3
	计算型c2

	密集计算型ic2
	计算型c1
	密集计算型ic1
内存型实例族	内存网络增强型m3ne
	内存型m3
	内存型m2
	内存型m1
高主频型实例族	高主频通用型hcg3
	高主频计算型hcc3
	高主频通用型hcg2
	高主频计算型hcc2
	高主频通用型hcg1
	高主频计算型hcc1
本地SSD型实例族	本地SSD型I2
	本地SSD型I1

通用型实例族

通用网络增强型g3ne

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:4
- 最高可支持22Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 数据分析和计算

- 各种类型和规模的企业级应用
- 中小型数据库系统、缓存、搜索集群
- 计算集群、依赖内存的数据处理
- 高网络包收发场景，如视频弹幕、电信业务转发等

实例规格

实例规格	vCPU	内存	内网带宽	网络收发包PPS
bcc.g3ne.c1m4	1 核	4 GiB	1.5 Gbps	30 万PPS
bcc.g3ne.c2m8	2 核	8 GiB	1.5 Gbps	30 万PPS
bcc.g3ne.c4m16	4 核	16 GiB	1.5 Gbps	50 万PPS
bcc.g3ne.c8m32	8 核	32 GiB	3 Gbps	100 万PPS
bcc.g3ne.c16m64	16 核	64 GiB	6 Gbps	160 万PPS
bcc.g3ne.c24m96	24 核	96 GiB	8 Gbps	200 万PPS
bcc.g3ne.c32m128	32 核	128 GiB	11 Gbps	250 万PPS
bcc.g3ne.c64m256	64 核	256 GiB	22 Gbps	400 万PPS

通用型g3

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:4等多种比例
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 数据分析和计算
  - 各种类型和规模的企业级应用
  - 中小型数据库系统、缓存、搜索集群
  - 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网带宽
bcc.g3.c1m4	1 核	4 GiB	1 Gbps
bcc.g3.c2m8	2 核	8 GiB	1.5 Gbps
bcc.g3.c2m12	2 核	12 GiB	1.5Gbps
bcc.g3.c4m16	4 核	16 GiB	1.5Gbps
bcc.g3.c4m12	4 核	12 GiB	1.5Gbps
bcc.g3.c8m12	8 核	12 GiB	2 Gbps
bcc.g3.c8m24	8 核	24 GiB	2 Gbps
bcc.g3.c8m32	8 核	32 GiB	2 Gbps
bcc.g3.c12m16	12 核	16 GiB	2.5 Gbps
bcc.g3.c12m32	12 核	32 GiB	2.5 Gbps
bcc.g3.c12m48	12 核	48 GiB	2.5 Gbps
bcc.g3.c12m64	12 核	64 GiB	2.5 Gbps
bcc.g3.c16m24	16 核	24 GiB	3 Gbps
bcc.g3.c16m48	16 核	48 GiB	3 Gbps
bcc.g3.c16m64	16 核	64 GiB	3 Gbps
bcc.g3.c24m96	24 核	96 GiB	4.5 Gbps
bcc.g3.c24m128	24 核	128 GiB	4.5 Gbps
bcc.g3.c32m128	32 核	128 GiB	6 Gbps
bcc.g3.c48m128	48 核	128 GiB	8.5 Gbps
bcc.g3.c48m192	48 核	192 GiB	8.5 Gbps
bcc.g3.c64m256	64 核	256 GiB	10 Gbps

通用型g2

规格族特点

- 处理器：Intel Xeon E5-2680 Broadwell（v4），主频 2.4GHz
- 处理器与内存配比为1:4等多种比例
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 数据分析和计算
  - 各种类型和规模的企业级应用
  - 中小型数据库系统、缓存、搜索集群
  - 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网带宽
bcc.g2.c1m4	1 核	4 GiB	1 Gbps
bcc.g2.c2m8	2 核	8 GiB	1.5 Gbps
bcc.g2.c2m12	2 核	12 GiB	1.5 Gbps
bcc.g2.c4m12	4 核	12 GiB	1.5 Gbps
bcc.g2.c4m16	4 核	16 GiB	1.5 Gbps
bcc.g2.c8m12	8 核	12 GiB	1.5 Gbps
bcc.g2.c8m24	8 核	24 GiB	1.5 Gbps
bcc.g2.c8m32	8 核	32 GiB	1.5 Gbps
bcc.g2.c12m16	12 核	16 GiB	2.5 Gbps
bcc.g2.c12m32	12 核	32 GiB	2.5 Gbps
bcc.g2.c12m48	12 核	48 GiB	2.5 Gbps
bcc.g2.c16m24	16 核	24 GiB	3 Gbps
bcc.g2.c16m48	16 核	48 GiB	3 Gbps
bcc.g2.c16m64	16 核	64 GiB	3 Gbps
bcc.g2.c32m128	32 核	128 GiB	6 Gbps
bcc.g2.c48m128	48 核	128 GiB	8.5 Gbps
bcc.g2.c48m192	48 核	192 GiB	8.5 Gbps

通用型g1

规格族特点

- 处理器：Intel Xeon Silver 4110 /Intel Xeon E5-2620 V4 /Intel Xeon E5-2650 V4 /Intel Xeon E5-2620 V3 /Intel Xeon E5-2620 V2
- 处理器与内存配比为1:4等多种比例
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
    - 数据分析和计算
    - 各种类型和规模的企业级应用
    - 中小型数据库系统、缓存、搜索集群
    - 计算集群、依赖内存的数据处理

实例规格

实例规格	vCPU	内存	内网带宽
bcc.g1.c1m4	1 核	4 GiB	1 Gbps
bcc.g1.c2m8	2 核	8 GiB	1.5 Gbps
bcc.g1.c2m12	2 核	12 GiB	1.5 Gbps
bcc.g1.c4m12	4 核	12 GiB	1.5 Gbps
bcc.g1.c4m16	4 核	16 GiB	1.5 Gbps
bcc.g1.c8m12	8 核	12 GiB	2 Gbps
bcc.g1.c8m24	8 核	24 GiB	2 Gbps
bcc.g1.c8m32	8 核	32 GiB	2 Gbps
bcc.g1.c12m16	12 核	16 GiB	2.5 Gbps
bcc.g1.c12m32	12 核	32 GiB	2.5 Gbps
bcc.g1.c12m48	12 核	48 GiB	2.5 Gbps
bcc.g1.c12m64	12 核	64 GiB	2.5 Gbps
bcc.g1.c16m24	16 核	24 GiB	3 Gbps
bcc.g1.c16m48	16 核	48 GiB	3 Gbps
bcc.g1.c16m64	16 核	64 GiB	3 Gbps

计算型实例族

密集计算型ic3

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
    - Web前端服务器
    - 大型多人在线游戏（MMO）前端
    - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.ic3.c1m1	1 核	1 GiB	1 Gbps
bcc.ic3.c2m2	2 核	2 GiB	1.5 Gbps
bcc.ic3.c4m4	4 核	4 GiB	1.5 Gbps
bcc.ic3.c8m8	8 核	8 GiB	2 Gbps
bcc.ic3.c12m12	12 核	12 GiB	2.5 Gbps
bcc.ic3.c16m16	16 核	16 GiB	3 Gbps

计算网络增强型c3ne

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:2
- 最高可支持22Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用
  - 高网络包收发场景，如视频弹幕、电信业务转发等

实例规格

实例规格	vCPU	内存	内网带宽	网络收发包
bcc.c3ne.c1m2	1 核	2 GiB	1.5 Gbps	30 万PPS
bcc.c3ne.c2m4	2 核	4 GiB	1.5 Gbps	30 万PPS
bcc.c3ne.c4m8	4 核	8 GiB	1.5 Gbps	50 万PPS
bcc.c3ne.c8m16	8 核	16 GiB	3 Gbps	100 万PPS
bcc.c3ne.c16m32	16 核	32 GiB	6 Gbps	160 万PPS
bcc.c3ne.c24m48	24 核	48 GiB	8 Gbps	200 万PPS
bcc.c3ne.c32m64	32 核	64 GiB	11 Gbps	250 万PPS
bcc.c3ne.c64m128	64 核	128 GiB	22 Gbps	400 万PPS

计算型c3

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.c3.c1m2	1 核	2 GiB	1 Gbps
bcc.c3.c2m4	2 核	4 GiB	1.5 Gbps
bcc.c3.c4m8	4 核	8 GiB	1.5 Gbps
bcc.c3.c8m16	8 核	16 GiB	2 Gbps
bcc.c3.c12m24	12 核	24 GiB	2.5 Gbps
bcc.c3.c16m32	16 核	32 GiB	3 Gbps
bcc.c3.c24m48	24 核	48 GiB	4.5 Gbps
bcc.c3.c32m64	32 核	64 GiB	6 Gbps
bcc.c3.c48m96	48 核	96 GiB	8.5 Gbps

计算型c2

规格族特点

- 处理器：Intel Xeon E5-2680 Broadwell（v4），主频 2.4GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.c2.c1m2	1 核	2 GiB	1 Gbps
bcc.c2.c2m4	2 核	4 GiB	1.5 Gbps
bcc.c2.c4m8	4 核	8 GiB	1.5 Gbps
bcc.c2.c8m16	8 核	16 GiB	1.5 Gbps
bcc.c2.c12m24	12 核	24 GiB	2.5 Gbps
bcc.c2.c16m32	16 核	32 GiB	3 Gbps
bcc.c2.c32m64	32 核	64 GiB	6 Gbps
bcc.c2.c48m96	48 核	96 GiB	8.5 Gbps

密集计算型ic2

规格族特点

- 处理器：Intel Xeon E5-2680 Broadwell（v4），主频 2.4GHz
- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高

- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.ic2.c1m1	1 核	1 GiB	1 Gbps
bcc.ic2.c2m2	2 核	2 GiB	1.5 Gbps
bcc.ic2.c4m4	4 核	4 GiB	1.5 Gbps
bcc.ic2.c8m8	8 核	8 GiB	1.5 Gbps
bcc.ic2.c12m12	12 核	12 GiB	2.5 Gbps
bcc.ic2.c16m16	16 核	16 GiB	3 Gbps

计算型c1

规格族特点

- 处理器：Intel Xeon Silver 4110 /Intel Xeon E5-2620 V4 /Intel Xeon E5-2650 V4 /Intel Xeon E5-2620 V3 /Intel Xeon E5-2620 V2
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码
  - 高性能科学和工程应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.c1.c1m2	1 核	2 GiB	1 Gbps
bcc.c1.c2m4	2 核	4 GiB	1.5 Gbps
bcc.c1.c4m8	4 核	8 GiB	1.5 Gbps
bcc.c1.c8m16	8 核	16 GiB	2 Gbps
bcc.c1.c12m24	12 核	24 GiB	2.5 Gbps
bcc.c1.c16m32	16 核	32 GiB	3 Gbps

密集计算型ic1

规格族特点

- 处理器：Intel Xeon Silver 4110 /Intel Xeon E5-2620 V4 /Intel Xeon E5-2650 V4 /Intel Xeon E5-2620 V3 /Intel Xeon E5-2620 V2

- 处理器与内存配比为1:1
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - Web前端服务器
  - 大型多人在线游戏（MMO）前端
  - 数据分析、批量计算、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.ic1.c1m1	1 核	1 GiB	1 Gbps
bcc.ic1.c2m2	2 核	2 GiB	1.5 Gbps
bcc.ic1.c4m4	4 核	4 GiB	1.5 Gbps
bcc.ic1.c8m8	8 核	8 GiB	2 Gbps
bcc.ic1.c12m12	12 核	12 GiB	2.5 Gbps
bcc.ic1.c16m16	16 核	16 GiB	3 Gbps

内存型实例族

内存网络增强型m3ne

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:8
- 最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用
  - 高网络包收发场景，如视频弹幕、电信业务转发等

实例规格

实例规格	vCPU	内存	内网带宽	网络收发包PPS	队列数
bcc.m3ne.c1m8	1 核	8 GiB	1.5 Gbps	30 万PPS	1
bcc.m3ne.c2m16	2 核	16 GiB	1.5 Gbps	40 万PPS	2
bcc.m3ne.c4m32	4 核	32 GiB	1.5 Gbps	80 万PPS	4
bcc.m3ne.c8m64	8 核	64 GiB	3 Gbps	150 万PPS	8
bcc.m3ne.c16m128	16 核	128 GiB	6 Gbps	280 万PPS	8
bcc.m3ne.c24m192	24 核	192 GiB	8 Gbps	400 万PPS	12
bcc.m3ne.c32m256	32 核	256 GiB	11 Gbps	500 万PPS	16

内存型m3

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.m3.c1m8	1 核	8 GiB	1 Gbps
bcc.m3.c2m16	2 核	16 GiB	1.5 Gbps
bcc.m3.c4m32	4 核	32 GiB	1.5 Gbps
bcc.m3.c8m64	8 核	64 GiB	2 Gbps
bcc.m3.c16m128	16 核	128 GiB	3 Gbps
bcc.m3.c32m256	32 核	256 GiB	6 Gbps

内存型m2

规格族特点

- 处理器：Intel Xeon E5-2680 Broadwell（v4），主频 2.4GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.m2.c1m8	1 核	8 GiB	1 Gbps
bcc.m2.c2m16	2 核	16 GiB	1.5 Gbps
bcc.m2.c4m32	4 核	32 GiB	1.5 Gbps
bcc.m2.c8m64	8 核	64 GiB	1.5 Gbps
bcc.m2.c16m128	16 核	128 GiB	3 Gbps

内存型m1

规格族特点

- 处理器：Intel Xeon Silver 4110 /Intel Xeon E5-2620 V4 /Intel Xeon E5-2650 V4 /Intel Xeon E5-2620 V3 /Intel Xeon E5-2620 V2
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能数据库、内存数据库
  - 数据分析与挖掘、分布式内存缓存
  - Hadoop、Spark群集以及其他企业大内存需求应用

实例规格

实例规格	vCPU	内存	内网带宽
bcc.m1.c1m8	1 核	8 GiB	1 Gbps
bcc.m1.c2m16	2 核	16 GiB	1.5 Gbps
bcc.m1.c4m32	4 核	32 GiB	1.5 Gbps
bcc.m1.c8m64	8 核	64 GiB	2 Gbps

高主频实例族

高主频通用型hcg3

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz
- 处理器与内存配比为1:4
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能Web前段服务器
  - 高性能科学和工程应用
  - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcg3.c2m8	2 核	8 GiB	1.5 Gbps
bcc.hcg3.c4m16	4 核	16 GiB	1.5 Gbps
bcc.hcg3.c8m32	8 核	32 GiB	2 Gbps
bcc.hcg3.c16m64	16 核	64 GiB	3 Gbps
bcc.hcg3.c24m96	24 核	96 GiB	4.5 Gbps
bcc.hcg3.c32m128	32 核	128 GiB	6 Gbps
bcc.hcg3.c48m192	48 核	192 GiB	8.5 Gbps
bcc.hcg3.c64m256	48 核	256 GiB	10 Gbps

高主频计算型hcc3

规格族特点

- 处理器：Intel Xeon Gold 6271C，主频 2.6GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能Web前段服务器
  - 高性能科学和工程应用
  - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcc3.c2m4	2 核	4 GiB	1.5 Gbps
bcc.hcc3.c4m8	4 核	8 GiB	1.5 Gbps
bcc.hcc3.c8m16	8 核	16 GiB	2 Gbps
bcc.hcc3.c16m32	16 核	32 GiB	3 Gbps
bcc.hcc3.c24m48	24 核	48 GiB	4.5 Gbps
bcc.hcc3.c32m64	32 核	64 GiB	6 Gbps
bcc.hcc3.c48m96	48 核	96 GiB	8.5 Gbps
bcc.hcc3.c64m128	64 核	128 GiB	10 Gbps

高主频通用型hcg2

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，睿频 3.1GHz
- 处理器与内存配比为1:4、1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能Web前段服务器
  - 高性能科学和工程应用
  - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcg2.c2m8	2 核	8 GiB	1.5 Gbps
bcc.hcg2.c2m16	2 核	16 GiB	1.5 Gbps
bcc.hcg2.c4m16	4 核	16 GiB	1.5 Gbps
bcc.hcg2.c4m32	4 核	32 GiB	1.5 Gbps
bcc.hcg2.c8m32	8 核	32 GiB	2 Gbps
bcc.hcg2.c8m64	8 核	64 GiB	2 Gbps
bcc.hcg2.c16m64	16 核	64 GiB	3 Gbps
bcc.hcg2.c16m128	16 核	128 GiB	3 Gbps
bcc.hcg2.c24m96	24 核	96 GiB	4.5 Gbps
bcc.hcg2.c24m192	24 核	192 GiB	4.5 Gbps
bcc.hcg2.c32m128	32 核	128 GiB	6 Gbps
bcc.hcg2.c32m256	32 核	256 GiB	6 Gbps
bcc.hcg2.c48m192	48 核	192 GiB	8.5 Gbps
bcc.hcg2.c48m256	48 核	256 GiB	8.5 Gbps

高主频计算型hcc2

规格族特点

- 处理器：Intel® Xeon® Skylake 6148，睿频 3.1GHz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
    - 高性能Web前段服务器
    - 高性能科学和工程应用
    - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcc2.c2m4	2 核	4 GiB	1.5 Gbps
bcc.hcc2.c4m8	4 核	8 GiB	1.5 Gbps
bcc.hcc2.c8m16	8 核	16 GiB	2 Gbps
bcc.hcc2.c16m32	16 核	32 GiB	3 Gbps
bcc.hcc2.c24m48	24 核	48 GiB	4.5 Gbps
bcc.hcc2.c32m64	32 核	64 GiB	6 Gbps
bcc.hcc2.c48m96	48 核	96 GiB	8.5 Gbps

高主频通用型hcg1

规格族特点

- 处理器：Intel Xeon E5-2667v4，睿频 3.2Ghz

- 处理器与内存配比为1:4
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能Web前段服务器
  - 高性能科学和工程应用
  - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcg1.c4m16	4 核	16 GiB	1.5 Gbps
bcc.hcg1.c8m32	8 核	32 GiB	2 Gbps
bcc.hcg1.c16m64	16 核	64 GiB	3 Gbps
bcc.hcg1.c24m96	24 核	96 GiB	4.5 Gbps

高主频计算型hcc1

规格族特点

- 处理器：Intel Xeon E5-2667v4，睿频 3.2Ghz
- 处理器与内存配比为1:2
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 适用场景
  - 高性能Web前段服务器
  - 高性能科学和工程应用
  - MMO游戏、视频编码

实例规格

实例规格	vCPU	内存	内网带宽
bcc.hcc1.c4m8	4 核	8 GiB	1.5 Gbps
bcc.hcc1.c8m16	8 核	16 GiB	2 Gbps
bcc.hcc1.c16m32	16 核	32 GiB	3 Gbps

本地SSD型实例族

本地SSD型I2

规格族特点

- 白名单开放，如有需要请提交工单申请
- 处理器：Intel® Xeon® Skylake 6148，主频 2.4GHz
- 处理器与内存配比为1:8
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 数据盘：NVMe SSD本地盘

- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	本地存储	内网带宽
bcc.l2.c4m32	4 核	32 GiB	930 GiB	1.5 Gbps
bcc.l2.c8m64	8 核	64 GiB	460/1860 GiB	2 Gbps
bcc.l2.c16m128	16 核	128 GiB	3720 GiB	3 Gbps

本地SSD型l1

规格族特点

- 处理器：Intel® Xeon® Skylake 6148&Intel® Xeon® E5-2620，主频 2.4GHz
- 实例网络性能与规格对应，规格越高网络转发性能越强，内网带宽上限越高
- 数据盘：NVMe SSD本地盘，本地盘大小可选，范围：5-500GiB
- 适用场景
  - OLTP、高性能关系型数据库
  - NoSQL数据库（如Cassandra、MongoDB等）
  - Elasticsearch等搜索场景

实例规格

实例规格	vCPU	内存	内网带宽
bcc.l1.c1m1	1 核	1 GiB	1 Gbps
bcc.l1.c1m2	1 核	2 GiB	1 Gbps
bcc.l1.c1m4	1 核	4 GiB	1 Gbps
bcc.l1.c1m8	1 核	8 GiB	1 Gbps
bcc.l1.c2m2	2 核	2 GiB	1.5 Gbps
bcc.l1.c2m4	2 核	4 GiB	1.5 Gbps
bcc.l1.c2m8	2 核	8 GiB	1.5 Gbps
bcc.l1.c2m12	2 核	12 GiB	1.5 Gbps
bcc.l1.c2m16	2 核	16 GiB	1.5 Gbps
bcc.l1.c4m4	4 核	4 GiB	1.5 Gbps
bcc.l1.c4m8	4 核	8 GiB	1.5 Gbps
bcc.l1.c4m12	4 核	12 GiB	1.5 Gbps
bcc.l1.c4m16	4 核	16 GiB	1.5 Gbps
bcc.l1.c4m32	4 核	32 GiB	1.5 Gbps
bcc.l1.c8m8	8 核	8 GiB	2 Gbps
bcc.l1.c8m12	8 核	12 GiB	2 Gbps
bcc.l1.c8m16	8 核	16 GiB	2 Gbps
bcc.l1.c8m24	8 核	24 GiB	2 Gbps
bcc.l1.c8m32	8 核	32 GiB	2 Gbps
bcc.l1.c8m64	8 核	64 GiB	2 Gbps
bcc.l1.c12m12	12 核	12 GiB	2.5 Gbps
bcc.l1.c12m16	12 核	16 GiB	2.5 Gbps
bcc.l1.c12m24	12 核	24 GiB	2.5 Gbps
bcc.l1.c12m32	12 核	32 GiB	2.5 Gbps
bcc.l1.c12m48	12 核	48 GiB	2.5 Gbps
bcc.l1.c12m64	12 核	64 GiB	2.5 Gbps
bcc.l1.c16m16	16 核	16 GiB	3 Gbps
bcc.l1.c16m24	16 核	24 GiB	3 Gbps
bcc.l1.c16m32	16 核	32 GiB	3 Gbps
bcc.l1.c16m48	16 核	48 GiB	3 Gbps
bcc.l1.c16m64	16 核	64 GiB	3 Gbps

型号及所在区域

BCC机器型号及其所在地域对应表如下：

规格族	华北-北京	华北-保定	西南-成都	华南-广州	华东-苏州	华中-武汉	中国香港
经济型e1	✔	✔		✔	✔	✔	
通用型g7、c7、m7	✔	✔			✔	✔	
通用型ga3、ca3、ma3	✔	✔			✔	✔	
通用型g6、计算型c6、内存型m6	✔				✔		
通用型g5、计算型c5、密集计算型ic5、内存型m5	✔	✔	✔	✔	✔	✔	✔
通用型g4、计算型c4、密集计算型ic4、内存型m4	✔	✔		✔	✔	✔	
通用型ga2、计算型ca2、密集计算型ica2、内存型ma2	✔	✔	✔	✔	✔		
通用型ga1、计算型ca1、密集计算型ica1、内存型ma1	✔	✔		✔	✔		
通用型gr1、计算型cr1	✔			✔	✔		
本地SSD型la2	✔	✔	✔		✔		
本地SSD型l5d	✔						
本地SSD型l5	✔	✔		✔			
本地SSD型l3d	✔	✔		✔		✔	
大数据型d1s					✔		
大数据型d1					✔		
GPU计算型GN5	✔			✔	✔		
GPU计算型GN3	✔						
GPU计算型GNA2	✔						
弹性裸金属服务器通用型g5		✔					
弹性裸金属服务器本地SSD型l5	✔	✔	✔	✔	✔	✔	✔
弹性裸金属服务器本地SSD型la2	✔	✔	✔		✔		
弹性裸金属服务器内存增强型l5me	✔			✔	✔		
弹性裸金属服务器GPU型lgn5	✔	✔			✔		

原机型与新机型名称对应关系

BCC机型下套餐的CPU内存的比值不同，适用于的业务场景也略有差异，为便于用户基于自身的业务应用，快速选中合适的机型，我们对机型的名称进行了修改。

> **说明：**机型名称修改，对机型性能等无任何影响，仅是按照CPU和内存的比值的不同，对规格名称进行重新划分。

原机型和新机型名称对应表如下：

原机型名称	新机型名称	cpu信息	主频
普通IV	通用型g4、计算型c4、密集计算型ic4、内存型m4	Intel Xeon Gold 6271C	2.6GHz
普通III（已停售）	通用型g3、计算型c3、密集计算型ic3、内存型m3	Intel Xeon(Skylake) Gold 6148	2.4GHz
网络增强型（已停售）	通用网络增强型g3ne、计算网络增强型c3ne、内存网络增强型m3ne	Intel Xeon(Skylake) Gold 6148	2.4GHz
普通II型（已停售）	通用型g2、计算型c2、密集计算型ic2、内存型m2	Intel Xeon E5-2680v4	2.4GHz
普通型（已停售）	通用型g1、计算型c1、密集计算型ic1、内存型m1	Intel Xeon Silver 4110 /Intel Xeon E5-2620 V4 /Intel Xeon E5-2650 V4 /Intel Xeon E5-2620 V3 /Intel Xeon E5-2620 V2	-
计算优化II型（已停售）	高主频计算型hcc2、高主频通用型hcg2	Intel Xeon(Skylake) Gold 6148	3.1GHz
计算优化型（已停售）	高主频计算型hcc1、高主频通用型hcg1	Intel Xeon E5-2667v4	3.2GHz
存储优化II型（已停售）	本地SSD型I2	Intel Xeon(Skylake) Gold 6148	2.4GHz
存储优化型（已停售）	本地SSD型I1	Intel Xeon(Skylake) Gold 6148&Intel Xeon E5-2520	2.4GHz

关于每个规格族的具体说明请参考实例规格。

产品定价

计费概述

计费项

一台云服务器BCC包括BCC实例、镜像、云磁盘、带宽等资源。涉及计费的BCC资源如下表所示。

计费项	计费说明	计费方式	参考文档
实例	以实例规格的形式提供，包括CPU和内存，收取实例规格费用。	• 包年包月（预付费） • 按量付费（后付费） • 抢占实例	• 包年包月（预付费）定价 • 按量付费（后付费）定价 • 按量付费（后付费）实例关机不计费 • 抢占实例
镜像	提供公共镜像、自定义镜像、服务集成镜像、云市场镜像、共享镜像。	云市场镜像提供免费和付费镜像，具体价格以售卖页显示为准。	-
云磁盘	根据选择的云磁盘的类型及容量收费。	• 包年包月（预付费） • 按量付费（后付费）	磁盘计费方式
公网带宽	根据您是否购买弹性公网IP进行收费。	• 包年包月计费 • 按使用流量计费 • 按使用带宽计费	弹性公网IP计费方式
快照	快照账单=快照链容量×快照保存时长×快照存储单价	按量付费	快照计费方式

计费模式

BCC提供包年包月（预付费）、按量付费（后付费）及抢占实例三种计费模式。并按照不同的规格类型分为不同的规格族，关于规格族信息，请参见实例规格。

计费模式	付费方式	计费周期	计费变更	变更配置
包年包月	预付费	按订单的购买周期计费	支持转为按量付费	支持CPU、内存、网络带宽等资源从低到高进行升级
按量付费	后付费	按分钟计费，每15分钟扣费，每小时出账	支持转为包年包月	支持CPU、内存、网络带宽等资源从低到高进行升级
抢占实例	后付费	按每分钟市场价计费，按小时扣费	不支持计费变更	不支持配置变更

支付方式

购买BCC实例时，您可以选择以下支付方式：

- 百度智能云账户余额，可在 管理控制台 - 财务总览 处查看当前余额。
- 在线支付。
- 代金券支付，请确保代金券的状态为使用中且订单类型为通用。 您可以登录在 管理控制台 - 财务中心 - 代金券 查看当前拥有的代金券列表。

> **注意**：云服务器 CDS 系统盘 40 GB 免费赠送政策已于 2020 年 4 月 8 日起取消。详情请见系统盘计费方式。

退款说明

百度智能云支持包年包月（预付费）BCC实例自主退订，当您想终止服务时，基于退订规则释放资源并退还相应的款项至您的账户。

百度智能云不支持BCC预留实例券退订。

以下为百度智能云包年包月（预付费）BCC实例退订详情描述：

退款规则

- 云服务器BCC退款额度每月有最大额度限制，额度以积分的形式呈现，每月积分根据您上个月的消费情况计算而来。具体积分以退订页面展示为准。当月退款额度超限后不能再操作退款，退款额度次月1日自动更新。百度智能云会为每个账户统一发送1500积分作为退款的初始额度，初始额度一直有效，不会次月清空。若您退款时发现因额度超限无法退款，可以提交工单或联系您的销售经理进行申请。
- 包年包月（预付费）实例退订后，实例和系统盘资源直接释放，无法恢复。挂载在该实例上的CDS数据盘和EIP自动解绑，继续保留计费。
- 若资源当前有未完成的订单(例如续费/变配/计费变更等)，不支持退订；
- 退订实例购买时使用的代金券不予退还；
- 退款期间产生的费用不退还，即提交退订申请至退订成功期间产生的费用不退还。 退款产品产生的按需计费账单不退还。
- 如出现疑似异常或恶意退货，百度智能云有权否决退货申请，百度智能云在法律范围保持最终解释权。

退款金额计算

使用时长	退款说明	举例
小于等于30天	退订金额=订单金额-已消费金额。 已消费金额=资源使用时长 x 资源单价(目录价) x 1.5。 不足1天按1天来计算。	您2023年1月1日10点购买云服务器BCC包年包月1个月，花费60元现金，1月11日15点操作退订申请后，应退金额=60-11天 x（60/31）x1.5=28.06元。
大于30天	退订金额=订单金额-已消费金额。 已消费金额=资源使用时长 x 资源单价(目录价)。 不足1天按1天来计算。	某规格云服务器BCC预付费1个月价格50元，1年8.3折后价格498，2年7折后价格840元。您2022年1月1日10点购买1台云服务器BCC包年包月2年，花费840元，2023年3月3日12点操作退订申请后，使用时长为1年2个月零3天，应退金额为=840-498x1-50x2-50/30x3=237元。

操作步骤

1.登录管理控制台，单击右上角的财务，选择 **退订管理**。

2.在 退订管理 页选择需退款的实例，并单击对应的 **退订** 按钮。

3.填写退订原因，勾选同意并阅读退款规则，并单击 **确认退订**。

4.再次确认无误后单击 **确认** 按钮即完成退订操作。

5.在 **退订记录** 栏可查看已退款进程。

计费方式

包年包月（预付费）

计费规则

包年包月（预付费）计费方式，需要提前一次性支付所选时间段内的费用。可选择1-9个月或1-3年进行付费，时长周期为实例创建日起的单位周期。这种模式价格相较于按量付费（后付费）模式更低廉，适合业务稳定的长期需求。

计费价格

实例包年包月（预付费）价格信息请参见BCC价格详情。

到期提醒和处理

- 到期提醒：BCC实例服务到期前7天，系统会给您发送即将到期提醒通知。
- 到期后处理：到期后立即停服，系统会发送到期停服通知。数据为您保留7天，期间不收取费用，7天内未充值则释放，释放前1天和释放时系统都会发送释放通知。

按量付费（后付费）

计费规则

按量付费（后付费）计费方式，是按实例的实际使用量进行收费，按分钟计费，不足一分钟按一分钟计费，每15分钟扣费，每小时出账。按量付费（后付费）不需要提前支付费用，但需要保证余额大于100元，这种模式比包年包月（预付费）模式价格高一些。适合业务瞬间有较大变化的场景，资源使用完即可立即释放，降低成本。

计费价格

实例按量付费（后付费）价格信息请参见BCC价格详情。

预留实例券

预留实例券介绍

预留实例券（Reserved instance，简称RI）是一种抵扣券，可以抵扣相关配置的按需付费BCC实例账单，也能够预留实例资源，支持多种购买周期，对比包年包月预付费，预留实例券与按量付费实例这种组合模式灵活性更好。

适用范围

RI 按适用范围可分为：可用区（AZ）级别的券与区域（Region）级别的券。

适用范围	概念	优势
可用区级预留实例券（当前已支持）	实例券生效后将自动匹配并抵扣相同AZ、实例规格以及EHC集群（仅适用EHC类型实例）的后付费BCC实例账单	支持资源预留，购买预留实例券后，百度云将保证您账号下在对应的AZ中至少能有与RI相同的实例规格*台数的后付费BCC资源。
区域级预留实例券（即将支持，敬请期待）	支持抵扣该Region所有可用AZ下同等算力的后付费BCC实例账单	灵活自动抵扣。

付费方式

RI 按付费方式可分为：全预付、部分预付、0预付。

> 0预付和部份预付会根据您在百度智能云产品使用情况开放，如您在控制台无法选择这两种计费方式请提交工单。

付费方式	概念	总支付金额
全预付	购买券时支付购买周期内对应算力的总额，价格一般与预付费一致。总价最低。	总价最低
部分预付	购买券时支付购买周期内对应算力50%的费用，其他费用将于有效期内每小时扣除（类似后付费）。总价居中。	总价适中
0预付	购买券时无需支付任何费用，费用将于有效期内每小时扣除（类似后付费）。总价最高。	总价最高

预留实例券+后付费实例计费 VS 后付费实例计费

	预留实例券+后付费实例	后付费实例
定义	通过预留实例券来抵扣后付费BCC实例账单。	购买BCC时的一种付费方式（先使用后付费），分钟计费，小时出账
特点	1. **价格优惠**：总成本比后付费实例低； 2. **资源保障**：AZ级实例券支持资源预留； 3. **灵活抵扣**：实例券不与某个实例绑定，一张实例券可抵扣同规格族的不同后付费BCC账单 4. **灵活支付**：支持按每小时支付（部分预付或0预付），避免一次性支付带来的资金链压力。	1. 即用即买，随时释放 2. 单价较高

资源抵扣

实例券生效后将自动匹配相同AZ、规格族的按需付费-后付费BCC实例进行抵扣，如您购买的是EHC实例规格，则实例券会自动匹配同EHC集群的实例。您无需再为被抵扣的实例额外付费。在预留实例券到期后，按需付费-后付费BCC实例将按照实例费用出账并恢复计费。例如：

> 举例：2c4g实例，后付费：1元/小时； 4c8g实例，后付费：2元/小时；一张 2c4g Region级预留实例券0预付：预留费用0.7元/小时；一张 4c8g AZ级预留实例券0预付：预留费用1.4元/小时。
>
> 用户买了2台4c8g实例、1张2c4gRegion级预留实例券和1张4c8gAZ级预留实例券。因1张4c8gAZ级预留实例券匹配一台4c8g实例，这台4c8g实例费用为0元/小时；1张2c4gRegion级预留实例券抵扣另1台4c8g实例的2c4g算力，这台费用为(2-1)=1元/小时，总费用为2张预留实例券与1台2c4g算力的费用相加，即：(0.7+1.4+1)=3.1元/小时。

快速入门

快速配置Linux云服务器

本文主要介绍如何使用控制台快速创建一台 Linux 云服务器。如果您想了解 Windows 云服务器的入门教程，可以参考快速配置 Windows 云服务器。

注册及实名认证

使用百度智能云云服务器BCC前，您需要拥有一个百度智能云账号并完成实名认证，具体操作如下：

1. 注册并登录百度智能云平台，请参考注册和登录。
2. 完成实名认证，操作细节请参考实名认证，实名认证后才可购买百度智能云云服务器BCC。

创建实例

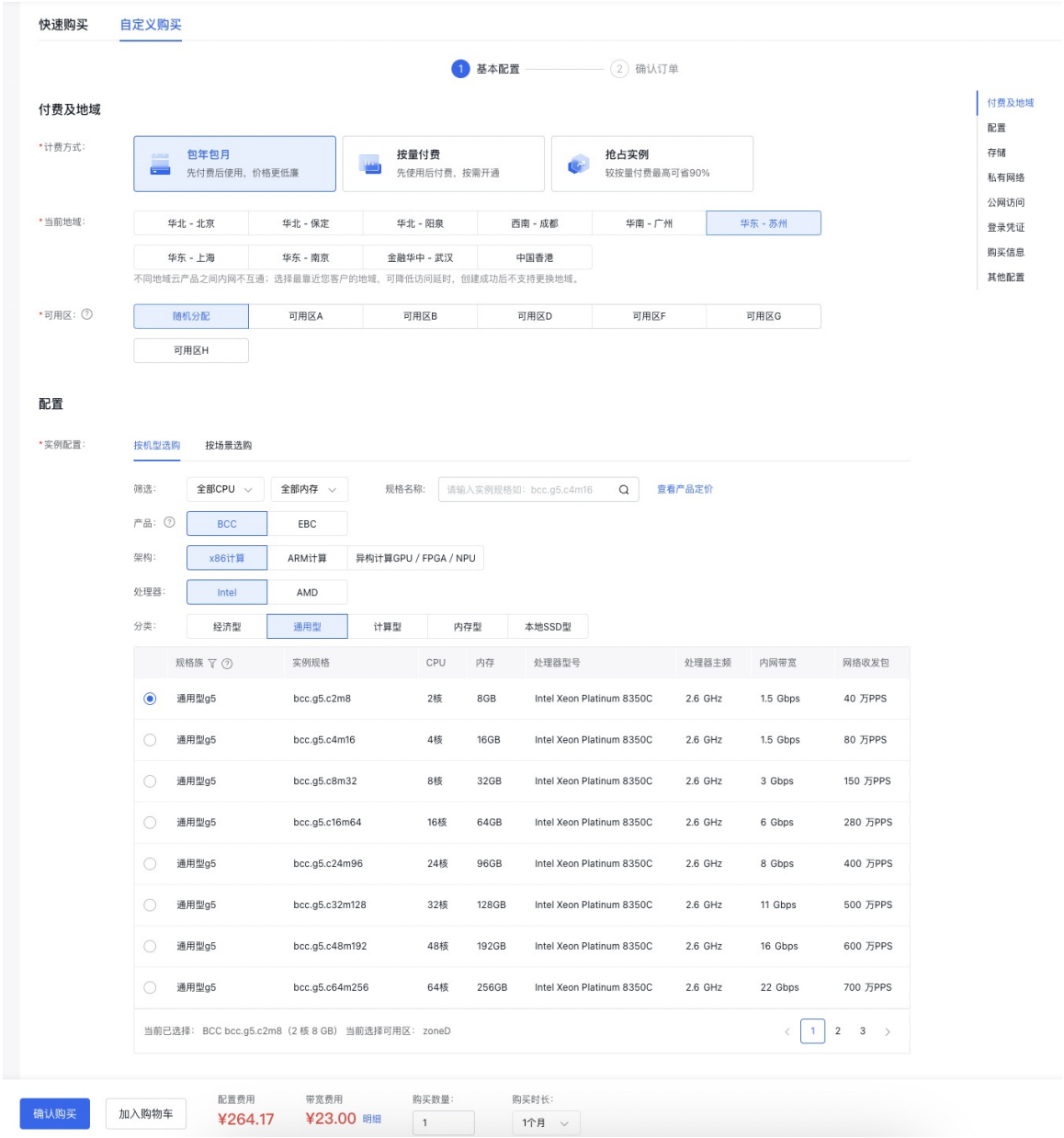
您可通过以下两种方式创建实例。

- 在 云服务器BCC产品页，单击 **立即购买**。
- 登录 管理控制台，选择 “产品服务 > 计算 > 云服务器 BCC”。在实例列表页单击 **创建实例**。

在购买产品页配置以下信息即可完成一台BCC实例的创建。

配置基本信息

1. 根据提示信息，在基本信息配置页根据需求选择相应配置。



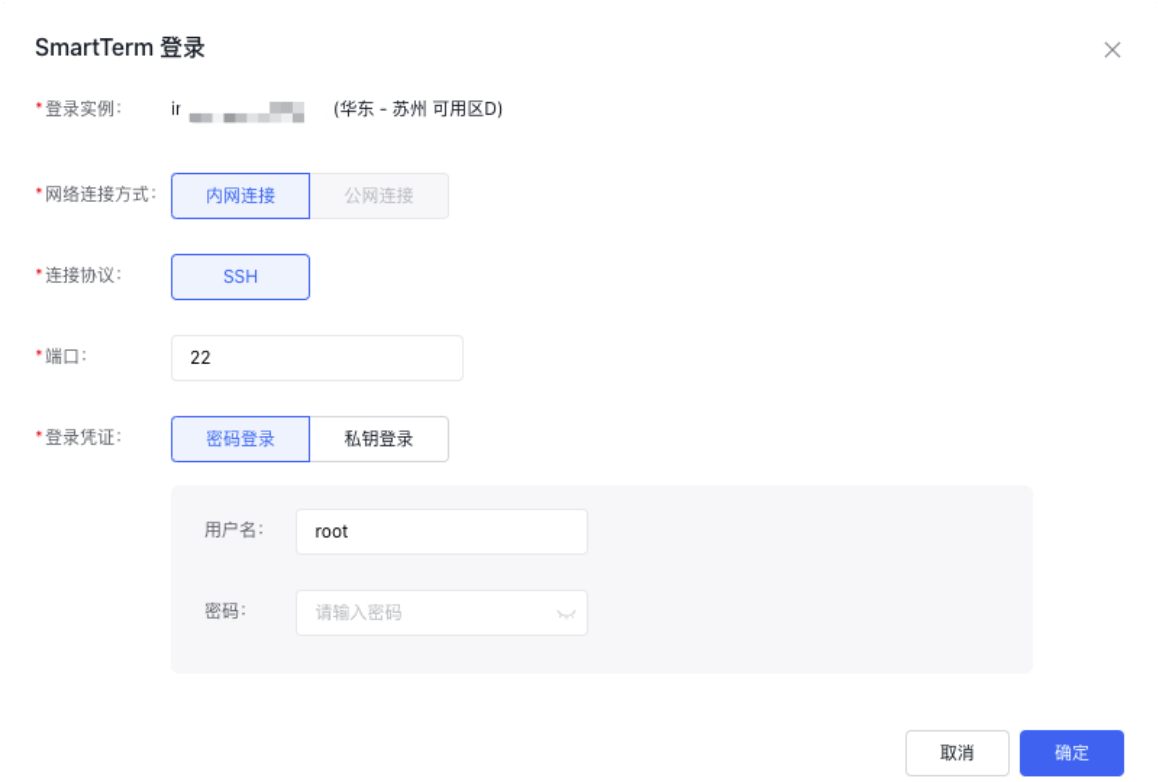
2. 单击 **确认购买**，在确认购买页确认购买信息后提交订单并完成支付，即可成功创建一台Linux云服务器。

登录云服务器

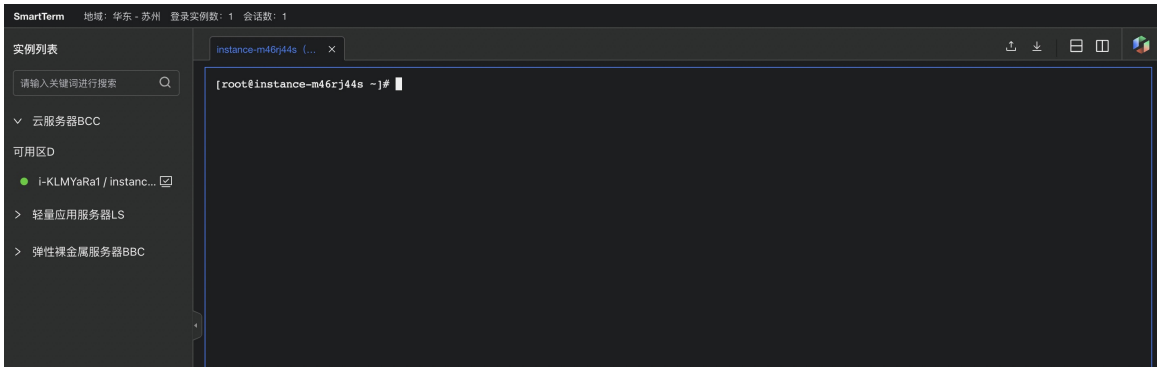
1. 登录云服务器BCC控制台，在实例列表中找到您刚购买的云服务器，在右侧操作栏中单击 **远程连接**。

2.登录方式可以选择SmartTerm或者VNC，可根据需求选择。以下操作以SmartTerm为例。

3. 在SmartTerm登录页面中，输入用户名及密码，点击 **确认**。



4. 显示[root@instance-name ~]#即为登录主机成功。

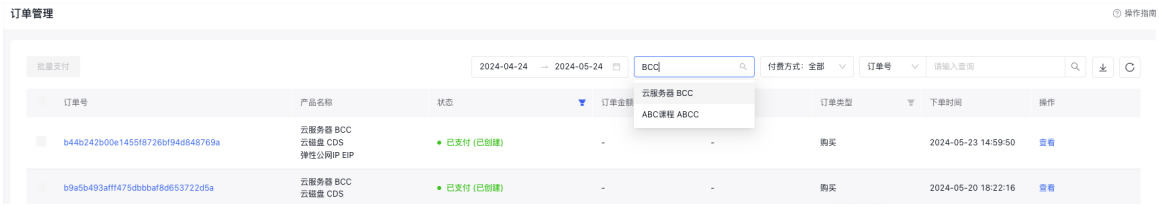


> 注意：>>只有状态为运行中的实例才可以进行远程登录。 您也可以通过其他方式登录BCC实例：参考登录方式。

查看费用账单

1. 您可在管理控制台顶部工具栏处点击 **财务-订单管理** 进入订单管理页面。

2. 在订单管理页选择云服务器BCC，并选择您购买的日期即可查看您购买的订单信息。



相关操作

- 了解云服务器BCC在售的实例规格族及适用场景：实例规格
- 了解更多创建云服务器BCC实例的方式：创建实例导航
- 了解云服务器BCC提供的API：API概览
- 了解Linux云服务器的相关操作：Linux系统搭建FTP服务

常见问题

- 忘记云服务器BCC登录密码？

  请参考 重置密码。

- 无法远程连接Linux服务器？

  请参考 无法远程连接Linux实例。

快速配置Windows云服务器

本文主要介绍如何使用控制台快速创建一台 Windows 云服务器。如果您想了解 Linux 云服务器的入门教程，可以参考快速配置 Linux 云服务器。

注册及实名认证

使用百度智能云云服务器BCC前，您需要拥有一个百度智能云账号并完成实名认证，具体操作如下：

1. 注册并登录百度智能云平台，请参考注册和登录。
2. 完成实名认证，操作细节请参考实名认证，实名认证后才可购买百度智能云云服务器BCC。

创建实例

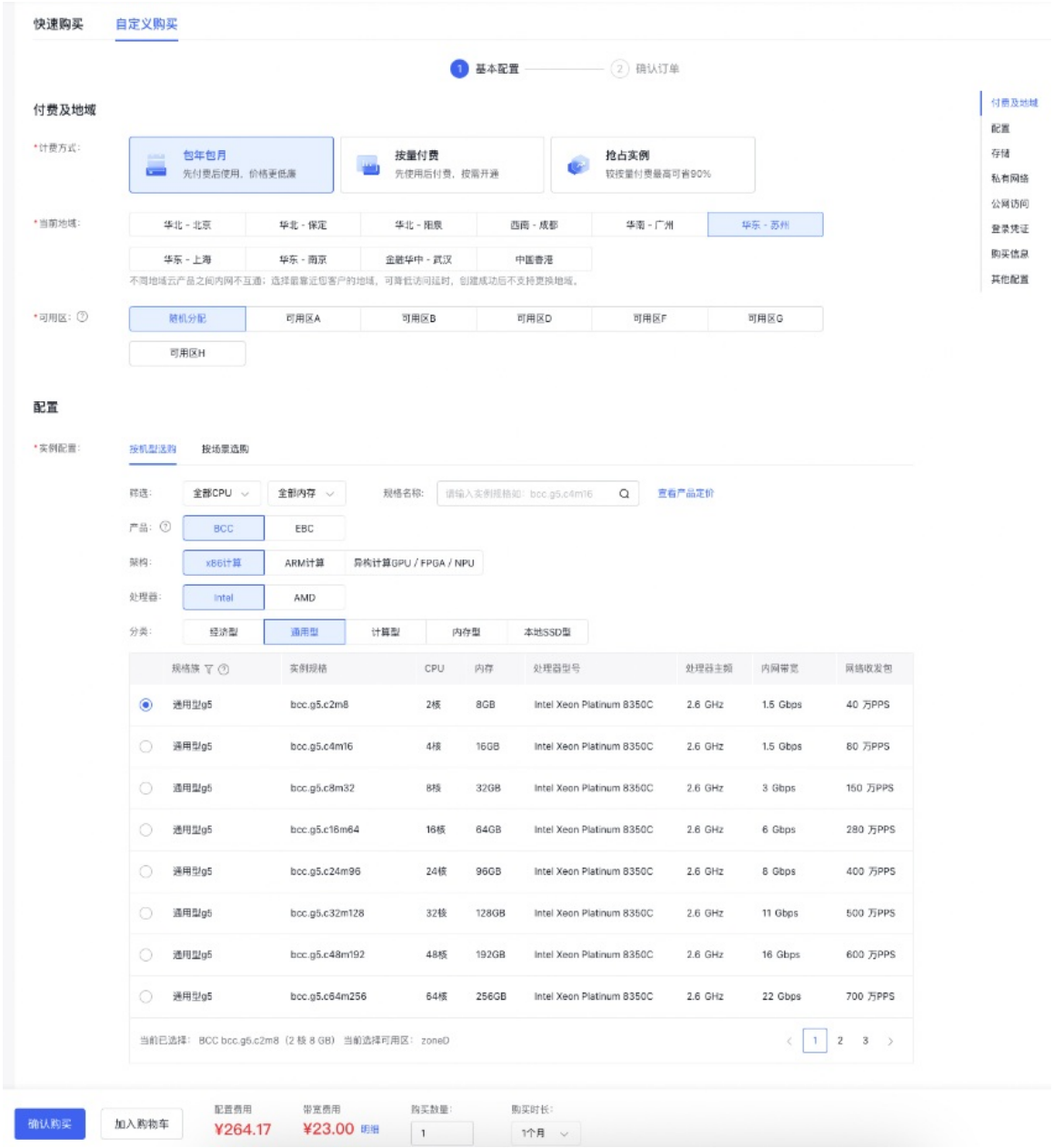
您可通过以下两种方式创建实例。

- 在 云服务器BCC产品页，单击 **立即购买**。
- 登录 管理控制台，选择 “产品服务 > 计算 > 云服务器 BCC”。在实例列表页单击 **创建实例**。

在购买产品页配置以下信息即可完成一台BCC实例的创建。

配置基本信息

1. 根据提示信息，在基本信息配置页配置以下信息。



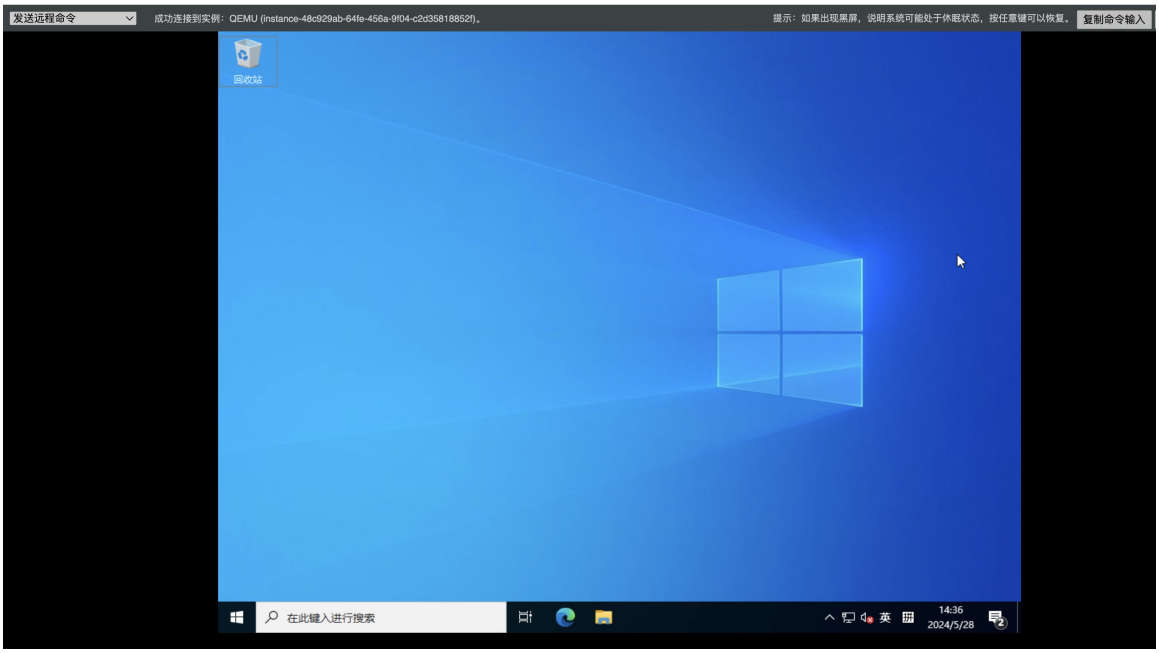
2. 单击 **确认购买**，在确认购买页确认购买信息后提交订单并完成支付，即可成功创建一台Windows云服务器。

登录云服务器

1. 登录云服务器BCC控制台，在实例列表中找到您刚购买的云服务器，在右侧操作栏中单击 **远程连接**，单击 **通过VNC登录**。

2. 在新页面中按“Ctrl + Alt + Delete” 或点击右上角登录。

3. 输入创建实例时设置的管理员密码登录主机。



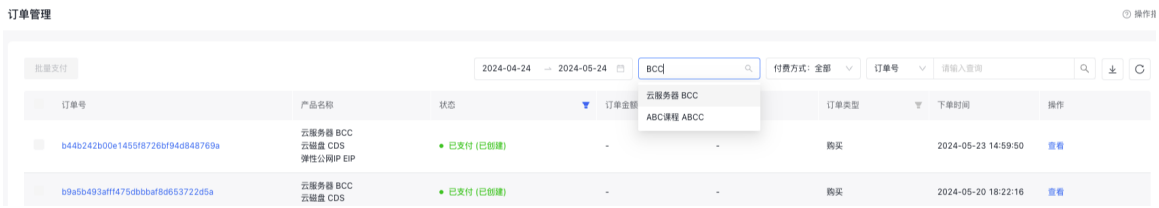
> **注意：**
>
> 只有状态为运行中的实例才可以进行VNC远程登录。

您也可以通过其他方式登录BCC实例：参考登录方式。

查看费用账单

1. 您可在管理控制台顶部工具栏处点击 **财务-订单管理** 进入订单管理页面。

2. 在订单管理页选择云服务器BCC，并选择您购买的日期即可查看您购买的订单信息。



相关操作

- 了解云服务器BCC在售的实例规格族及适用场景：实例规格
- 了解更多创建云服务器BCC实例的方式：创建实例导航
- 了解云服务器BCC提供的API：API概览
- 了解Windows云服务器的相关操作：Windows搭建IIS和FTP教程

常见问题

- 忘记云服务器BCC登录密码？

  请参考 重置密码。

- 无法远程连接Windows服务器？

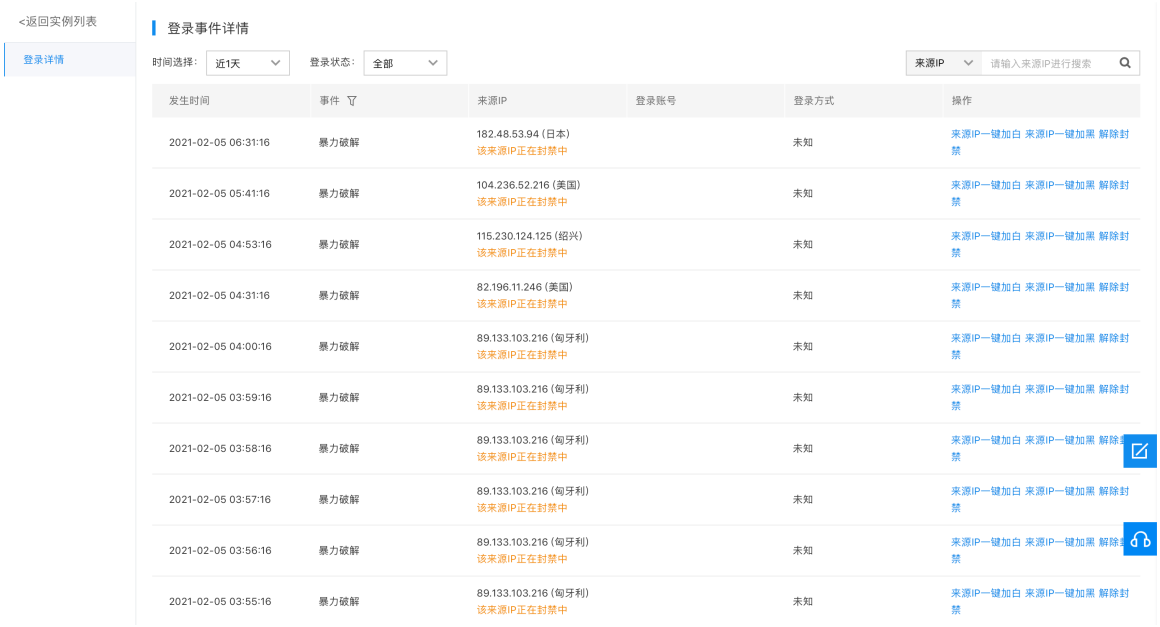
  请参考 无法远程连接Windows实例。

云服务器安全说明

安全隐患

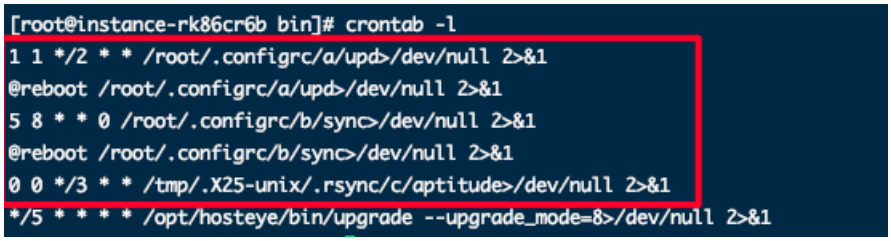
暴力破解

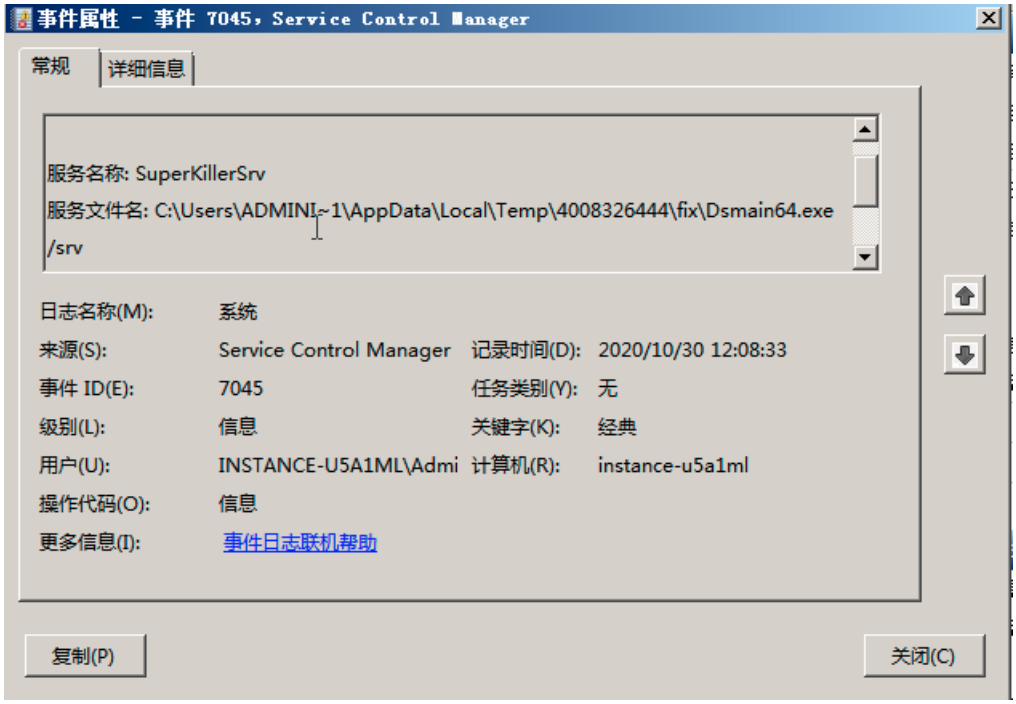
登录云控制台，在【安全-主机安全-登录管理】页面可以看到账户下所有服务器的安全防护情况，点击每台服务器最右侧有【登录详情】，可以看到暴力破解事件，百度智能云提供暴力破解的防护能力，短时间内连续登录失败6次就会被云平台自动封禁，因此这里看到的大部分事件是已经被云平台拦截了的记录，截图如下，也可以在【登录状态】里面选择状态为成功查看有无成功破解的记录。对于列出的异常IP，可以手动点击【来源IP一键加黑】从而彻底阻止该IP的访问。

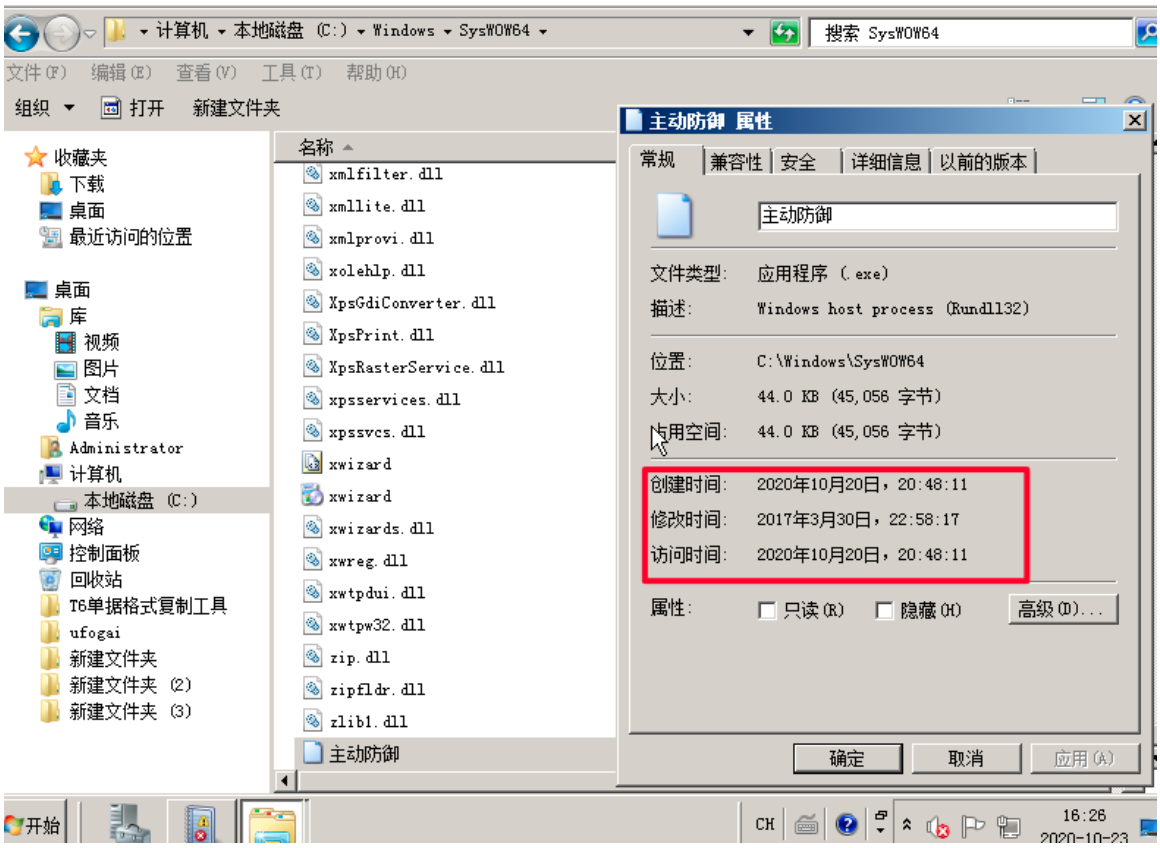


虚机被黑

服务器被黑后往往会植入恶意脚本，对于Linux系统可以执行crontab -l 命令查看是否有异常定时任务，此外也会遇到入侵者修改删除系统文件的场景，例如/usr下的子目录文件被修改，/usr/bin、/usr/sbin下的文件被清空；或者是/etc下的一些配置文件被修改，/etc/passwd和/etc/shadow被篡改等。对于Windows系统，一般系统文件中会有异常程序，或者查看系统日志会有异常报错信息。此类问题建议使用快照回滚数据或者重装操作系统来彻底清理。



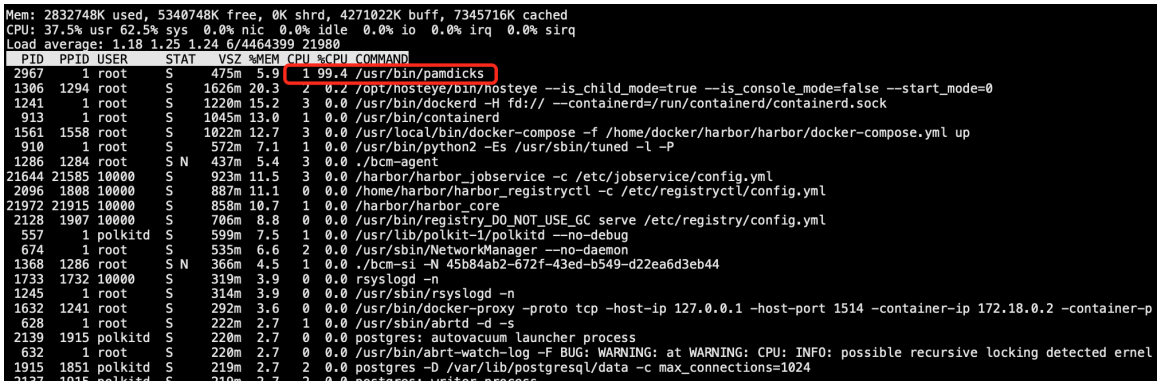




常见病毒

挖矿病毒

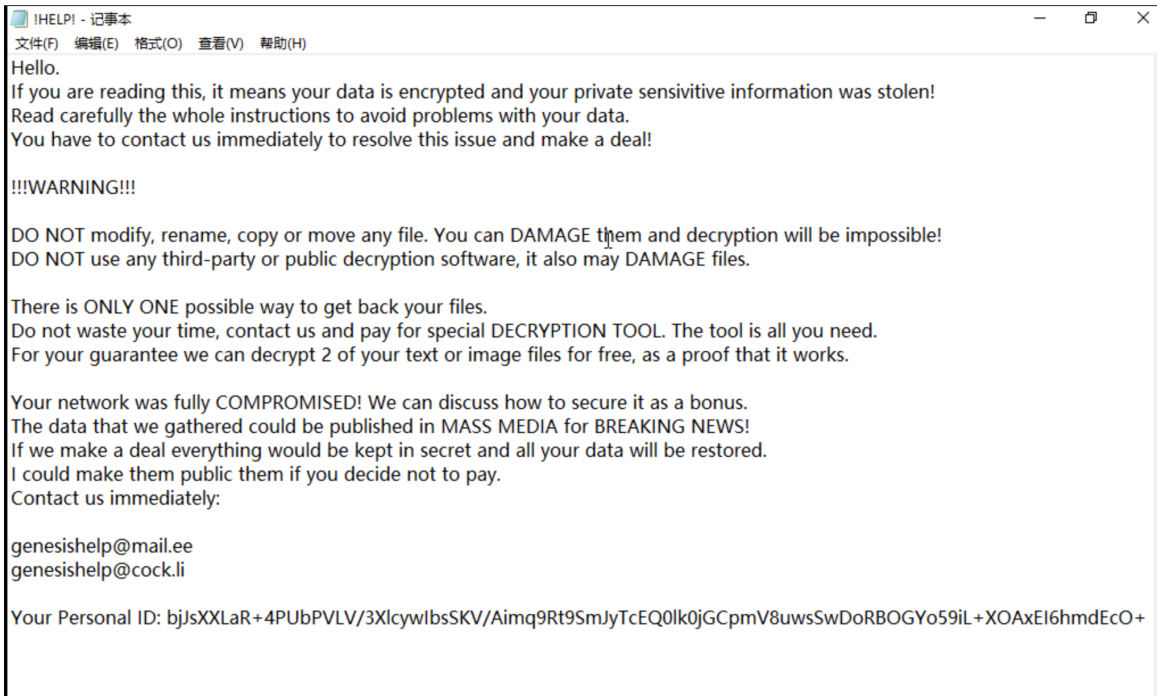
所谓的“矿”就是虚拟货币，有很多种，比如比特币、门罗币等等，这些货币充当虚拟世界的金钱，有很大价值。而挖矿病毒就是入侵者将复杂的木马程序植入被入侵机器中，通过借助傀儡机的计算能力来计算产生虚拟货币为自己牟利。一般中了挖矿病毒，最常见的现象就是cpu利用率经常无故打满，且持续100%，导致客户其他程序无法正常使用。一般挖矿病毒的进程会隐藏或者伪装成系统进程，发现后也很难彻底清理，此类问题建议使用快照回滚数据或者重装操作系统。



```
Tasks:  97 total,   2 running,  95 sleeping,   0 stopped,   0 zombie
%Cpu(s): 76.8 us,  1.7 sy,  0.0 ni, 20.6 id,  0.0 wa,  0.0 hi,  1.0 si,  0.0 st
KiB Mem :  4045256 total,   150120 free,  2939164 used,   955972 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   625032 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1229 root      20   0 2430396   1156    712 S  99.0  0.0 544:42.22 kswapd0
31141 root      20   0 5279240  91884   1724 S  57.1  2.3   4:07.50 tsm
32566 root      20   0  295968  14792   5164 S   1.0  0.4   3:29.08 docker-containe
 3376 root      20   0 1405296  69076   6040 S   0.7  1.7  99:56.43 hosteye
    9 root      20   0       0      0      0 R   0.3  0.0  25:26.74 rcu_sched
    1 root      20   0  125588   3012   1612 S   0.0  0.1   0:29.04 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:01.65 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.91 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0.0  0.0   0:04.85 watchdog/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:03.98 watchdog/1
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.84 migration/1
```

勒索病毒

勒索病毒主要以邮件、程序木马、网页挂马的形式进行传播，这种病毒利用各种加密算法对文件进行加密，被感染者一般无法解密，必须拿到解密的私钥才有可能破解。该病毒性质恶劣、危害极大，一旦感染将给用户带来无法估量的损失，基本上不可能逆向解密。加密完成后，还会修改壁纸，在桌面等明显位置生成勒索提示文件，指导用户去缴纳赎金。且变种类型非常快，对常规的杀毒软件都具有免疫性。此类问题也是建议使用快照回滚数据或者重装操作系统。



安全加固措施

使用安全组或ACL限制对外开放的IP和端口

在管理控制台上配置安全组来允许特定的IP和端口访问，或使用ACL来屏蔽特定的IP或IP段及端口访问。

创建和配置安全组请参考: 创建安全组

创建ACL访问控制规则请参考：ACL介绍

修改远程默认端口

攻击者往往先通过公网进行默认端口扫描，再根据扫描结果进行暴力破解攻击，所以修改远程默认端口号是有效的防御措施之一。

Windows操作系统请参考：Windows修改默认连接远程端口

Linux操作系统请参考：Linux修改默认远程连接端口

合理的密码策略

建议设置较为复杂的密码，字母、数字、特殊符号相结合，不要使用弱口令和易于破解的密码，诸如admin@123 1qaz!QAZ 此类，并定期修改密码。

使用快照与镜像

百度智能云平台提供快照和镜像两种数据备份的方法，其中快照对所有云磁盘都可创建，镜像只能对系统盘进行创建，用户定期创建服务器的数据备份，遇到安全隐患后可以直接通过数据回滚的方式进行恢复，简单高效，该措施可能会产生一些费用，因为目前快照是收取费用的，而镜像是免费的。

创建快照方法请参考：手动快照

创建自定义镜像请参考：创建自定义镜像

快照价格请参考：快照计费方式

使用百度智能云提供的主机安全工具

主机安全（HOSTEYE）是百度智能云为保护用户服务器上运行业务的连续性，而提供的一款主机安全产品。其具备高稳定性、轻量级、易部署的特点，可供小微企业用户快速构建主机安全防护体系，同时也可根据大中型企业用户的需求进行私有化、混合化场景进行落地部署。

官方产品文档：主机安全产品介绍

登录管理

定期在控制台上打开【主机安全客户端-登录管理-登录详情】，查看是否有暴力破解攻击服务器的恶意IP，如有的话可一键加入黑名单永久阻止其访问。

详细使用方法请参考：登录管理

病毒查杀

可以对几个特定目录进行实时防护，发现恶意进程会立即提示。特定目录包括/home、/root、/bin、/sbin、\windows\system、\windows\system32等。

详细使用方法请参考：病毒查杀

网站后门

实时防护发现的网站后门总数，需要开启实时防护功能。实时防护只针对Web目录。

详细使用方法请参考：网站后门

报警设置

用户能够设定安全事件的“暴力破解成功”和“异地登录行为”的告警通知方式和提醒时间。

详细使用方法请参考：系统管理

安全检测服务SRD

安全检测服务（Security Risk Detection，简称SRD）是百度智能云面向用户提供的安全风险检测服务，能够检测出SQL注入、XSS跨站脚本等多种常见Web漏洞，也能检测出系统软件中存在的安全漏洞。安全检测服务可以帮助用户快速的发现业务系统中存在的问题，及时修复，提升业务安全性和稳定性。

官方产品文档：安全检测服务SRD

云顾问 Cloud Advisor

云顾问 Cloud Advisor 产品可帮助客户更安全、更可靠、更低成本地用好云。基于百度智能云多年运维实践沉淀出的典型实践，对客户授权后，对客户使用百度智能云时可能遇到的问题与风险进行系统性检查与报告，并对于有风险的检查项给出解决建议和技术支持。云顾问可以理解为是云服务器的“体检报告”，便于全方位细致了解服务器存在的隐患。

官方产品文档：云顾问ACA

操作指南

实例

实例概述

实例简介

一个实例即是一台云服务器，包含CPU、内存、操作系统、网络、磁盘等基础计算组件。 云服务器BCC支持用户自定义CPU、内存、操作系统等配置，支持在线配置升级。支持挂载两种磁盘，临时数据盘和云磁盘CDS。

实例规格简介

根据不同的业务场景，云服务器BCC推出了不同的实例规格族。在同一个规格族中，根据CPU和内存的比值不同分为不同的规格类型。您可以根据业务需求选择合适的规格类型。

关于实例规格族的具体信息，请参见实例规格。如果您因为业务需求需要对已购买的实例变更配置，请参见实例配置变更。

实例镜像简介

镜像为BCC实例提供操作系统和预装的软件。百度智能云提供了多种公共镜像供您在创建BCC实例时选择。您还可以基于某一时刻的BCC系统盘创建您专属的自定义镜像，该镜像只能您自己看到。

百度智能云提供的镜像包括以下几种：

- 公共镜像：百度智能云官方提供的镜像，皆是正版授权，安全性高，稳定性强。公共镜像包含了Windows Server系统镜像和主流的Linux系统镜像，均为免费使用。
- 自定义镜像：您使用实例或快照创建的镜像，或是您从本地导入的自定义镜像。用于导入的镜像必须满足操作系统版本、镜像格式等要求。
- 共享镜像：由其他用户共享而来的镜像，仅能用作创建实例。
- 服务集成镜像：基于基础操作系统并集成有通用服务和应用软件环境的镜像。由百度或第三方服务商提供，经由百度智能云审核后发布。服务集成镜像可帮助您快速创建包含控制面板、PHP等常用运行环境的云服务器。
- 云市场镜像：由第三方服务商提供，经百度智能云审核后发布在云市场的镜像。包含丰富的服务运行环境和应用软件，可帮助您快速应对各类业务场景。

更多镜像介绍详见镜像概述。

实例存储简介

实例存储分为系统盘和数据盘：

系统盘：用于存储操作系统及核心配置，类似于Windows系统下的C盘。

数据盘：用于保存您的数据，类似于Windows系统下的D盘，支持扩容、挂载、卸载。

更多存储介绍详见存储概述。

常用操作

- 如果您还未创建云服务器BCC实例，可参见创建实例。
- 如果您创建实例时未设置密码或忘记密码，可参见重置密码。
- 如果您需要对当前实例的配置进行变更，可参见实例配置变更。

创建实例

创建实例导航

本文介绍创建云服务器实例的几种方式，包括基本的创建操作以及进阶的自定义功能等。

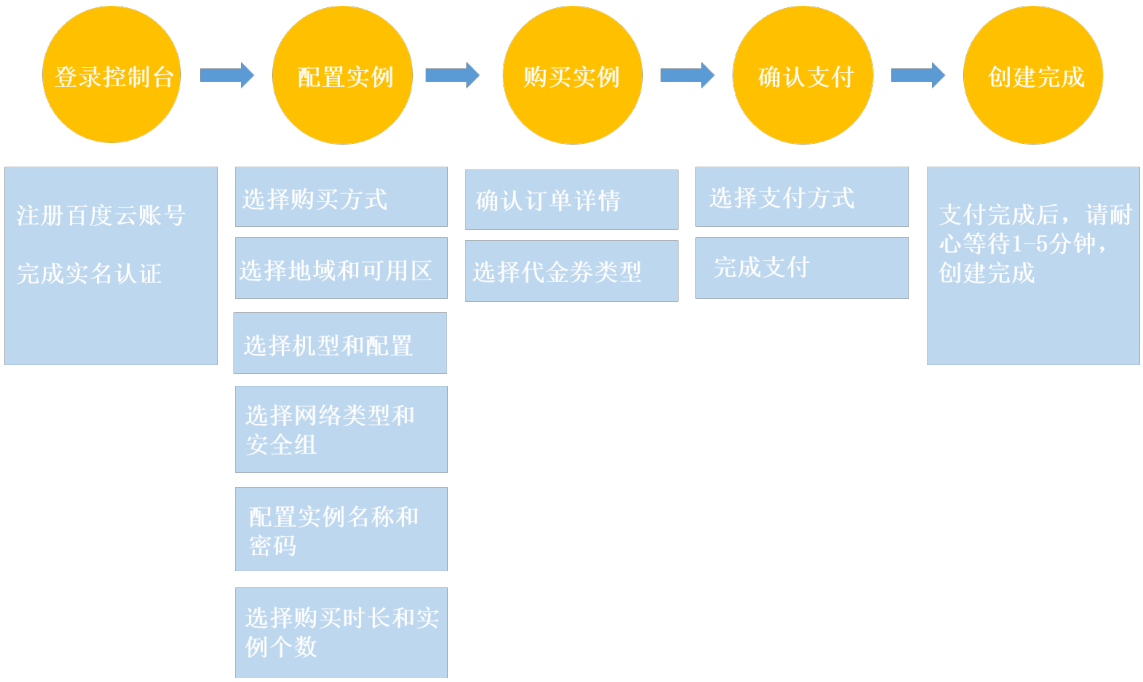
- 您可以按照云服务器控制台上，创建实例购买页的指引创建云服务器实例，可以灵活地选择配置项，确保满足业务的需求。详细操作请参见通过购买页创建实例。
- 如果您有常用的操作系统、应用程序等配置，可以先创建自定义镜像，再用自定义镜像创建实例，提高配置效率。详细操作请参见通过自定义镜像创建实例。
- 如果您需要一台和当前实例相同配置的实例，可以直接选择创建相同配置的实例，请参见购买相同配置实例。
- 启动模板是一项持久化BCC实例配置的功能。您可以使用启动模板快速创建实例，提升使用效率，详细操作请参见使用启动模板创建实例。

通过购买页创建实例

前提条件

- 创建云服务器实例之前必须完成实名认证。请根据自身情况，进行“企业认证”或者“个人认证”，具体请参考个人认证和企业认证。如果您在创建云服务器实例时仍未完成实名认证，你可以点击页面上的“**认证**”提示按钮到实名认证页面完成相关操作。
- 创建按量付费的云服务器实例需要您的账户余额大于或等于100元整，具体的计扣费方法请参考产品定价。

操作流程



操作步骤

1. 登录百度智能云官网，并进入管理控制台。
2. 选择“产品服务>>计算>>云服务器BCC”，在实例列表页面点击**创建实例**。
3. 进行**基本配置**的选择。

配置项	必选/可选	配置说明
地域	必选	不同地域间的云产品内网不互通；选择最靠近您业务的地域，可降低访问延时；创建成功后**不支持**更换地域。
可用区	必选	可用区是指在同一地域下，电力和网络互相独立的物理区域，故障会被隔离在一个可用区内。如果您需要购买多台云服务器，建议选择不同的可用区，实现容灾效果。
付费方式	必选	• **包年包月**：需要提前一次性支付所选时间段内的费用，这种模式价格相较于按量付费模式更加低廉。 • **按量付费**：按实例的实际使用量收费，不需要提前支付费用，但需要保证账户余额大于或等于100元，这种模式比预付费模式价格高一些。 • **抢占实例**：一种新运作模式，您在指定范围内出价，当指定实例规格的市场价格低于您的出价，且资源库存充足时，会成功创建抢占实例，并按照当前市场价计费。
实例配置	必选	根据CPU与内存比值的不同，云服务器提供了多种不同类型的实例规格族、。关于实例规格及适用的应用场景您可参考实例规格。
弹性高性能计算集群EHC	必选	加入同一个EHC集群的实例的RDMA网络可通信，不在同一EHC集群的实例的RDMA网络隔离，默认填充该可用区的默认集群参数。
		您可根据实际需求，选择适合的镜像类型及操作系统。 • **公共镜像**：由百度智能云官方提供，仅包含基础操作系统环境

镜像类型及操作系统	必须	• **公共镜像：**由百度智能云官方提供，仅包含基础操作系统环境。 • **自定义镜像：** 通过创建自定义镜像功能生成的镜像，包含系统盘基础操作系统、应用及软件环境的个性化配置。自定义镜像可以帮助您快速创建包含个性化配置的云服务器。 • **服务集成镜像：**在基础操作系统上集成有通用服务和应用软件环境的镜像。由百度或第三方服务商提供，经由百度智能云审核后发布。服务集成镜像可帮助您快速创建包含控制面板、PHP等常用运行环境的云服务器。 • **共享镜像：** 支持用户将自定义镜像共享给其他用户，被共享用户可以通过管理控制台或API查询到其他用户共享的镜像。被共享用户可以使用其他用户共享的镜像创建实例和重装操作系统。 • **云市场镜像：** 由第三方服务商提供的，经百度智能云审核后发布在云市场的镜像环境。包含丰富的服务运行环境和应用软件，可帮助您快速应对各类解决方案场景（目前暂不支持后付费镜像）。
主机安全(企业版)	可选	默认提供基础版主机安全，可额外开启企业版主机安全，提高云服务器的安全防护。主机安全详情可参考主机安全HOSTEYE
系统盘	必选	用于安装操作系统。非异构计算实例且镜像os是linux，默认大小20GB；若镜像os是windows，默认大小40GB；异构计算实例不区分操作系统默认均为40GB。 根据地域以及实例规格的不同，可选择的云盘类型也不同，以页面实际提供的云盘类型为准。
数据盘	可选	即挂载的数据盘，用于提高云服务器的存储容量，默认不选择。云磁盘挂载数量有一定上限，若您想挂载超出上限的云磁盘，请提交工单联系我们。当前，云服务器的系统盘和数据盘均为CDS云磁盘，关于CDS的磁盘类型和使用限制请查看磁盘类型和使用限制。
共享盘	可选	支持选择已有的文件系统，或单击创建新文件系统前往创建。文件系统最多可设置5个共享盘，如需更多请提交工单联系添加。
快照策略	可选	默认关闭绑定快照策略。通过快照，您可以实现磁盘数据备份，磁盘数据恢复以及磁盘镜像的制作。更多关于快照的使用方法和使用限制，请查看快照使用说明。快照目前是收费服务，请查看快照计费说明。

4. 进行**网络和带宽**配置的选择。

配置项	必选/可选	配置说明
网络类型	必选	如果您没有创建私有网络，可以选择默认私有网络。目前，百度智能云的私有网络VPC、子网、安全组、ACL、路由表免费，关于私有网络的更多介绍可参考私有网络VPC 。
安全组	必选	如果您没有创建安全组，可以选择默认安全组。
弹性资源	可选	如需公网访问，请购买弹性公网IP，或在实例购买成功后绑定已有的弹性公网IP。 公网带宽支持以下购买方式： • **包年包月计费：**需要提前一次性支付所选时间段内的带宽费用，购买包年包月的云服务器时，费用将合并在实例中收取。 • **按使用流量计费：**根据用户实际传输的数据量计费，流量使用没有上限，但可限定最大峰值带宽。 • **按使用带宽计费：**根据用户选择的固定带宽值进行计费，最大可购买200Mbps的带宽。

5. 进行**购买信息**的选择。

配置项	必选/可选	配置说明
购买数量	必选	选择实际想要购买的实例数量，默认为1台。
购买时长（包年包月）	必选	1-9个月或1-3年选择配置。时长周期为实例创建日起的单位周期。如2024年5月20日购买云服务器一个月，则使用周期为2024年5月20日-2024年6月20日。
自动续费（包年包月）	可选	默认不勾选，不进行自动续费。勾选后实例自动续费，自动续费周期与原订单周期一致。
自动释放（按量付费）	可选	默认不勾选，不进行自动释放。勾选后实例将在指定时间自动释放，具体可参考定时释放。

6. 进行**其他配置**的选择。

配置项	必选/可选	配置说明
标签	可选	当您拥有多台云服务器实例时，可以通过设置云服务器的标签，实现资源的分类管理。具体可参考标签功能进行设置。
资源分组	可选	资源分组让您能够按照不同维度对云上资源进行归类。可以选择已有的资源组，或者前往控制台创建资源分组，具体可参考资源分组。
部署集	可选	在指定部署集中创建云服务器实例时，会和处于同一部署集中的其他云服务器实例，按照物理服务器打散，保障发生硬件故障等异常情况下时的服务高可用性。具体可参考部署集进行设置。单个实例最多可加入2个部署集。
		• **实例名称：**您可自定义设置实例名称或由系统随机生成实例名称。 • **主机名：**您可自定义设置主机名（hostname）或由系统随机生成主机名。 **自定义命名策略说明：** ◦ 根据“名称前缀[begin_number,digits]名称后缀”的命名格式为多台云服务器实例统一设置实例名称或主机名。例如，设置实例名称为test-[2,3]-bcc，创建3台，则第一台云服务器实例名称为test-002-bcc，第二台云服务器实例名称为test-003-bcc，第三台云服务器实例名称为test-004-bcc。

系统信息	可选	◦ 名称前缀是必须项，名称后缀是可选项。 ◦ [begin_number,digits]：指定实例名称或者主机名称中数字的起始值，呈递增关系，是多台云服务器实例的实例名称或者主机名称的区别部分。[begin_number,digits] 字段中不能有空格。 ◦ begin_number：指定实例名称的开始数字，取值支持[0,999999]，默认值为0 ◦ digits：指定数字在实例名中的所占位数，取值支持[1,6]，默认值为6。 ◦ 若名称指定为[]或者[,]，则begin_number默认为0，digits会默认为6。 ◦ 若名称指定为[99]或者[99,]，则begin_number默认99，digits默认为6。 ◦ 若名称指定的begin_number位数大于digits的取值，digits默认为6。 ◦ 若名称指定的begin_number与新建的云服务器实例数量之和大于最大值999999，超过部分的云服务器实例都使用999999。 • **有序后缀：**用户选择打开此选项，则有序后缀从0001开始递增，最大不能超过9999。例如：InstanceName0001，InstanceName0002和HostName0001，HostName0002。 • **Domain开关：**用户打开后，主机名将带有domain后缀，支持DNS解析。 • **管理用户名：**Windows系统的管理员账号为Administrator，Linux系统的管理员账号为root。 • **管理员密码：**根据实例操作系统的不同，密码可供选择的设置方式也不同。 ◦ 用户自定义：自定义设置登录实例的密码。 ◦ 随机生成：购买成功后需登录控制台重置密码，可参考 重置密码。 ◦ 密钥对：Linux操作系统可以选择使用密钥对的方式连接云服务器，SSH密钥对是一种比常规密码更安全的登录云服务器的方式，具体可参考密钥对设置。 • **实例释放保护：** ◦ 默认关闭实例释放保护，开启实例释放保护的实例无法手动释放。

		◦ 开启释放保护不影响实例的定时释放，欠费释放以及实例过期释放。
高级配置	可选	• **用户数据注入**：此处输入的脚本将作为实例自定义脚本在启动实例时执行，实现自动化配置实例，例如获取并安装软件资源包、开启服务、打印日志、初始化服务环境等操作。Linux实例和Windows实例均支持数据注入功能： ◦ Linux实例可使用User-Data脚本。通过以下示例，实例创建成功后可在/root/test文件中看到 Welcome to Baidu AI Cloud。 示例： #!/bin/sh echo "Welcome to Baidu AI Cloud." | tee /root/test ◦ Windows实例可使用batch（首行必须是rem cmd）、powershell（首行必须是#ps1）以及bash脚本。通过以下示例，实例创建成功后可在C盘中看到新建的两个目录：cmd_create01和cmd_create02。 示例： rem cmd mkdir "C:\\cmd_create01" mkdir "C:\\cmd_create02"

7. 点击**确认购买**后，在**确认订单**页核对各项配置信息及费用明细，点击**提交订单**完成实例购买。

> **说明：**如果您有百度智能云代金券，可以使用代金券结算抵扣费用，如果代金券金额不足，不足部分可以使用银行卡完成支付操作，或者全部用银行卡支付。

8. 进入**管理控制台**的**实例列表**处，可查看实例的状态。
   - 创建时，实例状态为**创建中**。
   - 创建成功后，实例状态显示为**运行中**。
9. 实例创建成功后，系统将以短信形式发送给用户云服务器的信息，包括IP和用户名。为了安全因素考虑，密码将不再通过短信形式发送，如果您忘记创建实例时配置的密码可以选择**重置密码**。

常见问题

控制台找不到某台云服务器实例怎么办？

如果您在控制台界面查询不到相关实例，建议您进入控制台的实例页面，点击左上角的地域切换，查看其它地域内是否有您的服务器。

忘记服务器密码怎么办？

linux操作系统默认管理员用户为root，windows操作系统默认管理员用户为administrator，如果您忘记了密码，可以选择**重置密码**，具体可参考重置密码。

注意：执行重置密码操作会强制重启实例，建议您在业务低峰期执行重置动作。

通过自定义镜像创建实例

背景信息

您可以使用自定义镜像创建具有相同操作系统、应用程序和数据的云服务器实例，提高工作或交付效率。

前提条件

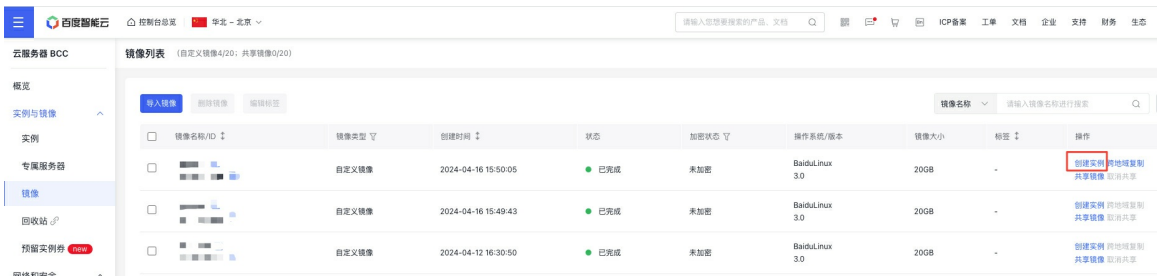
在需要创建实例的账号和地域中，已经创建自定义镜像。

如果您未创建自定义镜像，可参考以下方法：

镜像持有情况	方法
在本地设备或其他平台持有镜像	将本地设备或其他平台上的自定义镜像导入到云服务器中，详情可参考 导入自定义镜像。
未创建自定义镜像，但是有可作为模板的实例	使用实例创建自定义镜像。
未创建自定义镜像，但是有可作为模板的快照	使用快照创建自定义镜像。
在其它地域持有自定义镜像	将自定义镜像复制到需要创建实例的地域，详情可参考镜像跨地域复制。
在其它账号持有自定义镜像	将其他账号的自定义镜像共享给需要创建实例的账号，详情可参考共享自定义镜像。

操作步骤

1. 登录云服务器BCC控制台。
2. 在左侧导航栏选择**镜像**。
3. 找到需要使用的镜像，单击右侧操作列**创建实例**。



4. 配置相关实例信息并完成创建实例。

   地域和镜像信息已自动填充，您可根据需要对其他实例信息进行配置，配置信息说明请参考通过购买页创建实例。

> **注意：**如果您选择的自定义镜像中包含一个或多个数据盘快照，系统会自动根据这些快照创建相同数量的云磁盘作为数据盘，每个云磁盘的容量与对应的快照相同。您可以增加云磁盘容量，但不能缩小。

常见问题

- **使用自定义镜像创建云服务器时，是否对套餐大小有所限制？**

  使用自定义镜像创建云服务器时，对新服务器的各项参数（如CPU、内存）没有限制。

- **通过自定义镜像创建的云服务器支持修改系统吗？**

  支持，在实例详情处选择重装操作系统即可。

通过API使用自定义镜像创建实例

您可通过创建实例使用自定义镜像创建实例。

购买相同配置实例

背景信息

如果您需要在已购买的云服务器实例基础上，再次购买相同配置的实例，可以使用云服务器控制台提供的**购买相同配置**功能，提高在特定场景下横向扩展的效率。

操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。
2. 选择需要再次购买相同配置的实例，单击最右侧操作列 “更多>>购买相同配置”。

3. 核对自动选择的配置。

> 地域、实例配置及镜像信息已自动选择，支持修改已选择的配置信息。

4. 根据实际需要对配置信息修改，完成其他配置项的选择。
5. 提交订单并完成支付。

使用启动模板创建实例

背景信息

启动模板是一项持久化云服务器实例配置的功能。您可以通过使用启动模板快速创建实例，提升使用效率。启动模板中包含了用于创建实例的配置信息，可以存储除密码以外的任意配置信息。

操作步骤

关于使用启动模板创建实例的具体操作步骤可参考使用启动模板创建实例。

批量连续命名

操作场景

创建多台实例时，云服务器BCC支持批量创建实例后缀数字自动升序功能以及按规则命名功能。

当您需要购买 n 个实例并希望生成类似为 “BCC+序号” 的实例名称/主机名称时（即实例名称/主机名称为BCC0001、BCC0002、BCC0003 等实例），您可通过 **后缀数字自动升序** 实现。

当您需要创建 n 个实例并指定实例名称/主机名称带有序号且序号从 x 开始递增时，您可通过 **按规则命名** 实现。

适用场景

创建多台实例时实例名称/主机名称的设置。

后缀数字自动升序

可通过**开启有序后缀**的操作将批量购买的实例设置为前缀相同，仅序号递增的实例名称。

创建成功的实例默认序号从0001开始递增，且不能指定开始的序号。

> 若不开启有序后缀，系统会根据本次创建数量序号的数量级，依次递增，如果创建数量小于10，则序号从1~9递增，如果10<=创建数量<100，则从01~100递增。

按规则命名

根据**名称前缀[起始数字,位数]名称后缀** 的命名格式为多台云服务器实例统一设置实例名称或主机名。名称前缀是必须项，名称后缀是可选项。

[起始数字,位数]：指定实例名称或者主机名称中数字的起始值，呈递增关系，是多台云服务器实例的实例名称或者主机名称的区别部分。

[起始数字,位数] 字段中不能有空格，[]和,为英文字符。

起始数字：指定实例名称的开始数字，取值支持[0,999999]，默认值为0。

位数：指定数字在实例名中的所占位数，取值支持[1,6]，默认值为6。

若名称指定为[]或者[,]，则起始数字默认为0，位数会默认为6。

若名称指定为[99]或者[99,]，则起始数字默认99，位数默认为6。

若名称指定的起始数字位数大于位数的取值，位数默认为6。

若名称指定的起始数字与新建的云服务器实例数量之和大于最大值999999，超过部分的云服务器实例都使用999999。

举例一：

希望本次创建100台实例，以101数字开始，且数字在名字中间。按以下规则命名后，实例名称为：BCC-101、BCC-102 ... BCC-200。

举例二：

希望本次创建10台实例，以21数字开始，且数字在名字中间。按以下规则命名后，实例名称为：BCC21.baidu.cc、BCC22.baidu.cc ... BCC30.baidu.cc。

> 开启有序后缀和规则命名功能相互独立，若同时使用，则两者均会生效。比如，实例名称输入：bcc[21,10]baidu，同时也开启有序后缀，则实例名称会以bcc21baidu0001开始。

登录实例

登录方式概述

云服务器实例支持通过多种方式连接。您可以综合考虑操作系统、本地操作系统、鉴权方式等因素，选择合适的方式连接云服务器。

实例类型	连接方式
Linux实例	使用用户名密码验证连接Linux实例
	putty使用SSH密钥登录Linux实例
	xshell使用SSH密钥登录Linux实例
	SecrueCRT 使用SSH密钥登录Linux实例
	Linux/Mac使用SSH密钥登录Linux实例
	使用VNC远程登录云服务器
	使用SmartTerm远程登录云服务器
Windows实例	使用远程桌面登录Windows实例
	使用VNC远程登录云服务器

登录Linux实例

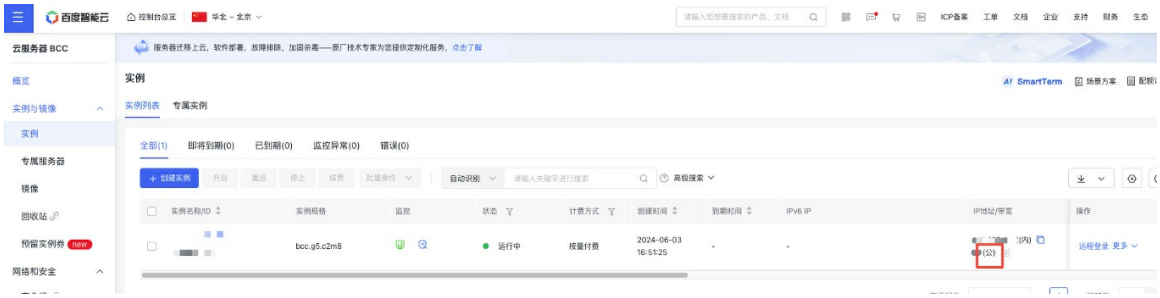
使用用户名密码验证连接Linux实例

前提条件

- 确认您的云服务器已绑定公网IP。
- 使用SSH登录Linux实例时，请确认您的云服务器已开放22端口（Linux实例创建时已默认开放22端口）。

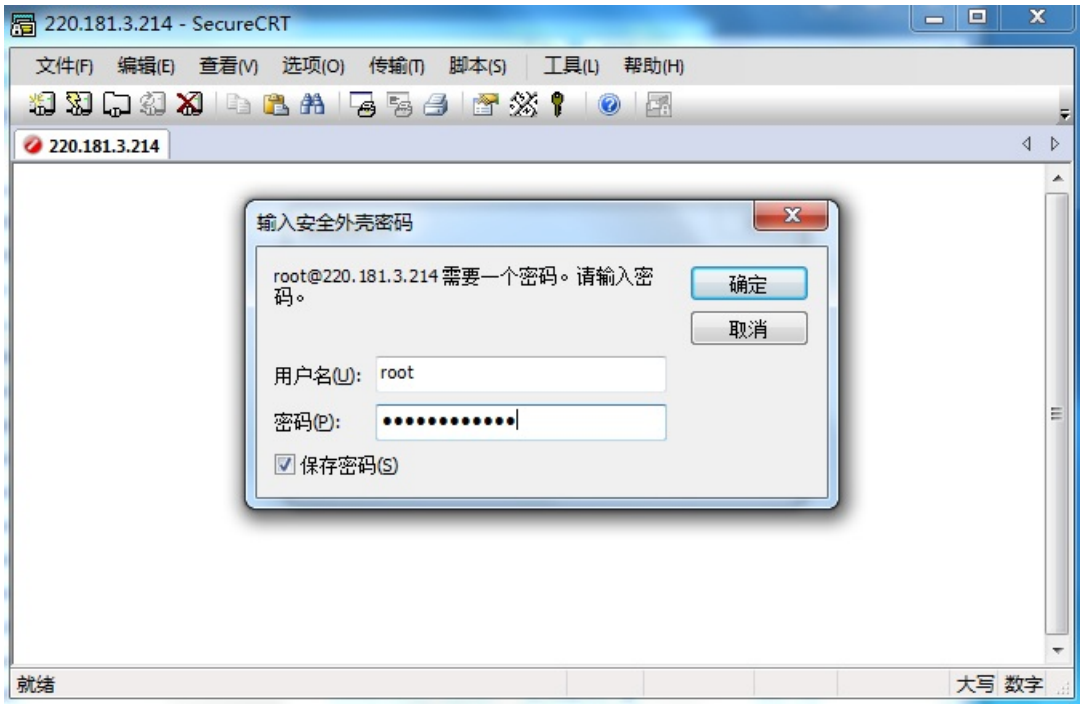
操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。查看云服务器的概要信息，找到其公网IP。



2. 在云服务器的状态处于**运行中**时，启动客户端进行连接。

对于Linux系统，可以通过SSH客户端（Putty、SecureCRT及Xshell等），使用SSH协议进行登录。



3. 输入用户名和密码进行验证。出于安全因素考虑，创建BCC实例后不再通过短信发送实例密码。如果您忘记了待操作实例的密码，可以重置密码。

4. 验证成功后，进入云服务器。

putty使用SSH密钥登录Linux实例

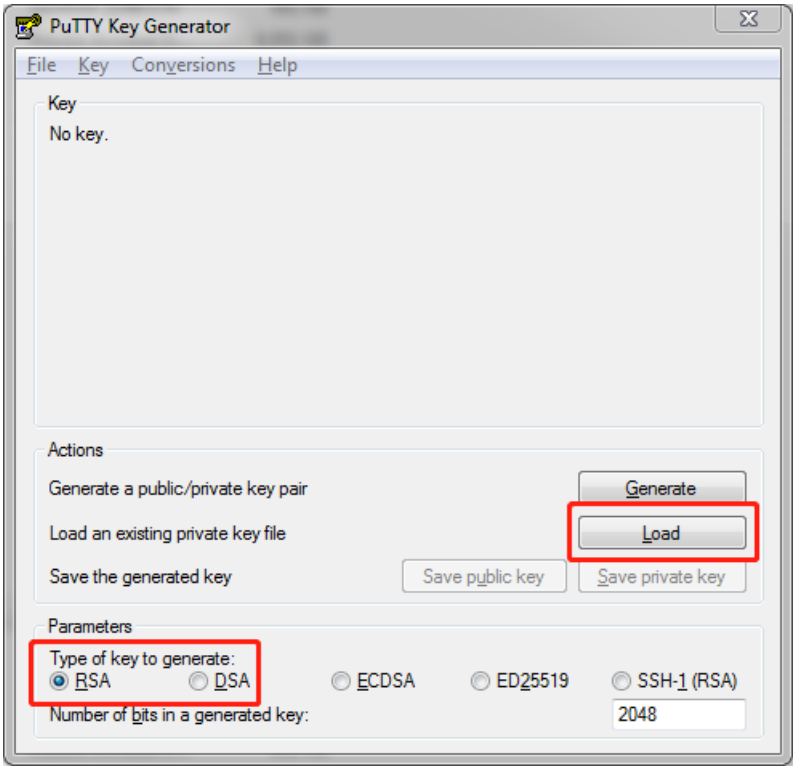
前提条件

- 下载并安装Putty和Puttygen工具：
  - 点击下载 Putty
  - 点击下载 Puttygen

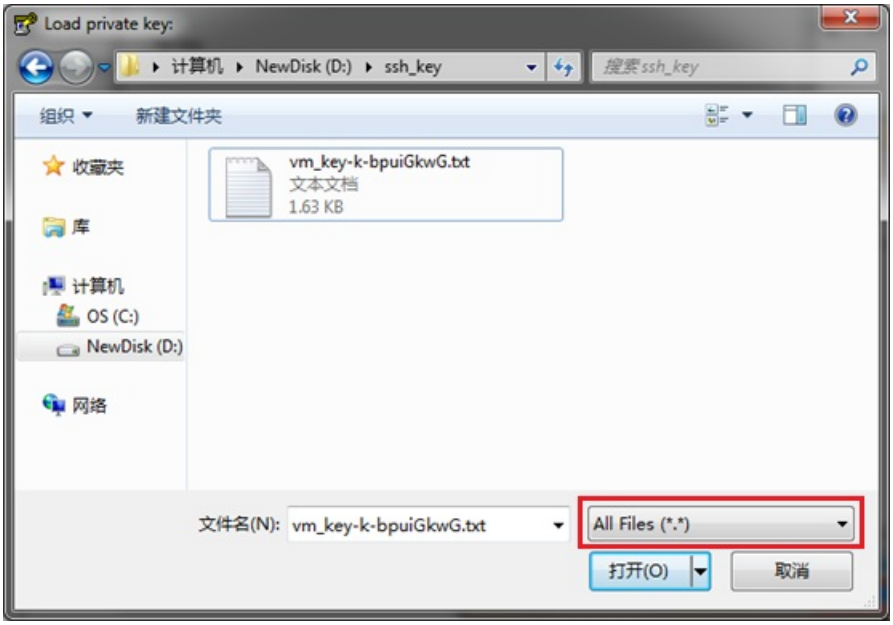
- 您拥有已绑定密钥对的BCC实例，如未绑定密钥对请参见绑定密钥对。
- 使用SSH登录Linux实例时，请确认您的云服务器已开放22端口（Linux实例创建时已默认开放22端口）。

使用puttygen创建私钥文件

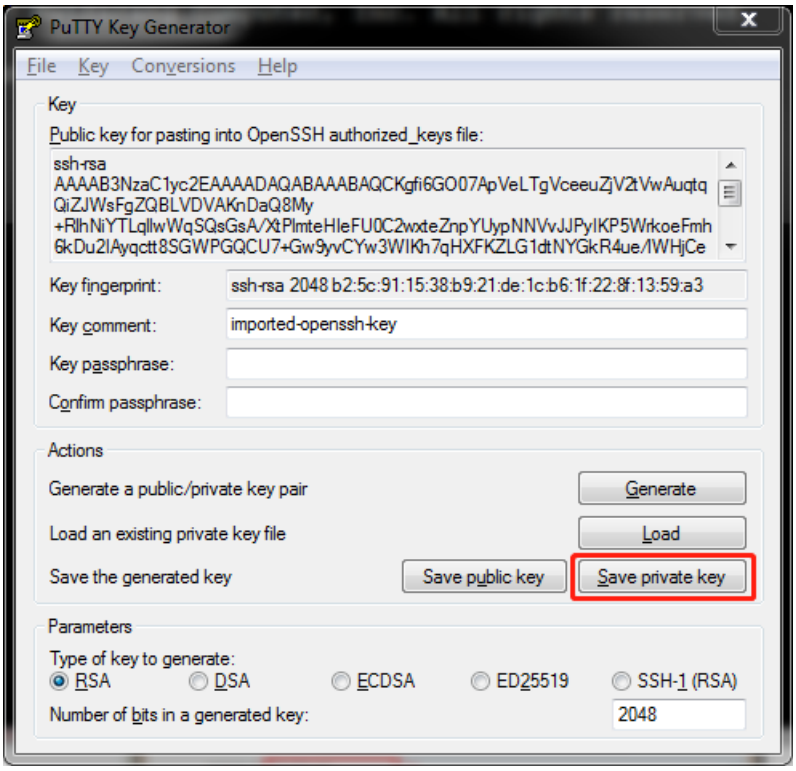
1. 打开puttygen，Type of key to generate 处选择RSA，然后单击 **Load**。



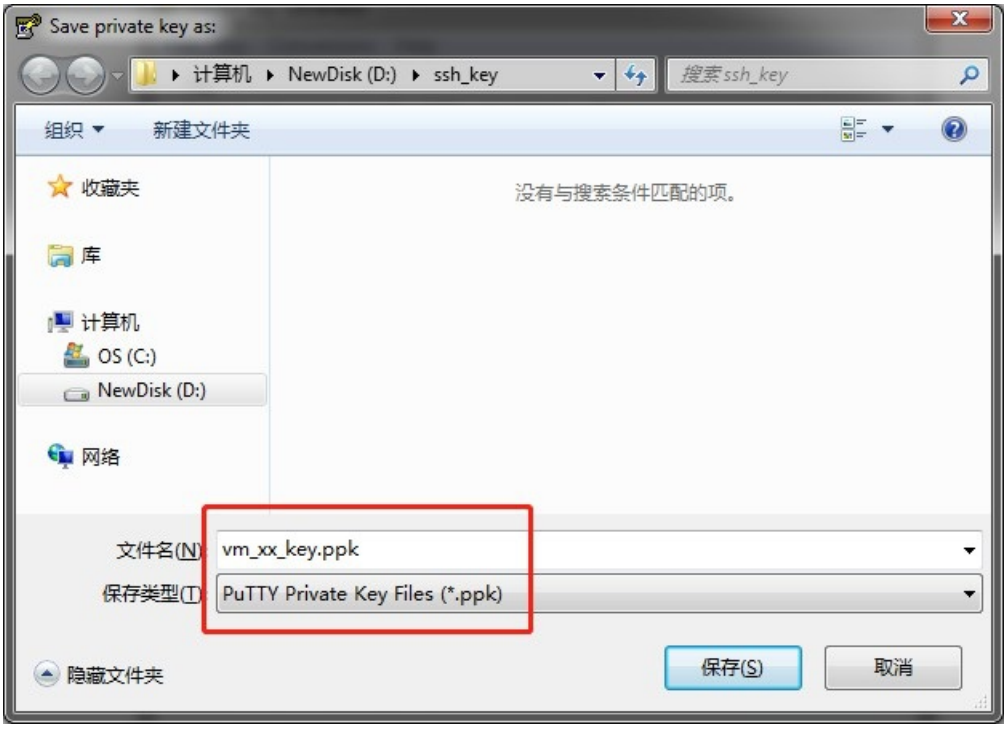
2. 选择显示所有类型的文件，选择创建密钥对时下载到本地的密钥对文件，单击 **打开**。



3. 然后点击**Save private key**，生成私钥文件。



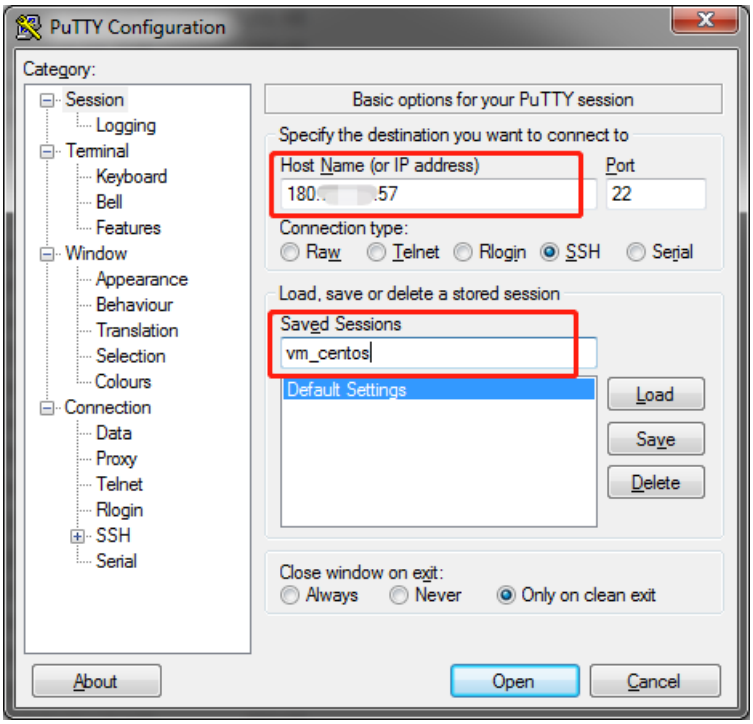
4. 选择私钥文件的保存路径，单击**保存**。



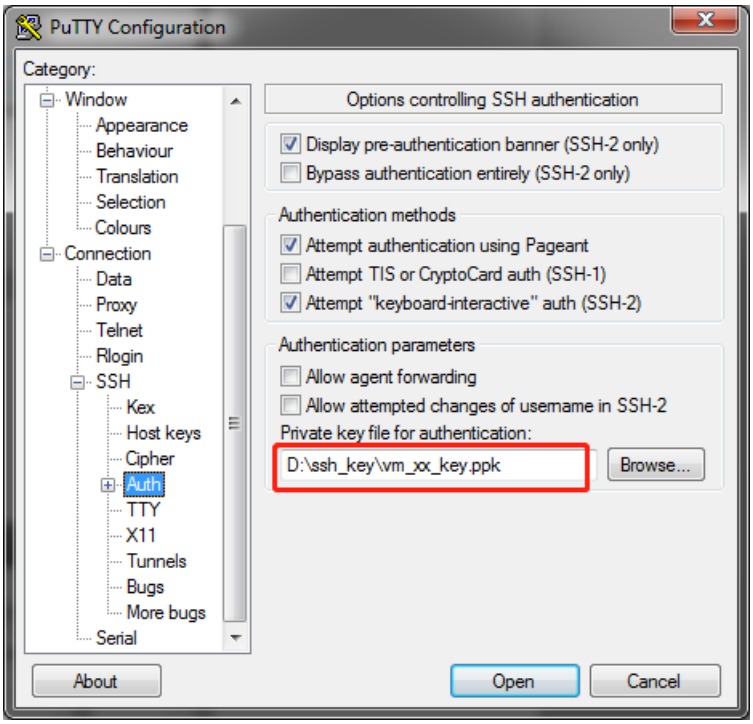
使用putty配置密钥登录

1. 打开putty，在Session的Host Name处输入远程主机的IP地址。

> **说明：**如果需要保存会话，请在**Saved Sessions** 处输入要保存会话的名称，然后单击**Save**



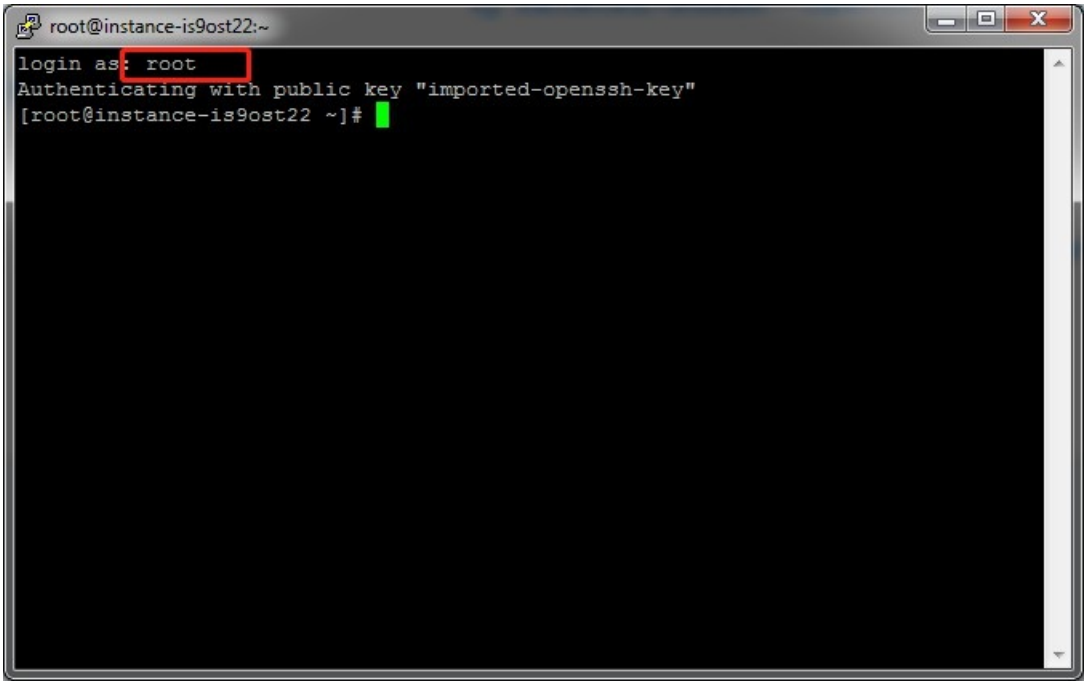
2. 在 SSH > Auth 处配置SSH的认证方式，单击**Browse**，选择使用puttygen创建的私钥文件。



3. 完成上述配置后，点击**Open**，使用私钥连接您的云服务器。

4. 第一次登录时提示putty未缓存Host key，单击**是**。

5. 输入用户名**root**，回车后即可登录到云服务器系统中。

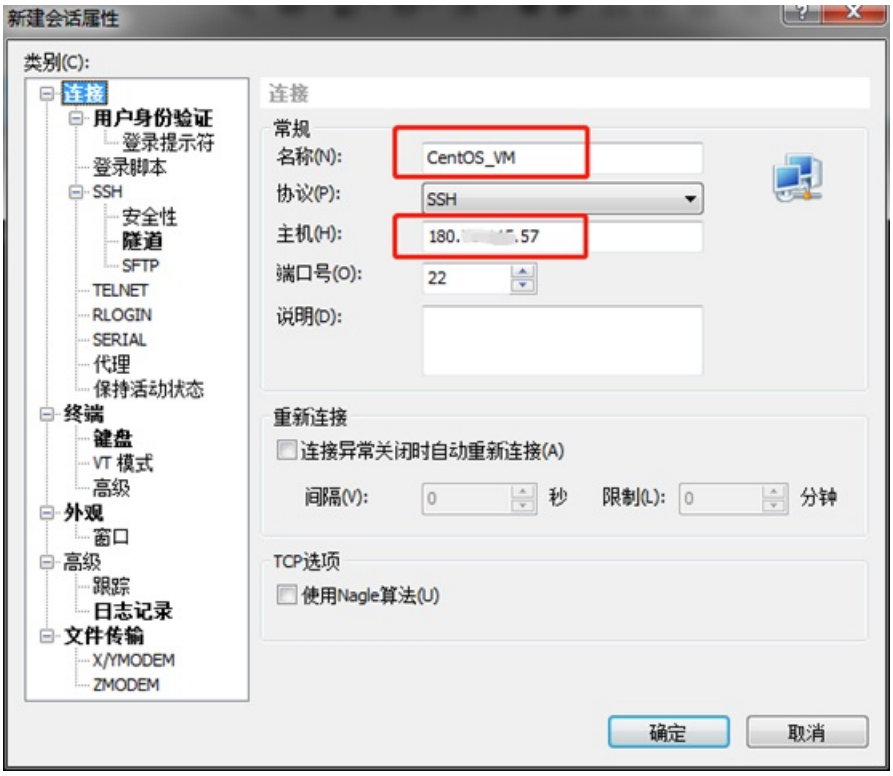


xshell使用SSH密钥登录Linux实例

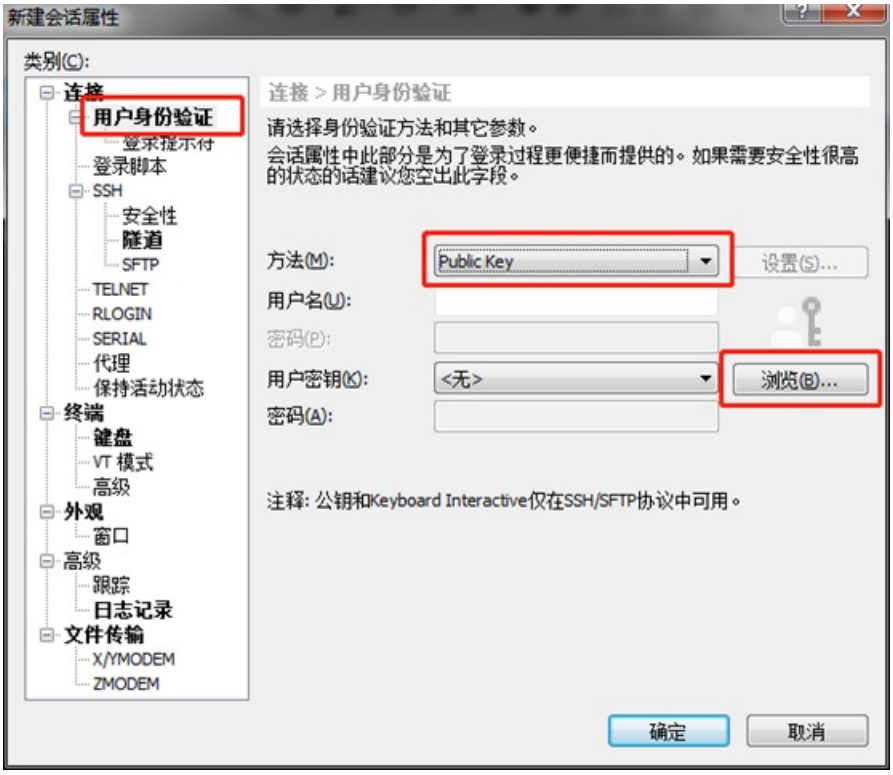
本文介绍百度智能云“使用密钥登录Linux服务器”的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《使用密钥登录Linux服务器》。

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

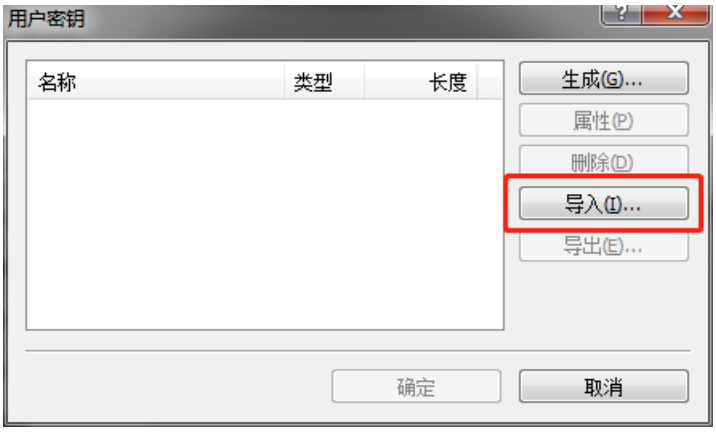
1. 在本地Windows系统中打开xshell。
   - 工具栏上打开**工具** > **新建用户密钥生成向导**
   - 在新建会话属性的**连接** 配置中输入会话名称和您的云服务器IP地址



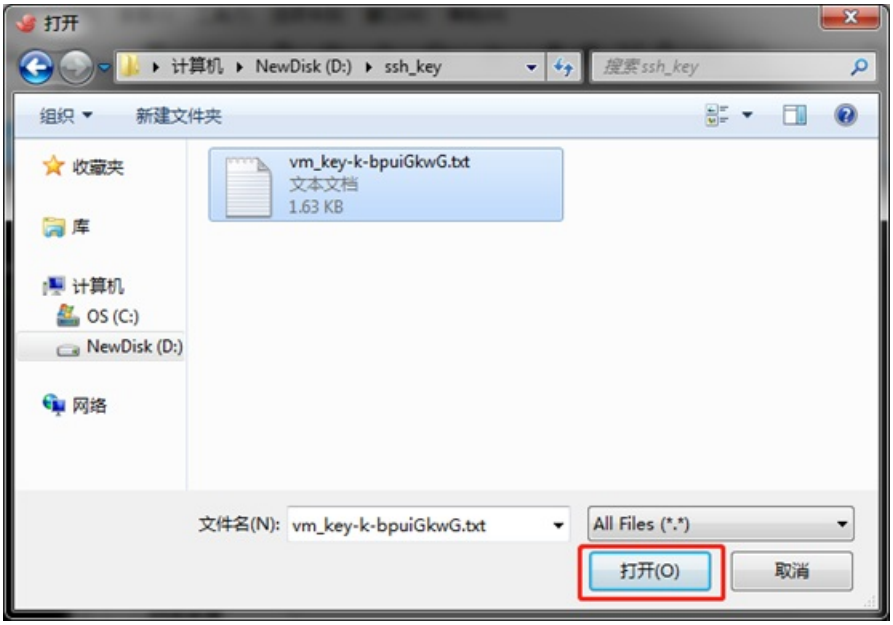
2. 点击**连接**中的用户身份验证，方法选择**Public Key**，然后在用户密钥处单击**浏览**。



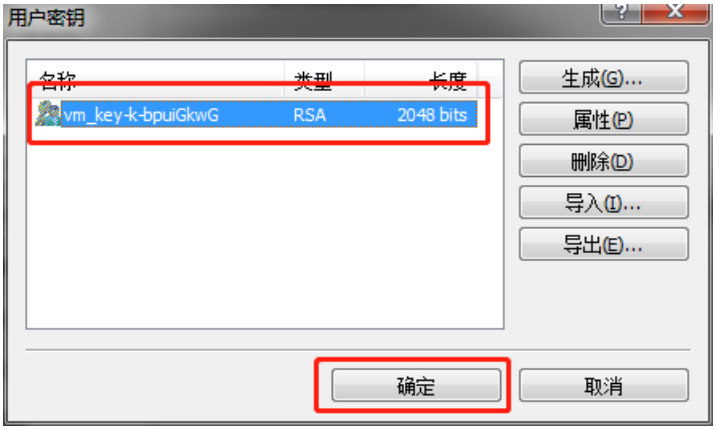
3. 在弹出的用户密钥界面，单击**导入**。



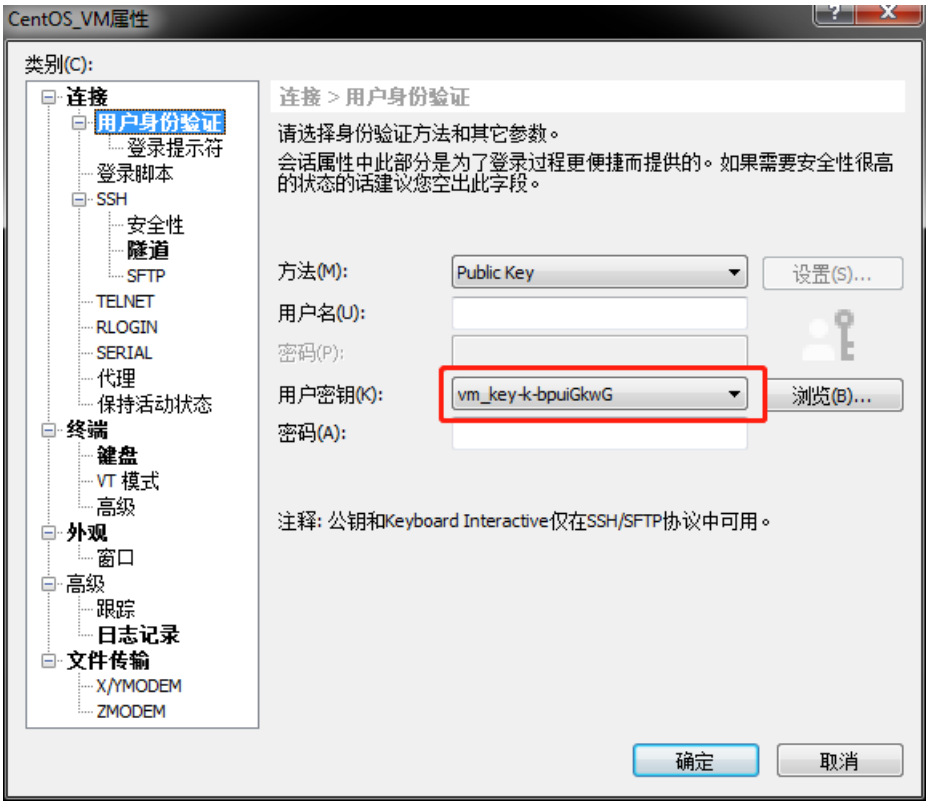
4. 选择您保存到本地系统的密钥对文件，然后单击 **打开**。



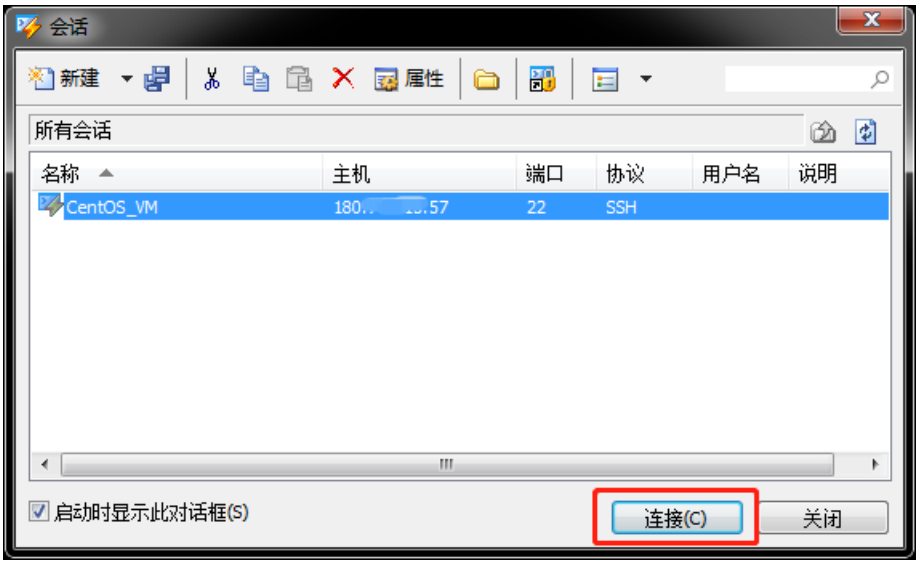
5. 点击生成的用户密钥，然后**确定**。



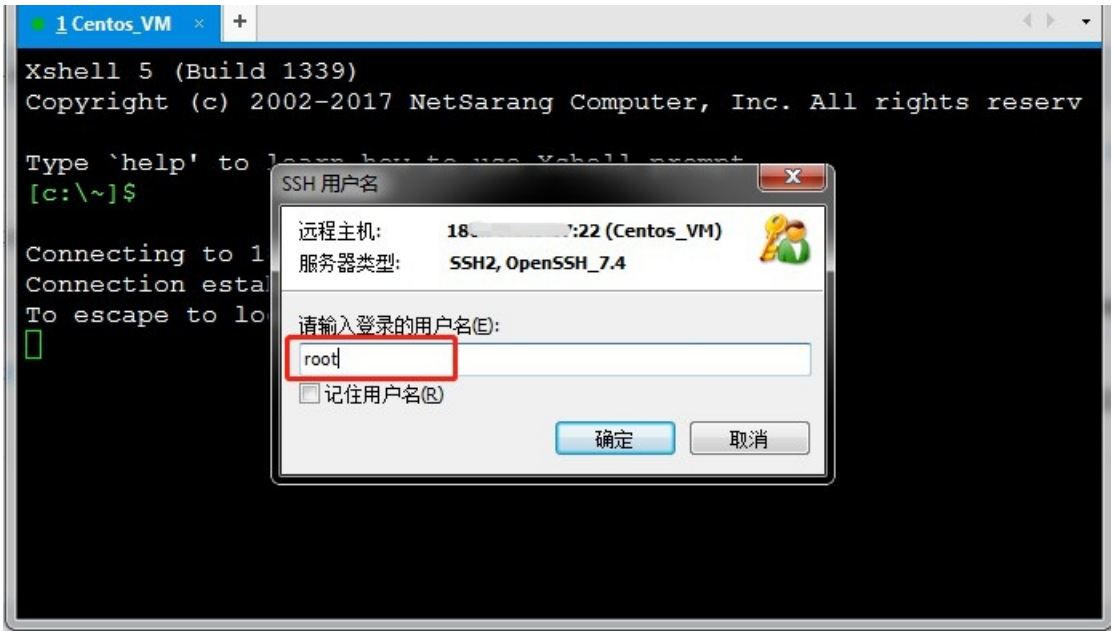
6. 此时可以看到用户身份验证界面，用户密钥已经是刚才创建的用户密钥，单击**确定**，保存连接会话。



7. 在会话界面，选择刚才创建的会话，然后点击**连接**。



8. 输入用户名**root**，即可通过密钥方式登录您的Linux云服务器。



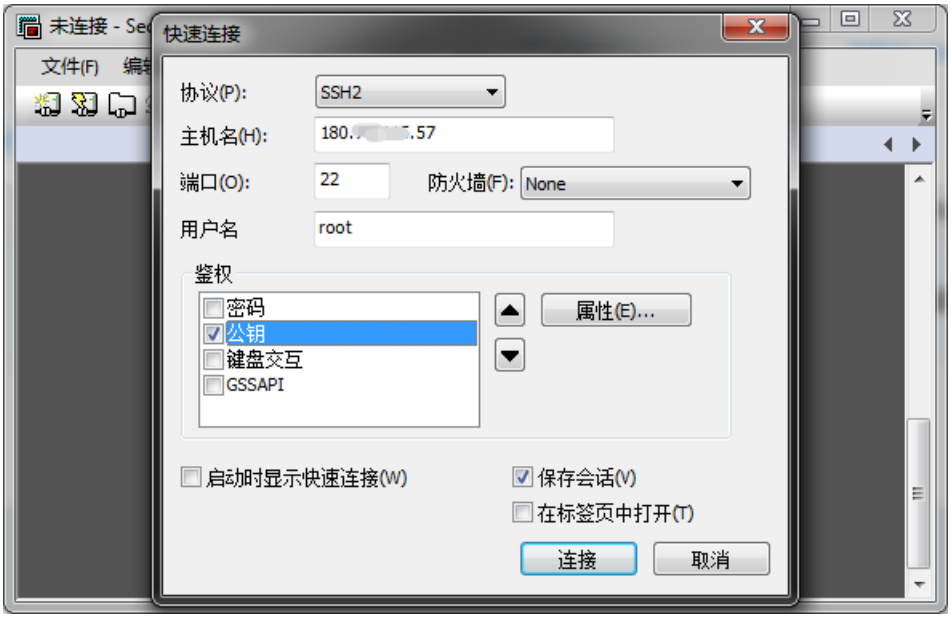
SecrueCRT使用SSH密钥登录Linux实例

前提条件

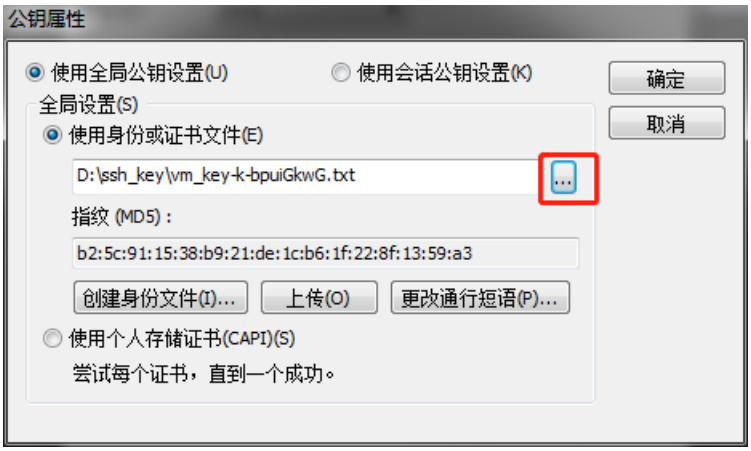
下载并安装SecrueCRT工具，点击下载SecrueCRT

操作步骤

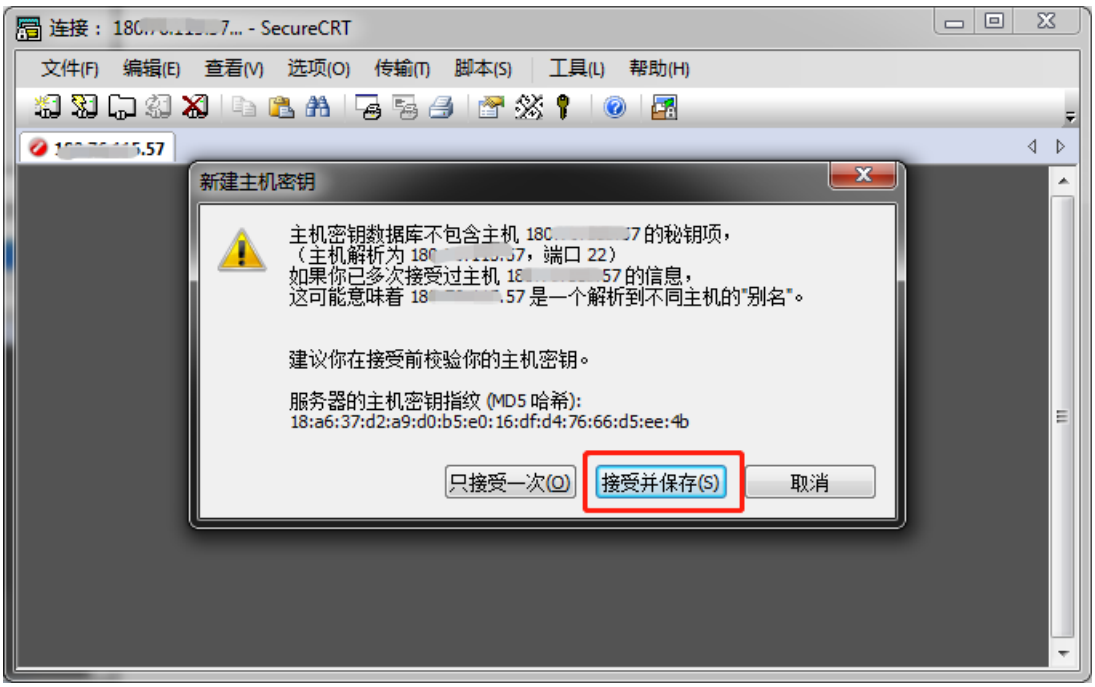
1. 打开SecrueCRT，打开**文件** >**快速连接**。
   - 在快速连接界面，主机名处输入您的云服务器的IP地址，用户名输入**root**
   - 鉴权处选择**公钥**，然后点击右侧的**属性**



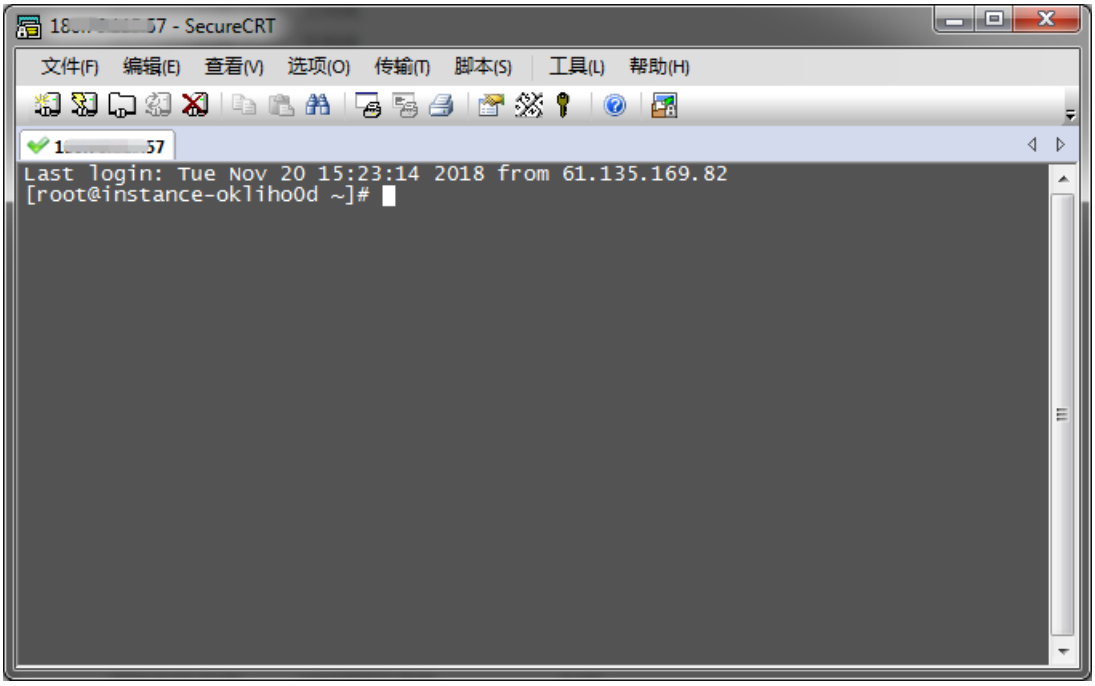
2. 在公钥属性界面的**使用身份或证书文件**，选择您保存到本地的密钥对txt文件，然后单击**确定**。



3. 保存完成后，在快速连接界面点**连接**，然后在弹出的新建主机密钥界面，点击**接受并保存**。



4. 此时，即可登录您的Linux云服务器。



Linux或Mac使用SSH密钥登录Linux实例

1. 找到您所下载的.txt密钥对文件在本地机上存储路径，例如：`/root/vm_key-k-bpuiGkwG.txt`（vm_key-k-bpuiGkwG.txt为您创

建密钥对时生成的txt文件。）

2. 打开Linux /MAC的终端，修改密钥对文件的属性：`chmod 400 /root/vm_key-k-bpuiGkwG.txt`

3. 在终端运行命令：`ssh -i /root/vm_key-k-bpuiGkwG.txt root@实例的弹性公网IP`，即可登录您的云服务器。

使用VNC远程登录云服务器

浏览器要求

VNC服务对浏览器有一定的要求，现在只支持如下版本的浏览器，使用之前注意VNC页面的提示。

名称	版本
Google Chrome	16+
Firefox	3.6+
iOS Safari	6.1+
Opera	11+
Internet Explorer	9+

操作步骤

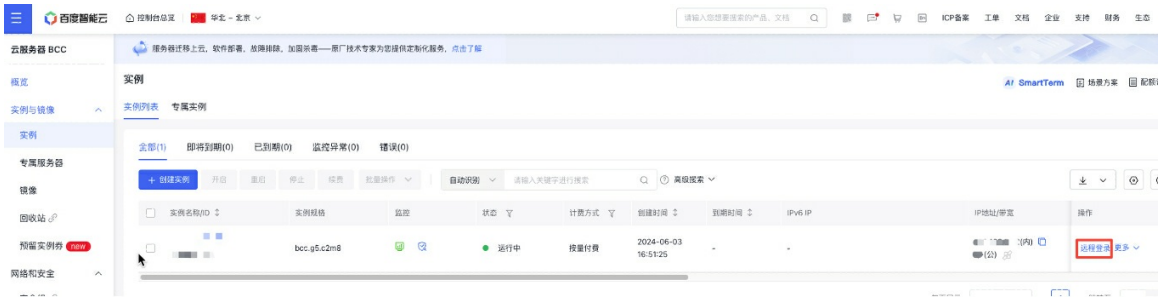
> **说明：**登录之前请确保云服务器处于“运行中”的状态，如果是“已关机”或“已停机”状态无法登录云服务器。

1. 登录云服务器BCC控制台进入实例列表页面。

2. 有两种方式可以选择VNC远程登录云服务器操作系统。

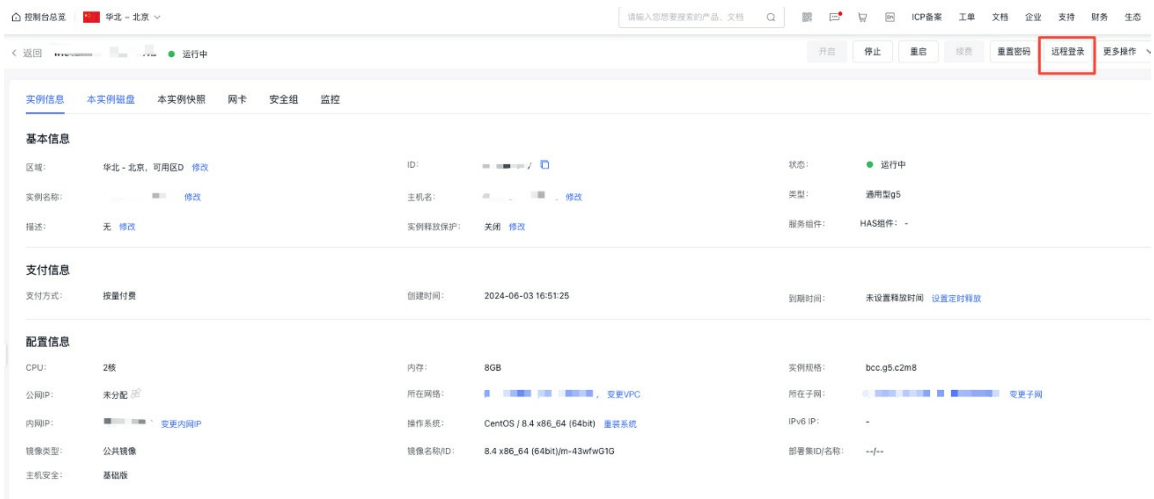
   - 方法一

     在“操作”列，选择“远程登录”后，选择“通过VNC登录”，进入VNC窗口。



   - 方法二

     在“实例详情”界面，选择右上角的“远程登录”后，选择“通过VNC登录”，进入VNC窗口。

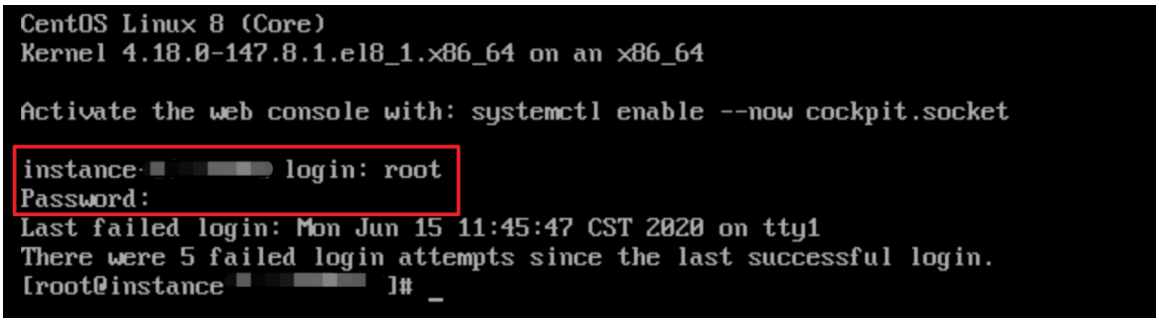


3.（可选）如果弹出VNC窗口黑屏，说明系统处于休眠状态，需要按键盘的任意键恢复系统。

4. 在VNC窗口的对话框中，在“login”后输入用户名，按**Enter**。

5. 在“Password”后输入密码，按**Enter**。

   输入的密码默认不显示，如下图所示：



6. 如果您忘记了实例的密码，可以对实例密码进行重置，具体请参考重置密码。

> 说明：
>
> - Linux系统的默认用户名为root，Windows系统的默认用户名为Administrator。
> - 如果密码进行过重置，需要使用最新的密码进行登录。
> - 为了安全考虑，VNC远程登录的网址只能使用一次。如果对同一VNC网址进行刷新的话，网址链接会失效。

5.（可选）用户登录操作系统后，需要执行如下红色框中的命令的话，可以点击下拉菜单中的按钮。

使用SmartTerm连接云服务器

概述

SmartTerm是百度智能云云服务器BCC提供的一款远程连接终端，帮助用户随时随地通过浏览器远程登录服务器管理业务，相比本地远程终端更轻量便捷，同时搭载AI助手，助力提升开发与运维的效率。

注意事项

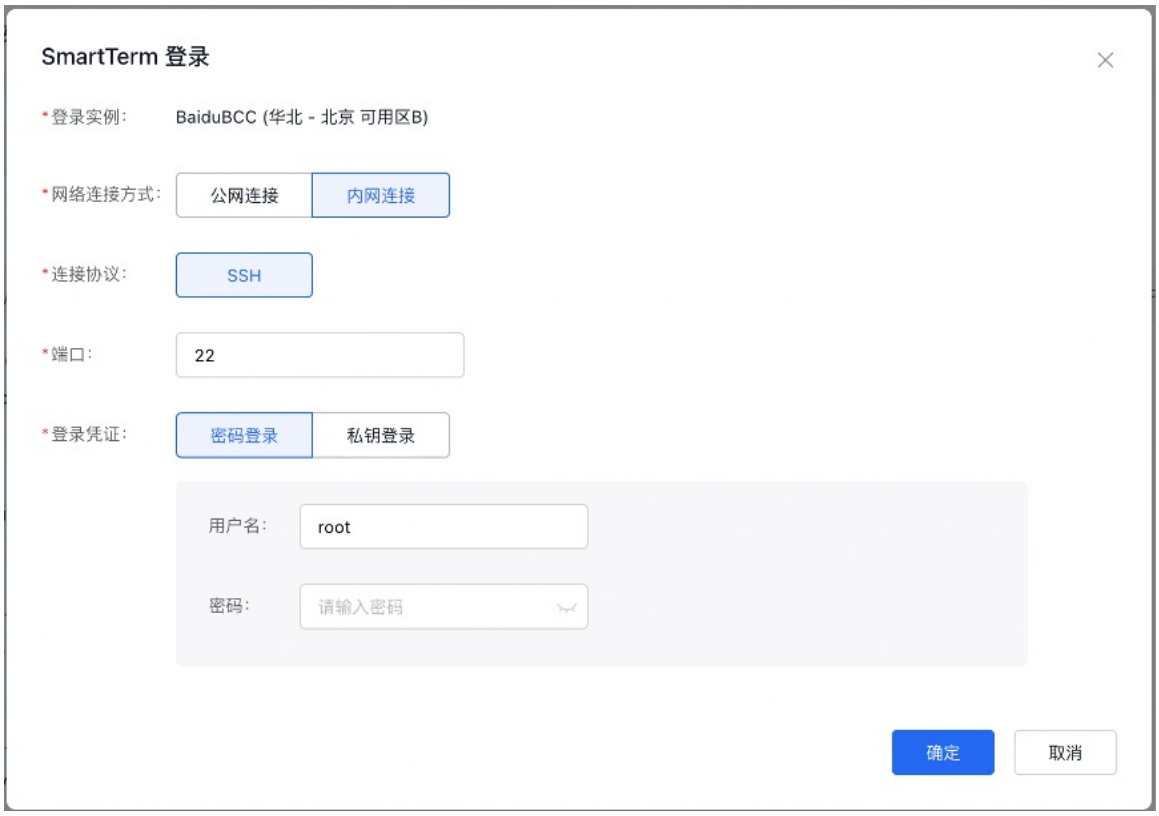
- 当前仅支持Linux系统的云服务器，Windows系统暂不支持。
- 实例状态需为**运行中**状态。

连接方式

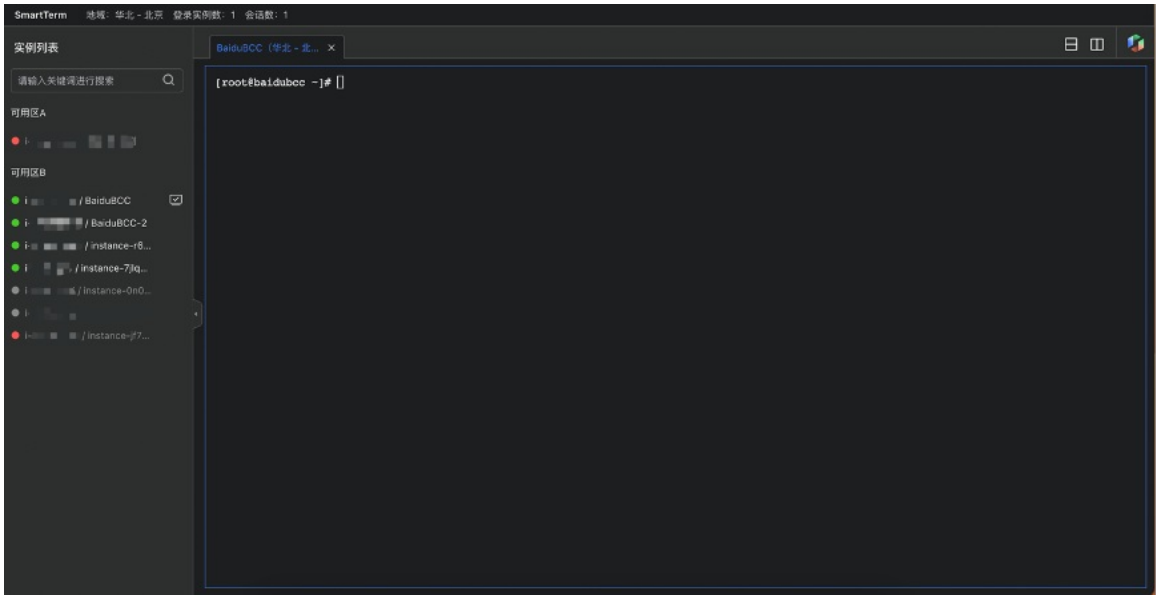
网络连接方式：支持公网连接和内网连接，即使云服务器没有绑定公网IP，也可以通过内网连接的方式登录云服务器。

> 说明：若您选择公网连接，则会通过您的公网IP连接云服务器，会产生一些网络流量。

登录方式：支持密码、密钥登录方式。

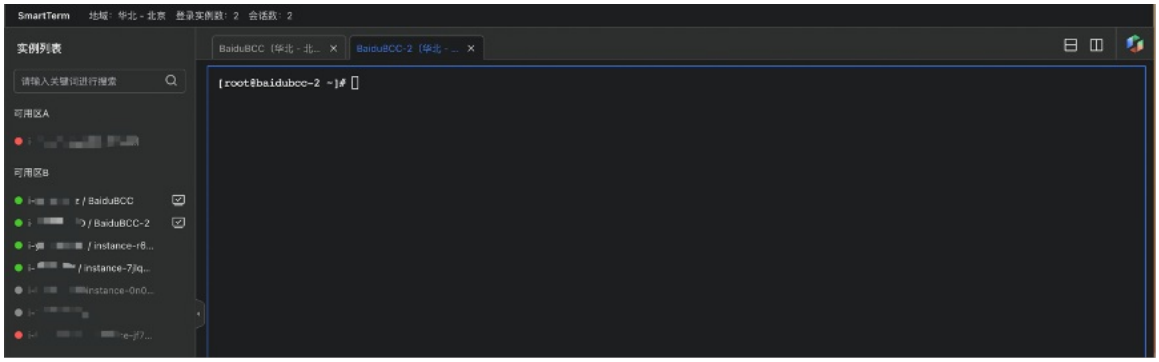


多实例管理 SmartTerm支持同时连接多台云服务器实例，您可以通过点击左侧栏里展示的实例登录，连接同地域下多台实例。

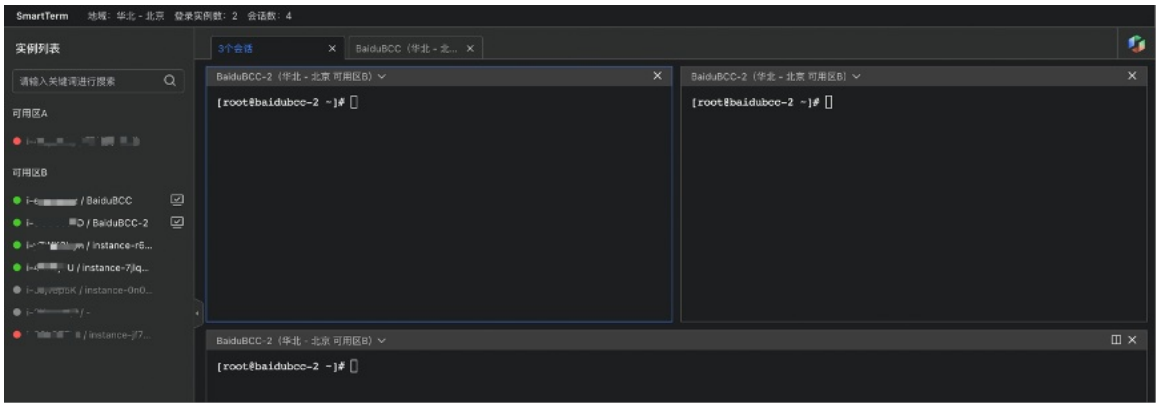


分屏 SmartTerm支持对单个实例开启多个会话，可以通过右上角分屏图标，进行横向或纵向分屏。

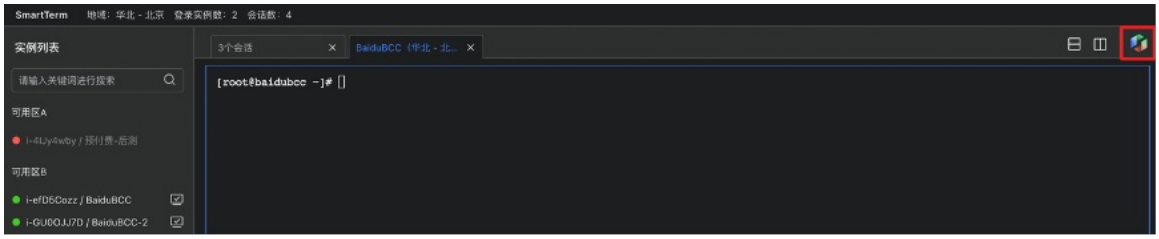
> 说明：最多支持同时4个分屏



如下图所示为3个分屏效果。同时也支持在分屏中切换成其他实例。



复制粘贴 支持使用快捷键Ctrl+C和Ctrl+V，或者通过选中内容并右键点击选择复制/粘贴，将选定的内容从本地设备复制到实例系统中，或从实例系统复制到本地设备。 **AI助手** AI助手是一款基于大语言模型的终端AI问答助手，能够帮助用户通过AI 获得帮助以便更高效地使用终端工具，提升用户终端操作效率。它能根据您输入的问题，提供对应的命令建议或方案，操作快速便捷易上手。 对于AI助手反馈的内容，支持一键复制或一键导入，简化操作，快速完成任务。 **AI问答界面** 1.单击页面右侧顶部AI助手图标，打开下图所示AI问答界面；



2.输入您的问题，Enter（回车键）或单击输入框右侧的发送图标，发送问题，AI将在聊天框中提供回答。

3.若AI的回答您认为正确，可以一键复制或一键导入；若想了解回答更多信息，可以点击“查看更多”，AI会返回更详细的解释说

明。若AI的回答您认为不太准确，可以“重新生成”。

4.若您觉得回答满意，赞一下；若您觉得回答不满意，踩一下。您的反馈能很好地帮助训练AI助手，让AI助手下次能更好、更准确地回答。

> 说明：AI生成的内容，仅供参考，请您谨慎采纳。

操作步骤 1.登录 云服务器BCC控制台 进入实例列表页面。

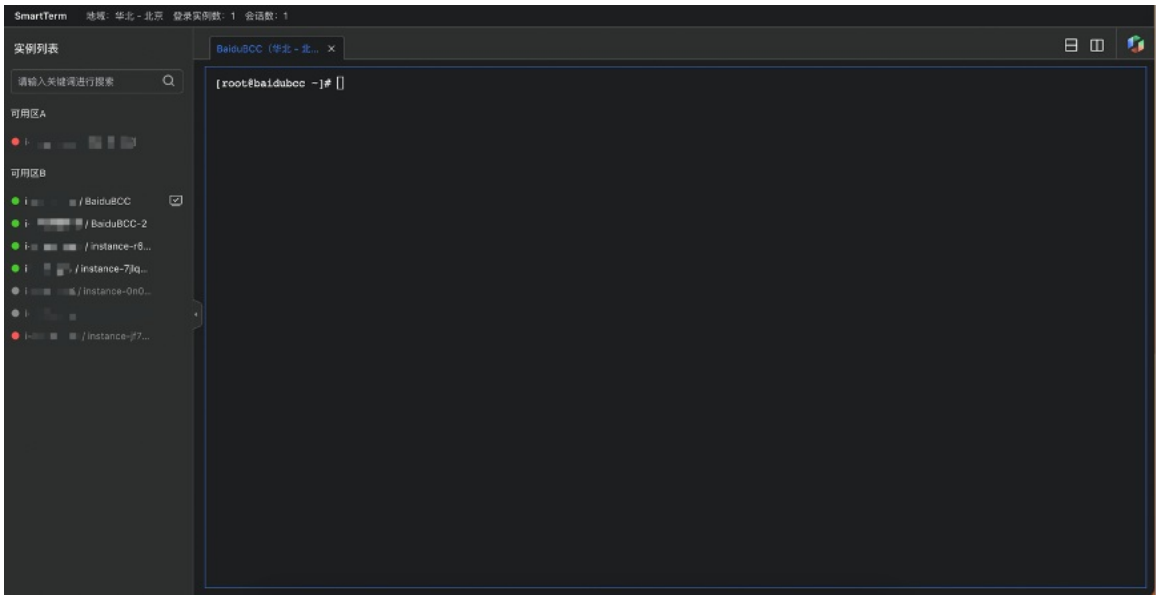
2.选择您要远程登录的实例，在右侧操作列单击 **远程登录** 。

3.在弹出的登录方式选择框中，单击 **通过SmartTerm登录** 。

4.输入密码，点击 **确定** 。

> 说明：网络连接方式默认为“内网连接”，可根据情况调整切换为“公网连接”。

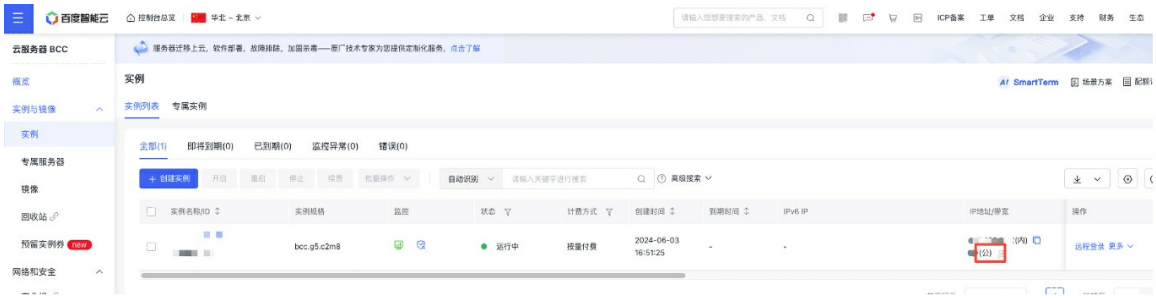
5.成功连接至实例系统内，可以进行所需的操作和任务。



登录Windows实例

使用远程桌面登录Windows实例

1. 登录云服务器BCC控制台进入实例列表页面。查看云服务器的概要信息，找到其公网IP。



2. 在云服务器的状态处于**运行中**时，启动客户端进行连接。

   - 对于Windows系统，使用Windows自带远程桌面或者通用的RDP远程桌面工具登录。

使用VNC远程登录云服务器

浏览器要求

VNC服务对浏览器有一定的要求，现在只支持如下版本的浏览器，使用之前注意VNC页面的提示。

名称	版本
Google Chrome	16+
Firefox	3.6+
iOS Safari	6.1+
Opera	11+
Internet Explorer	9+

操作步骤

> **说明：**登录之前请确保云服务器处于“运行中”的状态，如果是“已关机”或“已停机”状态无法登录云服务器。

1. 登录云服务器BCC控制台进入实例列表页面。

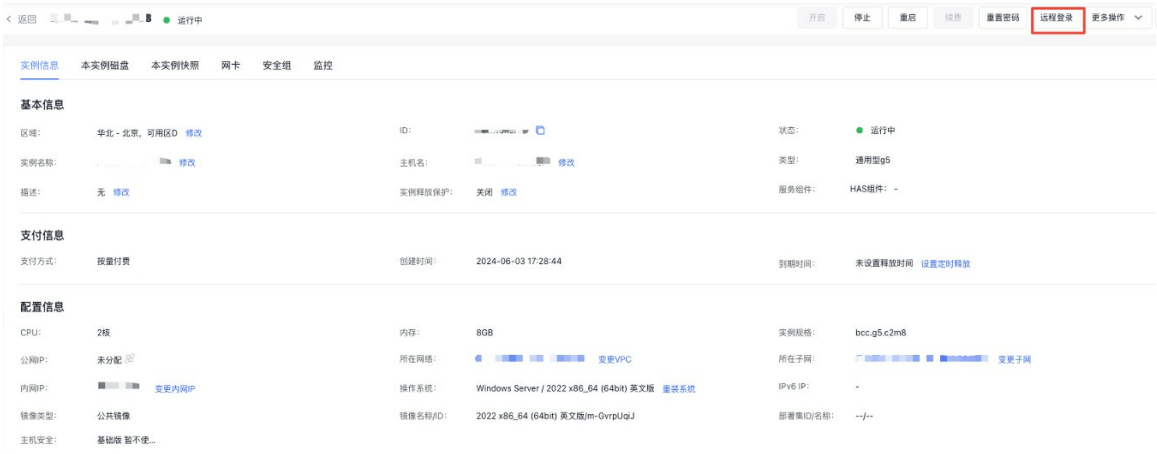
2. 有两种方式可以选择VNC远程登录云服务器操作系统。

   - 方法一

     在“操作”列，选择“远程登录”后，选择“通过VNC登录”，进入VNC窗口。

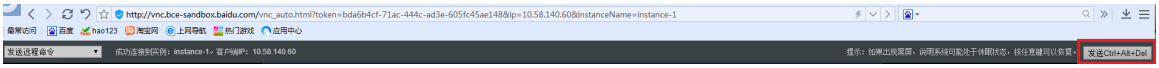
- 方法二

  在“实例详情”界面，点击右上角的“远程登录”后，选择“通过VNC登录”，进入VNC窗口。



3.（可选）如果弹出VNC窗口黑屏，说明系统处于休眠状态，需要按键盘的任意键恢复系统。

4.（可选）对于Windows操作系统，点击按钮“发送Ctrl+Alt+Del”，发送Ctrl+Alt+Del命令到操作系统。



5. 初次使用VNC登录实例需要输入该实例的用户名和密码。如果您忘记了实例的密码，可以对实例密码进行重置，具体请参考重置密码。

> 说明：
>
> - Linux系统的默认用户名为root，Windows系统的默认用户名为Administrator。
> - 如果密码进行过重置，需要使用最新的密码进行登录。
> - 为了安全考虑，VNC远程登录的网址只能使用一次。如果对同一VNC网址进行刷新的话，网址链接会失效。

6.（可选）用户登录操作系统后，需要执行如下红色框中的命令的话，可以点击下拉菜单中的按钮。

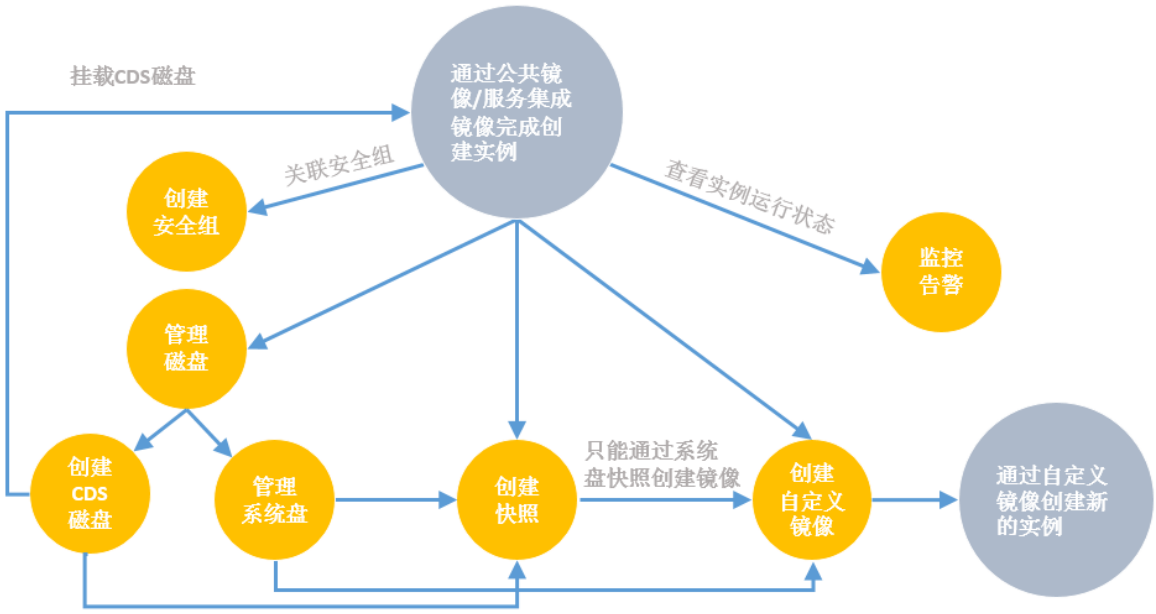
查看实例

背景信息

完成BCC实例创建后，您可以在实例列表中查看已经创建完成的BCC实例，可以按照实际需求完成关联安全组、创建快照、创建自定义镜像等操作。

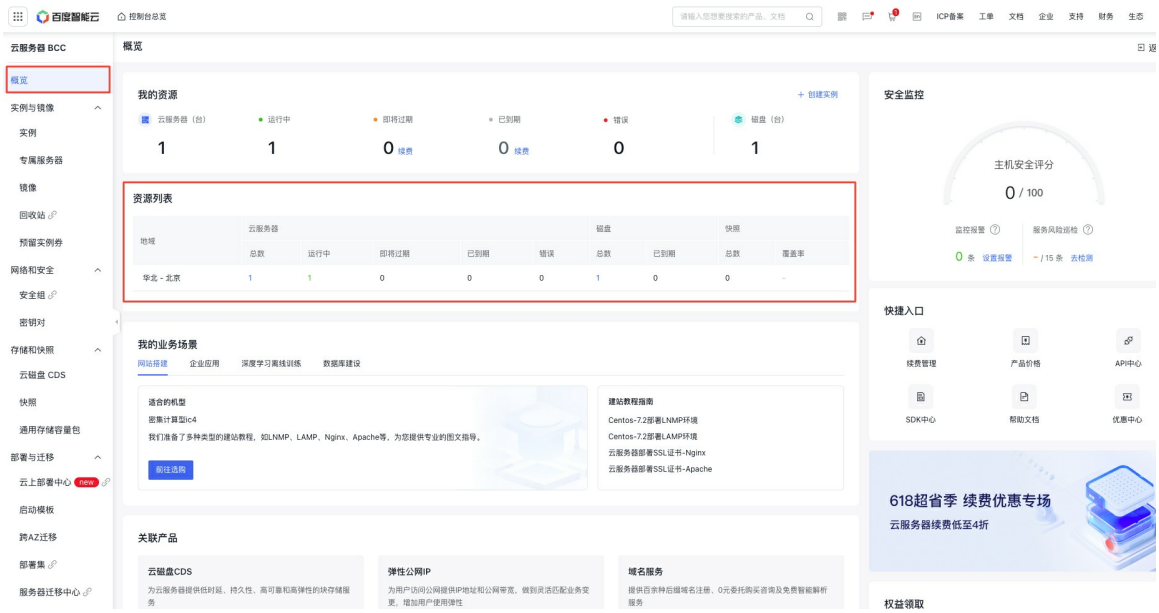
切换不同的区域，您可以查看到你在不同区域创建的实例。

第一次通过公共镜像或服务集成镜像完成创建BCC实例后，您可以完成更多与实例有关的拓展操作，通过实例或快照创建自定义镜像。

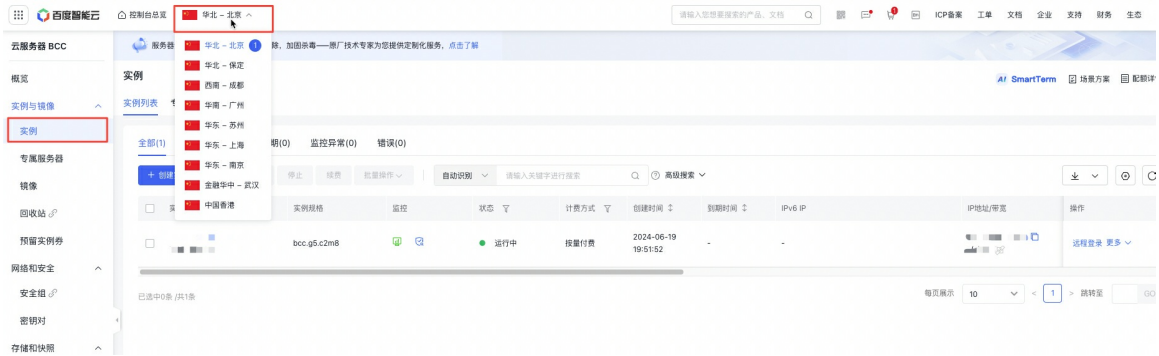


操作步骤

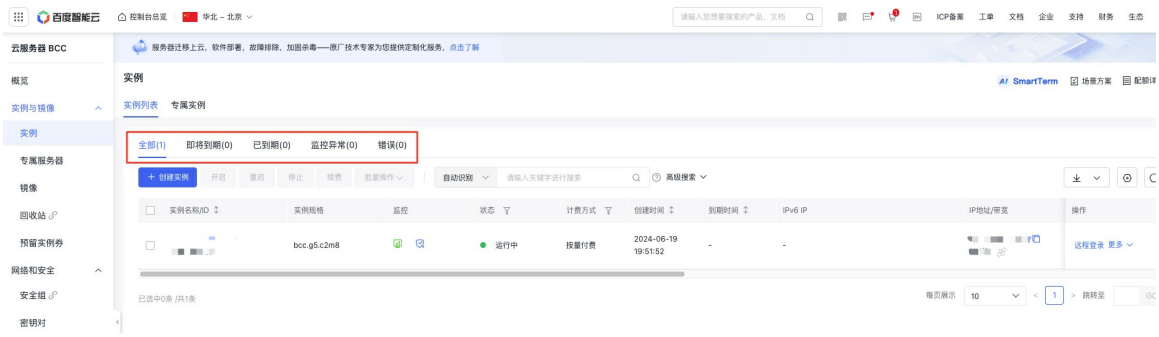
1. 登录 云服务器BCC控制台 进入“概览”页面，可以看到实例的整体分布情况。



2. 点击“实例”进入实例列表，点击左上角的“地域选择”，可以查看不同地域拥有的实例数量。



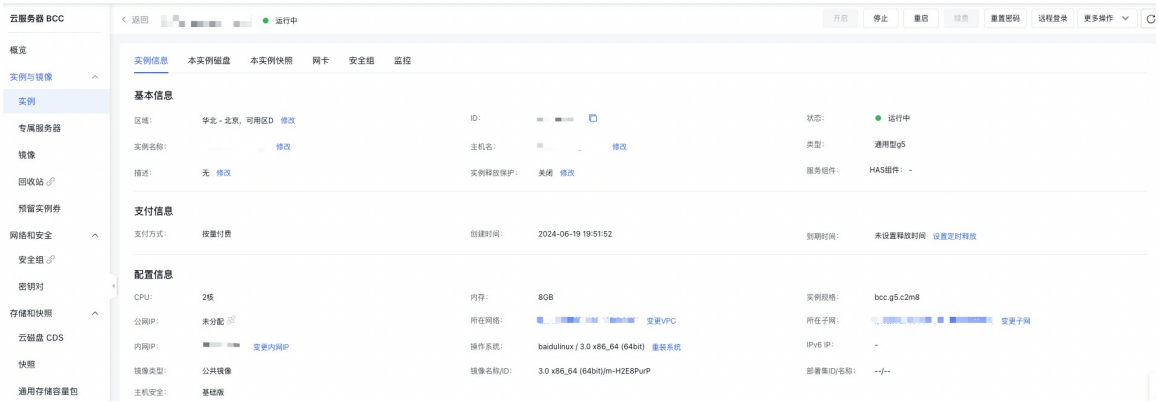
3. 选择地域后即可进入实例列表页面，可看到您已创建的云服务器列表，如下图所示。



4. 如果实例较多，您可以通过“实例名称”、“实例ID”、“实例内网IP”、“实例公网IP”等进行快速搜索，如下图所示。

5. 如需自定义查看实例列表项，您可以单击顶部右侧的 “自定义列” 按钮，自定义显示的实例列表项，如下图所示。

6. 您可在云服务器列表中“实例名称”列点击某一实例名称链接，进入“实例详情”页面，进行状态信息的查看，如下图所示。



- 在“**实例详情**”页签中，您可以查看该实例的基本信息、付费信息和配置信息。同时可以根据实际需求对实例信息进行修改或者升级配置等操作：变更实例名称、添加实例描述、重置密码、VNC远程、设置定时释放（按量付费实例）、配置升级、带宽调整、重装操作系统。

> **说明：**
>
> BCC实例通常拥有一个公网IP和一个内网IP，如果需要在BCC上部署互联网服务，监听端口需要绑定内网地址或0.0.0.0来发

> 布服务。BCC上的服务的客户端使用公网IP访问服务。

- 在“**本实例磁盘**”页签中，您可以查看挂载在该BCC实例中的系统盘和CDS磁盘的基本信息，包括磁盘名称、创建时间、类型、状态、容量、所属实例/IP等。同时可以完成创建CDS磁盘、创建快照、创建自定义镜像等操作。仅系统盘支持创建自定义镜像。
- 在“**本实例快照**”页签中，您可以查看该实例中已经创建的快照。同时可以进行回滚快照、创建快照、创建自定义镜像等操作。仅系统盘支持创建自定义镜像。
- 在“**安全组**”页签中，您可以查看当前应用的安全组规则。同时可进行查看与编辑当前安全组、关联安全组等操作。
- 在“**监控**”页签中，您可以查看当前实例的基本运行状态，包括CPU使用率、每秒磁盘IO读写次数/读写量等信息，可以根据您需求设置监控信息内容，具体可参考添加BCC报警策略。

通过API查看实例

- 您可通过 查询实例列表接口 查询所有实例的详细信息。
- 您可通过 查询指定实例详情接口 查询指定实例的详细信息。

启动实例

前提条件

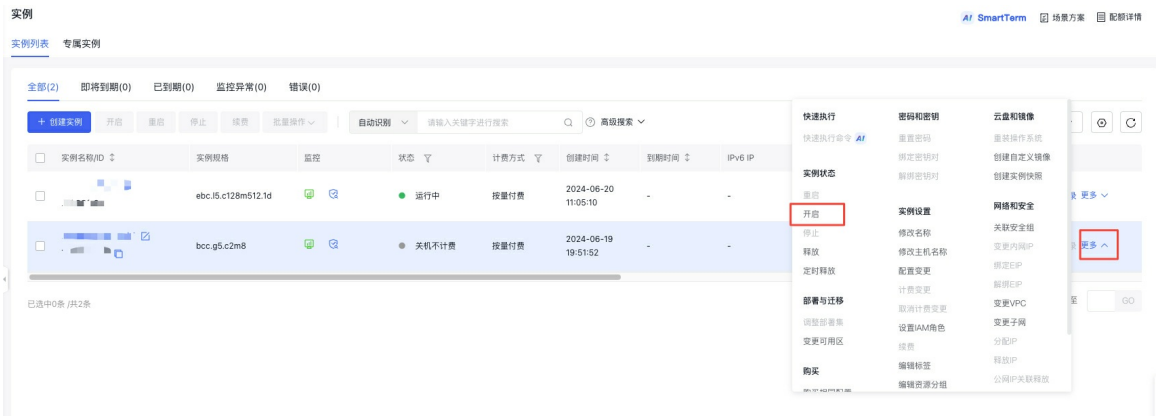
创建BCC实例后系统默认启动云服务器。只有当您已经 **停止** 该服务器后才能在实例列表中执行启动的操作。

操作步骤

有如下两种方式可以对选定的云服务器进行启动操作。

- 方法一

1. 登录云服务器BCC控制台进入实例列表页面。
2. 根据您的需求，选择不同的启动实例方式。
    - 启动单个实例：选择您要启动的实例，在右侧操作列单击 “更多 - 实例状态 - 开启”，如下图所示。或者勾选实例，在顶部单击“开启”。

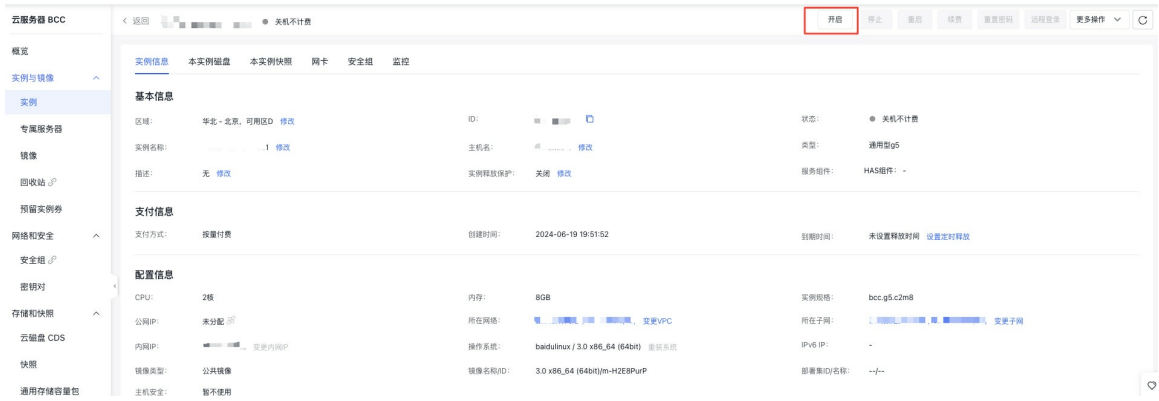


    - 启动多个实例：勾选您要启动的实例，在顶部单击 “开启”，如下图所示。

3. 在弹出的 **启动** 对话框中单击确认按钮，完成实例的启动操作。
4. 此时实例的状态变为 **重启中**，启动成功后实例的状态变更为 **运行中**。

- 方法二

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在“实例名称”列，选择要操作的一台云服务器的名称链接，进入“实例详情”页面。
3. 在“实例详情”页面，单击右上角 **开启** 按钮。



4. 在弹出的 **启动** 对话框中单击确认按钮，完成实例的启动操作。
5. 此时实例的状态变为 **重启中**，启动成功后实例的状态变更为 **运行中**。

通过API启动实例

您可通过 启动实例接口 启动指定的实例。

停止实例

停止实例

前提条件

实例必须处于**运行中**状态。

裸金属EBC实例不支持强制关机。

> 强制关机等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。

背景信息

当您需要停止实例时，可以进行停止操作。停止实例相当于本地计算机的关机操作。

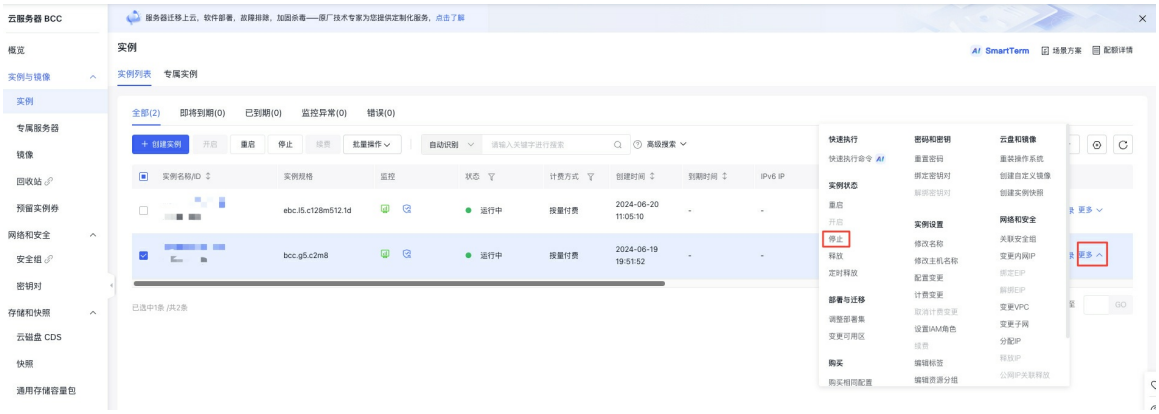
注意事项

- 停止实例后相关服务将不可访问，请谨慎使用。

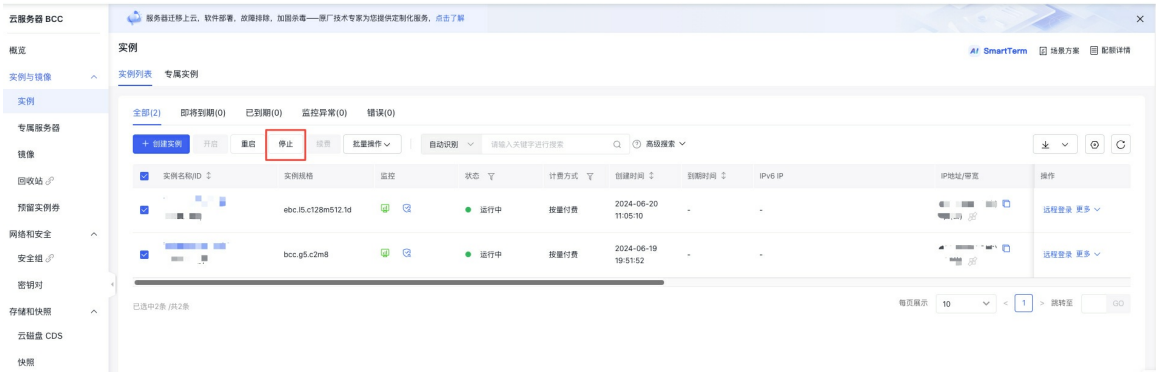
- 停止后实例仍然保留并继续收费，对于**BCC的按量付费实例**，您可选择**关机不计费**，具体参考按量付费实例关机不计费。
- 您可使用操作系统对应命令停止实例（Windows 系统下使用关机操作，Linux 系统下使用 shutdown 命令），也可以使用控制台停止实例。

操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。
2. 根据您的需求，选择不同的停止实例方式。
   - 停止单个实例：选择您要停止的实例，在右侧操作列单击 “更多 - 实例状态 - 停止”，如下图所示。或者勾选实例，在顶部单击“停止”。



   - 停止多个实例：勾选您要停止的实例，在顶部单击 “停止”，如下图所示。



3. 在弹出的**停止**对话框中，确认实例信息，设置停止模式，单击**确定**按钮，如下图所示。

> **说明：**关机不计费的操作可参考按量付费实例关机不计费。

4. 停止成功后，实例状态将变为**已关机**。

通过API停止实例

您可通过停止实例接口停止指定的实例。

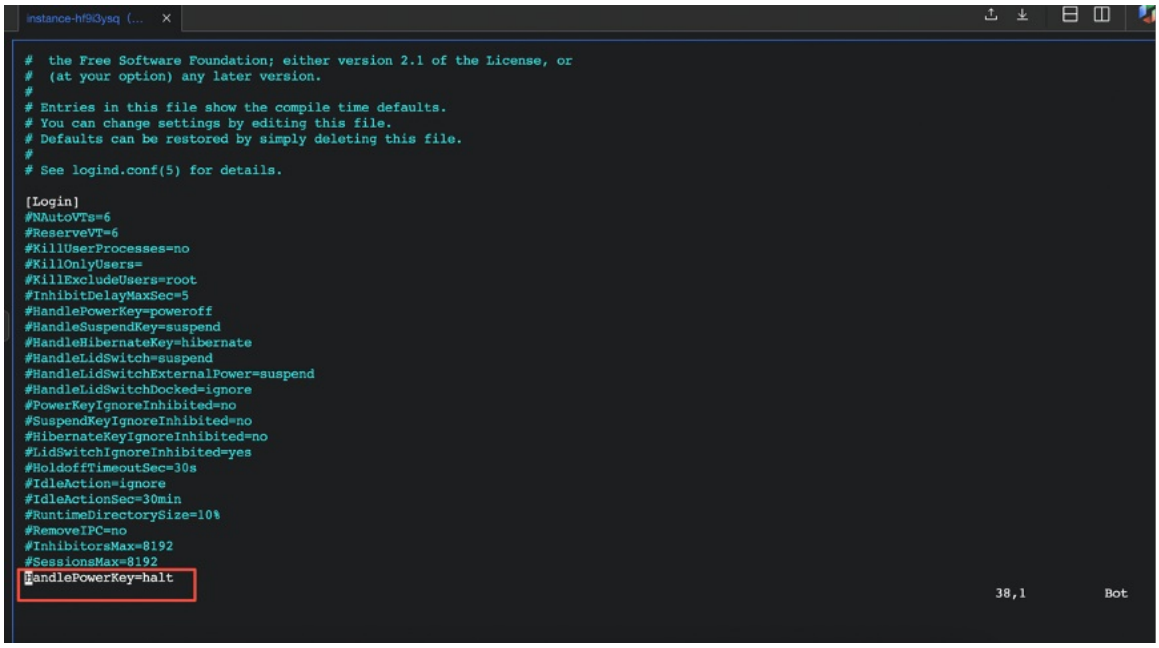
弹性裸金属实例与弹性高性能计算集群实例halt能力配置说明

背景信息

- 弹性裸金属实例ebc与弹性高性能集群实例ehc的关机能力，依赖实例支持halt能力。如果不支持halt，开机时，有一定概率启动失败。
- ebc与ehc最新发布的公共镜像已集成halt功能，如您使用最新的公共镜像创建了实例，可直接使用关机功能。
- 如实例不支持halt，可直接在系统中增加halt配置或使用云助手一键执行增加halt配置的命令。
- 如您使用的是自定义镜像，请在实例增加halt配置后，重新制作自定义镜像，确保新购实例可正常使用关机功能。

操作步骤

1. 检查ebc或ehc实例是否支持halt：进入实例，在/etc/systemd/logind.conf中查询是否有HandlePowerKey=halt。如已支持，可直接使用关机功能。



2. 如不支持halt，可直接在系统中增加halt配置：

- 进入实例中，执行以下命令：

```
echo "HandlePowerKey=halt" >> /etc/systemd/logind.conf
```

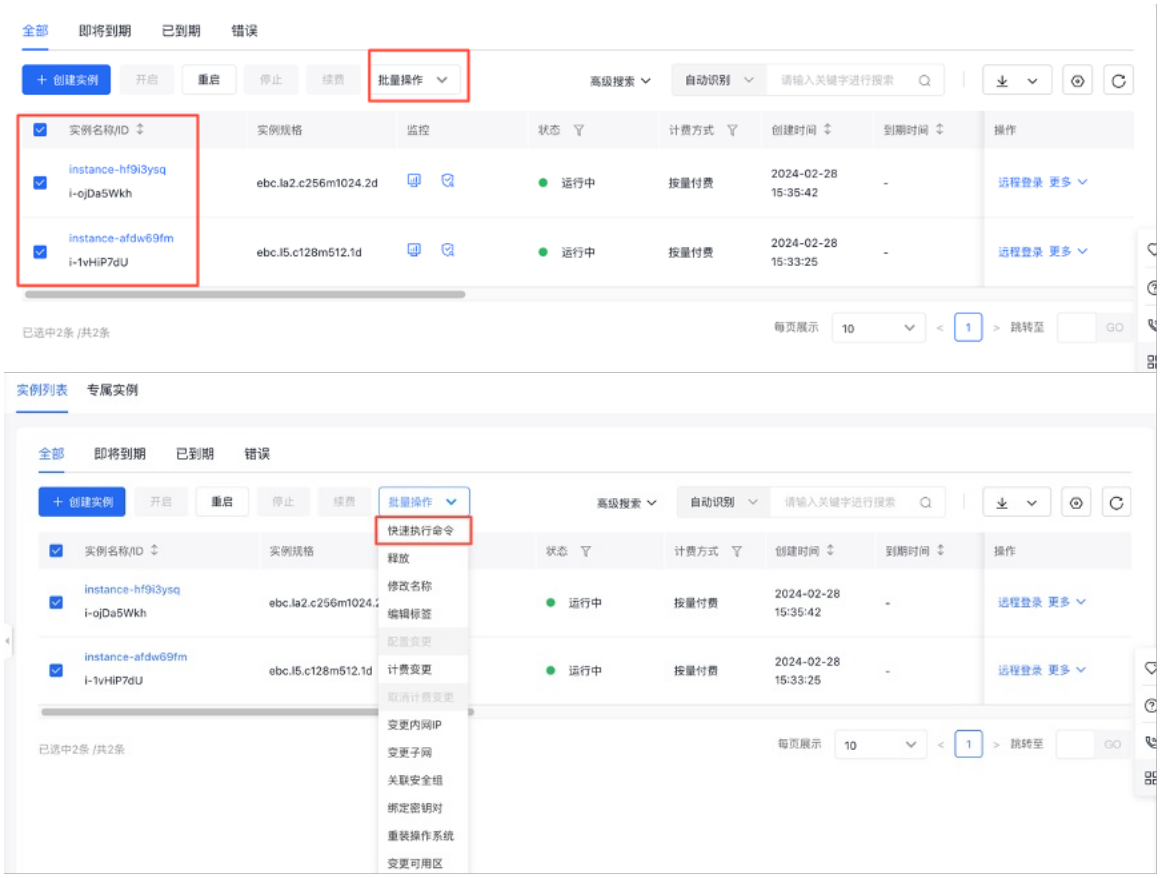
```
systemctl daemon-reload
```

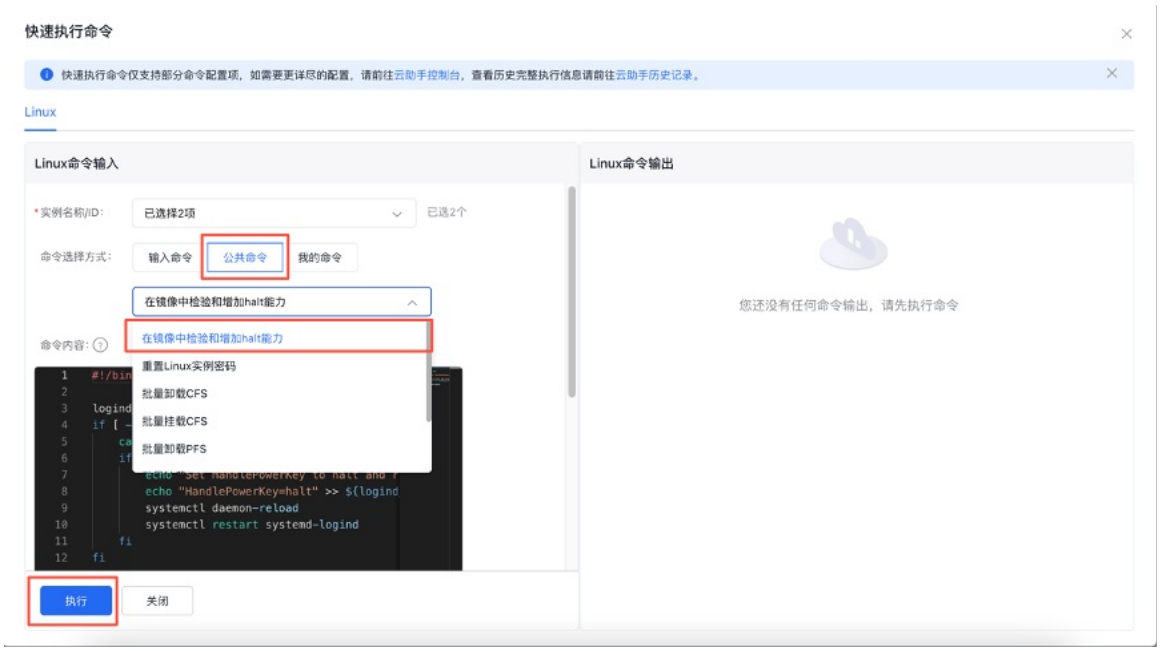
```
systemctl restart systemd-logind
```

- 执行命令后立即生效，不需重启实例，实例即可支持halt能力。

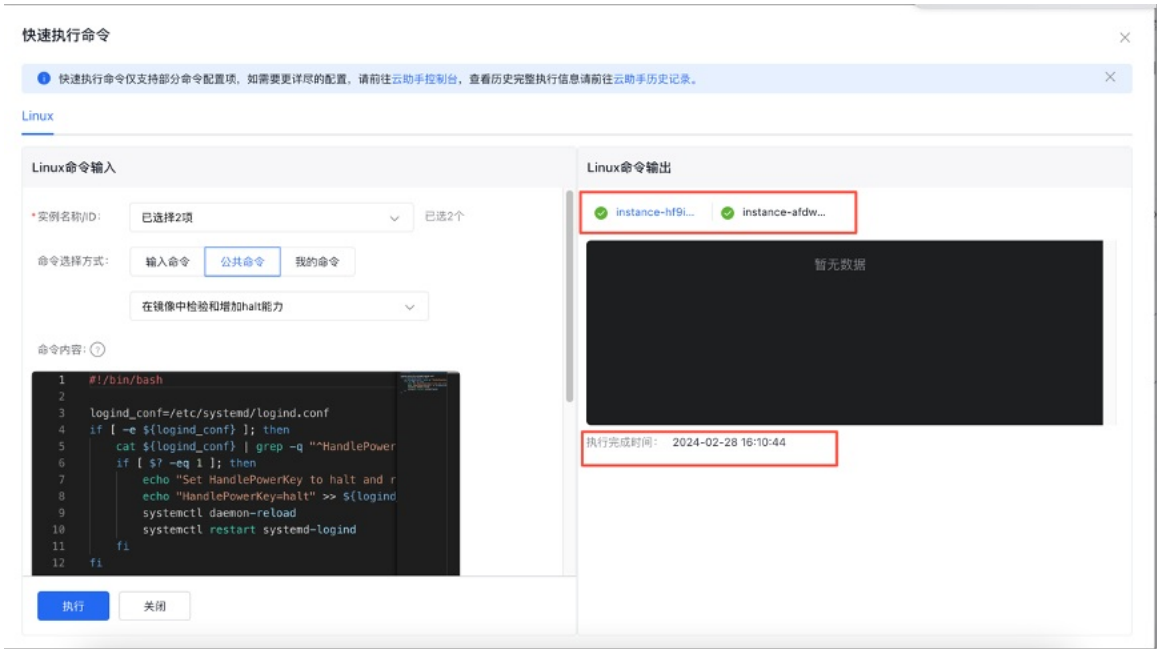
3. 使用云助手一键执行增加halt配置的命令：

- 使用与助手的快速执行命令功能，执行增加halt的公共命令：可选择批量操作实例。

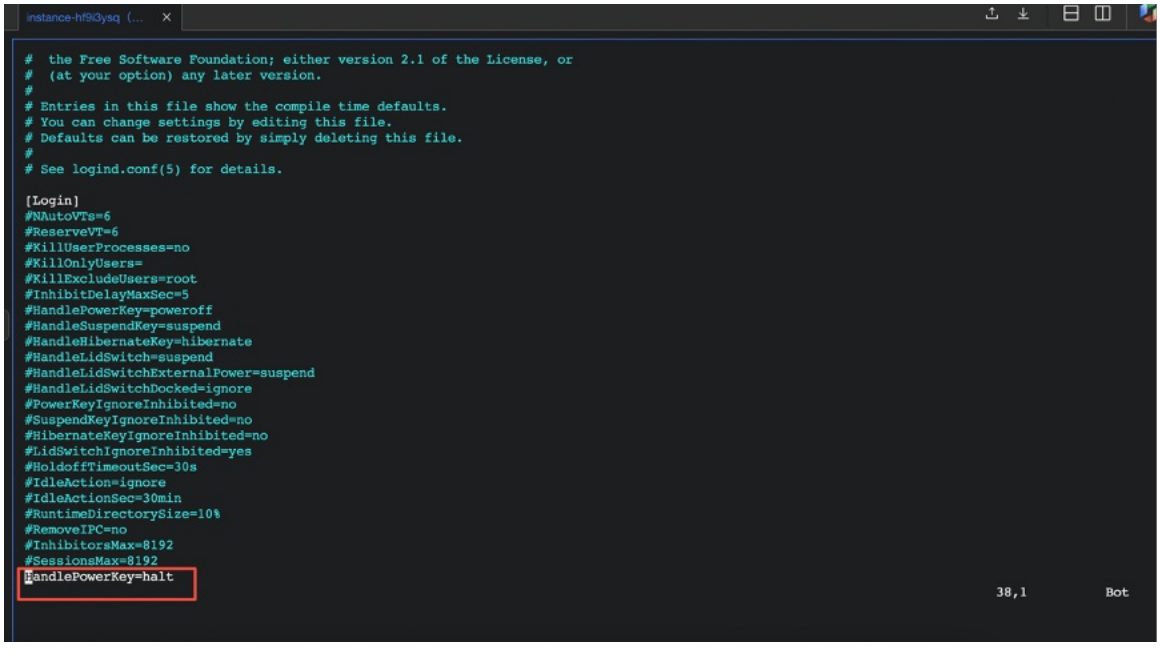




- 执行公共命令，待所有实例执行成功后，退出界面即可。



- 执行成功后，实例不需重启或其他操作，“HandlePowerKey=halt”自动被加入到“/etc/systemd/logind.conf”中。



4. 如您使用的是自定义镜像，通过增加halt后的实例，重新制作自定义镜像。用更新后的自定义镜像创建或重装的实例，均可正常使用关机功能。

重启实例

前提条件

实例必须处于**运行中**状态。 裸金属EBC实例不支持强制重启，仅支持正常重启。

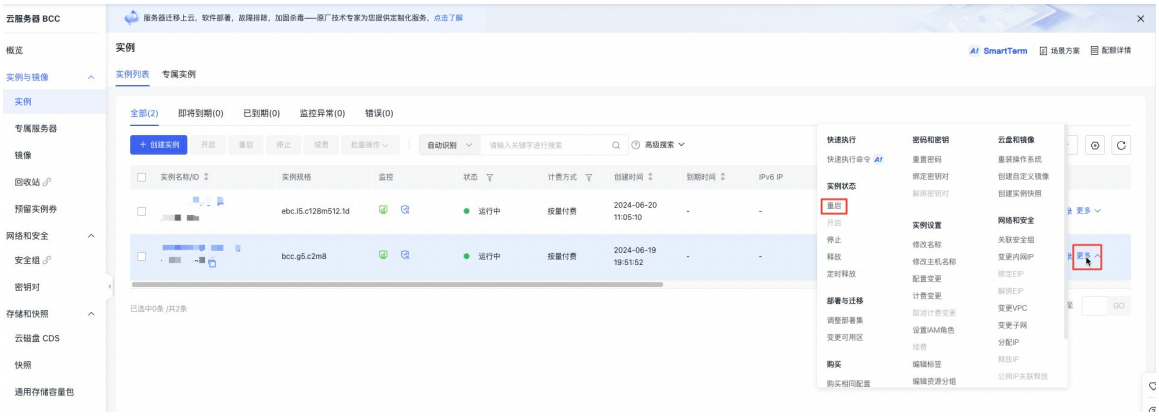
> 强制重启等同于断电重启，可能会丢失实例操作系统中未写入磁盘的数据。

注意事项

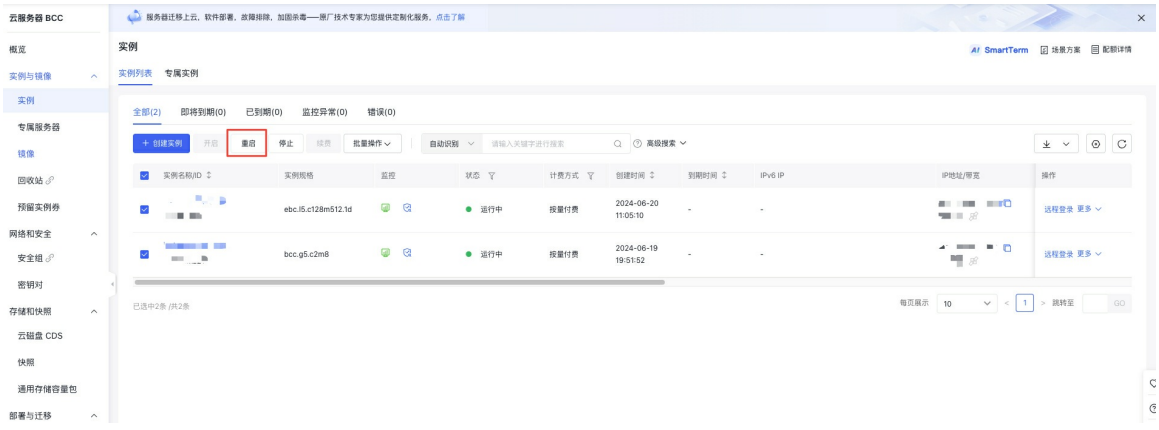
重启过程中，您的实例会停止工作，相关服务会**中断**，请谨慎操作。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 根据您的需求，选择不同的重启实例方式。
   - 重启单个实例：选择您要重启的实例，在右侧操作列单击 “更多 - 实例状态 - 重启”，如下图所示。或者勾选实例，单击"重启"。



   - 重启多个实例：勾选您要重启的实例，在顶部单击 “重启”，如下图所示。



3. 在重启对话框中，选择重启或者强制重启，然后单击**确定**。
   - 重启：正常重启。
   - 强制重启：可能会丢失实例操作系统中未写入磁盘的数据。

4. 确认重启后，实例状态将变成**重启中**。

其他方式

您也可以在运行中实例的实例详情页面，单击右上角的 **重启** 按钮执行重启。

通过API重启实例

您可通过重启实例接口重启指定的实例。

释放实例

当您不需要某个实例时，可以对实例进行释放操作。

前提条件

仅支持对**按量付费实例**和**已到期的包年包月实例**进行释放。

背景说明

对于包年包月实例和按量付费实例，提供不同的释放方式。

- 手动释放：对于已到期的包年包月实例和未欠费的按量付费实例，您可使用手动释放方式。

> 注意：已开启实例释放保护的实例无法手动释放。开启释放保护不影响实例的定时释放，欠费释放以及实例过期释放。

- 定时释放：对于按量付费实例，您可以选择指定时间自动释放实例。在定时释放前，您可以随时更改释放时间或取消定时释放。

注意事项

- 释放时，需要您输入验证码验证释放操作。
- 释放时，您可选择释放绑定的按量付费公网IP、云磁盘及快照。
- 释放时，提示报错“回收站的BCC实例已达到配额上限”，需要先进入回收站，将该区域已释放的BCC删除，然后再释放其他BCC实例。回收站BCC实例配额默认为50个。
- 实例释放后，已挂载的CDS磁盘自动卸载，基于此CDS磁盘的快照会保留。
- 实例释放后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

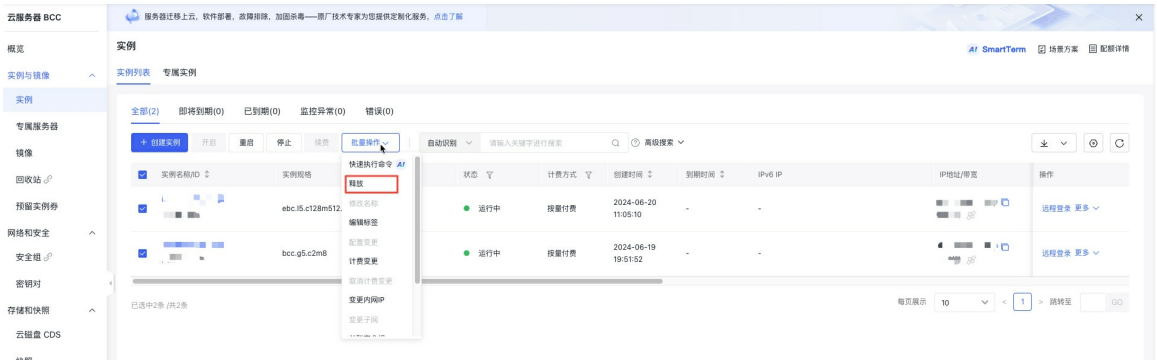
手动释放

1. 登录云服务器BCC控制台进入实例列表页面。

2. 根据您的需求，选择不同的释放实例方式。

   - 释放单个实例：选择您要释放的实例，在右侧操作列单击 “更多 - 实例状态 - 释放”，如下图所示。

   - 释放多个实例：勾选您要释放的实例，在顶部单击 “批量操作 - 释放”，如下图所示。



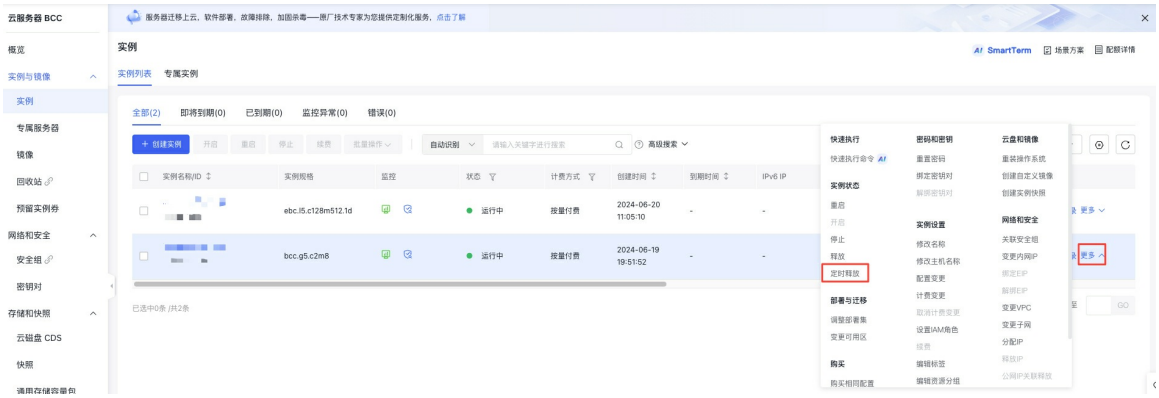
3. 在**释放前确认**框中确认要释放的实例，并根据您的需要选择**释放实例关联资源**以及**是否放入回收站中**。，如下图所示。

4. 发送手机验证码，单击**确定**，完成安全验证后实例将被立即释放。

定时释放

1. 登录云服务器BCC控制台进入实例列表页面。

2. 选择您要释放的**按量付费类型** 的实例，在右侧操作列单击 “更多 - 实例设置 - 定时释放”，如下图所示。



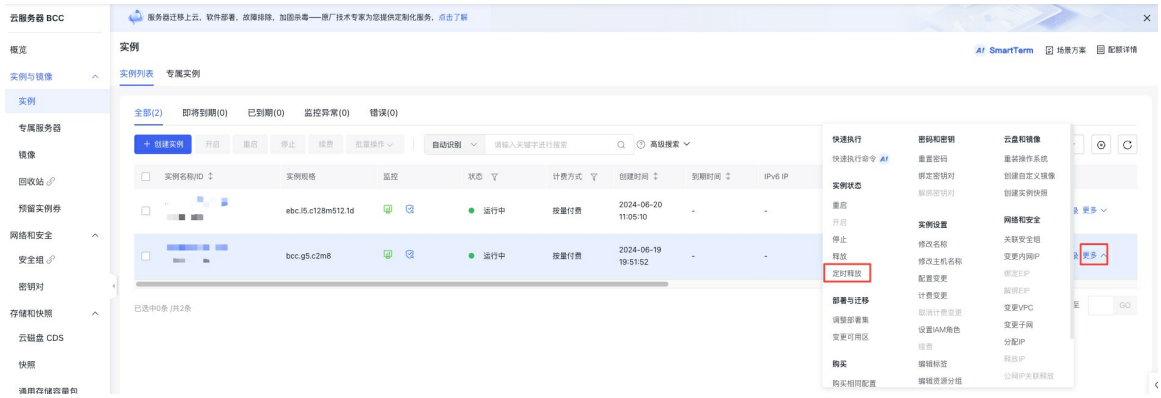
3. 在弹出的**创建释放计划**对话框，选择开启**自动释放设置**，并在**自动释放时间**项，选择您要释放的时间，精确到秒，单击**确定**，如下图所示。

> **注意： 释放时间必须晚于当前时间5分钟**。

4. 在**变更前确认**框完成变更确认，点击 **确定**。
5. 送手机验证码，单击**确定**，完成释放时间的设定。
6. 完成设定后，您可在右侧操作列，单击 “更多 - 实例设置 - 定时释放” 查看当前设定的定时释放时间。

取消/变更定时释放

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 选择您要变更定时释放的**按量付费类型**的实例，在右侧操作列单击“更多 - 实例设置 - 定时释放”，如下图所示。



3. 在弹出的**创建释放计划**对话框，选择**关闭自动释放**或**更改自动释放时间**，单击**确定**，如下图所示。

4. 在**变更前确认**框完成变更确认，点击**确定**。

5. 送手机验证码，单击**确定**，完成释放时间的**取消**或**变更**。

常见问题

能否释放公网IP？

用户可以自行释放空闲的EIP实例，节约租用成本，但以下三种情况不允许释放：

- 包年包月并且未到期的EIP实例；
- 已经绑定BCC或BLB实例的EIP；
- 因攻击导致暂停状态的EIP；

如需释放包年包月且未到期的EIP实例，请提交财务工单申请退款，具体是否允许退款，请以工单处理结果为准。

BCC释放后，自定义镜像和快照是否还存在？

- 自定义镜像：创建的自定义镜像只要不删除，将会一直保留。
- 快照：基于原系统盘的快照会自动删除。挂载的CDS磁盘若未同时释放快照，则基于此CDS磁盘的快照会保留。

通过API释放实例

您可通过释放实例接口释放指定的实例。

实例续费

背景信息

包年包月的云服务器实例，实例到期后会自动关机。到期后七天内，BCC将自动关闭并保留数据，同时给用户发送续费通知。续费后，系统将自动恢复原服务的使用。当到期超过七天，BCC数据资源将被清除，无法恢复。用户需要重新购买服务。

百度智能云提供两种续费方式，自动续费和人工续费。

自动续费

使用包年包月实例的用户，为了便于节省您的操作时间，并且避免服务到期时由于遗忘续费而导致服务被停止，建议您开通自动续费业务。自动续费定时为您的云资源进行续费，在实例到期前7天自动续费扣款，续费成功或失败都会通过短信和邮件进行提醒。

请注意下列详情说明，可能与您的业务相关：

- 开通自动续费业务，需要保证您的账户余额充足。
- 如果到期前7天由于账户余额不足导致自动续费失败，系统会在到期前3天再次进行自动续费扣款。如果扣款成功，则完成本周期内的自动续费业务；如果扣款失败，会在到期前1天再次扣款。
- 到期后，业务会自动停服，业务停服至保留资源期间不会自动续费，用户需要人工续费。
- 在一个周期内如果您已经对资源进行人工续费，自动续费将不再执行。

人工续费

- 包年包月的云服务器实例，实例到期后会自动关机，系统不会删除您的实例，默认保留7天，期间不收取费用，用户可选择在此期间人工续费。
- 按量付费的云服务器实例无需进行人工续费，系统将实时扣费，您可以在账单管理中参考您的消费账单。
- 在续费实例时，其挂载的CDS云磁盘和绑定的弹性公网EIP，可以和BCC实例同步关联续费。

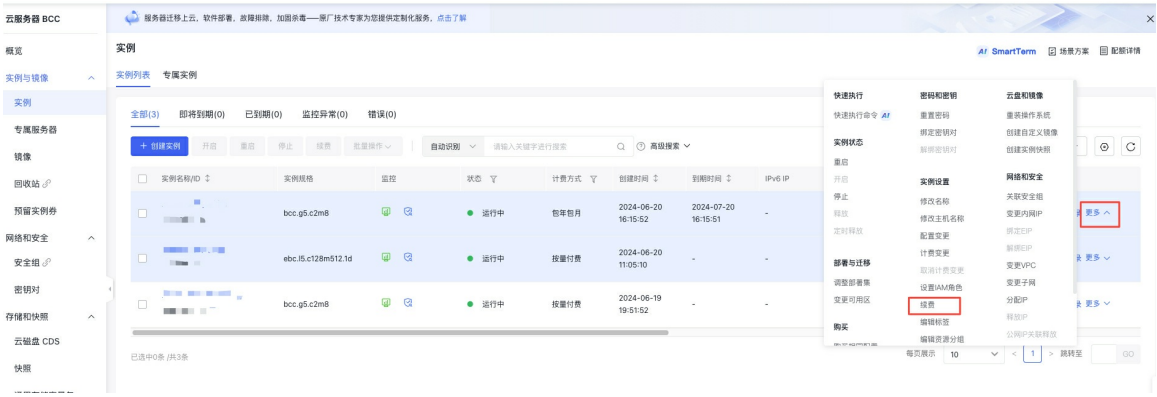
操作步骤

自动续费操作步骤

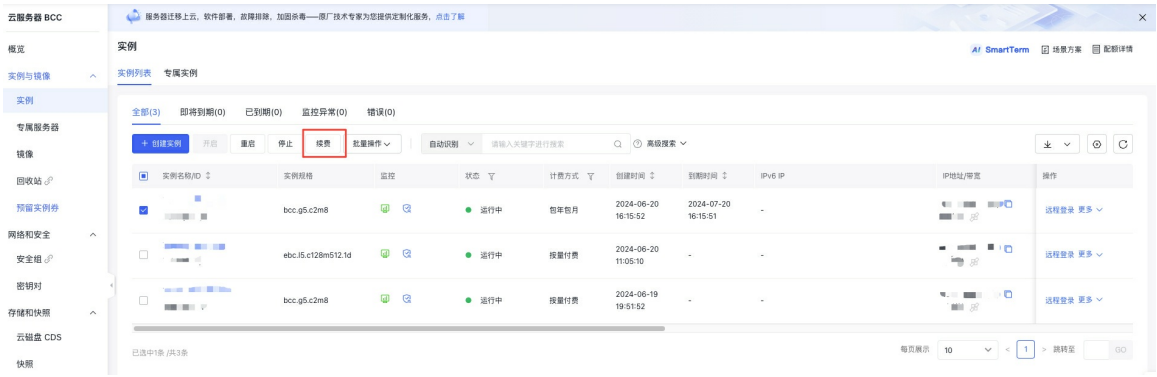
设置自动续费步骤请参考自动续费操作。

人工续费操作步骤

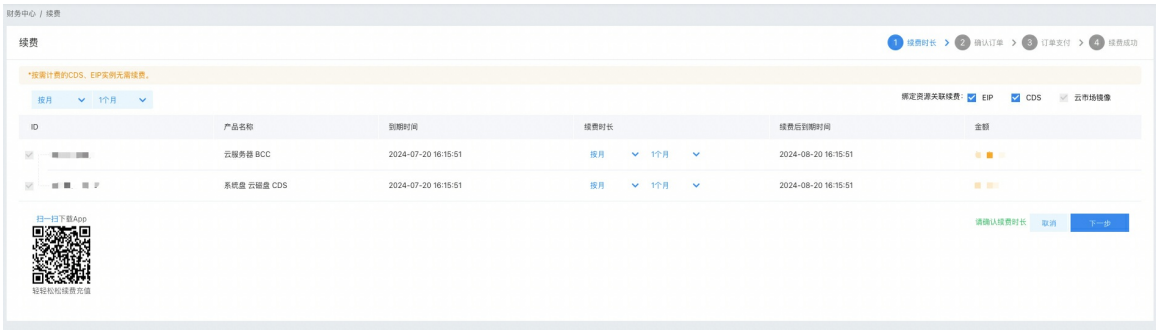
1. 登录云服务器BCC控制台进入实例列表页面。
2. 您可按照需要对一台或多台实例进行续费操作。
   - **单个实例续费**：选择要续费的实例，在右侧操作栏，单击 “更多 - 续费”，如下图所示。



   - **多个实例续费**：勾选要续费的实例，单击顶部的 “续费“ 按钮。



3. 选择续费时长，以及绑定的CDS和EIP实例的续费时长，点击“下一步”，进入“确认订单”页面。



4. 完成支付，实例续费成功。

其他方式

您可以在实例名称列，单击实例名称进入实例详情。在实例详情页中，单击续费按钮完成续费操作。

通过API续费实例

您可以使用实例续费接口为指定预付费实例续费。

重置密码

操作场景

如果您遗忘了密码或创建实例时未设置密码，您可以在控制台上重新设置实例的登录密码。

> **说明：** 重置密码会重启服务器，导致服务器停止几分钟，为保证你的服务不受影响，请在确认执行此操作前做好预案准备。

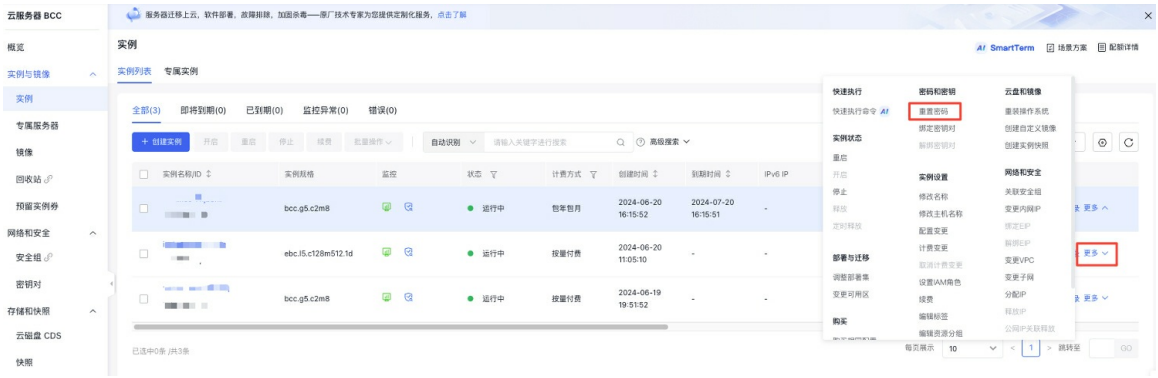
注意事项

- 实例必须处于**已停止**、**运行中**状态。
- 不能对加入伸缩组的实例重置密码。
- 不支持批量重置密码。

操作步骤

按以下步骤在控制台上修改实例的登录密码：

1. 登录百度智能云管理控制台，选择“产品服务 > 计算 > 云服务器BCC”。
2. 选择云服务器所在的区域；在云服务器列表中，勾选实例。
3. 在操作列选择“更多 > 密码和密钥 > 重置密码”。



4. 在弹出的对话框中，输入符合规则的新密码。

参数	规则	样例
密码	密码长度范围为8到32位字符。 密码必须同时包含这3类：字母、数字、字符仅使用!@#$%^*()。	Inst12#@

5. 点击“确定”确认密码。
6. 您需要输入手机验证码，进行二次验证，再次点击“确定”。
7. 新密码的生效：
    - 原**运行中**的云服务器，系统会自动重启，新密码生效。
    - 原**已停止**的云服务器，手动启动实例，新密码生效。

更多操作

在“实例详情”页，点击“重置密码”，一样可以重置实例的登录密码。

常见问题

为什么我的BCC无法重置密码？

出于安全方面的考虑，在总览页面重置密码时对密码的规则要求比较复杂，必须是：

- 8-32位字符
- 英文，数字和符号必须同时存在
- 符号仅限!@#$%^*()

请您检查您指定的新密码是否符合此规则。特别提醒：特殊符号只能使用规则中列举出来的几个。

通过API重置密码

您可以使用修改实例密码接口为指定实例重置密码。

视频参考

- BCC重置密码
- Linux通过单用户模式重置密码
- Windows系统BCC重置密码不成功

后付费实例关机不计费

按需付费实例关机不计费说明

实例关机不计费是指按量付费的实例通过停止操作使实例进入**关机不计费**状态。该实例的计算资源（CPU和内存等）将不再计费，而云磁盘（包括系统盘、数据盘）、弹性公网EIP（按带宽付费）等关联资源仍将继续收费。

使用限制

实例关机不计费功能仅适用于系统盘和数据盘均为云盘的按量付费的云服务器实例。

实例关机不计费功能不适用于以下场景：

- 不适用于在虚机系统内部开/关机。
- 不适用于搭载本地盘的实例。
- 不支持专属实例DCC。
- 不支持弹性裸金属实例EBC。
- 不支持弹性高性能计算集群实例EHC。
- 不支持抢占实例。
- 不支持CCE节点的实例。

若批量关机操作中同时包含**支持关机不收费**和**不支持关机不收费**的实例，则：

- 支持关机不收费的实例，关机后**停止CPU、内存等资源收费** 。
- 不支持关机不收费的实例，关机后继续正常收费。

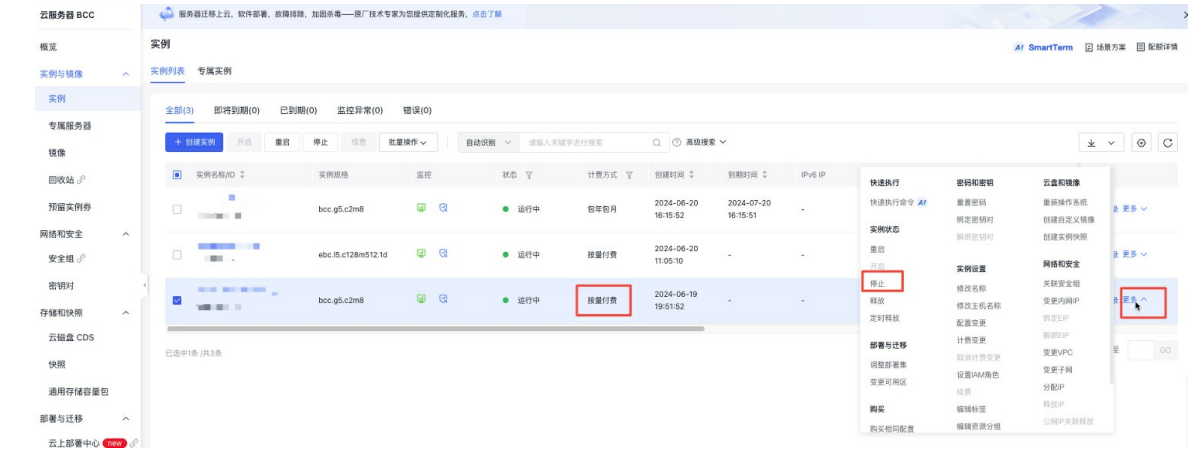
开启影响

启用实例关机不收费功能后，在实例关机后会有以下影响：

- 实例的CPU和内存将不再保留，所以再次启动实例时可能会失败。此时，您可以尝试再次启动，或者换一个时间再次启动。
- 关机期间无法支持除实例开机外的较多操作（如调整配置、调整硬盘、调整网络、重装系统、重启实例、重置密码、后付费转预付费、续费相关操作等），如需操作请开机后进行。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在实例列表页面，您可以按需选择对单个或多个实例进行操作。
   - 关机单个实例：选择需要关机的实例，在右侧操作列单击“更多>实例状态>停止”，或点击实例名称进入实例详情页面，点击上方的 **停止** 按钮。



* 关机多个实例：勾选所有需要关机的实例，在列表顶部单击 **停止** 按钮，即可批量关机实例。

![image.png](http://bjhw-bce-online-public0.bjhw.baidu.com:8109/proxy-external/?url=http://bce-cdn.bj.bcebos.com/doc/bce-doc/BCC/image_9fbd0f0.png)

3. 在弹出的**“停止”**窗口中，您可以选择**关机不收费**，如不支持，实例列表会提示不支持关机不计费。

4. 停止后支持关机不收费的实例状态将变成**关机不计费**，不支持关机不收费的实例状态将变成**已关机**，正常计费。

抢占实例

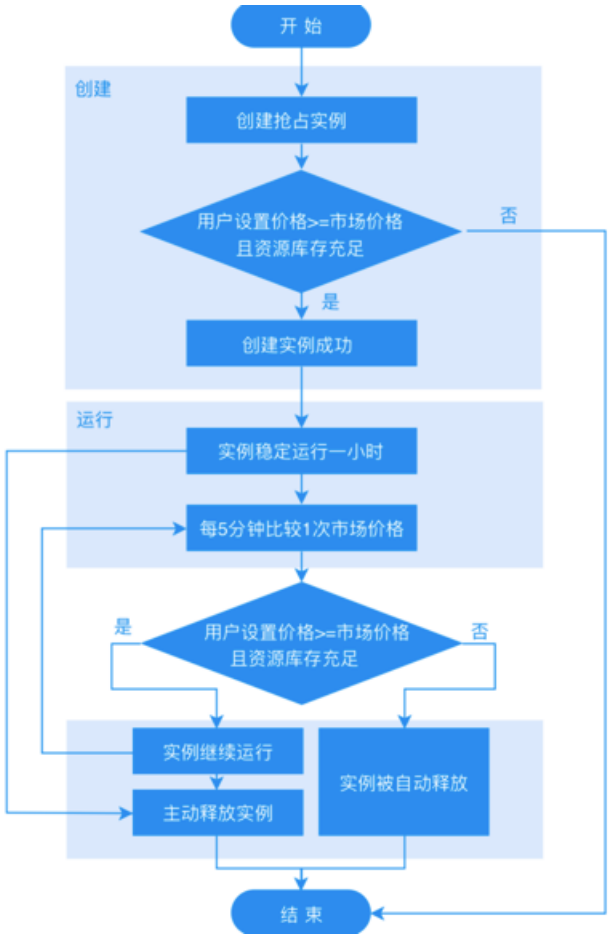
什么是抢占实例

抢占实例（也称竞价实例）是云服务器BCC实例的一种新运作模式。抢占实例性能与常规BCC实例无异，价格根据市场供需关系实时变化，相对于按量付费实例规格最高能节约90%的成本。抢占式实例在库存资源充足时可以被获取并使用，在库存资源不足时被中断回收。

抢占实例旨在为短期任务和高容错性应用提供经济高效的计算资源，适用于无状态、容错能力强、中断容忍度高的业务场景。

抢占实例运行流程

抢占实例运行流程如下图所示：



相关解释

项目	说明
抢占实例	一种按量付费实例，用户在指定范围内出价，当指定的实例规格的市场价格低于用户的出价，且资源库存充足时，客户可以成功创建抢占实例，并按照当前市场价计费。
市场价	根据用户选择的实例规格的库存和用户出价浮动，波动范围为选择配置的目录价格的1折～选择配置的目录价格，市场价格5分钟会更新一次。
折扣涉及部分	折扣仅涉及云服务器部分，其他部分如系统盘、数据盘、带宽、收费镜像等云服务器关联收费项均不受竞价折扣影响。
计费方式	抢占实例按分钟计费，成功创建实例时的市场价格为分钟价格。
自动出价	按此方式购买抢占实例，则默认实时匹配最新的市场价，实例仅会因为库存不足而被释放
最高价	按此方式购买抢占实例，当用户可接受的最高价格低于市场价或库存不足时，实例将会被释放。
一小时保护期	抢占实例一旦创建成功，就会拥有一个小时保护期，即在创建后第一个小时内，我们不会因为市场供需关系而释放您的实例，您可以在该抢占实例上正常运行业务。超过保护周期，即一小时后，我们每5分钟检测一次实例规格当前市场价格和库存，如果某一时刻的市场价格高于您的出价或资源库存不足，我们将释放您的抢占式实例。
抢占实例配额	每个用户在不同地域拥有最多创建20台抢占实例的配额，如需提升配额，请提交工单申请

> **注意：**
>
> 若市场价格高于您设置的最高价，释放时会提前5分钟短信通知用户，用户收到释放通知到释放的5分钟内，抢占实例按照上一轮的市场价格计费。

应用场景

抢占实例适用于无状态的应用场景，比如可弹性伸缩的Web站点服务、图像渲染、大数据分析和大规模并行计算等。您的应用程序的分布度、可扩展性和容错能力越高，使用抢占实例节省的成本越多。

您可以在抢占实例上部署以下常见业务：

- 实时分析业务
- 大数据业务
- 地理空间勘测分析业务
- 图像和媒体编码业务
- 科学计算业务
- 可弹性伸缩的业务站点、网络爬虫业务
- 图像和媒体编码业务
- 测试业务
- 有状态应用，比如数据库，不宜使用抢占式实例，因为竞价失败等因素导致实例被释放时，应用状态难以保存。

使用限制

- 抢占实例不支持转为预付费实例。
- 抢占实例不支持变更配置。
- 抢占实例不支持关机不计费。

创建抢占实例

前提条件

您可以在开通使用权限后，在控制台创建抢占实例。

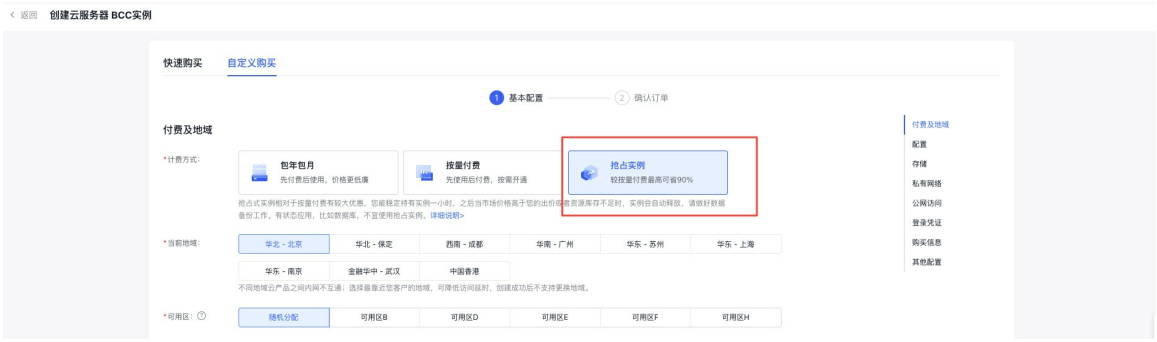
注意事项

在使用抢占实例时，您需要考虑以下内容：

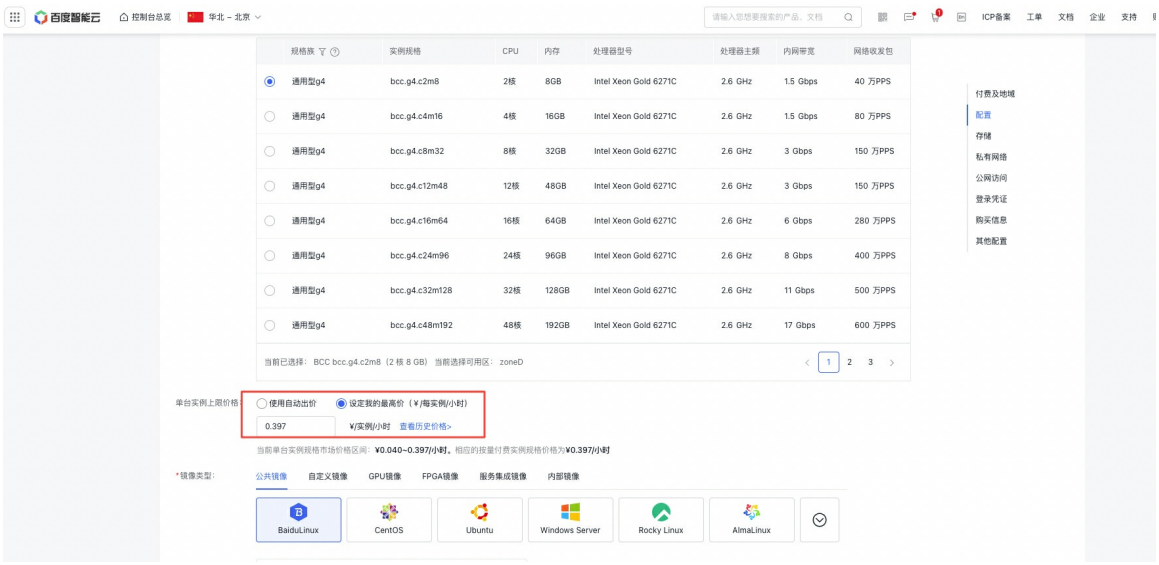
- 选择一个合理的出价：您的出价应该足够高，而且要充分考虑到市场价格的波动。这样，您的抢占请求才会被接受处理，而且创建后才不会因为价格因素被释放。另外，出价还必须符合您根据自身业务评估后的预期。
- 使用的镜像必须包含所需软件的配置，确保实例在创建完成后可以随时启动。
- 建议您使用不受抢占实例释放影响的存储介质来保存您的重要数据。例如，您可以使用独立创建的云盘（不能设置为随实例一起释放）、BOS等存储数据。
- 测试您的应用程序，确保它能很好地处理意外释放的实例。您可以使用按量付费的实例来运行该应用程序，然后自行释放这台实例，从而确认应用程序是否能处理自动释放的实例。

操作步骤

1. 登录百度智能云管理控制台，选择“产品服务>云服务器 BCC”，进入“实例列表”页面。
2. 选择地域，然后点击“创建实例”，进入“创建实例”界面。
3. 购买方式选择**抢占实例**。



4. 在**出价**模块设置单台实例价格。
   - 使用自动出价：选择跟随当前市场价格的模式，即表示始终接受实时的市场价格作为实例的计费价格 。
   - 设置最高价格：您必须为指定的实例设置一个价格上限，即您愿意为这个实例规格支付的最高价格 。



5. 选择您要使用的操作系统类型和网络等相关资源配置。

6. 确认配置信息后，点击**下一步**进入订单确认页，点击**去支付**开始创建抢占实例。

回收实例

回收站是一种云服务回收机制，本文档介绍实例的回收机制与恢复实例的操作方法。

哪些实例支持进入回收站

- 按量付费实例。
- 包年包月实例到期后7天续费期内进行释放。

> **注意：**
> **按量付费实例关机不计费时不支持进入回收站。**
> **按量付费实例定时释放时不会进入回收站。**
> **包年包月实例到期前财务退款不会直接进入回收站。**
> **竞价实例、弹性伸缩AS节点，和专属实例DCC，不支持进入回收站。**

包年包月实例进入回收站

- 保留时长：回收站内实例保留7个自然日。
- 过期处理：7个自然日后未进行续费，系统将释放资源，不可恢复。

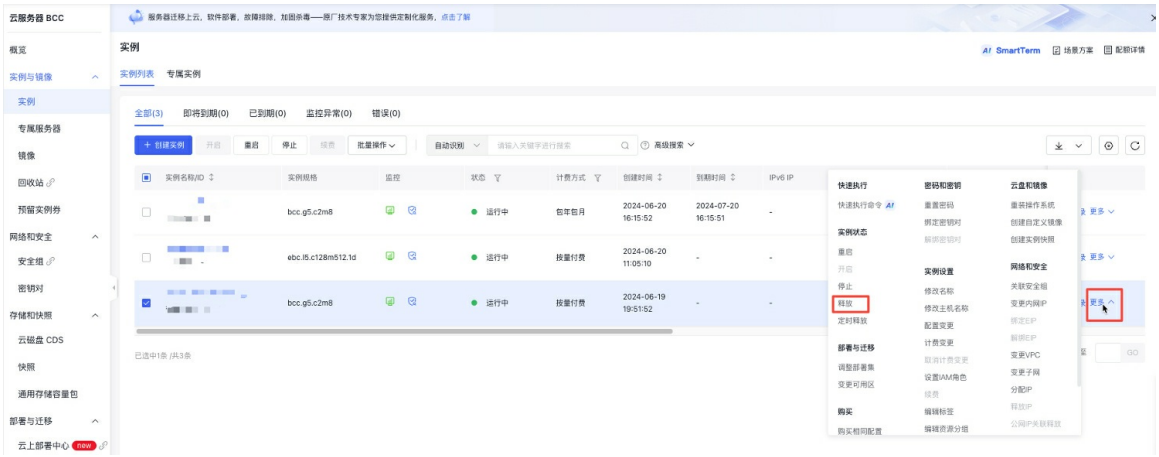
- 操作限制：回收站内实例，仅可进行 恢复计费 和 删除实例 操作。 **按量付费实例进入回收站**
- 保留时长：回收站内实例保留2个小时。
- 过期处理：2个小时后未进行续费，系统将释放资源，不可恢复。
- 关联实例：如绑定包年包月的EIP或者包年包月的CDS，云服务器实例进入回收站前会自动解绑，包年包月的EIP和CDS不会被回收。
- 操作限制：回收站内实例，仅可进行**“恢复计费”**和**“删除实例”**操作。

注意事项

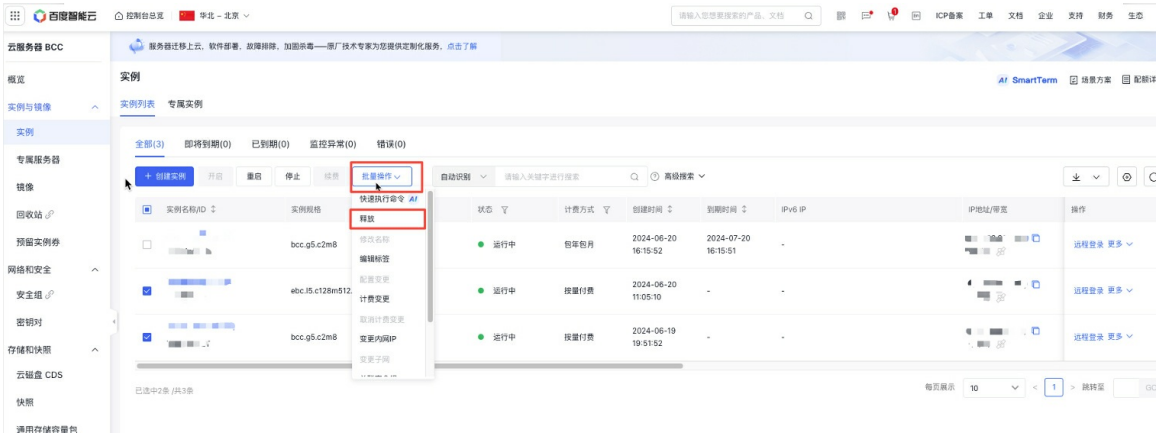
- 释放实例进入回收站时，需要您输入验证码验证释放操作。
- 释放时，您可选择释放绑定的按量付费的公网IP、云磁盘及快照。
- 实例释放后，已挂载的CDS磁盘自动卸载，基于此CDS磁盘的快照会保留。
- 实例释放后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

实例进入回收站操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。
2. 根据您的需求，选择需要释放进入回收站的实例。
   - 释放单个按量付费实例：选择您要释放的实例，在右侧操作列单击 “更多 - 实例状态 - 释放”，如下图所示。



   - 释放多个实例：勾选您要释放的实例，在顶部单击 “批量操作 - 释放”，如下图所示。



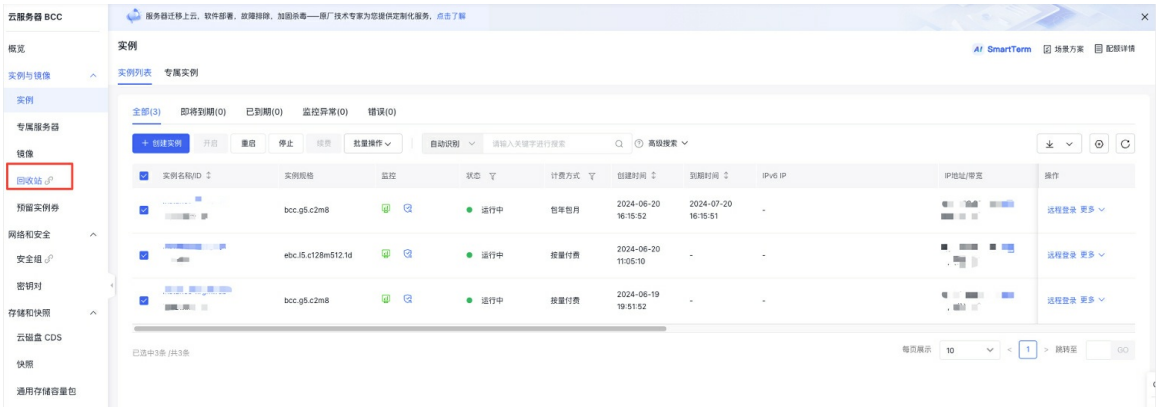
3. 在**释放前确认**框中确认要释放的实例，选择好关联释放的资源后，点击**下一步**。确认操作，**释放后实例进入回收站**会默认勾选好。勾选确认风险信息后，单击**确认**按钮，如下图所示。

4. 发送手机验证码，单击 **确定**，完成安全验证后实例将被立即释放。

回收站列表

1. 登录 云服务器BCC控制台 进入实例列表页面。

2. 选择左侧导航栏的 **回收站** 标签，点击进入。



3. 在跳转的 **回收站列表** 页面上，如下图所示，可以看到目前已经进入回收站的BCC实例。

> **注意： 查看回收站内某一类型的产品，需在页面左上方原产品那里先勾选**。

从回收站中恢复实例

1. 登录云服务器BCC控制台进入实例列表页面。选择左侧导航栏的 **回收站** 标签，点击进入回收站列表页。点击对应的实例后

面 **恢复计费** 按钮，如下图所示。

2. 在**恢复计费**的弹窗中，点击**“确认”**，恢复计费，如下图所示。

3. 完成上一步之后，恢复计费的BCC实例在回收站列表消失，同时会再次出现在BCC的实例列表中。

常见问题

云服务器实例进入回收站之后，绑定的IP和CDS是否会被释放

- 自定义镜像：创建的自定义镜像只要不删除，将会一直保留。
- 快照：基于原系统盘的快照会自动删除。挂载的CDS磁盘若未同时释放快照，则基于此CDS磁盘的快照会保留。

多个按量付费的实例释放时，报配额不足

回收站内支持最多同时容纳50个实例。若释放实例报配额不足时，可取消勾选加入回收站，直接释放。或者前往回收站清理可直接释放的实例后再操作。

跨AZ迁移

跨AZ迁移

注：仅针对华北-北京、华南-广州、华东-苏州地域待搬迁可用区中有资源的用户开放，以便进行资源跨AZ迁移

简介

支持实例跨AZ子网迁移，实例的私网IP不变。

- 实例在迁移过程中会关机停服，迁移前请务必进行重要数据的备份。
- 子网迁移前需先更改所在子网目标可用区，一个迁移任务支持选择子网下部分实例或全部实例同时迁移，已挂载的磁盘会随实例一同迁移。
- 迁移前请确保BCC实例均处于“运行中”或者“已关机”状态，否则无法执行迁移，关机不计费状态不支持迁移。
- 若迁移目标可用区无实例代系机型，会自动选择目标可用区存在的新代系机型；若没有相同配置，会自动选择最临近的配置

进行变配。

- 若待迁移BCC实例通过BLB对外提供服务访问，且BCC实例上配置了安全组规则或所属子网配置了ACL规则，而BLB上没有配置对应的安全组或者ACL规则，当BCC实例迁移后，通过BLB进入BCC的流量将不会遵循ACL和安全组规则进行控制，需要您手动在BLB上重新配置对应规则。

不支持场景说明

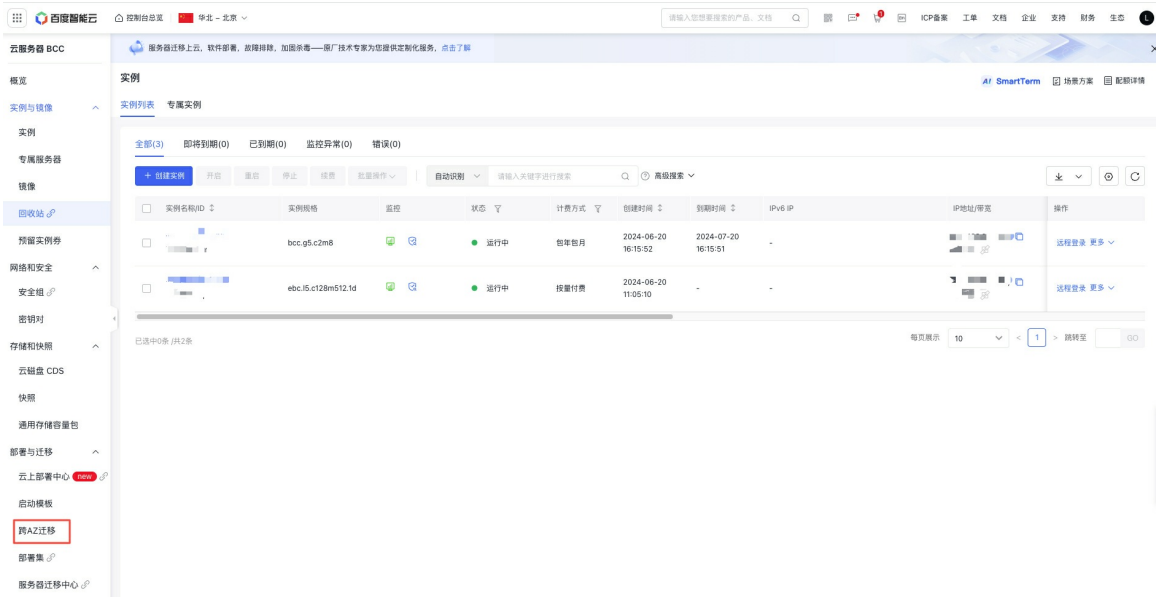
不支持跨AZ子网迁移的场景概述及处理建议如下：

场景概述	处理建议
本地盘实例迁移	BCC的系统盘/数据盘是本地盘，不支持跨AZ迁移。建议您对本地盘实例创建自定义镜像，在目标AZ用该镜像创建新的BCC实例。
ROCV1实例迁移	BCC的系统盘是ROCV1（实例规格为ic1/c1/g1/m1），不支持跨AZ迁移。建议您可先为BCC创建自定义镜像，在目标AZ用该镜像创建新的BCC实例。
已下架CDS盘（系统盘）迁移	BCC的CDS系统盘是上一代云磁盘，请执行CDS变配后，再进行子网迁移。
已下架CDS盘（数据盘）迁移	BCC的CDS数据盘是上一代云磁盘，请执行CDS变配后，再进行子网迁移。
已下架规格族（带本地数据盘）迁移	BCC有本地数据盘，不支持跨AZ迁移。请自行迁移后再进行子网的迁移。

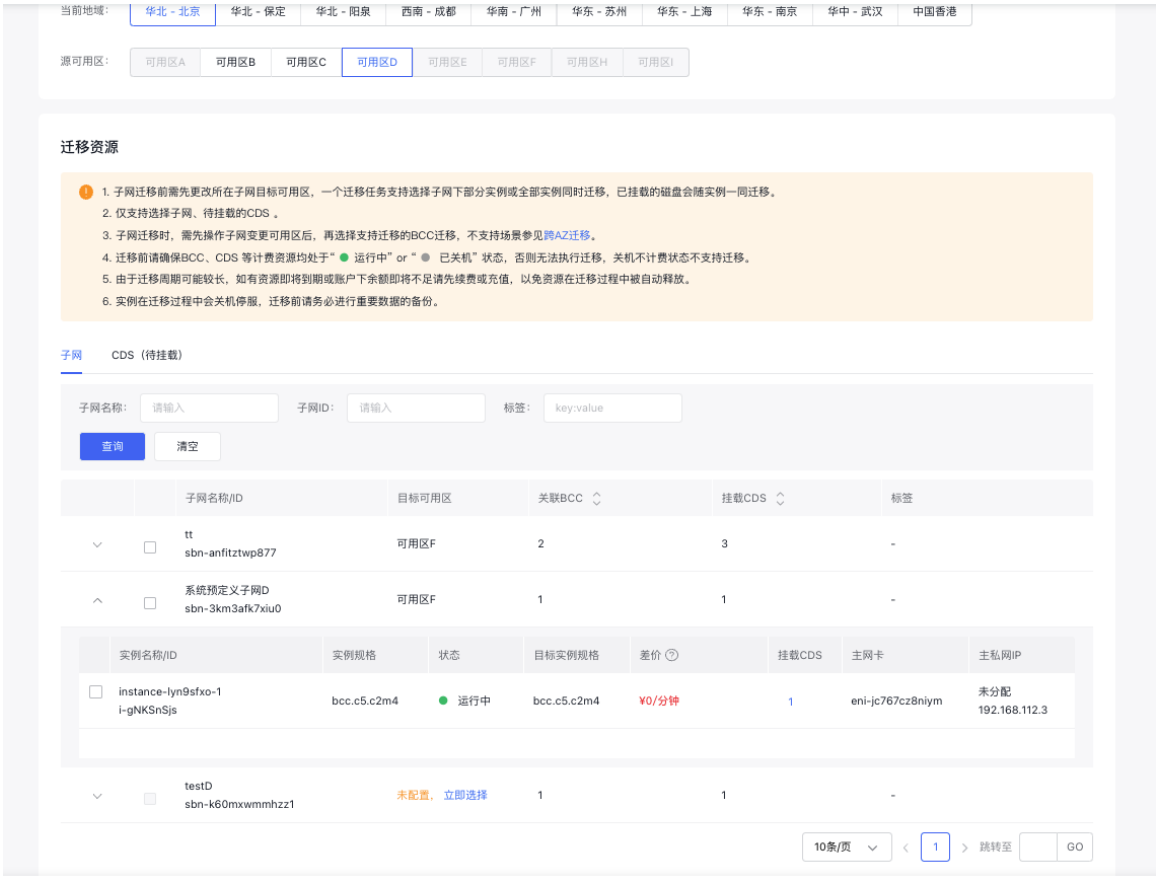
创建任务

注意事项：

1. 登录 百度智能云官网，点击右上角的“管理控制台”，快速进入控制台界面。
2. 选择“云服务器BCC -> 部署与迁移 -> 跨AZ迁移”，进入“任务列表”页面。



3. 点击“创建任务”，进入创建任务页面，选择要进行迁移的子网。若子网未变更目标可用区，需点击“立即选择”，变更可用区后再选择待迁移实例。



4. 选择起始时间，若不选择则代表创建任务后立即进行迁移。可选的最早开始时间=当前时间+1小时。

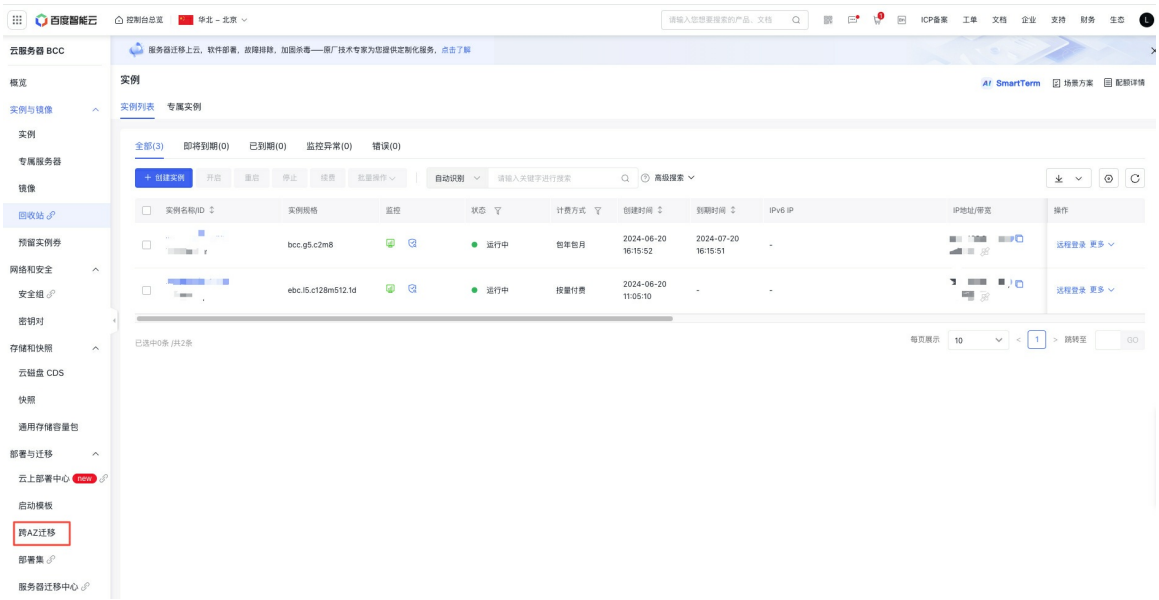
5. 完成配置后点击“确认”，进行待迁移资源的确认。

6. 返回“任务列表”页面。

查看任务

查看迁移任务信息

1. 选择“跨AZ迁移 -> 任务列表”，进入“任务列表”页面。



2. 点击迁移BCC/DCC/CDS数量，可进入“迁移资源”页面。

重试任务

1. 迁移失败的任务可以点击操作列表中的重试进行再次迁移。

撤销任务

1. 非立即开始的迁移任务可在操作列表中撤销任务。

常见问题

1. 一般迁移需要多长时间？

   答：影响迁移时长的因素有以下三个：

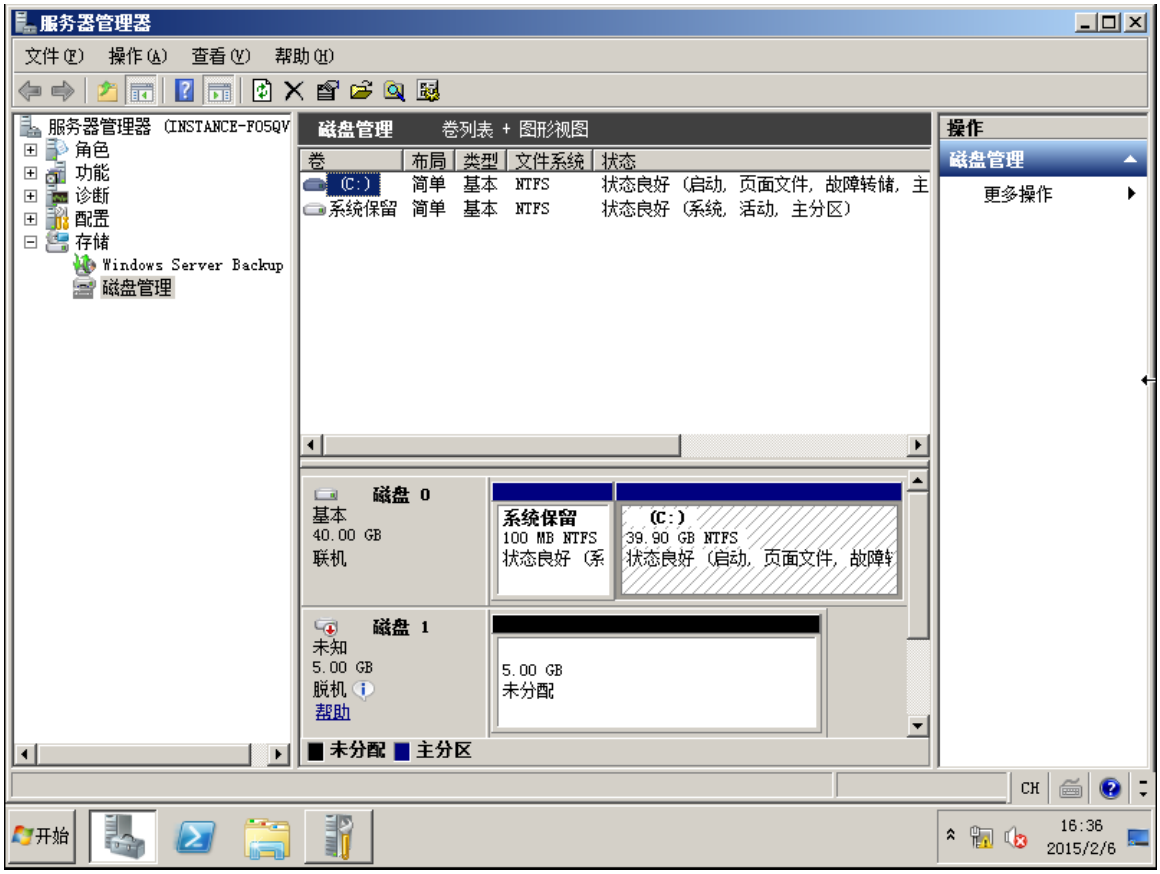
   1）当前机房中同时迁移的任务数量，当前我们支持200台BCC实例同时迁移，若此时迁移任务较多则需要排队

   2）与当前要迁移实例的所有CDS云盘已写入的容量有关，100g/3min

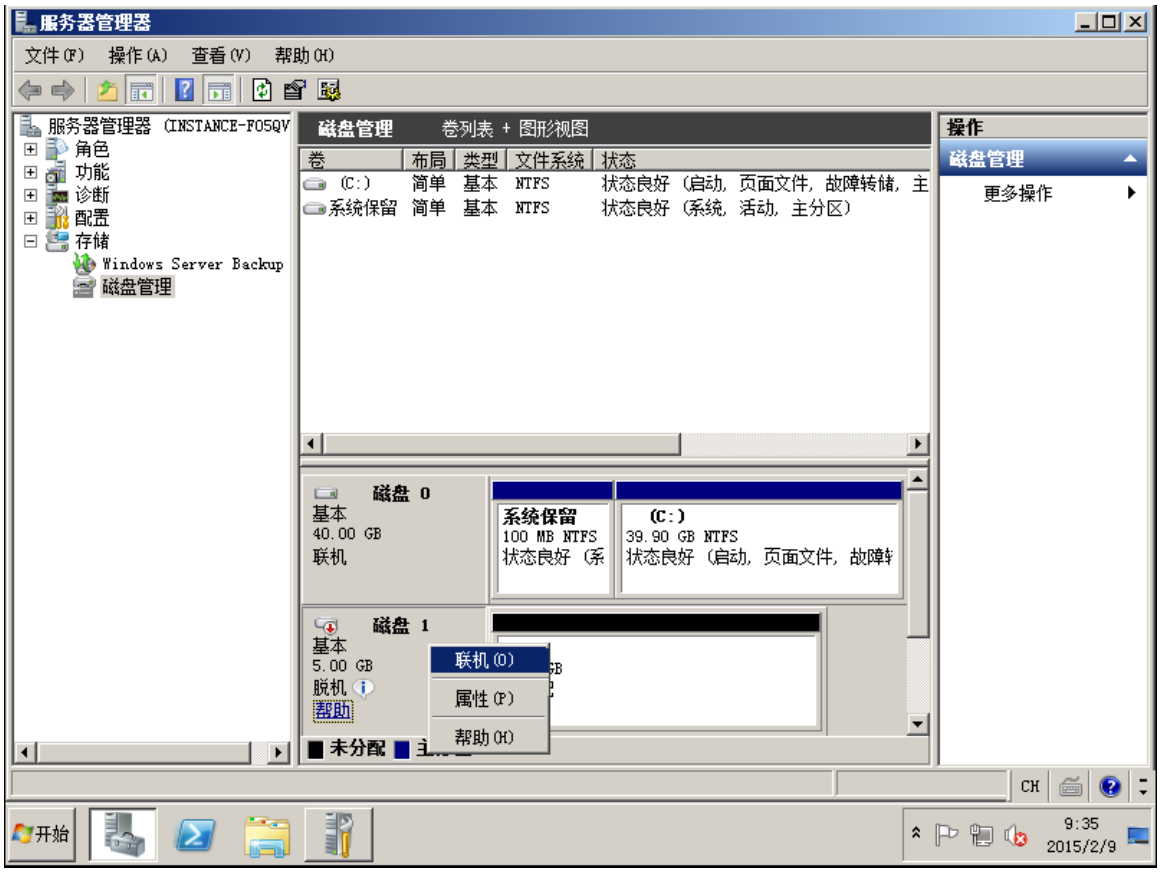
2. 为什么windows系统的主机使用工具迁移完成后，进入系统里面看不到数据盘了？

   答：部分存量用户的windows系统主机由于版本等问题使用跨AZ迁移工具迁移后，数据盘会自动脱机，需要手动联机后方能正常使用。磁盘联机的操作方法请参考如下步骤：

   1）登录windows主机后，点击左下角的“服务器管理器”，弹出“服务器管理器”窗口。



2）在页面下方找到需要联机的数据磁盘，例如“磁盘1”，单击右键，选择“联机”。



> 说明：如您有较多的windows主机需要跨AZ迁移，为了避免问题提高迁移效率，您也可以提交工单咨询相关问题的规避方法。

变更可用区

操作场景

当用户准备升级机型时，若当前AZ没有所需机型，可通过实例粒度的跨AZ迁移功能将实例迁移至其他AZ后再进行机型升级。

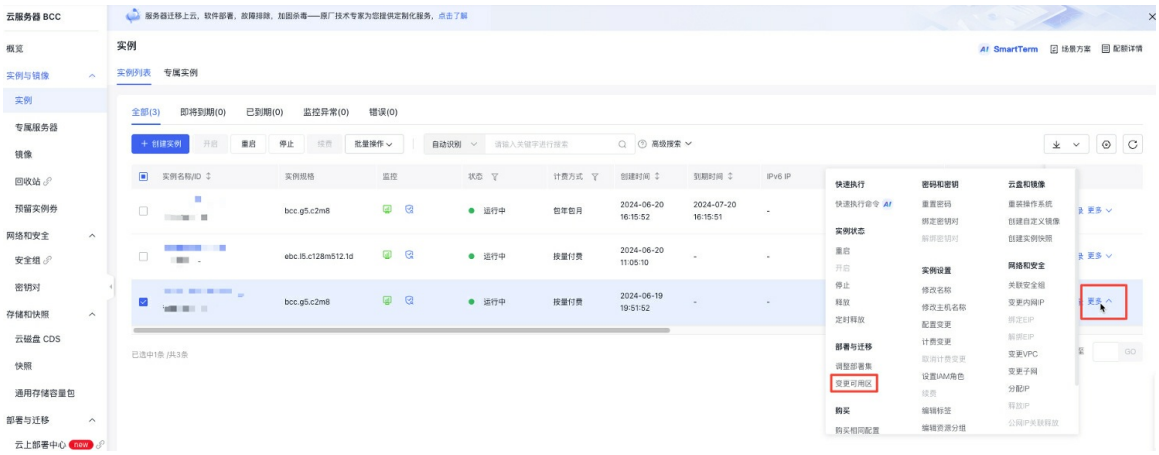
前提条件

- 本地盘实例不支持变更可用区
- 若实例有弹性网卡，请先将弹性网卡释放
- 若实例主网卡有辅助IP，请先将辅助IP释放
- 若实例有IPv6 IP，请先将IPv6 IP释放
- 变更可用区会重启云服务器，导致云服务器停止几分钟，为保证您的服务不受影响，请在确认执行此操作前做好预案准备

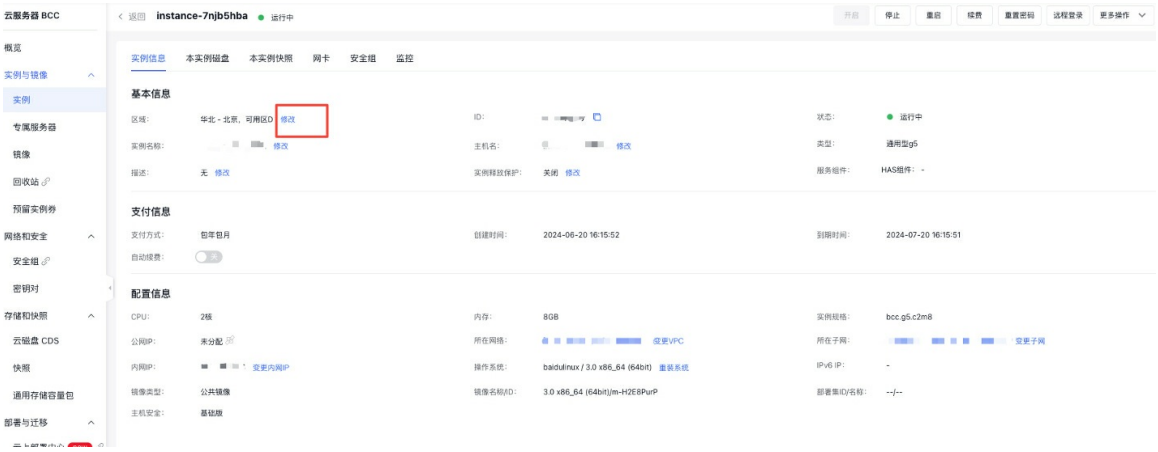
操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。
2. 对实例进行变更可用区操作，您可以根据需求选择不同的操作方式：

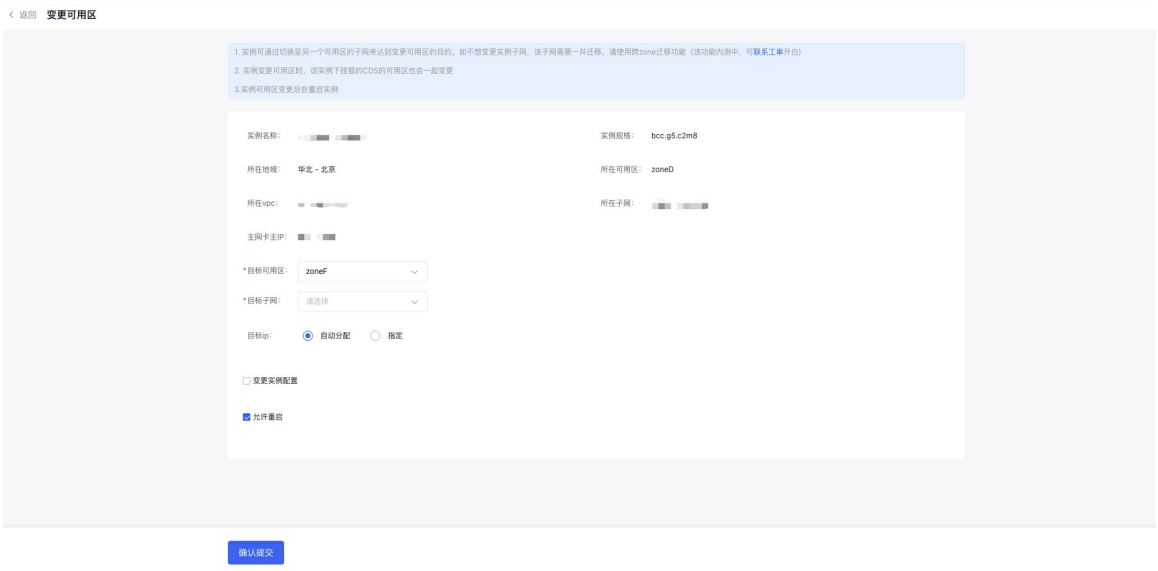
- 在右侧的操作栏，单击“更多>部署与迁移>变更可用区”。



- 点击实例名称进入实例信息页面，点击区域后的 **修改** 按钮。



3. 在操作页面选择相应的配置，点击**确认提交**，实例进入迁移状态。



4. 迁移完成后，实例会重新变为**运行中**或**已关机**状态。

修改实例名称

操作场景

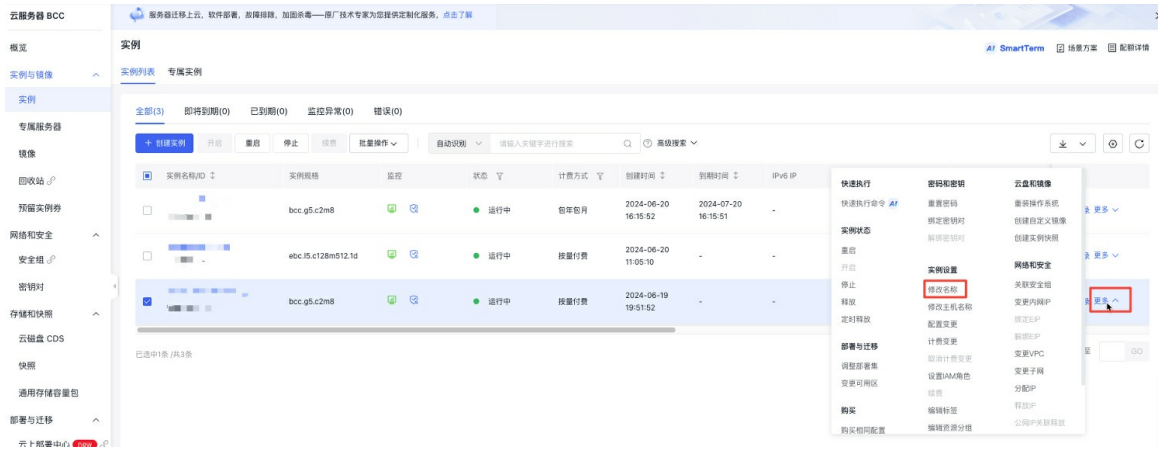
云服务器支持批量变更实例名称，名称以前缀+序号进行命名。例如在“修改实例名称”对话框，输入BaiduBCC，实例名称修改后为BaiduBCC-1、BaiduBCC-2、BaiduBCC-3等。

为了方便您在控制台上对实例的管理，云服务BCC支持为每台实例变更名称，您可单个变更名称或批量变更名称。

操作步骤

修改单个实例名称

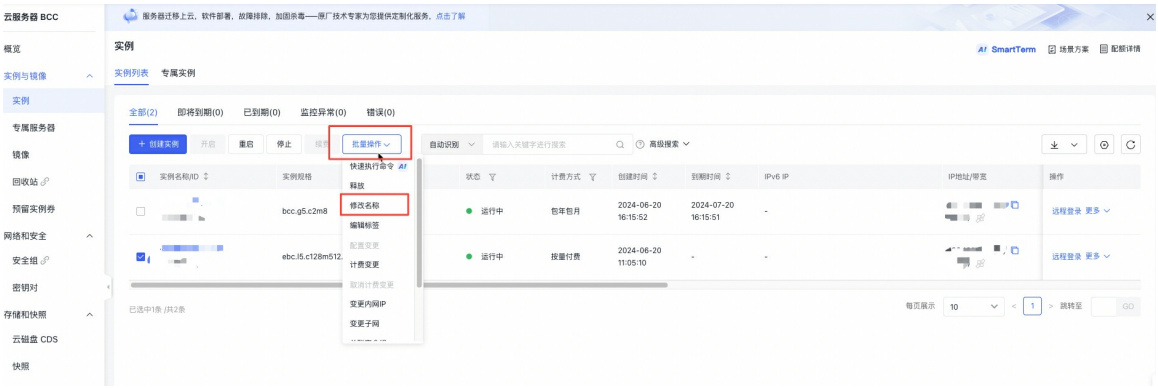
1. 登录云服务器BCC控制台进入实例列表页面。
2. 选择要变更名称的实例，点击实例名称右侧的 按钮，或点击操作列下“更多 - 实例设置 - 修改名称”，如下图所示



3. 在修改实例名称对话框中，输入您要修改的实例名称，点击确定，即可完成实例名称修改。

批量修改实例名称

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 勾选多个需要变更名称的实例，点击“批量操作 - 修改名称”。



3. 在修改实例名称对话框中，输入您要修改的实例名称前缀，点击确定，即可完成实例名称修改。

> **注意：**
>
> 批量修改名称中，名称以前缀+序号进行命名，例如：输入BaiduBCC，实例名称修改后为BaiduBCC-1、BaiduBCC-2、BaiduBCC-3等。

通过API修改实例名称

您可以使用 修改实例属性接口 为指定实例修改名称。

变更实例配置

您可以根据需求灵活的调整BCC实例的配置，包括CPU、内存、系统盘、弹性带宽等。

支持变配的的实例规格

> **注意：**仅通用型g4、计算型c4、密集计算型ic4、内存型m4支持在线变配（热升级），其他在线变配的限制请见在线变配（热升级）的限制。
>
> 其他规格族实例变配则需重启生效。

源实例规格族	可变配的目标实例规格族
通用型g7 计算型c7 内存型m7	• 通用型g7 • 计算型c7 • 内存型m7
通用型g6 计算型c6 内存型m6	• 通用型g6 • 计算型c6 • 内存型m6
经济型e1	• 通用型g7、通用型g6、通用型g5、通用型ga3、通用型ga2、通用型ga1 • 计算型c7、计算型c6、计算型c5、密集计算型ic5、计算型ca3、计算型ca2、计算型ca1 • 内存型m7、内存型m6、内存型m5、内存型ma3、内存型ma2、内存型ma1
通用型g5 计算型c5 密集计算型ic5 内存型m5	• 通用型g7、通用型g6、通用型g5、通用型ga3、通用型ga2 • 计算型c7、计算型c6、计算型c5、密集计算型ic5、计算型ca3、计算型ca2 • 内存型m7、内存型m6、内存型m5、内存型ma3、内存型ma2
通用型g4 计算型c4 密集计算型ic4 内存型m4	• 通用型g7、通用型g6、通用型g5、通用型g4、通用型ga3、通用型ga2 • 计算型c7、计算型c6、计算型c5、密集计算型ic5、计算型c4、密集计算型ic4、计算型ca3、计算型ca2 • 内存型m7、内存型m6、内存型m5、内存型m4、内存型ma3、内存型ma2
通用型ga3 计算型ca3 内存型ma3	• 通用型ga3 • 计算型ca3 • 内存型ma3
通用型ga2 计算型ca2 内存型ma2	• 通用型g5、通用型g4、通用型ga3、通用型ga2 • 计算型c5、密集计算型ic5、计算型c4、密集计算型ic4、计算型ca3、计算型ca2 • 内存型m5、内存型m4、内存型ma3、内存型ma2
通用型ga1 计算型ca1 内存型ma1	• 通用型g5、通用型g4、通用型ga3、通用型ga2、通用型ga1 • 计算型c5、密集计算型ic5、计算型c4、密集计算型ic4、计算型ca3、计算型ca2、计算型ca1 • 内存型m5、内存型m4、内存型ma3、内存型ma2、内存型ma1
通用型g3 计算型c3 密集计算型ic3 内存型m3	• 通用型g5、通用型g4、通用型ga2 • 计算型c5、密集计算型ic5、计算型c4、密集计算型ic4、计算型ca2 • 内存型m5、内存型m4、内存型ma2

单实例变更CPU和内存

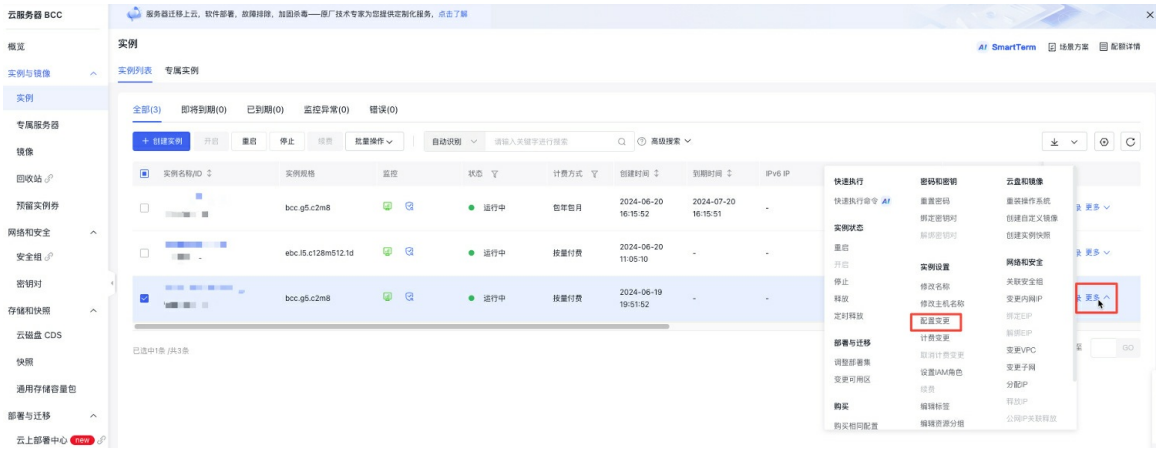
背景信息

- 对包年包月形式的BCC实例，支持CPU、内存、网络带宽等资源从低到高进行升级。
- 对于按量付费的具备公网IP的BCC实例，可动态调整CPU、内存、网络带宽等各项条目的增加或减少。
- CPU、内存配置改变后，默认情况下系统会在支付成功后自动重启以让新配置生效；仅改变带宽配置，系统不会重启。
- 如果选择在线变配，则会在不重启的情况下进行配置的变更。

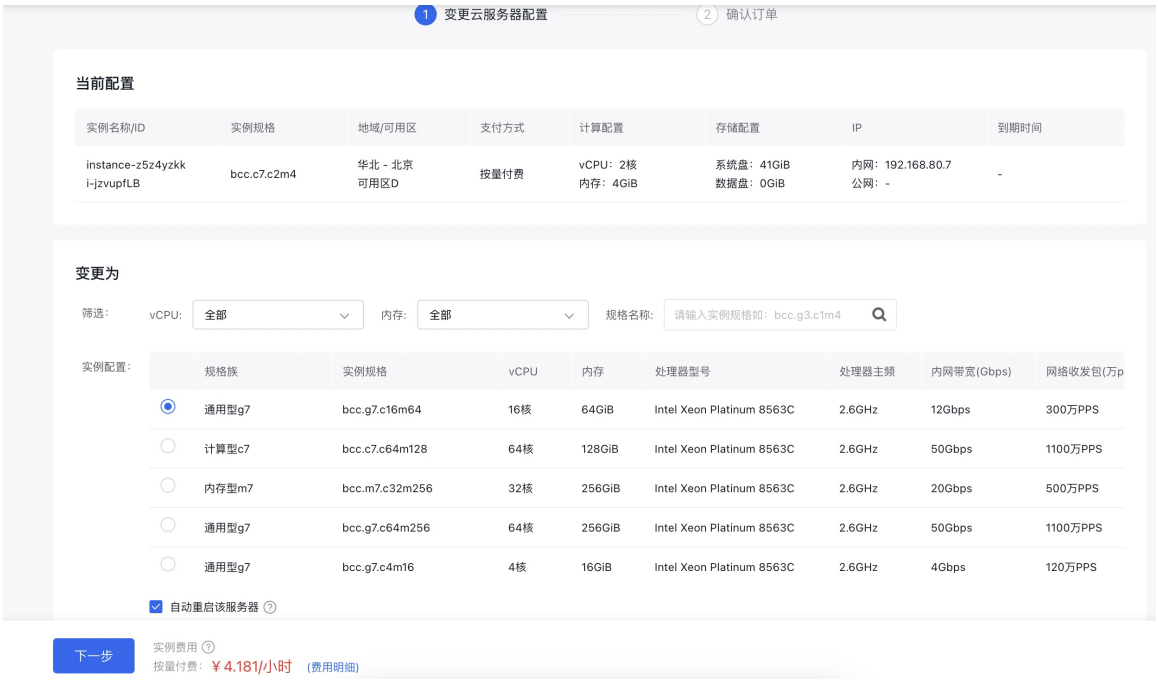
操作步骤

1.登录云服务器BCC控制台进入实例列表页面。

2. 选择您要变更配置的实例，在右侧的操作栏单击 “更多>实例设置>配置变更”，如下图所示。



3. 在配置变更页，选择相应的配置。若所选配置不支持在线变配，则需要勾选“自动重启服务器”。



> **注意：**不是所有配置都支持在线变配，具体支持情况请见在线变配（热升级）的限制。

4. 点击 “下一步”，并完成相应支付。

> **注意：**对于包年包月的BCC实例，在使用期内只支持升级内存和CPU并一次性扣费，不支持对CPU和内存进行降级，请根据

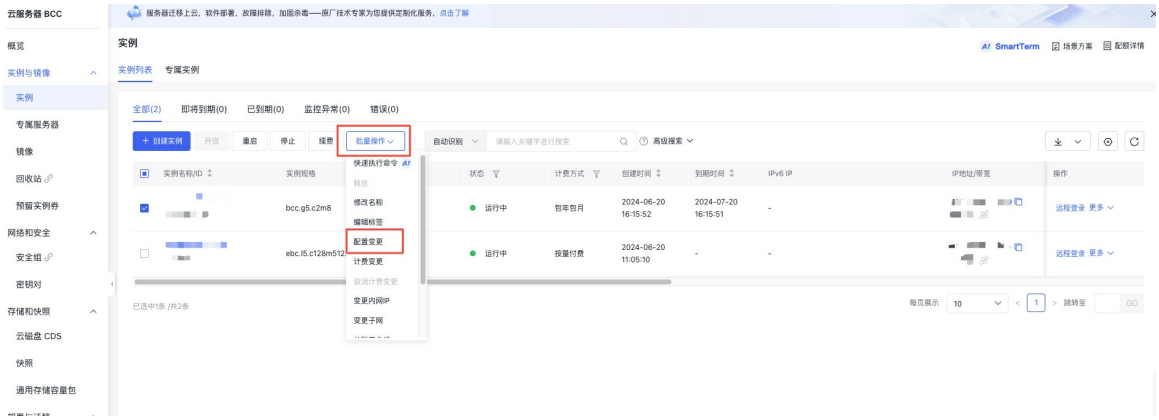
> 实际需求谨慎调整配置。

5. 变更配置成功后，实例状态会变为**资源扩展中**。

批量变更CPU和内存

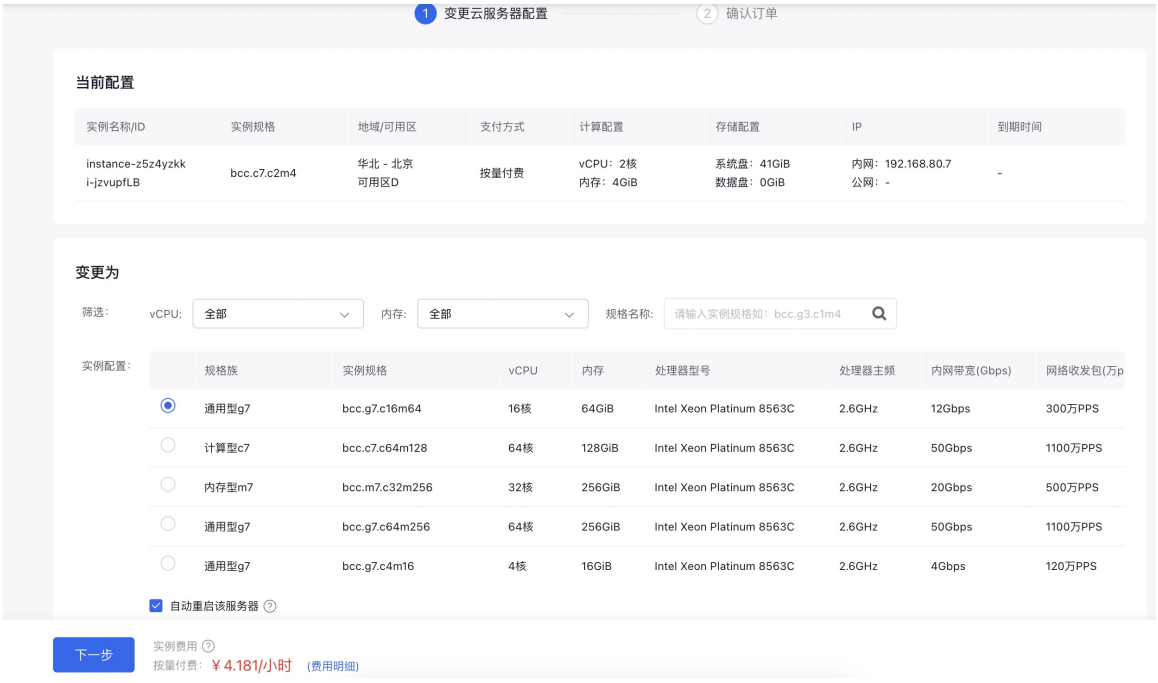
操作步骤

1. 登录云服务器BCC控制台进入实例列表页面。
2. 勾选您要变更配置的实例，在顶部的操作栏单击 “批量操作>配置变更”，如下图所示。



> **注意：**批量变更配置单次最多可支持操作50个实例。

3. 在配置变更页，选择相应的配置。若所选配置不支持在线变配，则需要勾选“自动重启服务器”。



> **注意：**不是所有配置都支持在线变配，具体支持情况请见在线变配（热升级）的限制。

4. 点击 “下一步”，并完成相应支付。

> **注意：**对于包年包月的BCC实例，在使用期内只支持升级内存和CPU并一次性扣费，不支持对CPU和内存进行降级，请根据实际需求谨慎调整配置。

5. 变更配置成功后，实例状态会变为**资源扩展中**。

系统盘扩容

详见存储文档下的系统盘扩容。

弹性调整带宽

前提条件

若云服务器BCC需要公网能力调整带宽的实例必须绑定弹性公网IP。

背景信息

- 对于包年包月的弹性公网IP，仅支持升级带宽。
- 对于按量付费的弹性公网IP，可随时动态调整带宽（带宽增加或减少）。

操作步骤

1. 登录云服务器EIP控制台进入EIP实例列表页面。
2. 选择要调整带宽的EIP实例，并点击操作栏的“调整带宽”。

3. 设置完相应的带宽，点击“确认订单”，进入“确认订单”页面，并完成变更。

> 注意：对于包年包月的BCC实例，在使用期内只支持升级带宽并一次性扣费，请根据实际需求谨慎选择。

变更计费方式

按量付费转包年包月

操作场景

创建一台按量付费实例后，你可以将实例转换成长期并稳定使用的包年包月实例，享受更大的价格优惠。

前提条件

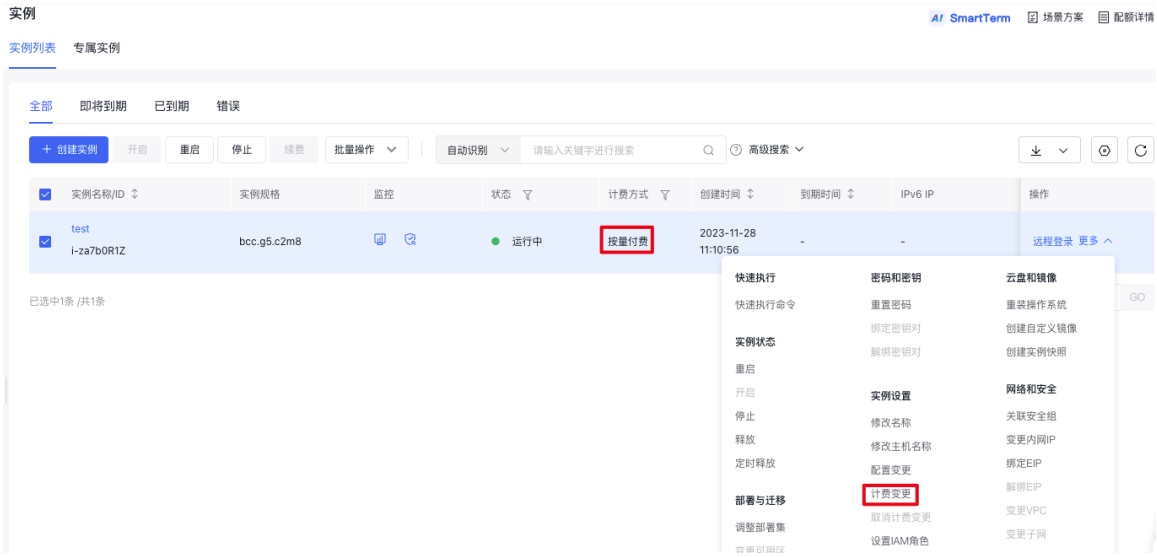
- 实例必须处于**运行中**、**已关机**状态。
- 不支持抢占实例

> 按量付费实例转换为包年包月后，实例系统盘以及实例上挂载的按量付费数据盘将转为包年包月。EIP和公网带宽的计费

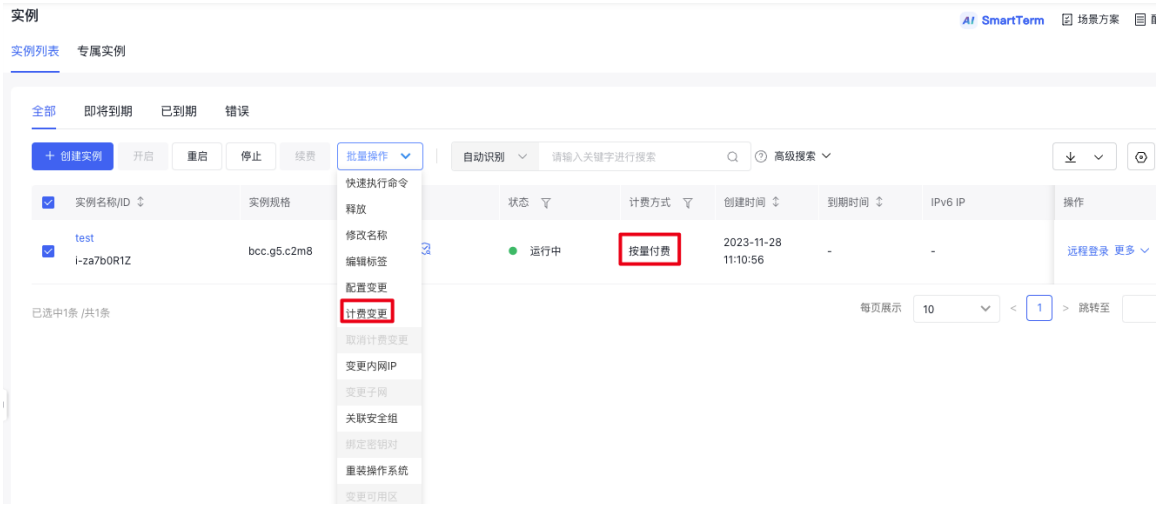
> 方式不变。

操作步骤

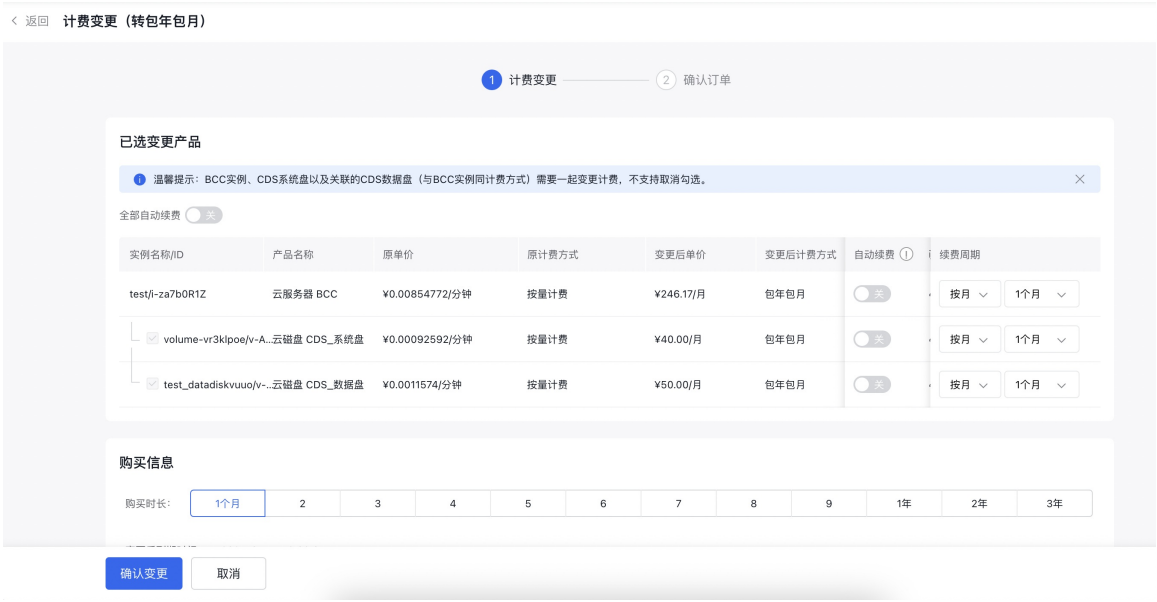
1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 根据您的需求，选择不同的计费变更操作。
   - 单个实例计费变更：选择要变更的按量付费实例，在右侧的操作栏，单击 “更多 - 计费变更”，如下图所示。



   - 多个实例计费变更：勾选要变更的按量付费实例，点击顶部的 “批量操作 - 计费变更”，如下图所示。



3. 在新页面中，选择预付费购买时长，点击下一步，如下图所示。



4. 完成支付，支付成功后，立即生效为包年包月方式。

通过API按量付费转包年包月

您可通过批量转包年包月接口变更实例计费方式为包年包月实例。

包年包月转按量付费

概述

创建一台包年包月实例后，您可以根据业务需求选择将包年包月的计费方式转为按量付费，更灵活地按需使用云服务器实例。转换为按量付费实例后，请确保账户余额充足以免发生欠费，影响实例的正常运行。

> 包年包月转按量付费，支持两种方式：
>
> - 立即转按量付费：按量计费模式立即生效。
> - 到期转按量付费：包年包月计费模式当前计费周期到期后，按量付费的计费模式才会生效。

前提条件

- 实例需处于运行中或已停止状态。
- 已过期的实例不支持计费变更。
- 已开通自动续费的实例无法变更为后付费。
- 包年包月转按量付费需要账号余额及代金券/现金券之和大于100元方可转换成功，若不足则会出现订单失败作废的情况。
- 若实例安装的操作系统类型为云市场镜像，不支持转为按量付费。

背景说明 包年包月实例转换为按量付费后，BCC实例、实例系统盘以及实例上挂载的包年包月数据盘将转为按量付费。EIP和公网带宽的计费方式不变。

立即转按量付费时退款规则如下：

- 退款会消耗退款额度，当月退款额度超限后不能再操作退款，次月1号自动清零。额度以积分的形式呈现，每月积分根据您上个月的消费情况计算而来，具体积分以页面展示为准。
- 若您退款时发现因额度超限无法退款，可以提交工单或联系您的销售经理进行申请。
- 仅退还实付金额到原付款渠道，已使用的代金券和优惠券不予退回。包年包月转按量付费的退款计算公式，请参见退款说

明。

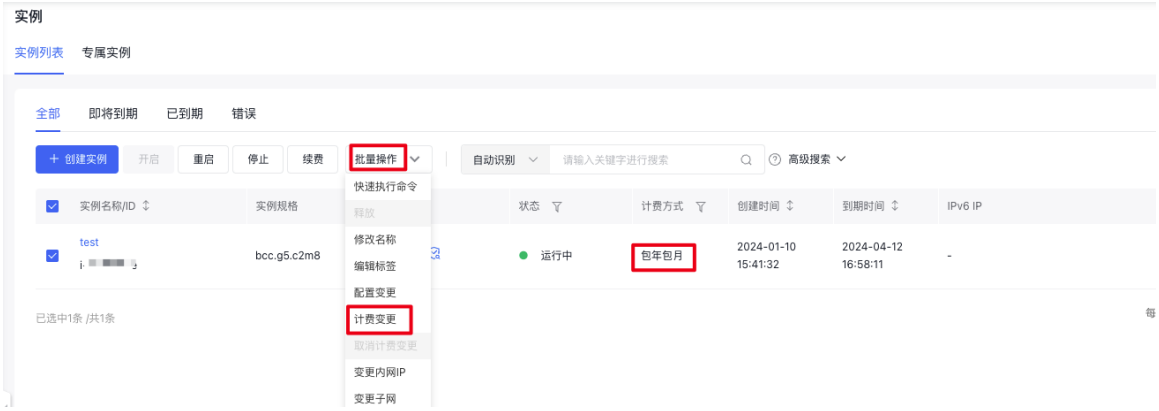
- 若包含未生效的续费订单，会全额退款。但已生效的订单仅退还部分款项。

操作步骤

立即转按量付费

1. 登录云服务器BCC控制台进入实例列表页面。
2. 根据您的需求，选择不同的操作方式。
   - 单个实例变更：选择要变更的包年包月实例，在右侧的操作栏，单击 “更多 - 计费变更”，如下图所示。

   - 批量实例变更：选择要变更的包年包月实例，在顶部的操作栏，单击 “批量操作 - 计费变更”，如下图所示。

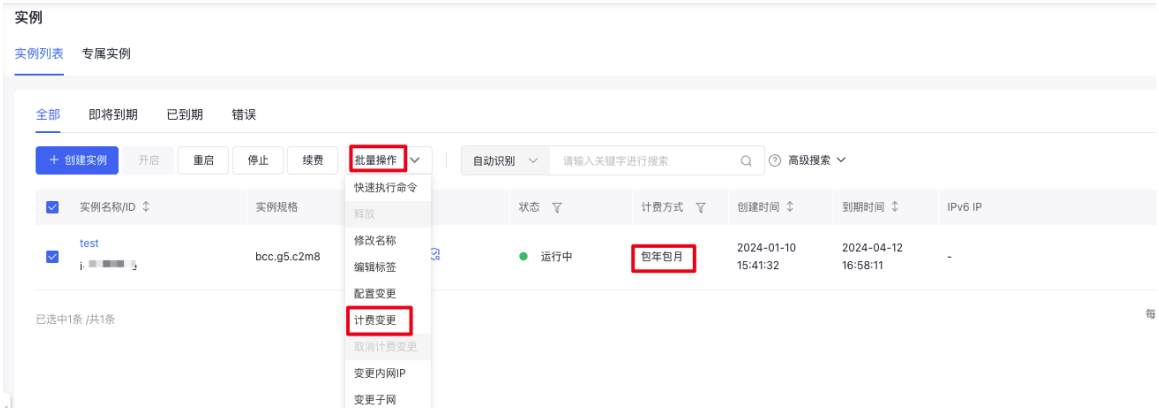


3. 在弹出的 **计费变更（转按量付费）** 对话框中，选择“立即转为按量付费”，核对退款金额后，勾选“我已阅读并同意《云服务器BCC包年包月转按量付费规则》”后单击**确认变更**按钮，完成实例的计费变更操作。

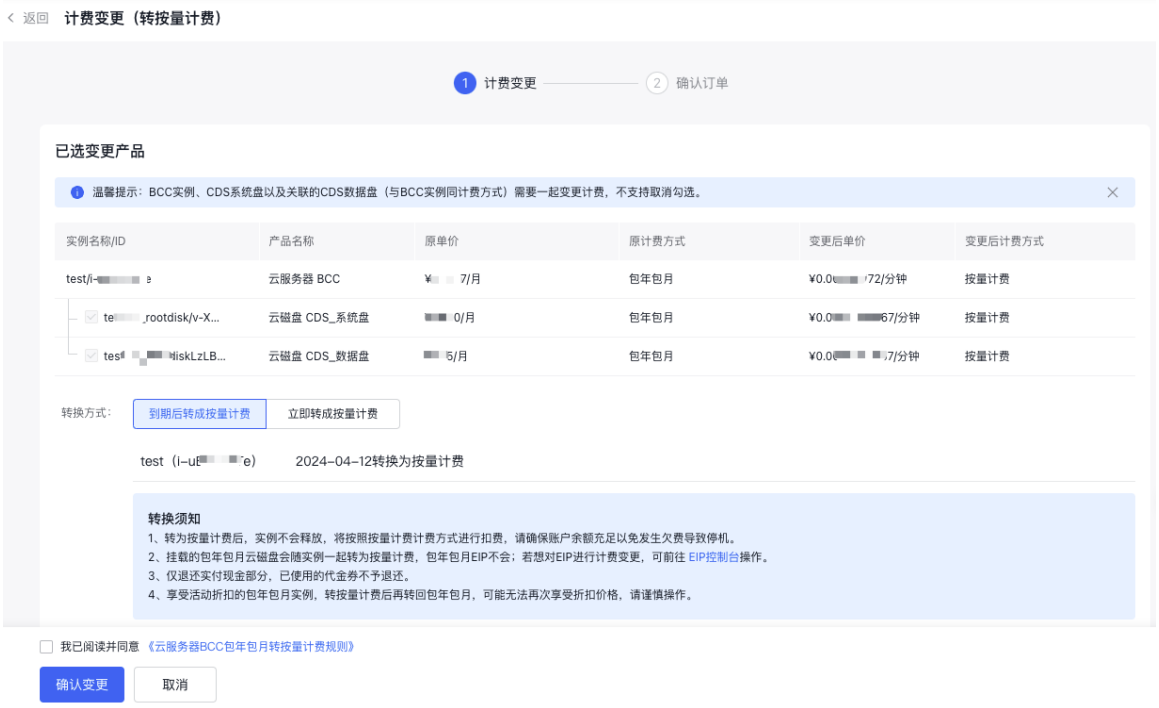
到期转按量付费

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 根据您的需求，选择不同的操作方式。
   - 单个实例变更：选择要变更的包年包月实例，在右侧的操作栏，单击 “更多 - 计费变更”，如下图所示。

- 批量实例变更：选择要变更的包年包月实例，在顶部的操作栏，单击 “批量操作 - 计费变更”，如下图所示。



3. 在弹出的**计费变更（转按量付费）**对话框中，勾选“我已阅读并同意《云服务器BCC包年包月转按量付费规则》”后单击**确认变更**按钮，完成实例的计费变更操作。



通过API包年包月转按量付费

您可以使用批量转按量付费接口为单个或多个包年包月实例转为按量付费计费方式。

设置IAM角色

IAM角色是一种虚拟身份，同用户身份一样，可以关联权限对云上资源进行操作。您可以将IAM角色关联到云服务器BCC实例，在实例内部基于STS临时身份凭证调用其他云产品的 API（临时身份凭证将周期性更新）。由于临时身份凭证仅可在实例内部获取，并且无需配置AKSK，这不仅确保了云账号AKSK的安全性，还能够通过IAM实现精细化的控制与权限管理。

本文介绍如何进行实例IAM角色管理，例如绑定、修改及删除实例角色。

使用限制

一台BCC实例只能被授予一个IAM角色。

临时身份凭证每5分钟进行更新，若角色解绑，临时身份凭证可能在最长5分钟后失效。

操作步骤

创建IAM角色 1、登录控制台，鼠标移入右上角头像，点击“多用户访问控制”。

2、进入“多用户访问控制”控制台，选择“角色管理”，点击“创建角色”。输入“角色名称”，载体选择云产品，载体实体选择“云服务器BCC”，策略管理选择需要对该角色赋予的权限。点击“确定”完成角色的创建。

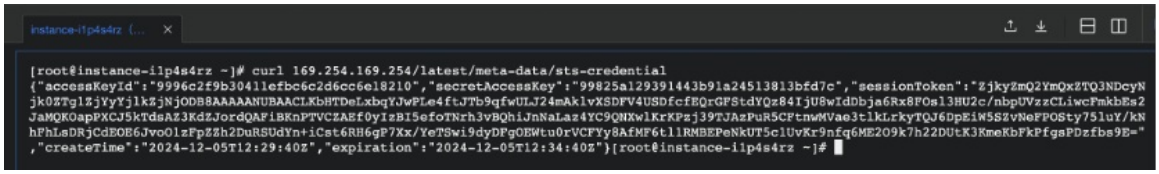
实例绑定IAM角色

1、进入云服务器BCC控制台，在实例列表选择您要绑定或者修改角色的实例，在右侧操作列单击 “更多 - 实例设置 – 设置IAM角色”，如下图所示。

2、在弹出的 **设置IAM角色** 对话框中点击选择IAM角色，单击确认按钮，实例绑定IAM角色成功。

获取实例IAM角色的临时身份凭证 您可以在BCC实例内部通过访问元数据（meta-dada） 获取到临时访问凭证。命令如下：

```
curl 169.254.169.254/latest/meta-data/sts-credential
```



使用实例IAM角色调用API 下文以部署在Linux BCC实例上的Go应用程序调用对象存储BOS下载文件接口为例，为您介绍BCC实例IAM角色的使用方法。

```
 import (
   "fmt"
   "github.com/baidubce/bce-sdk-go/auth"         //导入认证模块
   "github.com/baidubce/bce-sdk-go/services/bos" //导入BOS服务模块
   "github.com/baidubce/bce-sdk-go/services/sts" //导入STS服务模块
)

func main() {

   // 获取实例里的临时身份凭证
   accessKeyId = getAccessKeyId()
   secretAccessKey = getSecretAccessKey()
   sessionToken = getSessionToken()

   bosClient, err := bos.NewClient(accessKeyId, secretAccessKey, sessionToken)
   if err != nil {
      fmt.Println("create bos client failed:", err)
      return
   }
   stsCredential, err := auth.NewSessionBceCredentials(
      accessKeyId,
      secretAccessKey,
      sessionTokenn)
   if err != nil {
      fmt.Println("create sts credential object failed:", err)
      return
   }
   bosClient.Config.Credentials = stsCredential

   // 提供Bucket和Object，直接获取一个对象
   res, err := bosClient.BasicGetObject(bucketName, objectName)

   // 获取ObjectMeta
   meta := res.ObjectMeta

   // 获取Object的读取流（io.ReadCloser）
   stream := res.Body

   // 确保关闭Object读取流
   defer stream.Close()

   // 调用stream对象的Read方法处理Object
   ...
}
```

其他操作

修改实例的IAM角色 1、进入云服务器BCC控制台，在实例列表选择您要绑定或者修改角色的实例，在右侧操作列单击 “更多 - 实例设置 – 设置IAM角色”，如下图所示。

2、在弹出的 **设置IAM角色** 对话框中点击IAM角色下拉选择目标IAM角色，单击确认按钮，实例IAM角色修改成功。

收回实例的IAM角色 1、进入云服务器BCC控制台，在实例列表选择您要绑定或者修改角色的实例，在右侧操作列单击 “更多 - 实例设置 – 设置IAM角色”，如下图所示。

2、在弹出的 设置IAM角色 对话框中操作类型点击“收回”，单击确认按钮，实例IAM角色收回成功。

通过API授予IAM角色

您可通过绑定角色给指定实例绑定IAM角色。

您可以通过解绑角色收回实例的IAM角色。

您可以通过获取实例角色列表查看账户下全部角色名称。

镜像

镜像概述

什么是镜像？

镜像是一个包含了软件及必要配置的云服务器模板，至少包含操作系统，还可以包含应用软件（例：数据库）和私有软件。BCC镜像提供了创建和启动BCC和EBC实例所需的信息。创建实例时，必须选择镜像，并可以批量创建实例。

镜像类型	描述	费用
公共镜像	百度智能云官方提供的镜像，皆是正版授权，安全性好，稳定性高。公共镜像包含了Windows Server系统镜像和主流的Linux系统镜像。	Windows Server镜像会涉及资源计费，具体费用以创建实例时显示的信息为准。您购买的Windows Server公共镜像均已获得微软的正版授权。其余公共镜像免费。
自定义镜像	您使用实例或快照创建的镜像，或是您从本地导入的自定义镜像。用于导入的镜像必须满足操作系统版本、镜像格式等要求，请参见 镜像导入限制。只有自定义镜像的创建者可以使用、共享、复制和删除该镜像。 > 若实例系统盘为CDS云盘时，创建的自定义镜像不能用于创建本地盘系统盘实例。	均免费。
共享镜像	其他百度智能云账号共享给您的镜像。更多详情，请参见 共享自定义镜像。	如果您的共享镜像的最终来源为镜像市场时，以云市场第三方服务商提供的镜像计费方式为准。
服务集成镜像	基于基础操作系统并集成有通用服务和应用软件环境的镜像。由百度或第三方服务商提供，经由百度智能云审核后发布。服务集成镜像可帮助您快速创建包含控制面板、PHP等常用运行环境的云服务器。	以云市场第三方服务商提供的镜像计费方式为准。
云市场镜像	由第三方服务商提供的，经百度智能云审核后发布在云市场的镜像环境。包含丰富的服务运行环境和应用软件，可帮助您快速应对各类解决方案场景。	以云市场第三方服务商提供的镜像计费方式为准，目前暂不支持后付费镜像。

自定义镜像生命周期

创建自定义镜像

使用已有镜像创建实例后，您可以对实例进行个性化配置（如安装软件、部署项目等），并为更新后的实例创建自定义镜像。使用该自定义镜像创建的实例将包含已在该镜像中配置的自定义项。

您也可以使用如下方式创建：

- 已有的系统盘快照创建自定义镜像。具体步骤，请参见 创建自定义镜像。
- 从本地导入自定义镜像。具体步骤，请参见 导入自定义镜像。

共享和复制自定义镜像

镜像属地域性资源，例如，您在北京地域创建自定义镜像后，只能在北京地域使用该镜像创建实例。

- 当您共享该镜像给其他百度智能云账号使用时，该账号也只能在北京地域使用该镜像。如果您需要在其他地域共享该镜像，可以先复制镜像到目标地域，然后再将其共享给目标百度智能云账号。更多详情，请参见 共享自定义镜像。
- 当您需要在其他地域使用该镜像时，可以复制该镜像到目标地域。复制后的镜像独立存在，拥有唯一的镜像ID。更多详情，请参见 跨地域复制镜像。

更换镜像

创建实例后，如果您需要更换实例的操作系统，可以通过重装操作系统来实现。

删除自定义镜像

当您不再需要一个自定义镜像，可以将其删除。删除后，您不能再使用该镜像创建实例，请参见 删除自定义镜像。

创建自定义镜像

自定义镜像介绍

本文介绍百度智能云“创建自定义镜像”的方法（含操作指导视频），您可以参考下方文档或视频进行具体操作，请点击此处观看视频《BCC创建自定义镜像》。

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

当用户需要重新部署复杂初始化系统，多次部署同样配置的服务器时，或者需要进行大规模测试和软件开发商自有软件的快速部署时，都可选择创建自定义镜像。

自定义镜像是基于云服务器系统盘某一时刻的快照，实际是将正在使用的云服务器系统盘保存成私有镜像，该镜像只能用户自己看到。

操作准备

- 无需停止实例即可创建自定义镜像。创建镜像过程中，不能改变实例的状态，不要停止、启动或者重启实例，避免创建失败。
- 创建自定义镜像所需的时间，取决于实例磁盘的大小。

创建自定义镜像

1.左侧导航选择 **实例**，进入 **实例列表** 页面。

2.选择对应的实例，在操作栏点击 **更多**，点击 **云盘和镜像**，点击 **创建自定义镜像** 。

3.输入新镜像名称，开始新镜像的创建，约需1-10分钟。

> **说明：**
>
> 自定义镜像名称符合大小写字母、数字以及-_ /.特殊字符，必须以字母开头，长度1-65。

4.在左侧的导航中选择 **镜像**。

5.在 **自定义镜像** 列表，查看镜像状态为 **已完成**，创建自定义镜像成功。

后续操作

完成了自定义镜像的创建后，您可以在自定义镜像页签进行“创建实例”的操作。通过自定义镜像创建实例时，系统将默认选择实例镜像为该自定义镜像，您也可以根据自己的需求重新选择公共镜像或其他服务集成镜像。

通过API创建自定义镜像

您可以使用API **创建自定义镜像** 创建自定义镜像，API请参考 镜像相关接口。

删除自定义镜像

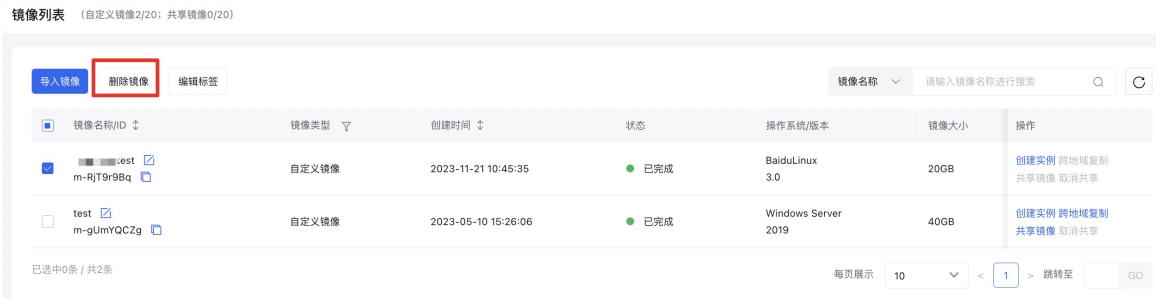
注意事项

- 如果自定义镜像正在共享中，则无法删除。
- 删除自定义镜像后，不能再使用该镜像创建、重置实例。
- 删除自定义镜像后，已使用该镜像创建的实例：
  - 仍可正常使用，并会继续产生费用。
  - 将无法重新初始化云盘。

删除自定义镜像

1.在左侧导航选择 **镜像**，进入镜像列表页面。

2.在镜像列表中，选中一个或多个镜像。



3.点击 **删除镜像**，系统弹出删除镜像对话框。

4.点击 **发送验证码**，系统将验证码发送至您绑定手机，请填写收到的验证码。

5.点击 **确认**，完成删除快照操作。

通过API删除自定义镜像

您可以使用 API删除自定义镜像，API请参考 删除自定义镜像接口。

共享自定义镜像

百度智能云BCC用户可以灵活将 **自定义镜像** 定向分享给其他用户，被共享用户可以使用共享的镜像创建BCC实例和重装操作系统。

适用场景

- 百度智能云用户可以将包含特定业务内容的BCC云服务器实例镜像分发给其他用户，帮助其构建业务系统，提高效率，加快扩展速度。
- 百度智能云用户的IT部门可以根据公司的要求制作BCC云服务器实例镜像，通过分享功能下发至各个部门，快速搭建业务系统。

使用影响

由于共享镜像可能涉及数据隐私和安全，用户需要自行承担使用镜像共享带来的风险。建议使用共享镜像前，登录BCC实例检

查确认是否有数据泄露风险。

背景信息

项目	说明
共享类型	自定义镜像
配额	• 共享用户：每个用户单地域最多可同时创建20个镜像，每个自定义镜像最多可以共享给50个其他用户。 • 被共享用户：每个被共享用户最多可以拥有20个共享镜像；共享镜像不占用自身创建镜像的配额。
使用区域	自定义镜像的共享需要在同一区域。
删除镜像	不可以删除已经共享给其他用户的自定义镜像，取消所有共享后方可删除。
共享用户	镜像不可以重复共享给其他用户，且不可以共享给自己本身。收到共享镜像的用户，只能使用共享镜像，不能主动取消共享或者删除镜像。
操作管理	支持管理控制台操作。

开启共享

自定义镜像跨用户共享的流程为：

1. 登录 云服务器BCC控制台。
2. 点击左侧导航中 **镜像**，进入镜像列表页。
3. 选择需要共享的镜像，镜像类型必须是 **自定义镜像**。
4. 点击 **共享镜像**，输入共享用户的百度账号、云账号或用户ID。

> 说明：
> 您可以将镜像共享给50位用户，如需共享请输入目标用户的账号信息。
> 镜像共享后，被共享的用户可以使用此镜像创建、重置实例。

5. 点击 **确定**,完成共享镜像操作。

取消共享

完成以下操作后，取消共享您的 **自定义镜像** 给其他用户。

> **说明：**
>
> 取消共享后，被共享方无法查看该镜像，同时无法基于此镜像创建实例。

1. 登录 云服务器BCC控制台。
2. 点击左侧导航中 **镜像**，进入镜像列表页。
3. 选择需要取消共享的镜像，镜像类型必须是"自定义镜像"。
4. 点击 **取消共享**，显示该镜像的共享用户列表。
5. 选择账户需要取消共享的百度账号、云账号或用户ID，点击"确定"，完成取消共享操作。

常见API

您可以使用 **API接口**

- 共享自定义镜像，API请参考 共享自定义镜像。
- 取消共享自定义镜像，API请参考 取消共享自定义镜像。

跨地域复制镜像

概述

镜像跨地域复制适用于跨地域部署实例、跨地域备份数据。您可以复制某一地域中的自定义镜像到其他百度智能云地域，并使用该镜像在相应地域创建BCC实例，实现在不同的地域部署环境相同的BCC实例。

应用场景

镜像是一种地域性资源，自定义镜像归属于其被创建时的地域。不同场景的使用方式如下表所示。

场景	操作步骤	描述
跨地域同账号复制	按下文描述操作即可	跨地域复制完成后，在目标地域生成一个新镜像，新镜像具有唯一的镜像ID。
跨地域跨账号复制	先按下文步骤进行操作，完成后再按 共享自定义镜像操作	跨地域复制镜像到目标地域后，再共享镜像给目标账号。

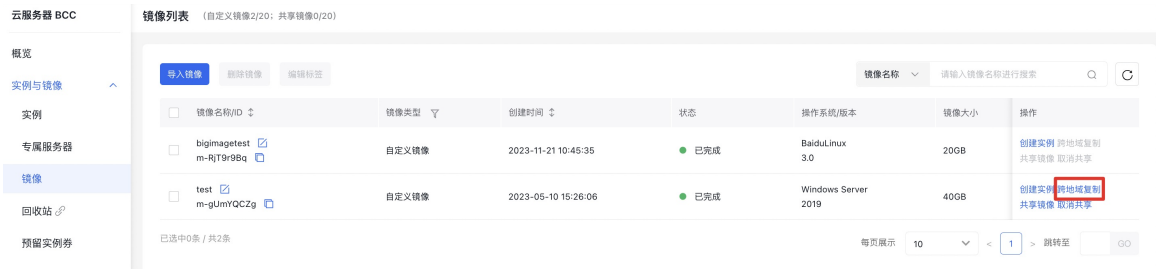
使用限制

- 跨地域复制镜像仅适用于自定义镜像。
- 关联数据盘快照的镜像不支持跨地域复制。
- 不同地域间的数据传输流量费。目前未对这部分流量收费，具体收费时间以百度智能云公告为准。

操作步骤

> **说明：**
>
> 复制镜像时间大概为10～30分钟。
>
> 状态为**复制中**时，您可以随时单击**取消复制**。取消任务后，目标地域的镜像列表中不再显示该自定义镜像。

1. 登录百度智能云平台，进入BCC产品页面
2. 在左侧导航栏中，选择 **镜像** ，进入镜像列表页面。
3. 选中需要复制的镜像，镜像类型必须是自定义镜像，在操作列中，单击 **跨地域复制**。



4. 在 **复制镜像** 对话框中，您可以看到选中的自定义镜像的名称。
5. 选择目标地域（地域支持多选），并单击 **确定**。
6. 切换到目标地域，查看自定义镜像的复制进度。当状态变为 **已完成** 时复制完成。

常用API

您可以使用 API接口跨区域复制自定义镜像，API请参考 跨地域复制自定义镜像接口。

导入自定义镜像

导入概述

在导入镜像前，您应该已经完成如下工作：

- 通读镜像导入说明，保证待导入的镜像满足各种限制和要求；
- 开通对象存储BOS。

镜像导入步骤

1.登录百度智能云对象存储控制台，上传您制作好的镜像文件【参考BOS控制台用户指南】 注：WEB上传不支持直接上传超过5G的文件，如果您的镜像文件实际大小超过5G，请安装【BOS桌面】进行上传；

2.登录 云服务器BCC控制台；

3.单击左侧导航窗格中的【镜像】；

4.单击【导入镜像】按钮；

5.填写要导入的镜像信息， 单击【开始导入】。

镜像导入限制

Linux系统类型镜像限制 |限制类型|介绍| |--|--| |操作系统 |支持CentOS、Ubuntu、Debian、OpenSUSE、SUSE发行版的镜像| |镜像格式| 支持qcow2、raw、vmdk和vhd格式| |镜像大小 | 镜像vsize不超过**500G**| |网络 |动态获取网络地址；暂不支持IPV6；不支持多个网卡| |驱动 |镜像必须安装虚拟化平台 KVM 的 virtio 驱动，详情参考 Linux系统检查virtio驱动| |cloud-init |镜像建议安装cloudinit，详情见安装安装cloud-init（Linux）| |文件系统 |Linux镜像暂时支持ext3、ext4（ext4不能带有metadata_csum 特性）、xfs文件系统（不含V5版xfs）；不支持LVM分区| |镜像文件URL |只能是与镜像要导入的区域是同一区域的BOS URL|

其它建议：

1.请勿修改/etc/issue*，否则将导致系统发行版无法被正常识别从而使得系统创建出现问题。
2.请勿修改/boot/grub/menu.lst，否则将导致系统无法启动。
3.请勿修改/etc/shadow为只读，否则将导致无法修改密码文件而使得无法登录系统。
4.请勿通过修改/etc/selinux/config来开启selinux，否则将导致系统无法启动。
5.请勿修改关键系统文件如/sbin, /bin, /lib*目录等。
6.请勿调整系统盘分区，目前只支持单个根分区（多分区可能会造成数据注入失败）。
7.请检查系统盘使用剩余空间，确保系统盘没有被写满。
8.在导入前，请确认文件系统的完整性。
9.在导入前，需要检查grub配置文件的启动参数，里面不可以有 "pcie_aspm=off" 的参数，不然会导致后续卸载磁盘失败，具体检查方法如下：

> CentOS6 查看/boot/grub/grub.conf
>
> CentOS7/8 查看/boot/grub2/grub.cfg

如图所示，以CentOS7为例，在menuentry中查看红框中所显示的部分是否存在 "pcie_aspm=off" 参数，如果存在则需要手动删除。

Windows系统类型镜像限制

限制类型	介绍
操作系统	Windows Server 2008 R2、 Windows Server 2012 R2、Windows Server 2016、Windows Server 2019、Windows Server 2022数据中心版
镜像格式	支持qcow2、raw、vmdk和vhd格式
镜像大小	镜像vsize不超过2T
网络	动态获取网络地址
驱动	镜像必须安装虚拟化平台 KVM 的 virtio 驱动。Windows系统默认未安装 virtio 驱动，在导入镜像前请先安装virtio驱动
cloudbase-init	镜像建议安装cloudbase-init，详情见安装cloudbase-init（Windows）
文件系统	仅支持使用 MBR 分区的 NTFS 文件系统；不支持 GPT 分区
镜像文件URL	只能是与镜像要导入的区域是同一区域的BOS URL

其它建议：

- 导入Windows操作系统的镜像前，请确认文件系统的完整性。
- 请检查系统盘的剩余空间，确保系统盘没有被写满。
- 请勿修改关键系统文件。

检查virtio驱动

Linux系统检查virtio驱动

1、执行命令“grep -i virtio /boot/config-$(uname -r)”命令检查当前操作系统的内核是否支持virtio驱动。

```
[root@localhost ~]# grep -i virtio /boot/config-$(uname -r)
CONFIG_VIRTIO_VSOCKETS=m
CONFIG_VIRTIO_VSOCKETS_COMMON=m
CONFIG_VIRTIO_BLK=m
CONFIG_SCSI_VIRTIO=m
CONFIG_VIRTIO_NET=m
CONFIG_VIRTIO_CONSOLE=m
CONFIG_HW_RANDOM_VIRTIO=m
CONFIG_DRM_VIRTIO_GPU=m
CONFIG_VIRTIO=m
# Virtio drivers
CONFIG_VIRTIO_PCI=m
CONFIG_VIRTIO_PCI_LEGACY=y
CONFIG_VIRTIO_BALLOON=m
CONFIG_VIRTIO_INPUT=m
# CONFIG_VIRTIO_MMIO is not set
[root@localhost ~]#
```

说明：

- 如果在输出信息中没有找到VIRTIO_BLK及VIRTIO_NET的信息，则表示该操作系统没有安装virtio相关驱动。需要在您的服务器平台重新编译Linux内核安装virtio驱动。
- 如果参数CONFIG_VIRTIO_BLK及CONFIG_VIRTIO_NET的值为y，表示系统包含了virtio驱动，可以导入镜像到百度云。
- 如果参数CONFIG_VIRTIO_BLK及CONFIG_VIRTIO_NET的值为m，需要进一步确认virtio驱动正确包含进了initramfs或initrd文件中。

2、执行命令：lsinitrd /boot/initramfs-$(uname -r).img | grep virtio，确认virtio驱动是否包含在initramfs或initrd中。

```
[root@localhost ~]# lsinitrd /boot/initramfs-$(uname -r).img | grep virtio
Arguments: -f --add-drivers ' virtio_blk virtio_pci'
-rw-r--r--   1 root     root         7740 Aug 14 18:43 usr/lib/modules/3.10.0-862.11.6.el7.x86_64/kernel/drivers/block/virtio_blk.ko.xz
drwxr-xr-x   2 root     root            0 Sep 19 02:47 usr/lib/modules/3.10.0-862.11.6.el7.x86_64/kernel/drivers/virtio
-rw-r--r--   1 root     root         4564 Aug 14 18:44 usr/lib/modules/3.10.0-862.11.6.el7.x86_64/kernel/drivers/virtio/virtio.ko.xz
-rw-r--r--   1 root     root         9680 Aug 14 18:44 usr/lib/modules/3.10.0-862.11.6.el7.x86_64/kernel/drivers/virtio/virtio_pci.ko.xz
-rw-r--r--   1 root     root         8276 Aug 14 18:44 usr/lib/modules/3.10.0-862.11.6.el7.x86_64/kernel/drivers/virtio/virtio_ring.ko.xz
[root@localhost ~]#
```

> **说明：** 如果 initramfs 或 initrd 中未找到上述 virtio 驱动相关信息，则需要判断是否已经编译到内核中。使用命令 grep -i virtio /boot/config-$(uname -r) 命令查看与 virtio 驱动相关的配置选项。值为 **y** 的配置选项表示相关驱动已经编译到内核中，并且正确安装，执行 lsinitrd /boot/initramfs-$(uname -r).img | grep virtio 命令不会显示对应的驱动模块信息；值为 **m** 的配置选项表示相关驱动已经添加到 initramfs 或 initrd 中，如果在 initramfs 或 initrd 中未发现相关驱动信息，需要手动添加驱动到 initramfs 或 initrd 中。

Windows系统检查virtio驱动

打开设备管理器，找到【磁盘驱动器】、【网络适配器】和【系统设备】，查看是否安装virtio驱动。如果未安装virtio驱动，则需要进行virtio驱动的安装。

安装virtio驱动

Linux系统安装virtio驱动

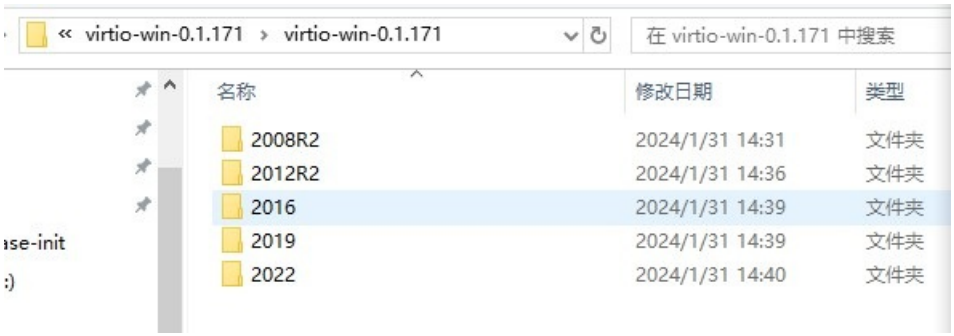
Centos 6 / 7系统

```
[root@localhost ~]# cp /boot/initramfs-$(uname -r).img  /boot/initramfs-$(uname -r).img.bak     # 先将initramfs进行备份（可选步骤）

[root@localhost ~]# mkinitrd -f --with=virtio_blk --with=virtio_pci /boot/initramfs-$(uname -r).img $(uname -r)
```

Ubuntu / Debian系统

```
[root@localhost ~]# echo -e "virtio_pci\nvirtio_blk" >> /etc/initramfs-tools/modules

[root@localhost ~]# update-initramfs  -u
```

Windows系统安装VirtIO驱动 下载virtio-win-0.1.171.zip到您的windows实例中，解压后打开文件夹，可以看到不同Windows操作系统版本对应的文件夹。每个文件夹中都包含amd64子文件夹，对应64位操作系统。



根据源服务器操作系统版本，选择对应的文件夹。例如，本示例中服务器的操作系统版本为Windows Server 2019 64位，则文件夹路径为C:\test\virtio-win-0.1.171\virtio-win-0.1.171\2019\amd64。

1）源系统为Windows Server 2012 R2、2016、2019、2022：

打开cmd命令窗口，进入该目录，执行命令pnputil -i -a *.inf。如果您当前的Windows实例登录用户为普通用户，需要以管理员权限运行该命令。如果是管理员用户，则可以直接运行该命令。

重启服务器。重启后，virtio驱动才会生效。

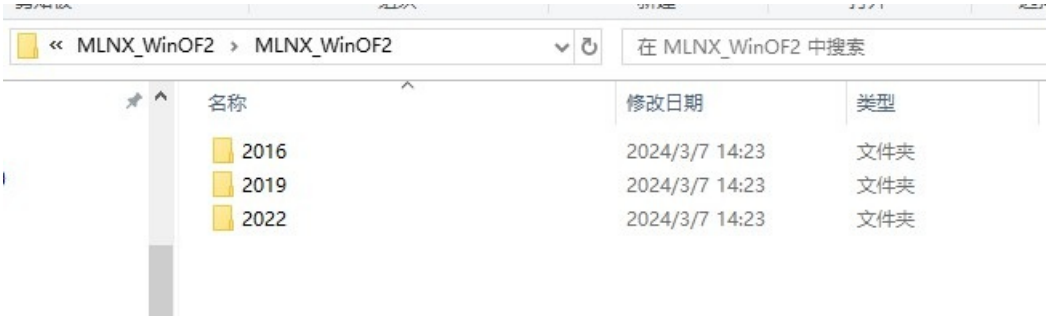
2）源系统为Windows Server 2008 R2时，按照1）的方法安装virtio驱动可能会导致实例启动异常，可以使用离线安装驱动的方

方，具体步骤可参考安装VirtIO驱动里Windows Server 2008 R2部分。

Windows系统安装MLNX_WinOF2网卡驱动（可选）

对于Windows Server 2016、2019、2022，建议安装MLNX_WinOF2网卡驱动，有些规格的实例（如bcc.ga2、bcc.ica2、bcc.ca2、bcc.ma2、bcc.la2等）需要安装该网卡驱动才能识别网络设备。

下载MLNX_WinOF2.zip到您的windows实例中，解压后打开文件夹，可以看到不同Windows操作系统版本对应的文件夹。



根据源服务器操作系统版本，选择对应的文件夹。双击exe程序默认安装即可。

安装cloudbase-init

安装cloud-init（Linux）

cloud-init 主要提供实例首次初始化时自定义配置（例：设置主机名）。

目前支持安装cloud-init的Linux发行版包括CentOS、Ubuntu、Debian、openSUSE。

如果您的镜像中没有安装cloud-init，请采用以下方式进行安装。

> 注意：建议您在做如下操作前对系统创建快照或者镜像备份，以便在出现意外错误时可以恢复系统。

下载及安装cloud-init工具

1. 下载cloud-init安装包

根据不同的操作系统版本，从如下链接下载安装包到服务器上。 **Ubuntu**

Ubuntu 24.04: cloud-init_18.5.6.1-0-u24.deb

Ubuntu 22.04: cloud-init_18.5.6.1-0-u22.deb

Ubuntu 20.04: cloud-init_18.5.6.1-0-u20.deb

Ubuntu 18.04: cloud-init_18.5.6.1-0-u18.deb

Ubuntu 16.04: cloud-init_18.5.6.1-0-u16.deb

CentOS Rocky Linux 9/AlmaLinux 9/CentOS Stream 9: cloud-init-18.5.6.1-1.el9.x86_64.rpm

CentOS 8/Rocky Linux 8/AlmaLinux 8/Centos Stream 8: cloud-init-18.5.6.1-1.el8.x86_64.rpm

CentOS 7: cloud-init-18.5.6.1-1.el7.x86_64.rpm

Debian

Debian 12: cloud-init-18.5.6.1-1-deb12.deb

Debian 10/11: cloud-init-18.5.6.1-1-deb10.deb

Debian 9: cloud-init-18.5.6.1-1-deb9.deb

openSUSE openSUSE 15.2: cloud-init-18.5.6.1-1-opensuse152.x86_64.rpm

2. 安装

Ubuntu及Debian安装

如果已经安装了当前版本或更高版本，重新安装前需要先卸载cloud-init，卸载命令：

```
apt-get --purge remove cloud-init
```

执行安装命令：

```
dpkg -i cloud-init_xxxxxx.deb
```

CentOS及openSUSE安装

如果已经安装了当前版本或更高版本，重新安装前需要先卸载cloud-init，卸载命令：

```
rpm -e cloud-init
```

执行安装命令：

```
rpm -i cloud-init-xxxxxx.rpm
```

配置cloud-init

根据不同操作系统，从以下链接下载 cloud_xxxxx.cfg到服务器上，替换掉/etc/cloud/cloud.cfg文件。

Ubuntu的cloud.cfg

CentOS/Rocky Linux/Almalinux/CentOS Stream的cloud.cfg

Debian的cloud.cfg

openSUSE的cloud.cfg

下载后执行`cp cloud-xxxxx.cfg /etc/cloud/cloud.cfg`即可。

安装cloudbase-init（Windows）

安装cloudbase-init后，重启实例会触发一次密码重置。创建实例时如果设置了密码，则会重置为该密码；如果没有设置密码，则会重置为随机密码。用户可通过控制台再次重置成新密码。

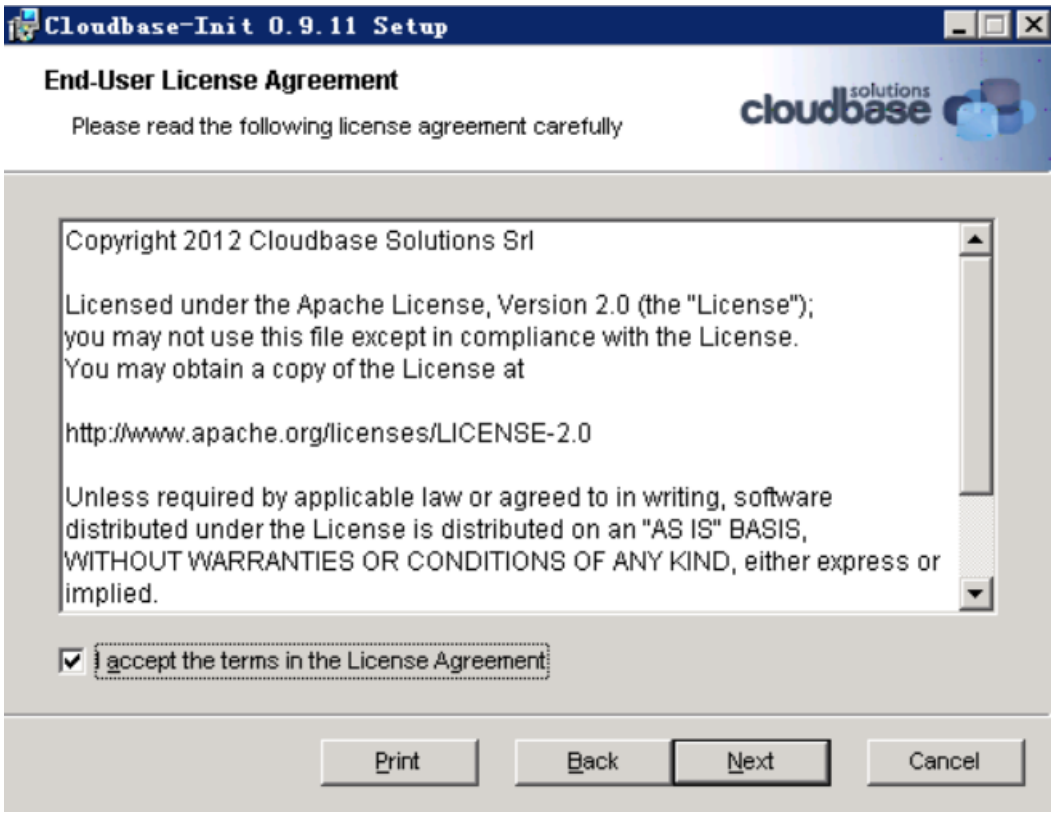
一键安装（推荐）

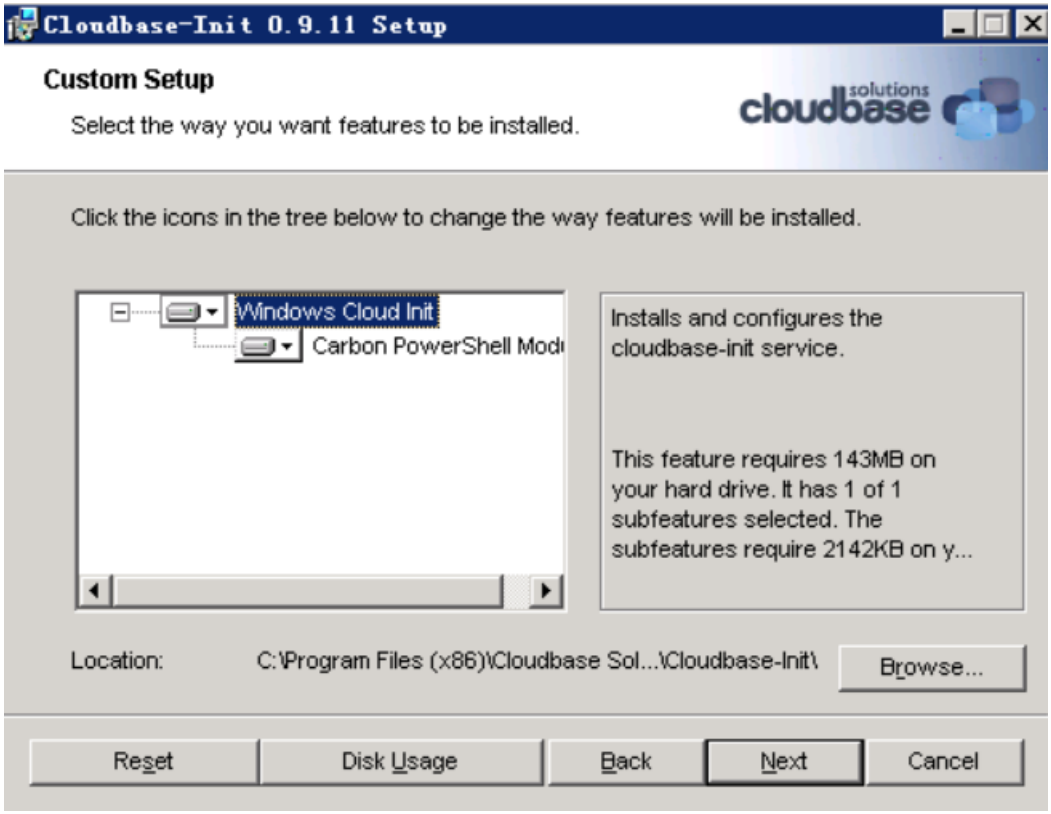
打开cmd命令窗口，运行以下命令：

powershell "IEX (New-Object Net.WebClient).DownloadString('http://mirrors.baidubce.com/baidu/windows/cloudbase-init/Cloudbase-Init-Install.ps1')"

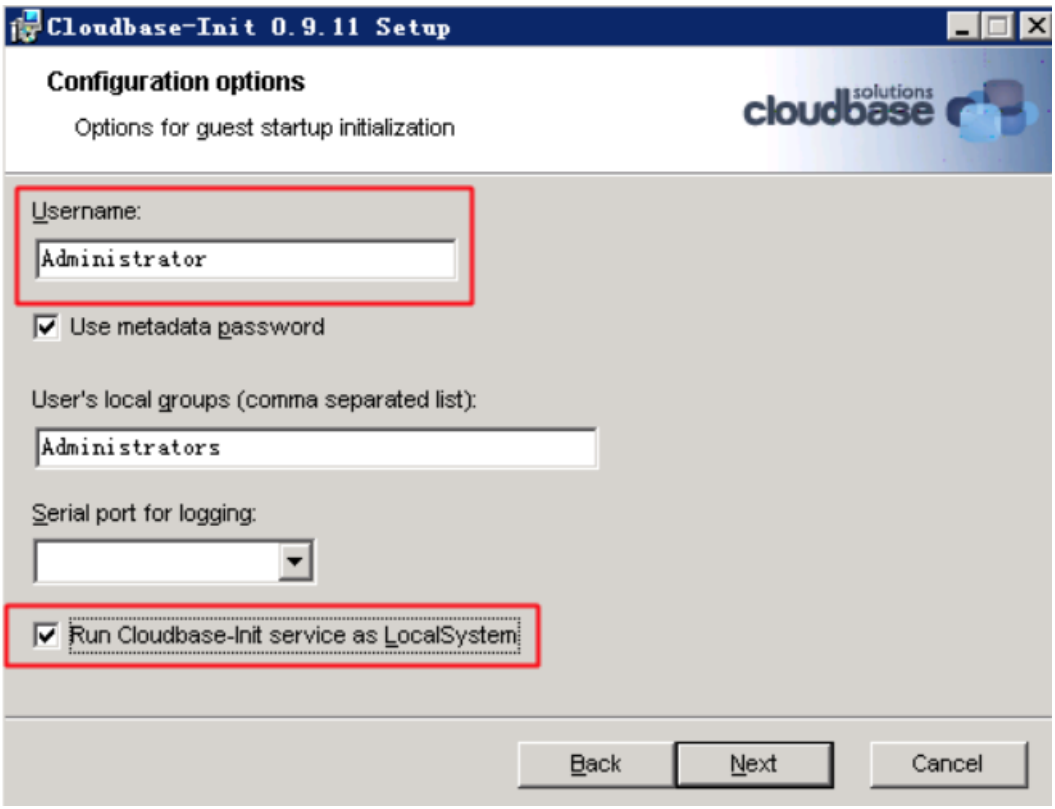
手动安装cloudbase-init

1. 下载[CloudbaseInit.zip]到您的windows实例中，执行安装过程。



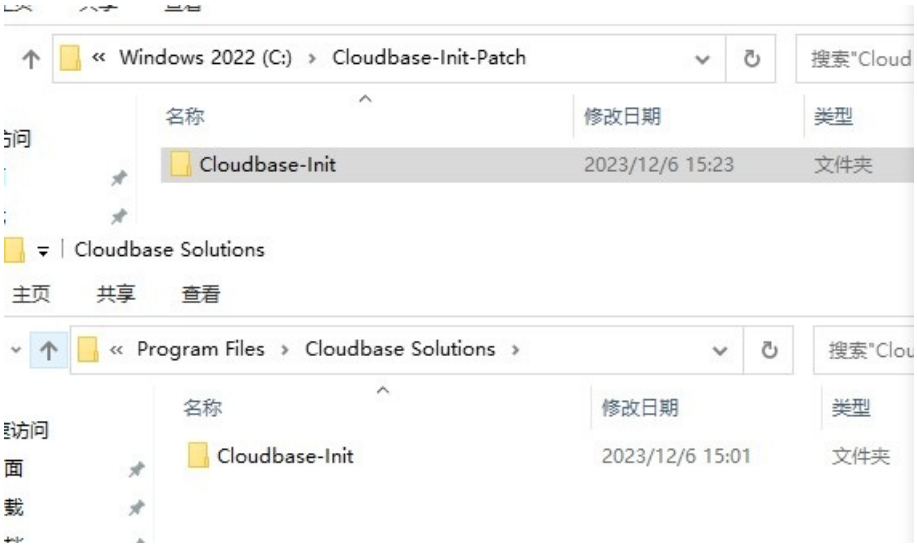


2. 需要将Username修改为：Administrator，并勾选上Use metadata password 和 Run Cloudbase-Init serveice as LocalSystem

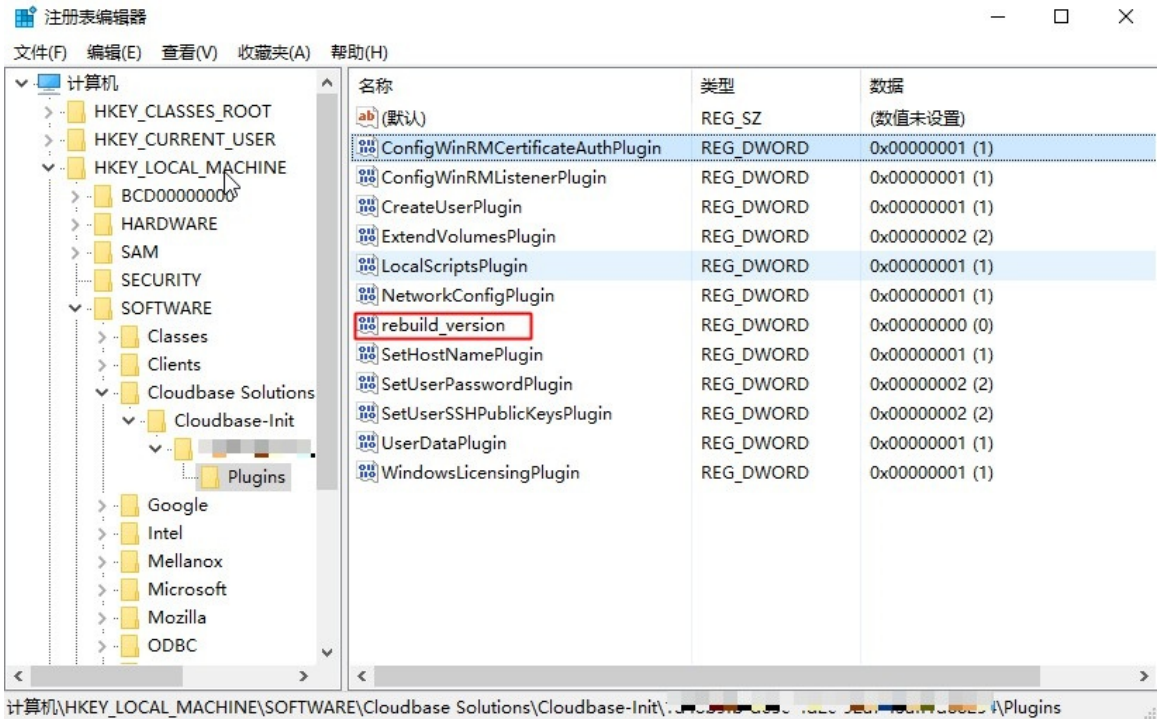


3. 安装完成时，不要勾选下图所示的选项（默认不选择）。

4. 下载[Cloudbase-Init-Patch.zip]到您的windows实例中，解压后将Cloudbase-Init文件夹复制到C:\Program Files\Cloudbase Solutions下，覆盖原有文件夹。



5. 检查注册表HKLM\SOFTWARE\Cloudbase Solutions\Cloudbase-Init\{vm uuid}\Plugins下的各键值是否和下图一致，如有不一致的改成下图中的值。



其中rebuild_version的值需要查询metadata服务器，在您的windows实例中打开链接http://169.254.169.254/openstack/latest/meta_data.json

检测网络配置

在镜像导入前，确保您的系统获取网络地址的方式为动态获取。

Linux系统检查网络获取方式

CentOS系统查看网络配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp            # 如果 BOOTPROTO=static 则需要修改为dhcp
```

Ubuntu/Debian查看网络配置方式

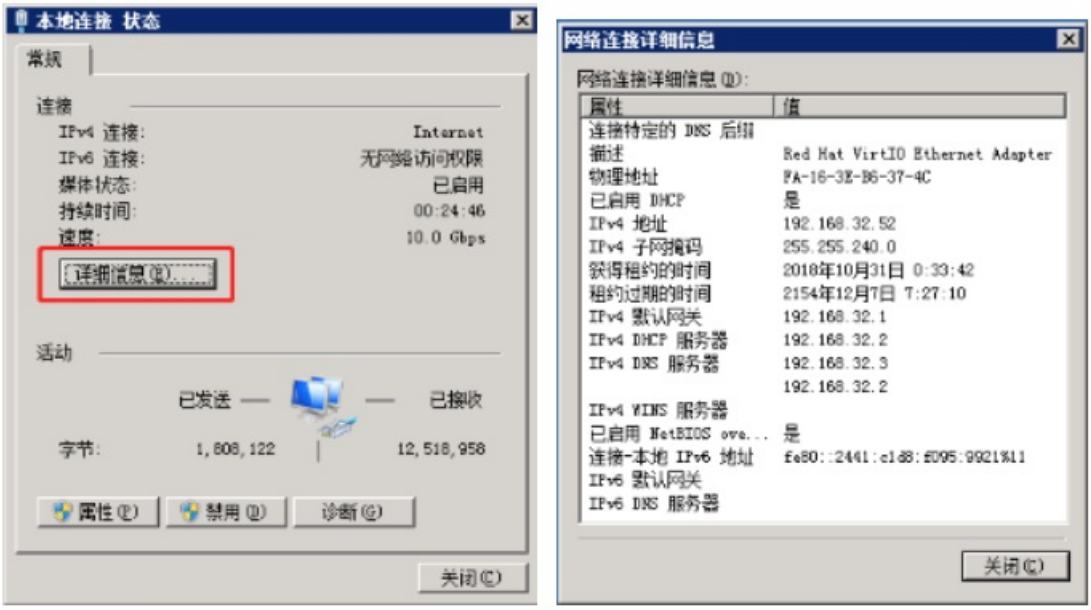
```
root@instance-x09zr2jc:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp            # 如果此处为static，则需要修改为dhcp
```

Windows系统查看网络配置

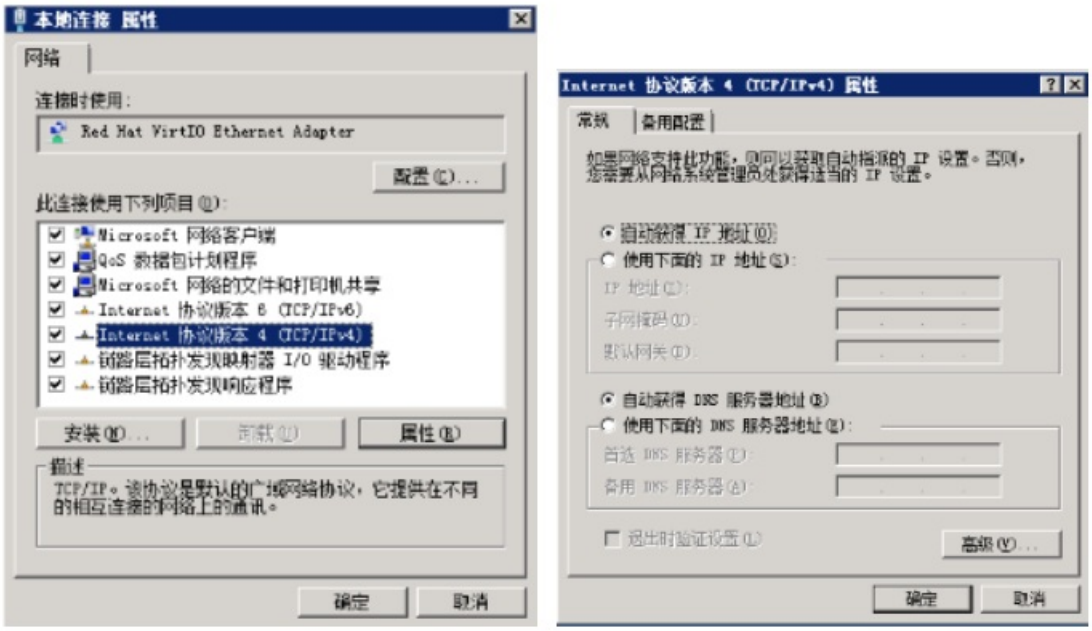
以Windows Server 2008 R2为例：

在Windows系统上依次选择【开始 】> 【控制面板】>【网络和Internet】>【网络和共享中心】，如图所示：

单击【本地连接】，查看网络连接属性，查看是否启用DHCP，如果已启用DHCP，则无需再进行设置。



如果没有启动DHCP，则需要设置本地连接的属性，点上图中的【属性】，设置Internet协议版本4，如下图所示：



系统优化操作（Linux）

修改fstab 文件磁盘标识方式为UUID

在您的系统上运行命令：blkid，返回结果如图示：

```
[root@instance-drtco0fm ~]# blkid
/dev/vda1: UUID="ce4f29f2-f1a9-4c31-95e6-5bc32a698d2b" TYPE="ext4"
```

cat /etc/fstab文件，查看挂载点 “/” 根目录一行，如下图所示，则无需修改。

```
UUID=eb448abb-3012-4d8d-bcde-94434d586a31 /                       ext4    defaults        1 1
```

如果第一列显示的是磁盘分区名称，如下图所示，需要编辑/etc/fstab文件，将磁盘分析名称/dev/vda1修改为【UUID=xxxxxxx】形式。

```
/dev/vda1               /                       ext3    defaults                1 1
```

修改grub 文件磁盘标识方式为UUID

使用导入的镜像创建实例，为了能够成功引导系统，强烈建议grub配置文件中，引导磁盘标识为UUID方式。

以centos 7为例，查看/boot/grub2/grub.conf文件，如果显示为root=UUID=ID方式则不需要修改。

```
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd
else
  search --no-floppy --fs-uuid --set=root e0ecc3fc-ed80-46a2-a92e-971da4e21998
fi
linux16 /boot/vmlinuz-3.10.0-862.el7.x86_64 root=UUID=e0ecc3fc-ed80-46a2-a92e-971da4e21998 ro crashkernel=auto LANG=en_US.UTF-8
initrd16 /boot/initramfs-3.10.0-862.el7.x86_64.img
```

如果显示为root=/dev/方式，则需要通过blkid命令查询对应磁盘分区的UUID，然后在/boot/grub2/grub.conf文件中修改为root=UUID=ID的形式。

```
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd
else
  search --no-floppy --fs-uuid --set=root e0ecc3fc-ed80-46a2-a92e-971da4e21998
fi
linux16 /boot/vmlinuz-3.10.0-862.el7.x86_64 root=/dev/sda1 ro crashkernel=auto LANG=en_US.UTF-8
initrd16 /boot/initramfs-3.10.0-862.el7.x86_64.img
```

检查grub启动参数

镜像导入百度智能云平台前，需要检查grub配置文件的启动参数：

CentOS 7 为例，查看/boot/grub2/grub.cfg

```
menuentry 'CentOS Linux (3.10.0-862.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuen
ced-e0ecc3fc-ed80-46a2-a92e-971da4e21998' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
      search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msd
    else
      search --no-floppy --fs-uuid --set=root e0ecc3fc-ed80-46a2-a92e-971da4e21998
    fi
    linux16 /boot/vmlinuz-3.10.0-862.el7.x86_64 root=UUID=e0ecc3fc-ed80-46a2-a92e-971da4e21998 ro crashkernel=auto LANG=en_US.UTF-8
    initrd16 /boot/initramfs-3.10.0-862.el7.x86_64.img
```

如图所示，在menuentry中，查看红框中所显部分，是否存在“console=tty0 console=ttyS0,115200” 参数。

如果参数不存在，需要编辑/boot/grub2/grub.cfg文件，在ro参数后面添加“console=tty0 console=ttyS0,115200”。

注意对于centos6系统，console=tty0 console=ttyS0,115200也需要在root参数后面，如下图所示：

```
title CentOS (2.6.32-754.3.5.el6.x86_64)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.32-754.3.5.el6.x86_64 ro root=UUID=8e644fc3-6868-4272-8df6-2315
d_NO_LUKS console=tty0 console=ttyS0,115200 LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
YPE=pc KEYTABLE=us rd_NO_LVM rd_NO_DM rhgb lapic=notscdeadline divider=10
        initrd /boot/initramfs-2.6.32-754.3.5.el6.x86_64.img
```

注：ubuntu系统的grub启动参数为 console=tty1 console=ttyS0,115200n8

配置DNS

为了保证您的镜像导入后，可以正常地访问网络，需要对DNS配置进行设置。

1. CentOS/RHEL

以网卡eth0为例，编辑/etc/sysconfig/network-scripts/ifcfg-eth0，增加以下设置：

RES_OPTIONS="rotate timeout:1"

2. Debian/Ubuntu

编辑/etc/resolvconf/resolv.conf.d/tail 文件，添加以下内容到文件末尾：

options rotate timeout:1

3. openSUSE

编辑/etc/sysconfig/network/config，找到“NETCONFIG_DNS_RESOLVER_OPTIONS”，将其修改为：

NETCONFIG_DNS_RESOLVER_OPTIONS="timeout:1 rotate"

清理网络规则文件

在您的服务器系统导入百度智能云前，请先确保/etc/udev/rules.d/70-persistent-net.rules文件为空。

在命令行运行： cat /etc/udev/rules.d/70-persistent-net.rules，如果返回信息中包含如下所示内容：

```
# PCI device 0x8086:0x100f (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:0c:29:73:b8:a6", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
```

请删除所有类似的信息，或者删除/etc/udev/rules.d/70-persistent-net.rules文件。

关闭防火墙

以CentOS系统为例。

Centos6系统在终端运行以下命令：

chkconfig iptables off

Centos7系统在终端运行以下命令：

systemctl disable firewalld

Centos系统关闭SElinux

编辑/etc/selinux/config文件，将SELINUX=enforcing修改为SELINUX=disabled。

镜像格式检查

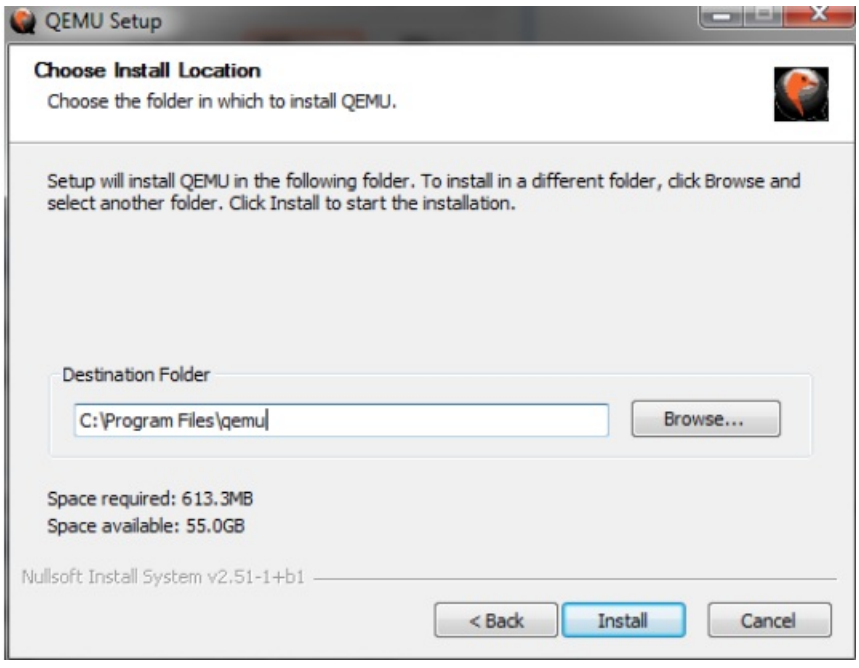
在镜像导入前，需要确认您的镜像格式为raw或qcow2(v2)。

安装qemu-img

- 如果检验平台为Ubuntu操作系统，运行: apt-get install qemu-utils
- 如果检验平台为CentOS操作系统，运行: yum install qemu-img
- Windows平台安装qemu-img，请先下载: qemu-img for windows，选择合适的版本进行下载。

windows下安装及设置qemu-img步骤如下：

1. 安装qemu-img，本示例安装路径为： C:\Program Files\qemu

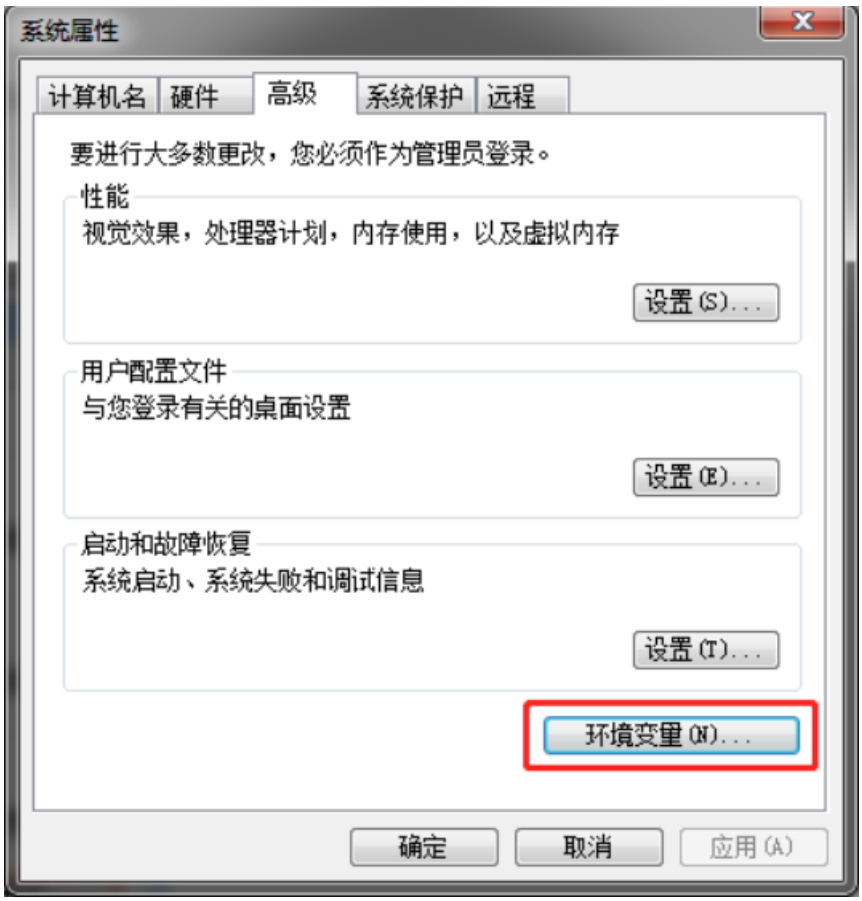


2. 为qemu-img配置环境变量：

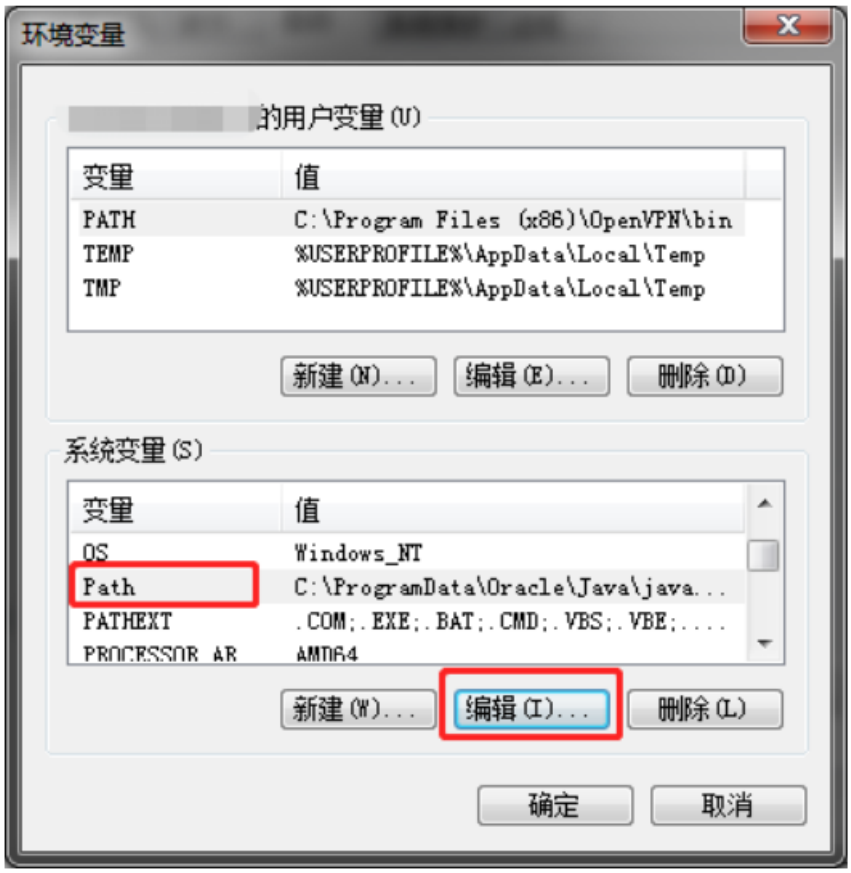
选择 开始 > 计算机，右键单击 属性。

在左侧导航栏里，单击 高级系统设置。

在 系统属性 对话框里，单击 高级 页签，并单击 环境变量。



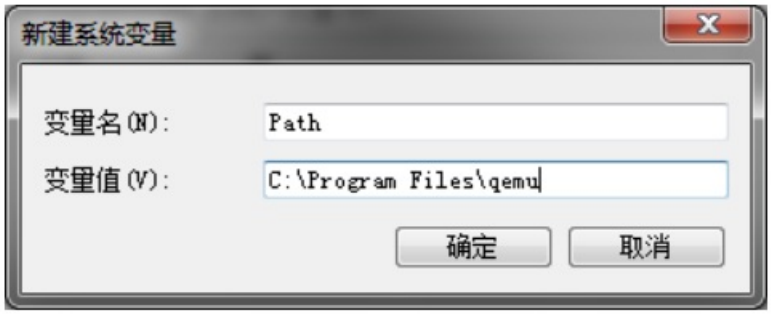
在 环境变量 对话框里，在 系统变量 部分找到 Path，并单击 编辑。如果 Path 变量不存在，单击 新建。



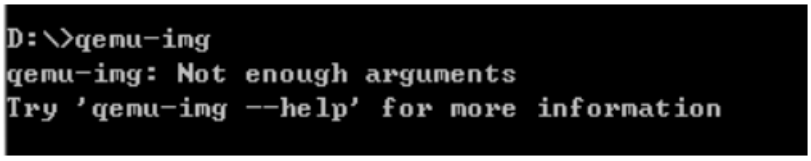
3. 添加系统变量值

如果是 编辑系统变量，在 变量值 里，添加 C:\Program Files\qemu，不同的变量值之间以半角分号（;）分隔

如果是 新建系统变量，在 变量名 处输入 Path，在 变量值 处输入 C:\Program Files\qemu。



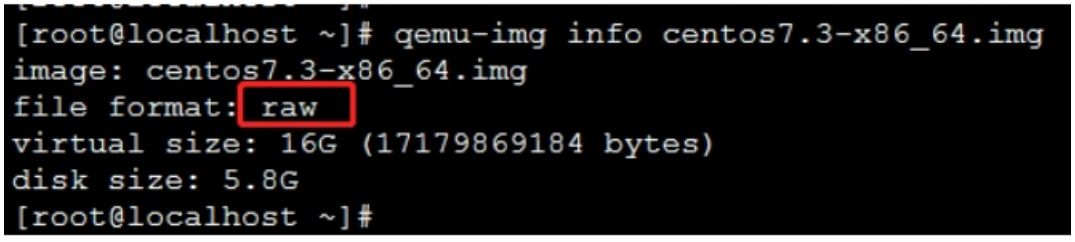
4. 设置完成后，可以在命令提示符中输入qemu-img命令进行验证环境变量是否设置成功，如果有关qemu-img的信息输出，则表明设置正确。

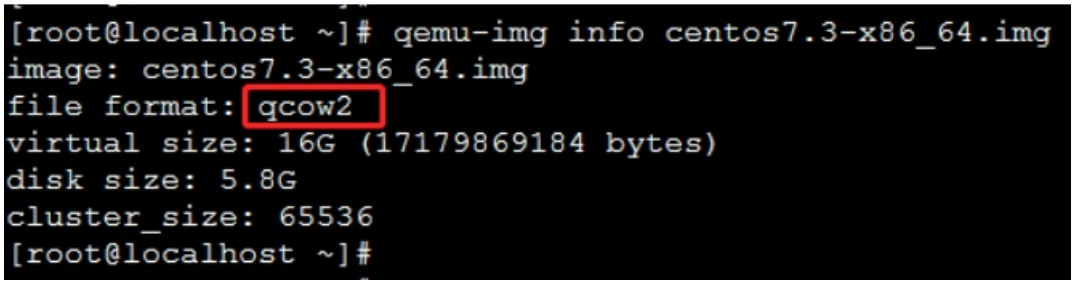


镜像格式检查

使用qemu-img info 命令检查镜像格式，确认导入的镜像文件格式为raw或qcow2。

命令：qemu-img info image_name





如图所示，如果返回信息中file format为raw或qcow2，表明该镜像符合标准。如果不是raw和qcow2格式，请先进行格式转换。

注：如果检测结果为qcow2，需要进一步检测兼容性。

qcow2镜像兼容性检查

将您的镜像复制到检测系统中，以Linux检测系统为例：

如果linux操作系统为CentOS6：

使用qemu-img info命令查询镜像的qcow2版本，如果输出信息如下所示，则表明qcow2镜像文件为v2版本，无需进行qcow2兼容性转换。

```
[root@localhost ~]# qemu-img info centos7.3-x86_64.img
image: centos7.3-x86_64.img
file format: qcow2
virtual size: 16G (17179869184 bytes)
disk size: 5.8G
cluster_size: 65536
```

如果输出信息如下所示，说明该qcow2镜像文件为高版本（V3），需要进行降级处理才能导入百度云。降级操作请参考格式转换章节。

```
[root@localhost ~]# qemu-img info centos7.3-x86_64.img
'image' uses a qcow2 feature which is not supported by this qemu version: QCOW version 3
Could not open 'centos7.3-x86_64.img': Operation not supported
[root@localhost ~]#
```

如果是在CentOS7，Ubuntu14/16/18，debain8/9等平台上使用qemu-img查看qcow2镜像的信息

如下图所示，返回结果中compat为0.10时，无需要进行qcow2的兼容性转换。

```
[root@localhost ~]# qemu-img info centos7.3-x86_64.img
image: centos7.3-x86_64.img
file format: qcow2
virtual size: 16G (17179869184 bytes)
disk size: 5.8G
cluster_size: 65536
Format specific information:
    compat: 0.10
[root@localhost ~]#
```

如果返回结果中compat为1.1时，需要对镜像文件进行qcow2的降级操作。

```
[root@localhost ~]# qemu-img info centos7.3-x86_64.img
image: centos7.3-x86_64.img
file format: qcow2
virtual size: 16G (17179869184 bytes)
disk size: 5.8G
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
[root@localhost ~]#
```

镜像格式转换

qemu-img工具支持raw、cow、qcow、vdi、vmdk、cloop、dmg、bochs、vpc、vvfat、qcow2格式之间相互转换。

转换为raw格式，以vmdk格式转换为raw为例：

```
qemu-img convert  -f  vmdk -O raw  centos.vmdk  centos.raw
```

转换为qcow2格式，以vmdk格式转qcow2为例：

```
qemu-img convert  -f  vmdk -O qcow2  -o compat=0.10  centos.vmdk  centos.raw
```

参数说明：

1.-f 代表源镜像文件和格式；

2.-O【大写】代表目标镜像的格式；

3.-o 【小写】转换为qcow2格式时需要的必要参数项，参数：compat=0.10表明目标镜像（qcow2）版本为兼容版本。

qcow2兼容性转换

qcow2镜像在导入前，如果需要进行兼容性转换，请运行以下命令：

```
qemu-img amend -f qcow2 -o compat=0.10  image_name.qcow2
```

使用镜像检测工具检测镜像

Linux系统镜像检测工具 在将自定义镜像导入至百度智能云之前，建议您使用镜像检测工具 `bd_diagnosis` 来检查镜像是否符合导入条件。这可以帮助确保自定义镜像的质量，从而顺利部署功能齐全且兼容性良好的实例。本文将介绍如何使用 `bd_diagnosis` 工具进行自定义镜像的检测。

bd_diagnosis 工具介绍 `bd_diagnosis` 是百度智能云提供的Linux镜像检测工具，用于扫描镜像中的系统和服务配置，并生成检测报告。该报告详细描述了镜像的系统信息以及检测到的异常情况。

当前不支持 FreeBSD 系统。

操作步骤

1. 使用 `root` 用户登录待检测的服务器内。

2. 依次运行以下命令，下载并解压 `bd_diagnosis` 工具。

```
wget https://wanjuan-docs.bj.bcebos.com/bd_diagnosis.tar.gz
tar xf bd_diagnosis.tar.gz
```

或者您也可以通过浏览器下载 bd_diagnosis 工具。

3. 执行以下命令，运行检测工具

```
cd bd_diagnosis
./main
```

检测工具主要检查以下配置项：

检测项	级别	检查项说明	修复建议
virtio driver	高	检测是否安装 virtio 驱动。实例运行需要相关虚拟化设备驱动支持，未安装该驱动会导致实例启动异常。	安装virtio。
cloud-init	高	检测是否安装 cloud-init。cloud-init 能在启动时动态完成系统初始化，包括主机名和 SSH 密钥等，同时执行实例自定义数据（User data）脚本。未安装导致系统初始化配置缺失。	安装Cloud-init。
fstab config	高	检测镜像中/etc/fstab文件系统磁盘设备挂载信息是否配置正确，错误的配置会导致系统启动异常，例如配置的挂载设备信息不存在、设备UUID错误等。	无
grub	高	检测镜像中grub引导文件是否正常。例如device设备配置异常、相关	无

grub	高	设备不存在、UUID错误等，会导致系统启动异常。	无
root	高	检测系统是否存在 root 用户。若不存在，管理权限受控，安全风险增加。管理权限受控，安全风险增加。	您可以通过修改 /etc/passwd 中的 root 用户配置为 root:x:0:0:root:/root:/bin/bash
sshd service	高	检测 sshd 服务的配置文件（通常是 /etc/ssh/sshd_config）是否正常。若异常，实例 SSH 连接失败。	无
passwd command	高	检查 /etc/shadow 和 /etc/passwd 文件是否存在及读写权限 若无读写权限，实例修改密码等操作无法生效。	您可以在 root 用户下通过执行 chmod +rw /etc/shadow /etc/passwd 恢复文件的读写权限。
DHCP	高	检测网络配置是否为 DHCP。如果镜像中设置为静态模式，会导致实例启动后网络配置异常。	需将网络设置为DHCP动态获取IP模式和清空70-persistent-net.rules文件。
disk size	高	检查系统中磁盘的大小是否超过 500GB。 百度智能云最大支持500GB系统盘。建议你的系统分区大小不要超过500G。否则会无法导入。	无
disk partition	高	检测系统分区类型是否正确。 BCC实例以 BIOS 模式启动的磁盘分区为 MBR。EBC裸金属实例一般以 UEFI 模式启动，磁盘分区为 GPT与系统不兼容的分区模式会导致启动故障。	无
file system type	高	检测文件系统类型。推荐使用 ext2/ext3/ext4 文件系统，其他类型的文件系统会导致系统异常。	无
selinux	中	检测是否禁用 SELinux，建议您关闭SELinux。开启有可能会导致系统启动异常。	需将SElinux置为disable。
Firewall	中	检测防火墙服务是否开启。建议您关闭系统中的防火墙服务，使用安全组统一管理您云上实例的出入流量。开启系统防火墙应用可能导致您访问实例失败。	关闭防火墙。
LVM	中	检测使用lvm分区及设置。会导致无法挂载任务失败，用户无法使用根区容器登录。不建议您使用 lvm 分区。	无
disk uuid	中	检查 Linux 系统的根分区是否通过 UUID 挂载。 当根分区以设备名称挂载，系统启动时名称变化，将导致无法找到根文件系统而启动失败。建议根分区通过UUID挂载。	您可以执行 blkid id 获取分区 UUID，并用 UUID 替换 /etc/fstab 中的分区表示符。
mlx5 driver	中	检测是否安装Mellanox网卡驱动。若未安装，无法使用依赖 Mellanox 网卡设备的实例规格（如bcc.ga2、bcc.ica2、bcc.ma2、bcc.la2 等）。	若您镜像应用实例规格依赖该网卡驱动，请安装Mellanox 网卡驱动。
Kernel config	中	检测 Linux 系统上的内核配置信息，确保系统的关键功能（如硬件热插拔）是否开启。关闭不影响实例运行，但系统不支持内核特性（如硬件热插拔）。比如若实例属于g4、c4、ic4、m4规格族，在线变配后系统内不生效，需重启生效。	**此项异常不影响实例运行**，可镜像导入后按需配置。
halt	中	检测是否配置halt。EBC裸金属实例关机依赖halt能力。若镜像用于创建EBC裸金属实例，建议配置halt。	**此项异常不影响实例启动**，可导入后按需配置。
SshPort	低	检测 22 端口是否启用。22端口关闭，将无法通过该端口执行登录和数据传输等操作。	**此项异常不影响实例启动**，可导入后按需配置。
online resize fs	低	检测镜像是否支持根分区自动扩容。例如您的镜像虚拟磁盘空间大小为10 GB，您购买实例的系统盘为 200 GB，如果您安装了cloud-init、growpart等组件，相关组件在实例初始化时会将根分区及文件系	**此项异常不影响实例运行**，可导入后按需配置。

fs		统扩容，扩容至您系统盘的大小 200 GB。	
graphic	低	检测ubuntu系统是否开启图形桌面。若ubuntu系统用于EBC裸金属实例，不建议开启图形桌面。开启可能会导致睡眠，进而影响到 ssh 登录失败。	若用于EBC裸金属实例，建议关闭图形桌面。
bsm agent	中	检测是否安装bsm-agent。bsm-agent是为实现云服务器BCC、BBC等实例自动化运维打造的工具。建议您安装bsm-agent。	**此项异常不影响实例启动**，只是无法使用云助手、运维编排等能力。若需要，请镜像导入完成后创建实例后，再安装bsm-agent。
bcm agent	中	检测是否安装bcm-agent 。BCM-Agent 作为用户侧的采集程序，承担了采集和传输重要监控数据的任务，便于用户及时感知系统状态。建议您安装 bcm-agent。	**此项异常不影响实例启动**，只是实例无法监控指标数据。若需要，请镜像导入完成后创建实例后，安装bcm-agent。
has agent	中	检测是否安装has-agent。has-agent 能够确保硬件组件的健康状态得到实时监控，并在发现异常时上报，从而帮助运维人员快速定位问题并采取措施。建议您安装 has-agent 便于硬件监控服务。	**此项异常不影响实例启动**，只是无法及时检测到裸金属实例的硬件故障。若需要，请镜像导入完成后创建实例后，安装has-agent。
hosteye	中	检测是否安装hosteye客户端。主机安全监控端（Hosteye）是针对用户的服务监控与行业安全的延续，提供主机安全服务，其具备高稳定性、轻部署的特点，云主机安全端更加强大的防护能力。建议您安装 hosteye 客户端，并应用相关规则。	**此项异常不影响实例启动**，只是云上安全防护性降低。若需要，请镜像导入完成后创建实例后，再安装hosteye。

等待检测工具检测完成，检测结果类似如下信息。

```
-------------------------------------------------------
OS: AlmaLinux 9.2
Arch: x86_64
Kernel: 5.14.0-362.8.1.el9_3.x86_64
BootMode: BIOS(Legacy)
RTC-Mode: utc
-------------------------------------------------------
[ check virtio driver ]                    [ OK ]
[ check cloud-init ]                       [ OK ]
[ check fstab config ]                     [CRITICAL]
[ check grub ]                             [ OK ]
[ check sshd service ]                     [ OK ]
[ check root ]                             [ OK ]
[ check passwd command ]                   [ OK ]
[ check dhcp ]                             [ OK ]
[ check disk size ]                        [ OK ]
[ check disk partition ]                   [ OK ]
[ check file system type ]                 [ OK ]
[ check selinux ]                          [ OK ]
[ check firewall ]                         [ OK ]
[ check LVM ]                              [ OK ]
[ check disk uuid ]                        [WARNING]
[ check mlx5 driver ]                      [ OK ]
[ check kernel config ]                    [WARNING]
[ check halt ]                             [ OK ]
[ check ssh port ]                         [ OK ]
[ check online resize fs ]                 [ OK ]
[ check graphic ]                          [ OK ]
[ check bsm agent ]                        [ OK ]
[ check bcm agent ]                        [ OK ]
[ check has agent ]                        [WARNING]
[ check hosteye ]                          [ OK ]
-------------------------------------------------------
There are 3 warning(s), 1 critical(s).
The reslult file is: AlmaLinux_dignosis_result_20241205102943.txt
Please check result file for the critical/warning information.
-------------------------------------------------------
```

检测工具会根据检测项的等级给出OK、CRITICAL或WARNING检测结果。 **检测结果说明**

- OK：检测项均符合要求。

- CRITICAL：强烈建议修复项。使用该自定义镜像创建的实例会出现**无法正常启动**、**网络异常**等严重问题，**建议您修复报错项后再导入镜像**。
- WARNING：建议优化项。**不影响使用该自定义镜像创建的实例启动**，但会缺少云助手自动化运维等能力。您可忽略报错项继续导入镜像，但建议您在镜像导入创建实例后按需进行优化，以提高您云上的运维效率。

Windows系统镜像检测工具 检查项如下：

检测项	说明	修复建议
VirtIO驱动	百度云支持KVM虚拟化技术，导入百度云的镜像需要安装KVM virtio驱动	如何为系统安装virtio 驱动，请参见安装virtio驱动
磁盘分区类型	百度云BCC默认支持MBR磁盘分区类型	无
磁盘大小	百度云服务器最大支持2TB系统盘	无
cloudbase-init	建议安装cloudbase-init，用来设置hostname、密码等	如何为系统安装cloudbase-init，请参考安装cloudbase-init
MLNX_WinOF2网络驱动	对于Windows Server 2016、2019、2022，建议安装MLNX_WinOF2网卡驱动，有些规格的实例（如bcc.ga2、bcc.ica2、bcc.ma2、bcc.la2等）需要安装该网卡驱动才能识别网络设备	如何为系统安装MLNX_WinOF2驱动，请参考安装MLNX_WinOF2驱动

下载检测工具 check-windows.ps1 到您的系统中，打开powershell窗口，执行该脚本：

使用导入的自定义镜像创建实例

使用自定义镜像创建实例步骤

镜像成功导入后，进入您的云服务器控制台，点击左侧的镜像，可以在镜像列表看到导入的自定义镜像。使用该自定义镜像，创建实例。

成功创建实例后，如果你的镜像系统属于百度智能云平台支持的准备镜像，建议进行进一步定制化。

> 导入的自定义镜像不支持创建EBC实例。

Linux系统定制操作

Linux系统设置软件源（建议）

如果您的云服务器操作为百度智能云支持的标准系统，建议您将软件源设置为百度源。

下载软件源设置脚本到您的服务器上，然后命令行执行：bash setrepo.sh，进行软件源的设置。

Linux系统设置NTP（建议）

如果您的Linux云服务器启用了NTP服务，建议增加百度NTP服务器地址到您的云服务器配置中。

1、编辑/etc/ntp.conf文件，在文件示尾增加一行内容，如下：

```
server ntpsr.baidubce.com
```

2、编辑文件：/etc/ntp/step-tickers（如果存在），将文件内容修改为以下：

```
ntpsr.baidubce.com
```

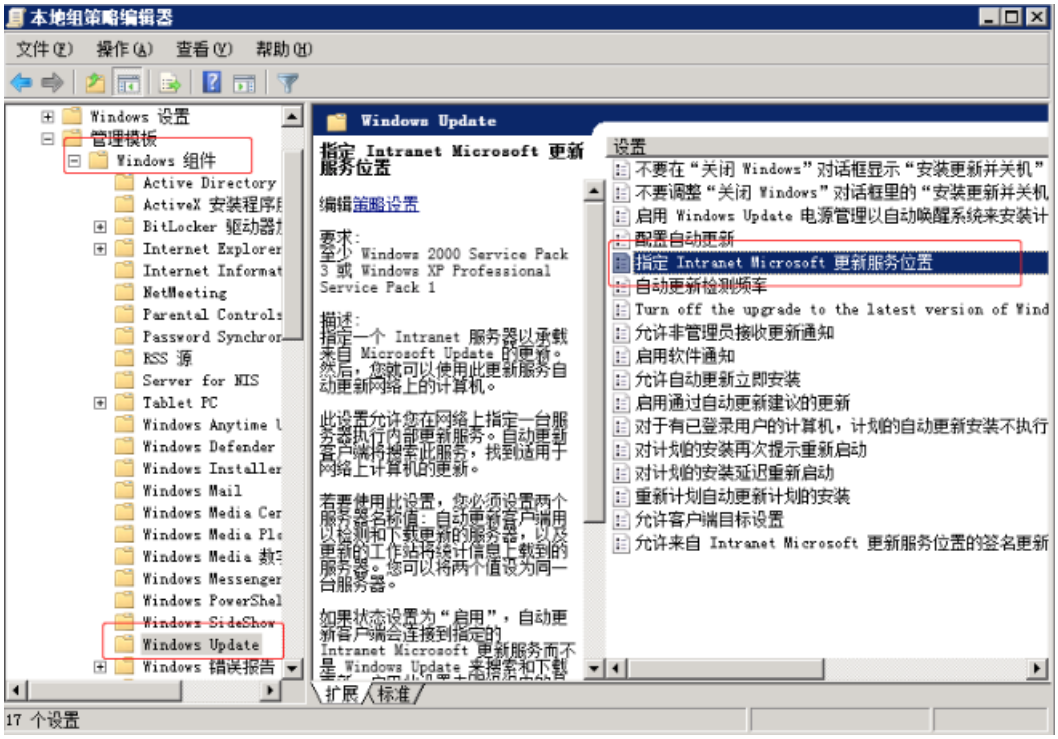
Windows系统定制操作

修改更新服务器地址

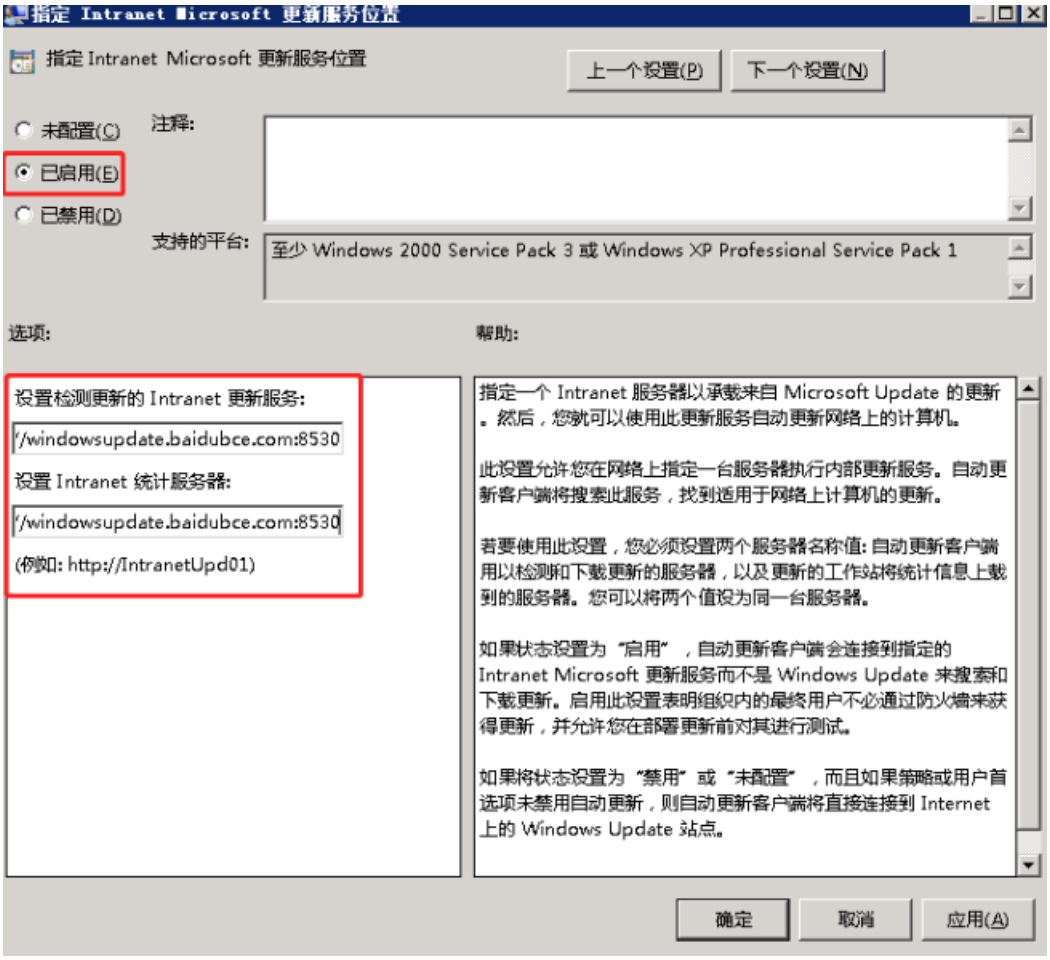
以Windows Server 2008 R2为例：

1、虚拟机 运行gpedit.msc打开【本地组策略编辑器】。

2、找到本地计算机策略--计算机配置--管理模板–Windows组件–Windows Update/Windows更新。



打开右侧的【指定Intranet Microsoft更新服务位置】。



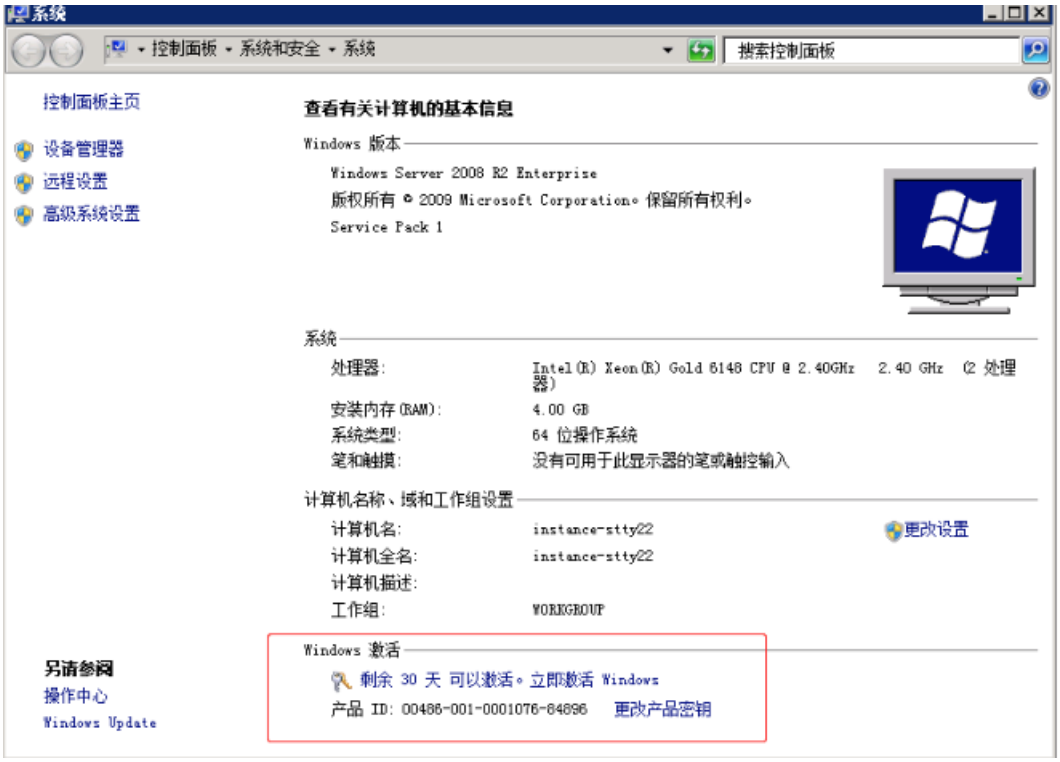
1）指定Intranet Microsoft更新服务位置选择【已启用】。

2）在设置检测更新的Intranet更新服务和设置Intranet统计服务器这两处输入：http://windowsupdate.baidubce.com:8530，然后确定。

3、设置完成后，即可使用百度智能云的更新服务器对Windows系统进行更新补丁操作。

修改激活服务

使用自定义镜像创建的Windows服务器在百度智能云平台上如果处于未激活状态，需要修改虚拟机的激活服务器配置以激活操作系统。



找到命令提示符，然后在命令提示符中运行以下命令：

1. slmgr /ckms # 清除KMS服务器设置

2. slmgr /rearm # 重置服务器的状态，此命令执行后重启系统

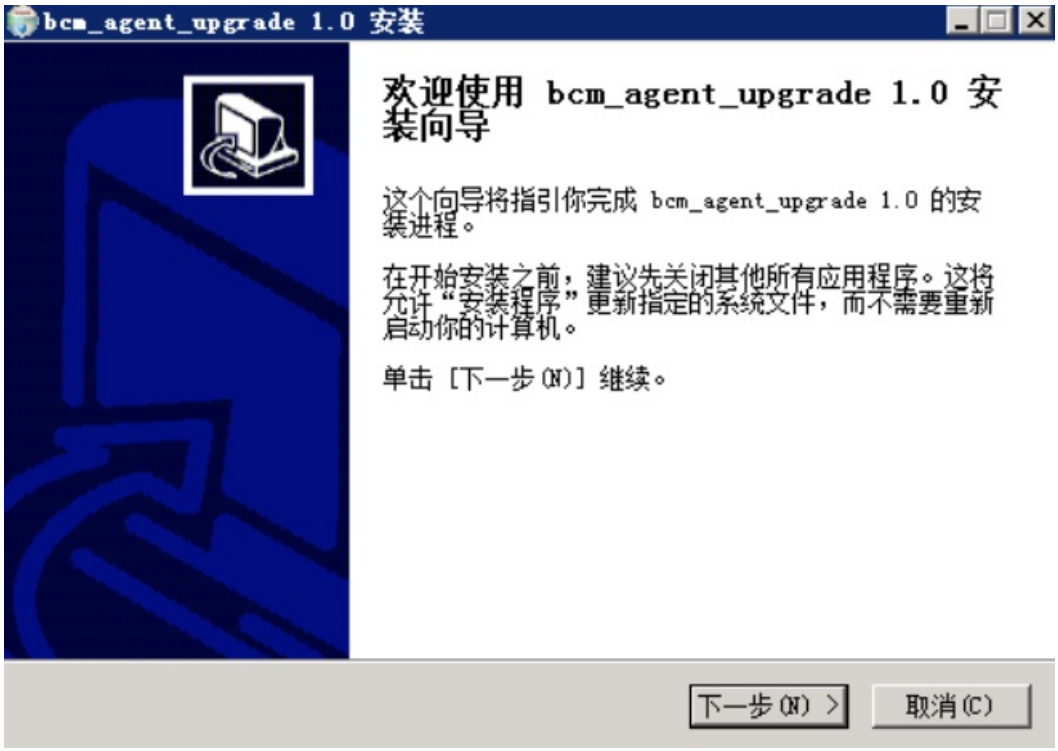
3. slmgr /skms kms.baidubce.com # 设置KMS服务器的IP地址

4. slmgr /ato # 激活 此命令执行成功后会提示操作系统已成功激活

Windows系统安装Baidu bcm-agent（建议）

1、下载bcm-agent到您的云服务器；

2、双击安装文件，进行默认安装即可。



Windows系统安装Baidu bsm-agent（建议）

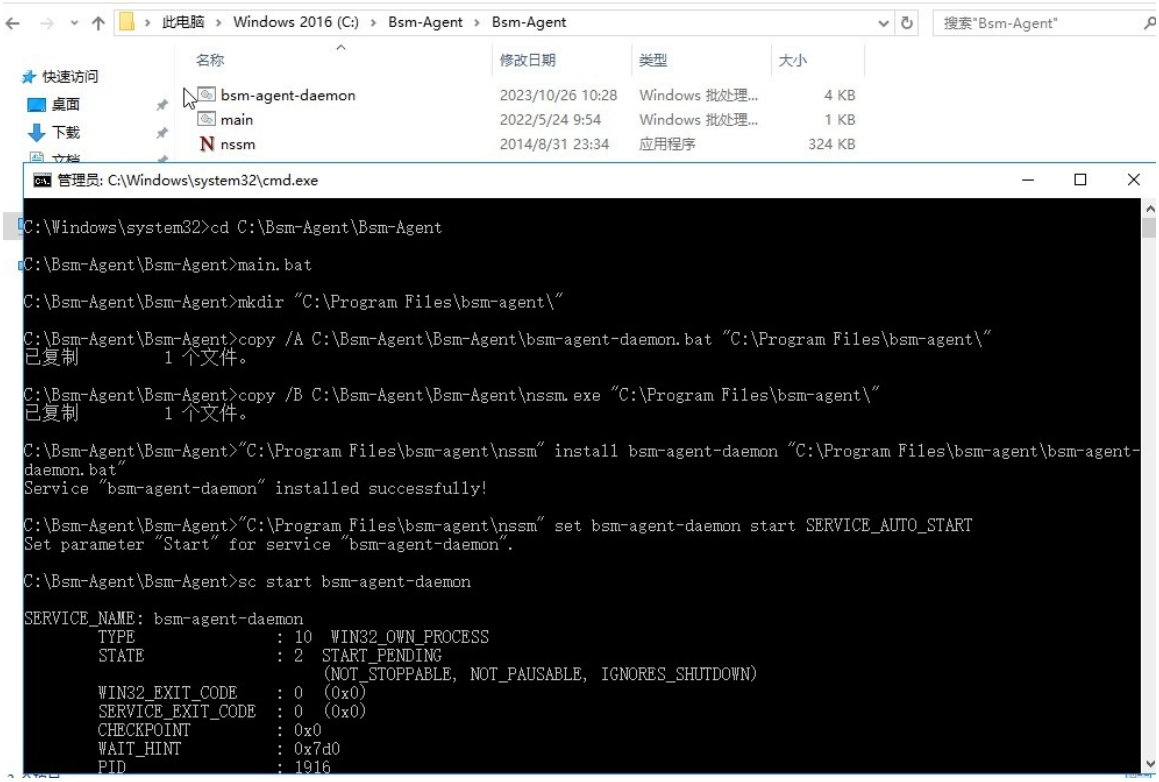
1、下载bsm-agent到您的云服务器；

2、解压文件后，cmd命令行进入该目录，执行安装命令并启动服务：

```
cd C:\Bsm-Agent\Bsm-Agent
```

```
main.bat
```

```
sc start bsm-agent-daemon
```



Windows系统安装Baidu HostEye（建议）

1、下载HostEye到您的云服务器；

2、双击安装文件，进行默认安装即可。

重装操作系统

前提条件

- 关机不计费实例不可重装操作系统。
- 系统盘或数据盘是变配优化中状态的不支持重装操作系统。

背景信息

当您遇到以下几种场景时，可使用重装操作系统操作：

- 需要将实例恢复至初始状态；
- 实例出现系统故障，无法正常启动；
- 需要更换新版本的操作系统。

云服务器BCC支持以下两种类型的重装：

- **同操作系统互换**，如：Linux系统更换为Linux系统。
- **不同操作系统互换**，如：Linux系统更换为Windows系统。

注意事项

重装操作系统会清除系统盘内数据且不可恢复（云磁盘数据不受影响），建议您提前为系统盘创建快照或者制作自定义镜像，以做好数据备份。重装系统后云服务器服务会中断几分钟，请做好相关的备份与服务切换工作，**谨慎选择重装操作系统**。

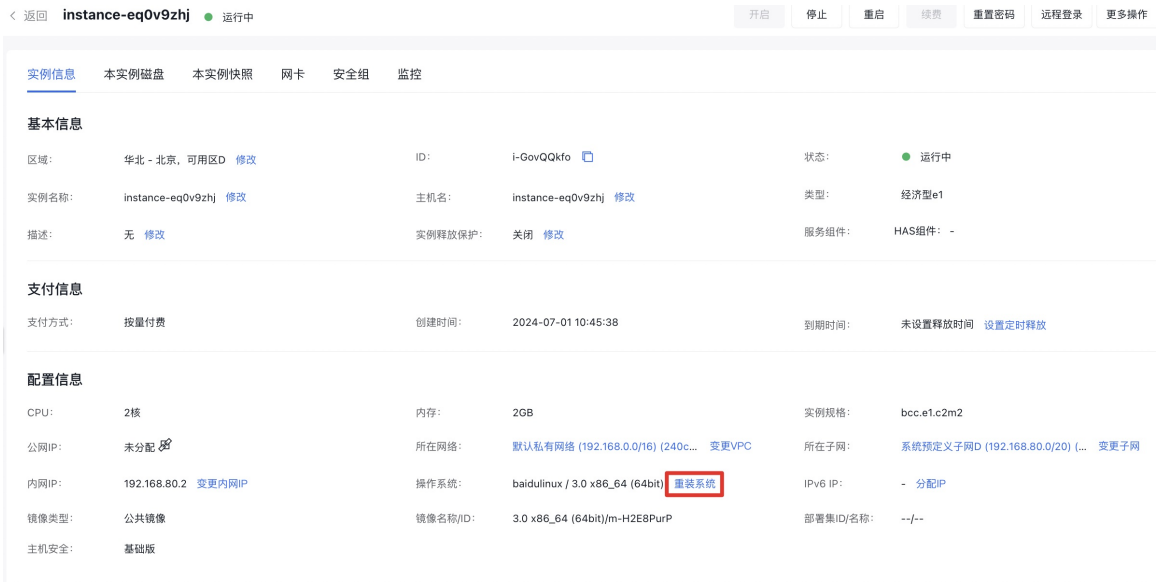
操作步骤

单个实例重装操作系统

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 您可通过以下种方式选择一个实例重装操作系统：

   方式一：在实例列表页选择要重装系统的实例，单击操作栏下的“更多 > 云盘和镜像 > 重装操作系统”，如下图所示。

方式二：在“实例名称”列，选择该云服务器的ID链接，进入“实例详情”页面。点击页面中的“重装操作系统”，进行系统的重装，如下图所示。



3. 在弹出的 “重装操作系统” 窗口中，选择您要使用的镜像类型及操作系统，设置密码，点击 **确定**。

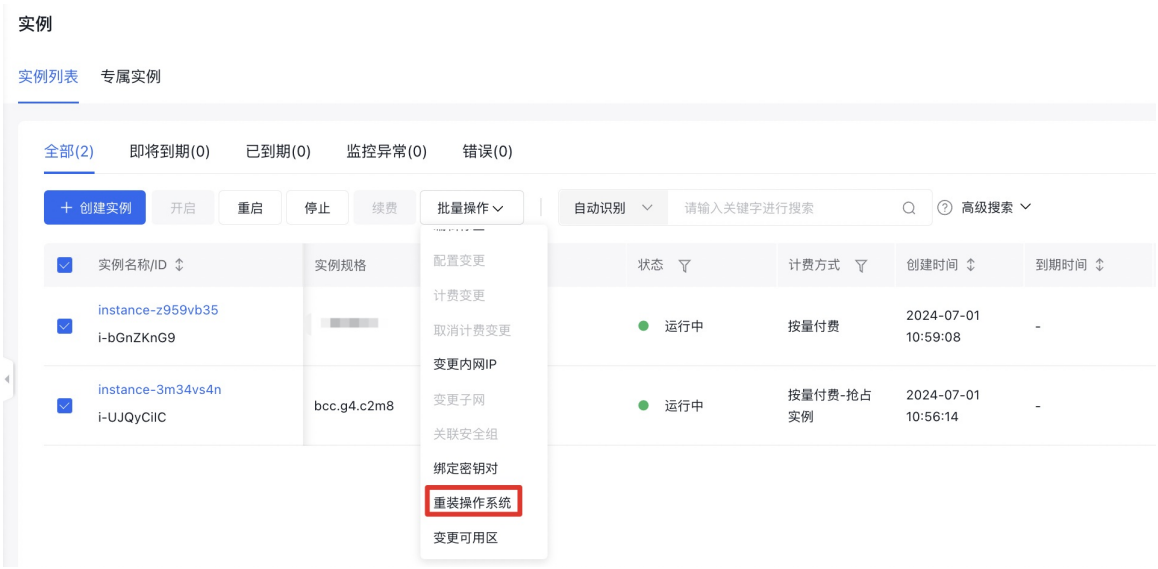
> **注意：**
>
> 1. 您可选择三种镜像，包括公共镜像、自定义镜像、服务集成镜像。
> 2. Windows Server 系统只支持密码验证，Linux 系统可以使用密码验证或SSH密钥对验证。

4. 点击确定后，实例状态将会变成 **重建中**。

多个实例重装操作系统

1. 登录 云服务器BCC控制台 进入实例列表页面。

2. 在实例列表页勾选您要重装系统的操作实例，点击“批量操作 - 重装操作系统”，如下图所示。



3. 在弹出的 “重装操作系统” 窗口中，选择您要使用的镜像类型及操作系统，设置密码，点击 **确定**。

> **注意：**
>
> 1. 您可选择三种镜像，包括公共镜像、自定义镜像、服务集成镜像。
>
> 2. Windows Server 系统只支持密码验证，Linux 系统可以使用密码验证或SSH密钥对验证。

4. 点击确定后，实例状态将会变成 **重建中**。

通过API重装操作系统

您可以使用 重装实例接口 为指定实例重装操作系统，也可以通过 批量重装实例接口 对多个实例重装操作系统。

基于本地系统创建linux镜像

应用场景

主要应用于用户期望在本地创建符合平台要求的自定义镜像。

本文介绍如何基于ubuntu16.04系统制作自定义镜像。

操作步骤

准备工作

由于百度智能云要求导入的镜像在格式、驱动等方面存在限制，请详细了解镜像导入限制，并在必要时提前配置系统，百度智能云提供了镜像检测工具可以帮助您快速检查相关的配置是否符合平台要求。

查找系统分区

完整的linux系统镜像至少需要包含mbr（主引导记录）、根分区（/目录挂载的地方，包含文件系统主体），否则将无法进入系统。

boot分区（挂载/boot目录，保存系统内核和其他启动过程需要的文件）、home分区（挂载/home目录，包含用户目录），如果包含这两个分区，也需要一并拷贝。

执行mount命令查看挂载状况：

```
    sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
   proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
   udev on /dev type devtmpfs (rw,nosuid,relatime,size=1002644k,nr_inodes=250661,mode=755)
   devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
   tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204784k,mode=755)
   /dev/sda1 on / type ext4 (rw,relatime,data=ordered)
   securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
   tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
   tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
   tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
   cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
   pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
   cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
   cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
   cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
   cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
   cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
   cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
   cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
   cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
   cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
   cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
   mqueue on /dev/mqueue type mqueue (rw,relatime)
   hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
   debugfs on /sys/kernel/debug type debugfs (rw,relatime)
   systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=35,pgrp=1,timeout=0,minproto=5,maxproto=5,direct)
   fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
   lxcfs on /var/lib/lxcfs type fuse.lxcfs (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
   tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=204784k,mode=700)
```

可以看出，并不包含boot、home分区，只需要拷贝mbr和根分区即可，即拷贝/dev/sda。

制作镜像

使用 VMWare vCenter Converter 或 Virtual box 等虚拟化平台的导出镜像工具，详情请参考各平台的导出工具文档。

使用 qemu-img 命令 或 使用 dd 命令方式导出镜像

1. 使用qmeu-img制作镜像：

如果没有qemu-img命令，则运行如下命令进行安装：

```
    apt-get update
   apt-get install qemu
```

执行以下命令生成镜像，其中test.qcow2为生成的镜像，镜像存储到新的磁盘或网络目录/mnt/test/下进行保存, -O指定输出镜像格式（可以根据需求选择格式类型，包括qcow2、vpc、vmdk、raw格式等，目前百度智能云只支持raw和qcow2格式镜像导入）。

```
`qemu-img convert -f raw -O qcow2 /dev/sda /mnt/test/test.qcow2`
```

操作耗时几分钟，完成后执行以下命令进行兼容性转换

`qemu-img amend -f qcow2 -o compat=0.10 test.qcow2`

使用qemu-img info查看镜像信息如下：

```
  image: test.qcow2
file format: qcow2
virtual size: 40G (42949672960 bytes)
disk size: 3.6G
cluster_size: 65536
Format specific information:
compat: 0.10
refcount bits: 16
```

2.使用dd命令制作镜像：

执行如下命令拷贝磁盘

```
dd if=/dev/sda of=/mnt/test/test.img bs=1M
```

以上命令为全盘拷贝，如果需要只拷贝某个分区，可以使用count参数进行指定：（其中bs的值*count的值=需要拷贝的大小）

```
dd if=/dev/sda of=/mnt/test/test.img bs=1M count=1024
```

需耗时几分钟，完成后使用qemu-img info查看镜像信息：

```
 image: test.img
file format: raw
virtual size: 40G (42949672960 bytes)
disk size: 40G
```

格式转换

当前百度智能云支持raw、qcow2格式的镜像导入，如使用其他格式如vmdk、vpc等，请参照格式转换进行转换。

后续操作

生成的镜像即可以导入百度智能云，进一步创建实例。

镜像快速指南

公共镜像快速指南

本文为基于LAMP的WordPress教程，从零开始带领您创建BCC实例，配置基础的LAMP (Linux+Apache+Mysql+PHP) 服务，而后基于此构建您的第一个WordPress博客。

1. 登录百度智能云官网。
   - 未注册，须先注册百度账号。
   - 已注册，直接登录。
2. 完成实名认证。

   您可根据自身情况，进行“企业认证”或者“个人认证”。对于“包年包月实例”，当购买公网带宽大于0时，需要实名认证。对于“按量付费实例”，则必须完成实名认证。

   如果您在创建BCC实例的时候仍未完成实名认证，可单击页面上的**认证**提示按钮到实名认证页面完成操作。
3. 创建BCC实例。

   BCC实例分为包年包月和按量付费两种创建方式，详见创建BCC实例。

   Linux镜像请选择Ubuntu 14.04.1 LTS (64 bit)。创建成功后，返回实例列表，获取公网IP。
4. 登录BCC。

打开远程登录工具，IP Name框中输入公网IP并用root用户登录。

5. 配置Apache。

```
 # apt-get install apache2       //安装apache
# vim /etc/apache2/apache2.conf
 //添加 ServerName localhost
# service apache2 restart       //启动apache服务
# chmod a+w /var/www            //赋予当前用户写权限
```

6. 配置PHP和Mysql。

```
 # apt-get install php5 php5-mysql
# apt-get install mysql-server
# mysql -u root -p
> mysql
# exit
```

7. 配置Wordpress。

```
 # cd /var/www/html
# wget http://dlsw.baidu.com/sw-search-sp/soft/fd/26373/wordpress_V4.2.2_zh_CN.1436773043.zip
# apt-get install unzip
# unzip
# ls /var/www/html/wordpress
 index.php        wp-admin             wp-content        wp-load.php      wp-signup.php
 license.txt      wp-blog-header.php   wp-cron.php       wp-login.php     wp-trackback.php
 readme.html      wp-comments-post.php wp-includes       wp-mail.php      xmlrpc.php
 wp-activate.php  wp-config-sample.php wp-links-opml.php wp-settings.php
```

8. 打开"<公网IP>/wordpress"浏览WordPress博客。

服务集成镜像快速指南

主机宝CentOS6.5（64位）版

服务提供商：徐州八方网络科技有限公司

操作系统：CentOS6.5 64位

集成软件：Nginx1.4.7、MySQL5.1.73、PHP5.2.17/PHP5.4.24、PHPMyAdmin4.0.10、PureFtpd1.0.36、Memcached1.4.18、主机宝Linux版2.1.3

功能亮点：

1. 系统预装主机宝管理面板，可快速灵活部署安全的WEB环境，轻松管理和备份站点、数据库、FTP等。

2. 针对网站环境和系统安全设置进行优化，提升访问速度和负载能力，支持PHP5.2到PHP5.4的一键切换。

3. 7x24全天候在线售后支持全力为您的应用保驾护航！

售后说明：

邮箱：baidubce@admin5.com （7×24小时） 电话：13305205345 （7×24小时）

使用文档：

主机宝CentOS6.5（64位）建站操作文档

> **说明**：

> 使用主机宝CentOS镜像时，需要在安全组入站规则中增加TCP 21、TCP 80、TCP 9999端口 和TCP 35000-50000 端口，配置方法可参考使用文档或参考安全组配置文档。

开源软件版权声明：

徐州八方开源软件版权声明

百度智能云法律声明：

百度智能云对商户提供使用的镜像不拥有任何权利，其知识产权归该镜像合法拥有者所有。更多声明

主机宝Windows Server 2008（64位）版

服务提供商：徐州八方网络科技有限公司

操作系统：Windows Server 2008 64位

集成软件：ASP、.NET2.0/3.5/4.0、PHP5.2.17/PHP5.4.21、MySQL5.1.73、PHPMyAdmin4.0.10、FileZilla 0.9.41、主机宝2.0.9

功能亮点：

1.系统预装主机宝管理面板，可快速灵活部署安全的WEB环境，轻松管理和备份站点、数据库、FTP等。 2.针对网站环境和系统安全设置进行优化，提升访问速度和负载能力，支持PHP5.2到PHP5.4的一键切换。 3.7x24全天候在线售后支持全力为您的应用保驾护航！

售后说明：

邮箱：baidubce@admin5.com （7×24小时） 电话：13305205345 （7×24小时）

使用文档：

主机宝Windows Server 2008（64位）建站操作文档

> **说明：**
>
> 使用主机宝Windows镜像时，需要在安全组入站规则中增加TCP 21、TCP 80和TCP 6801-6802 端口，配置方法可参考使用文档或参考安全组配置文档。

开源软件版权声明：

徐州八方开源软件版权声明

百度智能云法律声明：

百度智能云对商户提供使用的镜像不拥有任何权利，其知识产权归该镜像合法拥有者所有。 更多声明

基于CentOS6.5（64位）的LAMP环境

服务提供商：百度智能云

操作系统：CentOS6.5 64位

集成软件：Apache 2.2.15、MySQL 5.1.73、PHP 5.3.3、vsftp 2.2.2

功能亮点：

1.百度智能云官方提供，安全稳定。 2.系统环境采用源码编译安装，集成FTP服务便于用户上传站点文件。 3.购买完成后，用户可参照使用文档轻松创建站点，避免繁杂的配置工作。

售后说明：

由百度智能云提供技术支持

使用文档：

基于CentOS6.5（64位）的LAMP环境使用说明

> **说明**：
>
> - 使用基于CentOS6.5（64位）的LAMP环境镜像时，需要在安全组入站规则中增加TCP 21和TCP 80端口，配置方法可参考使用文档或参考安全组配置文档。
> - 本镜像中的LAMP环境使用官方yum源默认软件版本安装，当特定软件存在安全漏洞而官方未更新yum源时，请您随时关注百度智能云官网安全公告并进行升级修复。

开源软件版权声明：

百度智能云开源软件版权声明

百度智能云法律声明：

百度智能云对商户提供使用的镜像不拥有任何权利，其知识产权归该镜像合法拥有者所有。更多声明

基于CentOS6.5（64位）的WordPress环境

服务提供商：百度智能云

操作系统：CentOS6.5 64位

集成软件：Apache 2.2.15、MySQL 5.1.73、PHP 5.3.3、vsftp 2.2.2、phpMyAdmin 4.0.10、WordPress 3.9

功能亮点：

1.百度智能云官方提供，安全稳定。 2.系统预装WordPress博客平台程序，用户可参照文档轻松进行博客平台的配置和开通。 3.默认安装了phpmyadmin管理平台，方便管理数据库。

售后说明：

由百度智能云提供技术支持

使用文档：

基于CentOS6.5（64位）的WordPress环境使用说明

> **说明**：
>
> - 使用基于CentOS6.5（64位）的WordPress环境镜像时，需要在安全组入站规则中增加TCP 21和TCP 80端口，配置方法可参考使用文档或参考安全组配置文档。
> - 本镜像中的LAMP环境使用官方yum源默认软件版本安装，当特定软件存在安全漏洞而官方未更新yum源时，请您随时关注百度智能云官网安全公告并进行升级修复。

开源软件版权声明：

百度智能云开源软件版权声明

百度智能云法律声明：

百度智能云对商户提供使用的镜像不拥有任何权利，其知识产权归该镜像合法拥有者所有。更多声明

镜像列表

公共镜像

注意： 百度智能云会基于开源社区的最新发布不断更新公共镜像，装有老镜像的BCC实例若要使用新特性，需重装系统。

操作系统类型	操作系统版本
BaiduLinux	3.0 x86_64 (64bit) 2.0 x86_64 (64bit)
CentOS	8.4 x86_64 (64bit) 8.3 x86_64 (64bit) 8.2 x86_64 (64bit) 8.1 x86_64 (64bit) 8.0 x86_64 (64bit) 7.9 x86_64 (64bit) 7.8 x86_64 (64bit) 7.7 x86_64 (64bit) 7.6 x86_64 (64bit) 7.5 x86_64 (64bit) 7.4 x86_64 (64bit) 7.3 x86_64 (64bit) 7.2 x86_64 (64bit) 7.1 x86_64 (64bit) 6.8 x86_64 (64bit) 6.5 x86_64 (64bit) 6.5 i386 (32bit) 8.5 aarch64 (64bit) 7.9 aarch64 (64bit) 7.6 aarch64 (64bit)
AlmaLinux	9.2 x86_64 (64bit) 9.0 x86_64 (64bit) 8.8 x86_64 (64bit) 8.6 x86_64 (64bit)
Debian	12.0.0 amd64 (64bit) 11.7.0 amd64 (64bit) 11.1.0 amd64 (64bit) 11.0.0 amd64 (64bit) 10.10.0 amd64 (64bit) 10.9.0 amd64 (64bit) 10.8.0 amd64 (64bit) 10.7.0 amd64 (64bit) 10.6.0 amd64 (64bit) 10.5.0 amd64 (64bit) 10.4.0 amd64 (64bit) 10.3.0 amd64 (64bit) 9.13.0 amd64 (64bit) 9.9.0 amd64 (64bit) 9.1.0 amd64 (64bit)
	9 x86_64 (64bit)

CentOS Stream	9 x86_64 (64bit) 8 x86_64 (64bit)
Fedora	32 x86_64 (64bit)
FreeBSD	11.4 amd64 (64bit)
OpenSUSE	15.2 x86_64 (bit) 42.3 x86_64 (64bit)
Rocky Linux	9.4 x86_64 (64bit) 9.3 x86_64 (64bit) 9.2 x86_64 (64bit) 9.1 x86_64 (64bit) 9.0 x86_64 (64bit) 8.8 x86_64 (64bit) 8.7 x86_64 (64bit) 8.6 x86_64 (64bit) 8.5 x86_64 (64bit)
Ubuntu	24.04 LTS amd64 (64bit) 22.04 LTS amd64 (64bit) 20.04 LTS amd64 (64bit) 18.04 LTS amd64 (64bit) 16.04 LTS amd64 (64bit) 16.04 LTS i386 (32bit) 14.04.1 LTS amd64 (64bit) 14.04.1 LTS i386 (32bit) 12.04.4 LTS amd64 (64bit) 12.04.4 LTS i386 (32bit) 22.04 LTS aarch64 (64bit) 20.04 LTS aarch64 (64bit)
Windows Server	2022 x86_64 (64bit) 中文版 2022 x86_64 (64bit) 英文版 2019 x86_64 (64bit) 中文版 2019 x86_64 (64bit) 英文版 2016 x86_64 (64bit) 中文版 2016 x86_64 (64bit) 英文版 2012 R2 x86_64 (64bit) 中文版 2012 R2 x86_64 (64bit) 英文版 2008 R2 x86_64 (64bit) 中文版 2008 R2 x86_64 (64bit) 英文版 1909 x86_64 (64bit) 中文版 1909 x86_64 (64bit) 英文版

服务集成镜像

- 主机宝CentOS 6.5（64位）版
- 基于CentOS 6.5（64位）的LAMP环境
- 基于CentOS 6.5（64位）的WordPress环境
- Access VPN CentOS 6.5（64位）版

使用方法请参考镜像快速指南。

云市场镜像

详见云市场。

windows镜像制作

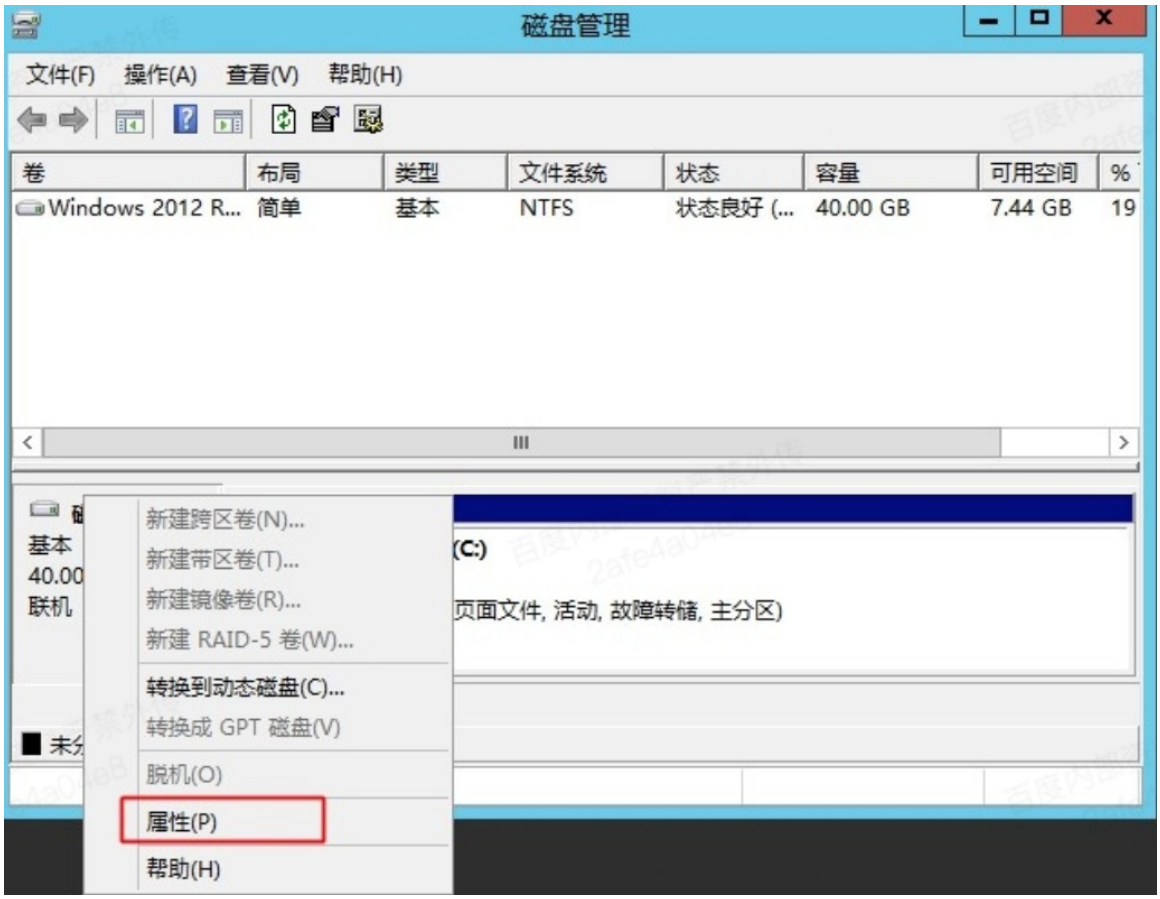
本文以Windows Server 2012 R2操作系统为例，介绍适合云服务器BCC的Windows镜像制作过程。

可行性检查

检查系统分区

1. 命令行 或 PowerShell 中，输入 diskmgmt.msc，打开“磁盘管理”工具；

2. 打开磁盘属性窗口；



3. 检查卷的磁盘分区形式；

4. 目前只支持MBR分区，不支持GPT分区；

准备工作

卸载软件

1. 卸载电脑管家、安全类软件，以防导致系统蓝屏或系统不稳定；

2. 卸载有冲突的驱动（比如VMware tools，Xen tools等）。

安装cloudbase-init

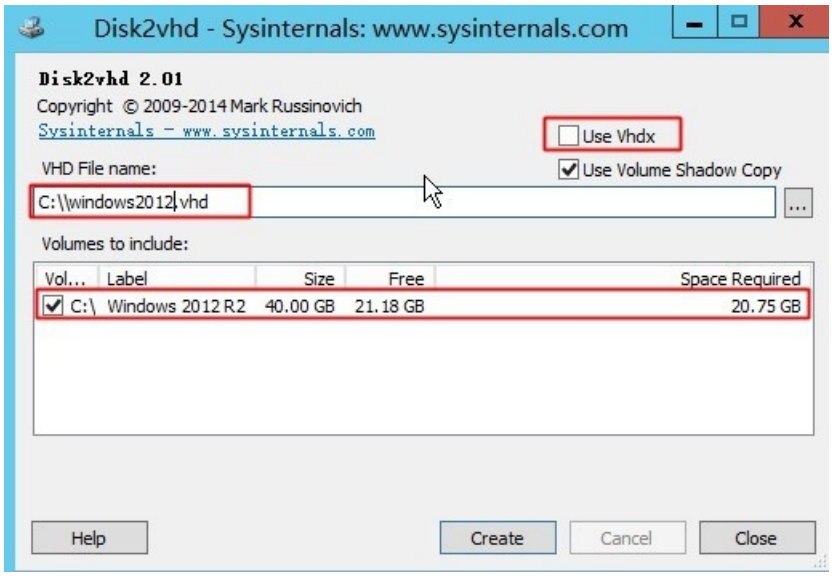
详情请参考 安装cloudbase-init

导出镜像

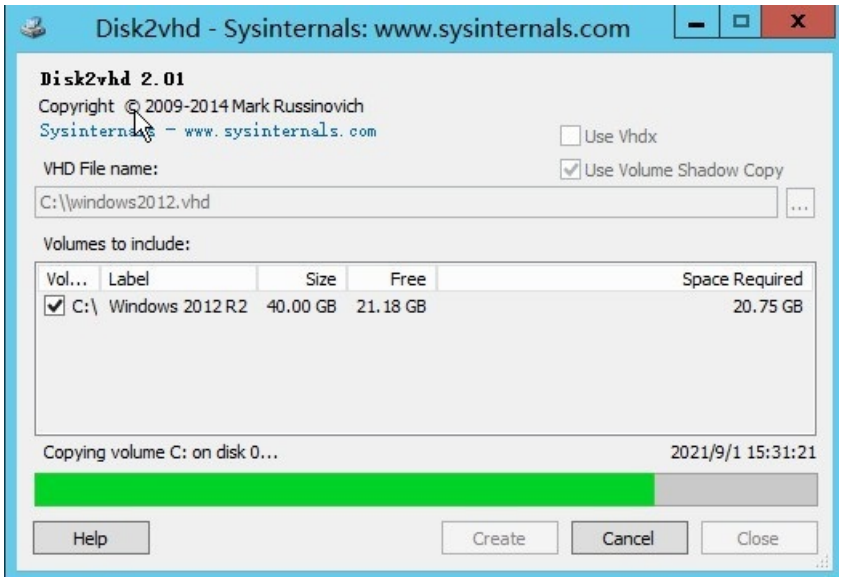
微软官方有镜像导出工具disk2vhd.exe可以使用。

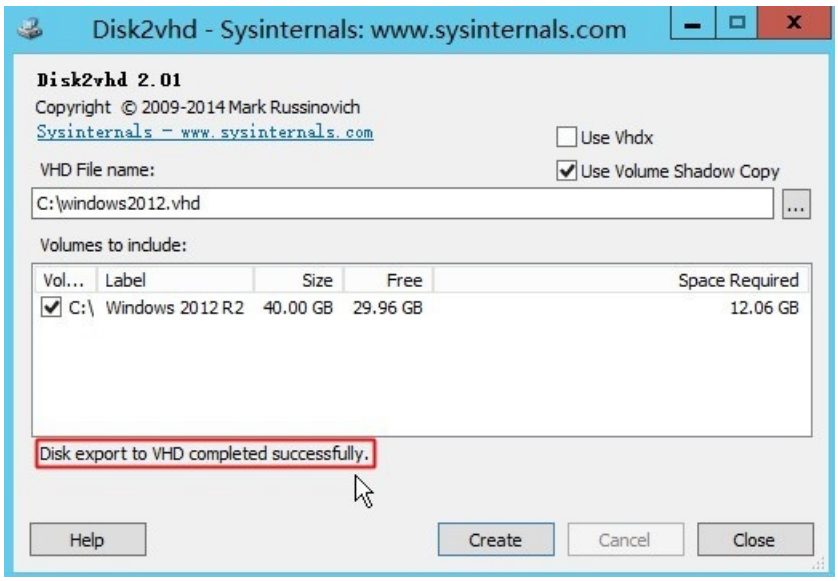
1. 下载并运行disk2vhd.exe；

2. 选择系统分区所在卷，并填写导出镜像路径；



3. 点击“Create”开始提取镜像，等待完成；

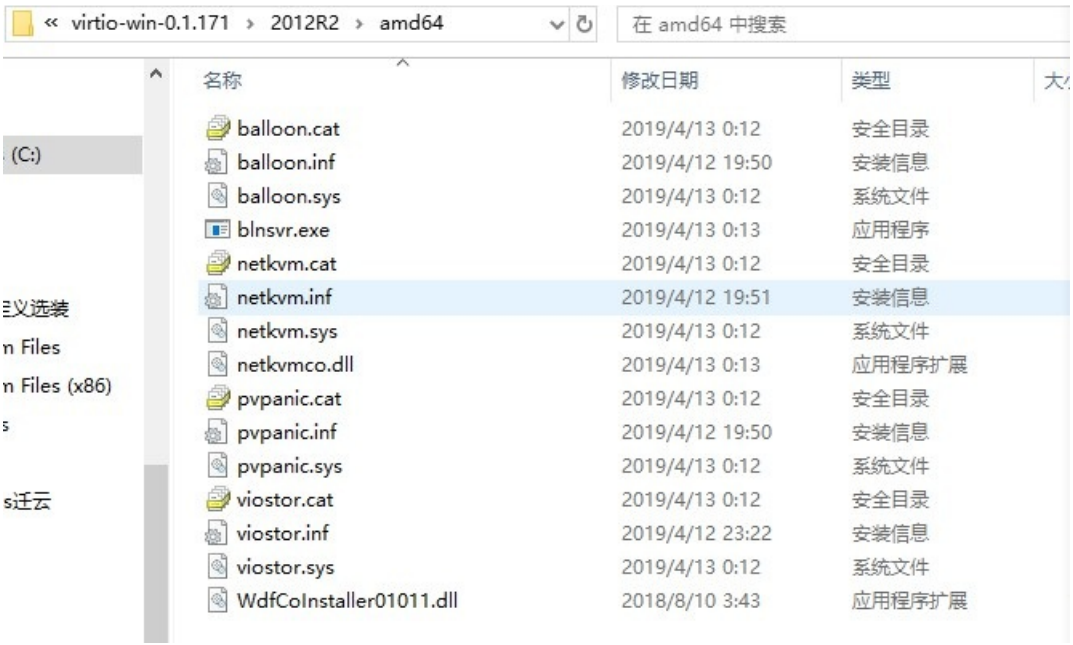




安装VirtIO驱动

1. 下载virtio驱动文件并解压；

根据当前操作系统类型，选择驱动版本，比如Windows Server 2012 R2的驱动文件在C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64；



2. 挂载镜像并离线安装驱动；

1）源系统为Windows Server 2012 R2、2016、2019、2022时，可以直接在源系统上打开cmd命令窗口，依次执行以下命令：

```
mkdir c:\mount
```

```
dism /mount-image /imagefile:c:\windows2012.vhd /index:1 /mountdir:c:\mount
```

```
dism /image:c:\mount /add-driver /driver:C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64
```

```
dism /unmount-image /mountdir:c:\mount /commit
```

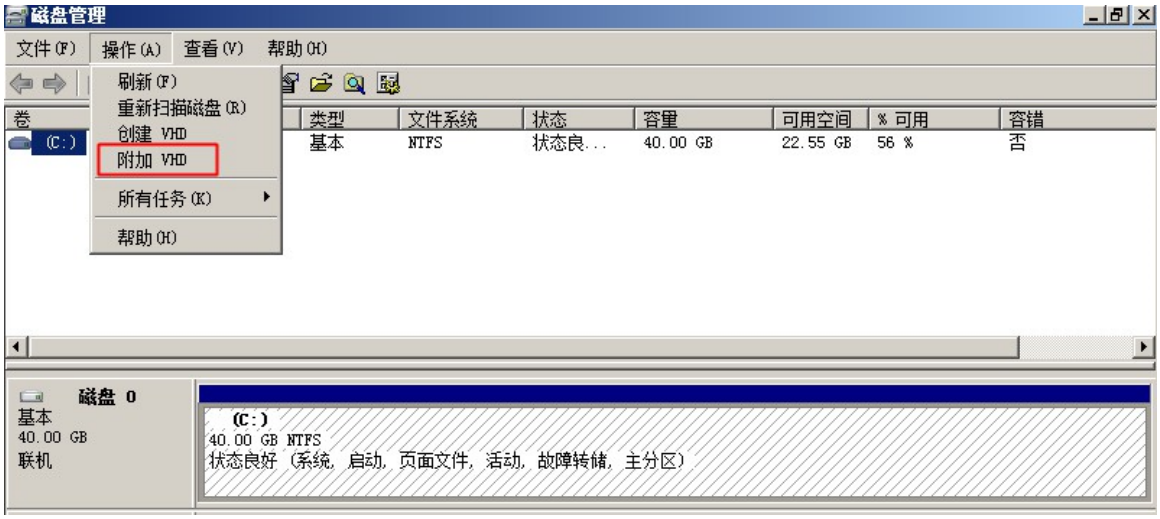
```
C:\Users\Administrator>mkdir c:\mount

C:\Users\Administrator>dism /mount-image /imagefile:c:\windows2012.vhd /index:1
/mountdir:c:\mount

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Mounting
[==========================100.0%==========================]
The operation completed successfully.

C:\Users\Administrator>dism /image:c:\mount /add-driver /driver:C:\virtio-win-0.
1.171\virtio-win-0.1.171\2012R2\amd64

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Image Version: 6.3.9600.17031

Searching for driver packages to install...
Found 4 driver package(s) to install.
Installing 1 of 4 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64\balloo
n.inf: The driver package was successfully installed.
Installing 2 of 4 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64\netkvm
.inf: The driver package was successfully installed.
Installing 3 of 4 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64\pvpani
c.inf: The driver package was successfully installed.
Installing 4 of 4 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2012R2\amd64\viosto
r.inf: The driver package was successfully installed.
The operation completed successfully.

C:\Users\Administrator>dism /unmount-image /mountdir:c:\mount /commit

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Unmounting
[==========================100.0%==========================]
The operation completed successfully.
```

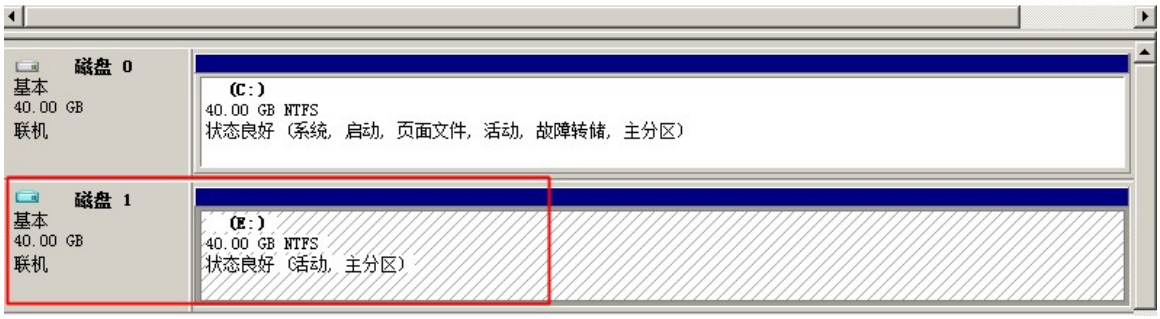
2）源系统为Windows Server 2008 R2时，需要先将vhd文件拷贝到另一台Windows虚机上（中转系统）；

如果中转系统是Windows Server 2012 R2、2016、2019、2022时，可参考1）直接执行命令；

如果中转系统也是Windows Server 2008 R2时，可通过“磁盘管理”工具，附加生成的VHD镜像文件（因为需要安装驱动，所以

以非只读形式附加）：





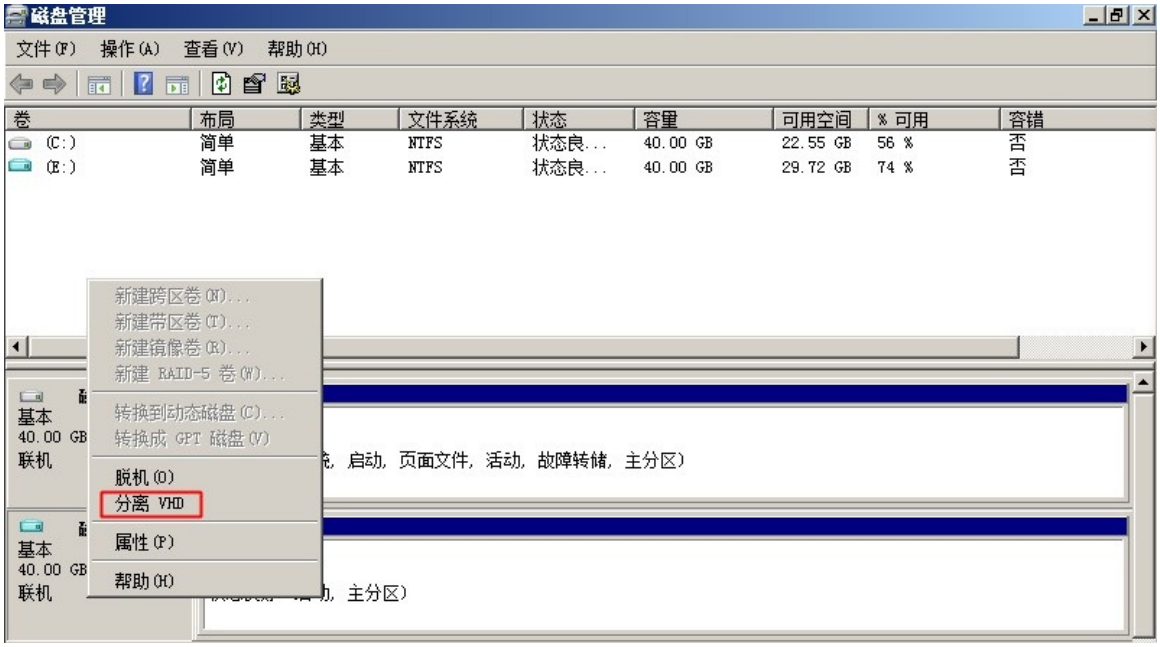
使用DISM工具离线安装驱动：

dism.exe /Image:{vhd镜像所在分区} /add-driver /driver:{驱动路径} /recurse /forceunsigned

例如：dism.exe /Image:e: /add-driver /driver:C:\virtio-win-0.1.171\virtio-win-0.1.171\2008R2\amd64 /recurse /forceunsigned

```
C:\Users\Administrator>dism /image:e: /add-driver /driver:C:\virtio-win-0.1.171\
virtio-win-0.1.171\2008R2\amd64 /recurse /forceunsigned

部署映像服务和管理工具
版本: 6.1.7600.16385

映像版本: 6.1.7600.16385

正在搜索要安装的驱动程序包...
找到 4 个要安装的驱动程序包。
正在安装第 1 个，共有 4 个 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2008R2\amd
64\balloon.inf: 驱动程序包已成功安装。
正在安装第 2 个，共有 4 个 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2008R2\amd
64\netkvm.inf: 驱动程序包已成功安装。
正在安装第 3 个，共有 4 个 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2008R2\amd
64\pvpanic.inf: 驱动程序包已成功安装。
正在安装第 4 个，共有 4 个 - C:\virtio-win-0.1.171\virtio-win-0.1.171\2008R2\amd
64\viostor.inf: 驱动程序包已成功安装。
操作成功完成。
```

分离VHD镜像盘：



3. 至此，镜像制作完成；

> 目前BCC支持VHD格式镜像导入，如果想做格式转换，请参考 镜像格式检查 和 镜像格式转换 。

如何设置网卡多队列属性

背景

随着计算机网络带宽不断提升，单个CPU处理网络中断存在瓶颈，不能完全满足网卡的需求，在BCC云服务器存在多个CPU的情况下，通过开启网卡多队列，可以将云服务器中的网卡中断分散给不同的CPU进行处理，从而提升网络PPS和带宽性能。

支持网卡多队列的镜像列表

网卡多队列的支持情况和实例规格、镜像的操作系统有关。

镜像	是否支持多队列	是否默认开启多队列
CentOS 6.5/6.8/7.*/8.*	是	是
CentOS Stream 8/9	是	是
Rocky Linux 8.*/9.*	是	是
AlmaLinux 8.*/9.*	是	是
Ubuntu 14.04/16.04/18.04/20.04/22.04/24.04	是	是
Debian 8.*/9.*/10.*/11.*/12.*	是	是
OpenSUSE 42.3/15.*	是	是
Windows server 2012及以上	是	是

手动配置网卡多队列

查看网卡队列数

```
 [root@instance-0wazpxeh ~]# ethtool -l eth0  #查询网卡eth0的队列数
Channel parameters for eth0:
Pre-set maximums:
RX:             0
TX:             0
Other:          0
Combined:       4  #表示此网卡最多支持设置开启4个队列
Current hardware settings:
RX:             0
TX:             0
Other:          0
Combined:       4 #表示当前开启的是4个队列
```

如果查询结果中，两个Combined字段取值相同，则表示弹性网卡已开启支持多队列。

否则，使用以下命令开启多队列：

ethtool -L eth0 combined 4

自动配置网卡多队列

以centos 7 为例，创建网卡多队列服务：

新建 /usr/lib/systemd/system/virt-net-init.service

```
 usr/lib/systemd/system/virt-net-init.service
[Unit]
Description=Virt IO Network Init
After=local-fs.target
Before=network-pre.target

[Service]
Type=oneshot
ExecStart=/usr/libexec/virt-net-init.sh
RemainAfterExit=True

[Install]
WantedBy=multi-user.target
```

创建网卡多队列配置服本：

新建/usr/libexec/virt-net-init.sh

```
 ##### !/bin/bash

rps_flow_cnt=4096
##### get specified mask
function get_specified_cpumask(){
    local cpu_num=$1
    quotient=$((${cpu_num}/32))
    remainder=$((${cpu_num}-32*$quotient))
    if [ ${quotient} -gt 0 ]; then
        res_tail=""
        res_head="80000000"
        while [ $quotient -gt 1 ]
        do
            res_tail="${res_tail},00000000"
            ((quotient--))
        done
        if [ $remainder -ne 0 ];then
            res_tail="${res_tail},00000000"
            res_head=$((1<<($remainder-1)))
            res_head=`printf "%x" ${res_head}`
        fi
        result="${res_head}${res_tail}"
    else
        result=$((1<<(${cpu_num}-1)))
        result=`printf "%x" $result`
    fi
    echo $result
}

##### get all mask for rps/xps
function get_all_cpus() {
    local cpu_nums=$(grep -c processor /proc/cpuinfo)
    quotient=$((${cpu_nums}/32))
    remainder=$((${cpu_nums}-32*$quotient))
    if [ ${quotient} -gt 0 ];then
        res_head="ffffffff"
        res_tail=""
        while [ $quotient -gt 1 ]
        do
            res_tail="${res_tail},ffffffff"
            ((quotient--))
        done
        if [ $remainder -ne 0 ];then
            res_tail="${res_tail},ffffffff"
            res_head=$(((1<<$remainder)-1))
            res_head=`printf "%x" ${res_head}`
        fi
        result="${res_head}${res_tail}"
    else
        result=$(((1<<${cpu_nums})-1))
        result=`printf "%x" $result`
    fi
    echo $result
}

##### set xps
function set_xps() {
    dev=$1
    for xps_file in `ls /sys/class/net/$dev/queues/tx-*/xps_cpus`
    do
        xps_cpus=$(get_all_cpus)
        echo "[INFO] set ${xps_cpus} into ${xps_file}." | logger -i -t 'virt-net-init'
```

```
        echo ${xps_cpus} > ${xps_file}
    done
}

##### set rps/rfs for multicore when multiqueue is disbale
function set_rps_and_rfs(){
    rps_cpus=$(get_all_cpus)
    dev=$1
    queues=`ls -ld /sys/class/net/$dev/queues/rx-* | wc -l`
    num=0
    while [ $num -lt $queues ]
    do
        echo ${rps_cpus} > /sys/class/net/$dev/queues/rx-$num/rps_cpus
        echo "[INFO] set ${rps_cpus} into /sys/class/net/$dev/queues/rx-$num/rps_cpus." | logger -i -t 'virt-net-init'
        echo ${rps_flow_cnt} > /sys/class/net/$dev/queues/rx-$num/rps_flow_cnt
        echo "[INFO] set ${rps_flow_cnt} into /sys/class/net/$dev/queues/rx-$num/rps_flow_cnt." | logger -i -t 'virt-net-init'
        ((num++))
    done

}

function set_smp_affinity(){
    local dev=$1
    pci_dbdf=$(ethtool -i $dev | grep bus-info | cut -d' ' -f2)
    if [ -z $pci_dbdf ]; then
        echo "[ERR] No LiquidIO NIC detected" | logger -i -t 'virt-net-init'
    else
        dir=$(find /sys/devices/ -type d | grep $pci_dbdf | grep "/net$")
        result=${dir%/*}
        # get virtioX
        virtio_num=${result##*/}

        cpunums=`cat /proc/cpuinfo | grep processor | wc -l`
        if [ ${cpunums} -lt 2 ];then
            echo "[INFO] don't need set affinity for net interrupt!!" | logger -i -t 'virt-net-init'
        else
            irq_input=$(grep ${virtio_num}-input /proc/interrupts | sed "s/: .*//g" | sed "s/^ *//g")
            cur_cpunum=0
            for irqnum in ${irq_input}; do
                num=$((${cur_cpunum}%$cpunums+1))
                mask=`get_specified_cpumask $num`
                #echo "[INFO] irq:${irqnum} bind to cpu:$mask." | logger -i -t 'virt-net-init'
                echo "[INFO] echo $mask > /proc/irq/$irqnum/smp_affinity" | logger -i -t 'virt-net-init'
                echo $mask > /proc/irq/$irqnum/smp_affinity
                ((cur_cpunum++))
            done

            irq_output=$(grep ${virtio_num}-output /proc/interrupts | sed "s/: .*//g" | sed "s/^ *//g")
            cur_cpunum=0
            for irqnum in ${irq_output}; do
                num=$((${cur_cpunum}%$cpunums+1))
                mask=`get_specified_cpumask $num`
                #echo "[INFO] irq:${irqnum} bind to cpu:$mask." | logger -i -t 'virt-net-init'
                echo "[INFO] echo $mask > /proc/irq/$irqnum/smp_affinity" | logger -i -t 'virt-net-init'
                echo $mask > /proc/irq/$irqnum/smp_affinity
                ((cur_cpunum++))
            done
        fi
    fi
}

##### set multiqueue
```

```
function set_multiqueue() {
    local dev=$1
    local pre_set=`ethtool -l ${dev} | grep Combined | head -n 1 | cut -f 2`
    local cur_set=`ethtool -l ${dev} | grep Combined | tail -n 1 | cut -f 2`
    #virtio-net will enable multiqueue by default
    if [ ${pre_set} -gt ${cur_set} ];then
        ethtool -L ${dev} combined ${pre_set} 2>&1 | logger -i -t 'virt-net-init'
        if [ "${PIPESTATUS[0]}" -ne 0 ]; then
            echo "[WARN] set multiqueue[chnum = ${cur_set}] for $dev failed. Ignore this if multiqueue is already set or
chnum = 1." | logger -i -t 'virt-net-init'
        fi
    else
        echo "[INFO] multiqueue[chnum= ${cur_set}] for $dev is enabled." | logger -i -t 'virt-net-init'
    fi
}

function main(){
    for (( i = 0; i < 5; i++ )); do
        dir_list=$(find /sys/devices/ -type d | grep virtio | grep "/net$")
        if [[ ! -z "${dir_list}" ]]; then
            break
        fi
        sleep 0.5
    done

    total_queues=0
    for dir in ${dir_list[*]}; do
        dev=$(ls $dir)
        if [[ -n "$dev" ]]; then
            #check if multiqueue is available
            queue_nums=`ethtool -l $dev 2>&- | grep "Combined" | head -n 1 | awk '{print $2}'`
            if [ "$?" == "0" ] && [ ${queue_nums} -gt 1 ];then
                #set multiqueue and affinity
                #echo "set multiqueue for ${dev},and rps/rfs don't need to set."
                set_multiqueue ${dev}
                set_smp_affinity ${dev}
                #continue
            else
                #set rps and rfs
                set_rps_and_rfs ${dev}
                queues=`ls -ld /sys/class/net/$dev/queues/rx-* | wc -l`
                total_queues=$((${total_queues}+$queues))
                set_xps ${dev}
            fi
        else
            echo "[WARN] net device isn't exist." | logger -i -t 'virt-net-init'
        fi
    done

    if [ ${total_queues} -ne 0 ];then
        rps_sock_flow_entries=$((total_queues*${rps_flow_cnt}))
        echo ${rps_sock_flow_entries} > /proc/sys/net/core/rps_sock_flow_entries
        echo "[INFO] set ${rps_sock_flow_entries} for rfs." | logger -i -t 'virt-net-init'
    fi
}
main
```

为多队列配置脚本添加可执行权限：

chmod + x /usr/libexec/virt-net-init.sh

增加网卡多队列配置开机自动启动：

systemctl enable virt-net-init.service | logger -i -t 'virt-net-init'

百度智能云镜像服务协议

在您使用百度智能云镜像服务之前，请您仔细阅读本服务协议的全部内容。您使用本服务的任何行为，即视为您已阅读并同意下列协议、规则等的约束。

一、协议的主体、生效和适用范围

1.1 本协议是百度智能云与镜像服务商（以下简称“镜像商”或“服务商”，及使用服务商镜像的用户（以下简称“用户”）之间的具有法律效力的有效合约。

1.2 当用户浏览本网站镜像服务的内容时，百度智能云将通过网页提醒并有效指引用户阅读本服务条款的全部内容。

1.3 本协议内容包括协议正文以及所有百度智能云就镜像服务事宜已经发布的或未来可能发布的各类业务规范、活动规则、通知、公告、说明等（以下统称为“规则”）。所有规则为协议不可分割的组成部分，与协议正文具有同等法律效力。

1.4 当用户使用百度智能云镜像服务目前或将来提供的特定服务项目时，用户同时应接受适用于那些服务的条款、准则和条件（下称“特殊条款”），百度智能云将以书面协议、网页提示、网站公告等形式向用户明示这些特殊条款。如本协议与特殊条款的约定不一致的，则以特殊条款的约定为准。

1.5 用户理解并同意，镜像服务提供仅是百度智能云网站向您提供的所有服务中的其中一类。因此，用户在使用镜像服务提供的同时，还应遵守百度智能云网站服务条款的其他规则和规定。

1.6 百度智能云有权根据需要，不定时制定、修改本协议和/或各类规则，且有权将本协议项下的权利义务转让给第三方，而无须获得用户的事先同意。如有变更，百度智能云将以网站公示、邮件通知或其他适当的方式进行告知。变更后的协议和规则，及/或权利义务转让，一经公布即自动生效。如用户不同意相关变更及/或转让，应当立即停止使用镜像服务；如用户继续使用镜像服务的，则视为用户接受修改后的协议和规则及/或转让。

1.7 服务商与用户之间，因用户订阅使用服务商所提供的镜像的行为而产生的法律关系及法律纠纷，由服务商与用户自行处理，不属于本协议的约定范围。

二、定义

除非本协议中另有定义，下列词语在本协议中均应具有以下特定涵义：

2.1 百度智能云镜像服务：由百度智能云网络平台建立，可向用户提供特定镜像的服务。

2.2 镜像服务商（或称“镜像商”、“服务商”、“合作伙伴”）：是指在百度智能云镜像服务这一网络平台上提供其镜像及服务的合作伙伴。

2.3 镜像服务用户（或称“用户”）：是指通过百度智能云使用镜像服务的个人，法人或者其他组织。（简称用户）

2.4镜像：镜像是一种包含软件配置的模板，其中集成了启动实例所需的软件配置（操作系统、应用程序服务器和应用程序）。镜像可以包含基础的操作系统，也可以在此基础上整合具体的软件环境。

2.5服务集成镜像：基于基础操作系统并集成有通用服务和应用软件环境的镜像。由百度或第三方服务商提供，经由百度智能云审核后发布。服务集成镜像可帮助用户快速创建包含控制面板、PHP等常用运行环境的云服务器。

2.6云服务器实例：是基于百度智能云创建的虚拟计算环境。用户可以从单一镜像启动不同硬件配置的实例，并基于实例运行用户所需要的应用程序。用户对其启动的实例有完全的控制权。

三、百度智能云镜像服务内容

3.1 百度智能云为用户提供基于镜像服务的各项网络和技术服务，包括但不限于：包含特定软件环境的服务集成镜像、包含特定控制面板的服务集成镜像、客服及其他支持支持。

3.2 除非法律法规明确要求，或出现以下情况，否则，百度智能云没有义务对服务商及/或用户的信息数据、应用（服务）信息，以及与其在百度智能云镜像服务上所进行的操作有关的其它事项进行事先审查：

3.2.1 百度智能云有合理的理由认为特定服务商及/或用户及具体操作事项可能存在重大违法或违约情形。

3.2.2 百度智能云有合理的理由认为特定服务商及/或用户在镜像服务上的行为涉嫌违法或不当。

3.3 无论百度智能云审核或不审核这些资料、数据或信息，均不会导致百度智能云将对此承担任何责任。

四、用户权利与义务

4.1 您可以根据需要自行选择使用任意镜像。您使用具体镜像之前，请审慎了解镜像的功能、要求等详细内容，如果您对具体镜像有异议，则请勿选择使用。因使用镜像等产生的任何问题，请与服务商进行咨询解决。

4.2百度智能云没有责任和义务对于服务商发布在百度智能云镜像上的任何不准确或不正确的内容承担任何责任，无论该等不准确是由服务商所造成的，还是由于用户的误解所造成的。

4.3 百度智能云对服务商提供的镜像内容不拥有任何权利，其著作权等知识产权归该镜像的合法拥有者所有。百度智能云不对服务商通过本平台提供的镜像的知识产权作任何明示和默示的担保。

4.4百度智能云也不承担用户因使用这些资源对自己和他人造成任何形式的损失或伤害。

4.5 服务商及用户了解并同意，百度智能云有权依照法律法规的规定，应政府部门（包括司法及行政部门）及或权利人的要求，向其提供服务商及/或用户提供给百度智能云的身份信息和操作记录等必要信息。如服务商及/或用户涉嫌侵犯他人知识产权等合法权益，则百度智能云亦有权在初步判断涉嫌侵权行为存在的情况下，向权利人提供使用者必要的身份信息。

4.6 如存在下列情况，百度智能云以普通或非专业人员的知识水平标准对相关内容进行判别，可以认为这些内容或行为具有违法或不当性质的，百度智能云有权删除相关信息或停止对该服务商或用户提供服务，并追究相关法律责任：

4.7 第三方通知百度智能云，认为某个服务商或某项镜像可能存在重大问题；

4.8 百度智能云有权对服务商及/或用户的信息数据及操作行为进行查阅，发现信息数据或操作行为中存在任何问题或怀疑，均有权向服务商及/或用户发出询问或要求改正的通知，或者直接做出删除等处理。

4.9 服务商及用户了解并同意，百度智能云不对因下述任一情况而导致的任何损害赔偿承担责任，包括但不限于利润、商誉、使用、数据等方面的损失或其它无形损失的损害赔偿 (无论百度智能云是否已被告知该等损害赔偿的可能性)：

4.9.1 使用或未能使用镜像服务。

4.9.2 第三方未经批准的使用服务商及/或用户的账户或更改其数据。

4.9.3 百度智能云之外的第三方对“服务”的声明或关于“服务”的行为；

4.9.4 服务商及/或用户对镜像服务的误解。

4.9.5 任何非因百度智能云的原因而引起的与镜像服务有关的其它事宜，包括疏忽。

4.10 在任何情况下，百度智能云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括使用镜像服务而遭受的利润损失，承担任何责任（即使已被告知该等损失的可能性）。

4.11 不论在何种情况下，百度智能云均不对由于计算机病毒、信息网络正常的设备维护，信息网络连接故障，电脑、通讯或其他系统的故障，电力故障，罢工，劳动争议，暴乱，起义，骚乱，生产力或生产资料不足，火灾，洪水，风暴，爆炸，战争，政府行为，司法行政机关的命令或第三方的不作为而造成的不能服务或延迟服务承担责任。

五、镜像服务商行为规范

5.1 服务商理解并同意，在接受并遵守本服务协议的所有条款的基础上，还应遵守《百度智能云线上服务协议》的通用条款和管理规范。《服务商合作协议》和《百度智能云线上服务协议》的效力适用于全部服务商。

5.2 服务商声明并保证其发布的镜像内容：

5.2.1 真实、合法、准确、完整，不会有任何欺诈成份；

5.2.2 不会侵犯任何第三者对该应用享有的合法权利或权益，包括但不限于第三方知识产权、物权等；

5.2.3 不会违反任何法律、法规、条例或规章 (包括但不限于关于规范互联网站、互联网信息、不正当竞争的法律、法规、条例或规章)；

5.2.4 不会含有诽谤（包括商业诽谤）、非法恐吓或非法骚扰的内容；

5.2.5 不会含有淫秽、或包含任何色情内容；

5.2.6 不会含有蓄意毁坏、恶意干扰、秘密地截取或侵占任何系统、数据或个人资料的任何病毒、伪装破坏程序、电脑蠕虫、定时程序炸弹或其他电脑程序；

5.2.7 不会含有任何已知的漏洞、后门、恶意软件或其他可能对用户的系统安全造成损害的不安全内容；

5.2.8 不会直接或间接与下述各项内容链接：任何法律、法规、条例或规章所禁止的商品或服务；或无权链接或包含的商品或服务。 如服务商违反上述声明及保证，百度智能云有权随时暂停直至终止其在百度智能云平台提供镜像服务的资格，并保留追究其违约责任的权利。

5.3 服务商同意接收来自百度智能云及其合作伙伴发出的邮件、信息。

六、镜像用户使用规则

6.1 用户应对其用以登录百度智能云用户名和密码的安全负全部责任，同时对以其用户名进行的所有活动和事件承担全部责任。

6.2 用户在使用百度智能云镜像服务的过程中，承诺遵守以下约定：

6.2.1 百度镜像服务上的所有行为均遵守国家法律、法规等规范性文件及《百度智能云网站服务协议》、百度智能云镜像的各项规则的规定和要求，不违背社会公共利益或公共道德，不损害他人的合法权益，不违反本协议及相关规则。

6.2.2 用户对其所发布信息的真实、安全或合法性负责，不得利用百度智能云提供的网络服务上传、展示或传播任何虚假的、骚扰性的、中伤他人的、辱骂性的、恐吓性的、庸俗淫秽的或其他任何非法的信息资料；不得侵犯其他任何第三方的专利权、著作权、商标权、名誉权或其他任何合法权益；不得违背社会公共利益或公共道德；不得存在其他涉嫌违法或违反本协议及各类规则、或百度智能云认为不适合在镜像服务上发布的内容。

用户如果违反前述承诺，产生任何法律后果的，用户应以自己的名义独立承担所有的法律责任，并确保百度智能云免于因此产生任何损失。

6.3 用户理解并同意：

6.3.1 用户使用包含镜像内容的BCC实例时，其因使用BCC实例而与百度智能云之间所形成的法律关系，应按照百度智能云与用户之间所达成《百度智能云线上服务协议》的约定履行。

6.3.2 用户使用的服务商镜像的服务期间，与用户通过该镜像所创建的BCC实例的服务期限保持高度一致和完全同步；因用户的BCC实例的数据释放及/或服务终止所导致的镜像服务同步终止，用户无权向百度智能云提出任何主张、要求或索赔。

6.4 用户声明并承诺：用户对服务商提供的镜像内容，不会进行修改、改编或翻译、反向工程、分拆、制作衍生作品、或散发，亦不会移除、遮蔽或更改镜像产品所附随的内容； 如用户违反上述声明及保证，百度智能云有权随时终止为其提供服务，并保留追究其违约责任的权利。

6.5 用户同意接收来自百度智能云、服务商及百度智能云合作伙伴发出的邮件、信息。

七、知识产权和隐私权政策

7.1 百度智能云镜像服务的所有内容，包括但不限于文字、图表、标识、商标、按钮图标、图像、声音文件片段、数字下载、数据编辑和软件、网站架构、网页设计、档案、咨询、资料等，都是百度智能云或其内容提供者的财产，受中国和国际版权法的保护，均由百度智能云或其他权利人依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。未经百度智能云或其内容提供者的合法授权或许可，不得以任何目的对百度智能云镜像服务或其任何部分进行擅自使用、修改、复制、复印、仿造、出售、转售、访问、公开传播、散布、发行或以其他方式加以利用。

7.2 服务商及用户了解并同意，百度智能云自身发布的各类信息（包括但不限于身份信息、镜像及服务信息、评论等），拥有

永久的免费使用权，并有权在全球范围内全部或部分地使用、复制、修订、改写、发布、翻译和展示。但服务商及/或用户在发布信息的同时，声明全部或部分保留的，百度智能云的使用行为不得违反其声明保留的范围。

7.3 尊重知识产权是百度智能云遵循的基本准则，镜像服务商和用户均应遵守前述义务，不得侵犯他人的合法权利。如有违反，侵权人应向被侵害的合法权利人，及百度智能云承担损害赔偿责任。

7.4 百度智能云声明并承诺：

7.4.1 不会向任何人出售或出借用户的个人信息，除非事先得到用户的许可。

7.4.2 不允许任何第三方以任何手段收集、编辑、出售或者无偿传播用户的个人信息。

7.4.3 为服务用户之目的，百度智能云可能通过使用用户的个人信息，向其提供服务，包括但不限于向用户发出产品和服务信息，或者与百度智能云合作伙伴共享信息以便他们向用户发送有关其产品和服务的信息（后者需得到用户的事先同意）。

7.5 百度智能云将在下述情况下部分或全部披露用户的个人信息：

7.5.1 经用户同意，向第三方披露；

7.5.2 根据法律的有关规定，或者行政或司法机构的要求，向第三方或者行政、司法机构披露；

7.5.3 如果用户出现违反中国有关法律或者网站政策的情况，需要向第三方披露；

7.5.4 为提供用户所要求的应用，而必须和第三方分享用户的个人信息；

7.5.5 其它百度智能云根据法律或者网站政策认为合适的披露；

八、通知送达

8.1 百度智能云发送给服务商及/或用户的所有通知，均可通过网页公告、电子邮件、手机短信或常规的信件传送等方式进行；该等通知于发送之日视为已送达收件人。

8.2 服务商及/或用户发送给百度智能云的通知，应当通过百度智能云对外正式公布的通信地址、传真号码、电子邮件地址等联系信息进行送达。

九、法律、规则的适用与纠纷管辖

9.1 本服务协议的订立、执行和解释及争议的解决均应适用中华人民共和国法律。本协议中涉及到的任何一方对服务存有异议，可与其他各方协商解决，协商不成的，有权向北京市海淀区人民法院起诉。

9.2 本服务协议根据《百度智能云用户服务协议》制定，服务商及用户在使用百度智能云镜像服务的过程中，同时应遵守《百度智能云用户服务协议》的全部规定。

9.3 本服务协议为百度智能云镜像服务的基本规则，包含了服务商及/或用户使用百度智能云镜像服务时所需遵守的一般性规范。

9.4 用户在使用百度智能云镜像服务的某项具体服务过程中，还需遵守适用于该服务的特殊条款（例如：用户与百度智能云签署的其他协议或文件、百度智能云另行发布的规则等）；服务商在与百度智能云进行某项具体合作的过程中，还需遵守百度智能云与服务商就具体合作事项签署的协议条款的约定。

数据与个人信息保护声明

数据与个人信息保护声明

Personal Data Protection Act.

1. This Clause shall apply where Party B undertakes any Processing of data, documents, materials or information in or from Singapore（as “Services”）.

   当乙方接受甲方指示在新加坡或从新加坡承接任何数据、文件、材料或信息的处理时(以下简称服务)，适用本声明。

2. Both [Party A]2 and [Party B]2 acknowledge and agree that with regard to the Processing of Personal Data, [Party B] is a Data Intermediary (as defined in the PDPA) acting on behalf of [Party A], and [Party B] Processes any Personal Data disclosed by [Party A] to [Party B] for and on [Party A]'s behalf and that this Agreement constitutes contract evidenced or made in writing for such purposes. [Party A] agrees and undertakes that it is fully responsible for complying with applicable law, requirements and obligations, including under the PDPA, in relation to any Personal Data which it may require Party B to Process pursuant to this Agreement, and any additional costs or expenses to be incurred by Party B in connection with meeting any requirements arising under the PDPA shall be borne exclusively by Party A.

【甲方】与【乙方】均承认并同意，就个人数据处理而言，【乙方】是代表【甲方】行事的数据中介机构(定义见PDPA)，【乙方】为【甲方】代表【甲方】处理【甲方】向【乙方】披露的任何个人数据，且本声明构成为此目的以书面形式证明或订立的合同。【甲方】同意并承诺，就其可能要求乙方根据本声明处理相关的任何个人数据，其完全负责遵守适用的法律、要求和义务，包括《个人数据保护法》项下的规定，且乙方因满足《个人数据保护法》项下产生的任何要求而发生的任何额外成本或费用均由甲方全权承担。

3. Without limiting the generality of the foregoing, [Party A] acknowledges and agrees that in its use of the [Services]:

在不限制前述一般性原则的前提下，【甲方】承认并同意在使用【服务】时:

（a）where [Party A] discloses Personal Data to [Party B], [Party A] has the sole responsibility for the accuracy, quality, and legality of Personal Data and the means by which [Party A] acquired Personal Data, and [Party A] represents and warrants to [Party B] that [Party A] has obtained all necessary consents and/or notices under applicable law, including the PDPA, from each individual to whom the Personal Data relates for the disclosure of such individual's Personal Data to [Party B] and for [Party B]'s collection, use, disclosure and/or Processing, and that such consents and/or notices have not been withdrawn;

当【甲方】向【乙方】披露个人数据，【甲方】应对个人数据的准确性、质量和合法性以及【甲方】获取个人数据的手段负全部责任，【甲方】向【乙方】声明并保证，【甲方】已根据适用法律(包括PDPA)获得所有必要的同意和/或通知，向【乙方】披露个人数据以及【乙方】收集、使用、披露和/或处理该等个人数据，且该等同意和/或通知尚未撤回；

（b）[Party A] represents and warrants to [Party B] that [Party A] has at all times complied with and will continue to comply with the requirements of the PDPA, in respect of the collection, use, disclosure, Processing and/or other handling of Personal Data, including complying with any applicable requirement to provide notice to individuals of the use of [Party B] as Data Intermediary;

【甲方】向【乙方】声明并保证，在个人数据的收集、使用、披露、处理和/或其他处理方面，【甲方】一直并将继续遵守PDPA的要求，包括就委托【乙方】作为数据中介服务商对个人数据的使用已向个人发出通知并承诺遵守任何适用的要求；

（c）Personal Data received via or from Singapore, may be transferred to, and collected, used, disclosed and/or Processed in the country in which [Party B], its affiliates and/or its sub-contractors maintain facilities for the [Services], and [Party A] represents and warrants to [Party B] that [Party A] has obtained all necessary consents and/or notices under applicable law, including the PDPA, from each individual to whom the Personal Data relates for the transfer of Personal Data out of Singapore in connection with [Party A]'s use of the [Services], and that such consents and/or notices have not been withdrawn.

通过新加坡或从新加坡接收的个人数据，可能会被转移到【乙方】其关联公司和/或其分包商为履行【服务】而维护设施的国家或地区进行收集、使用、披露和/或处理，【甲方】向【乙方】声明并保证，【甲方】为【甲方】使用【服务】而将个人数据转移出新加坡的需求已根据适用法律(包括PDPA)获得所有必要的同意和/或通知，且该等同意和/或通知尚未撤回。

4. [Party A] represents, warrants and undertakes not to transmit and/or upload, in its use of the [Services] any data, documents, materials or information which is or may be (a) subject to any controls, restrictions or prohibitions relating to its communication, dissemination or export from Singapore, including without limitation, under the Official Secrets Act 1935, the Strategic Goods (Control) Act 2002, nor (b) defamatory, abusive, obscene, threatening, malicious or otherwise objectionable or illegal or in breach or violation of any applicable laws, regulations, licensing requirements or codes of conduct, or which infringes the rights of any party, including without limitation, contractual, moral and/or privacy rights. Party A shall fully indemnify Party B against any loss, claim, penalty or damage Party B may suffer on account of any breach of this Clause.

【甲方】声明、保证并承诺在使用【服务】时，不传输和/或上传任何受到或可能受到(A)与从新加坡的通信、传播或出口有关的任何控制、限制或禁止的数据、文件、材料或信息，包括但不限于《1935年官方保密法》、《2002年战略物品(管制)法》规定的控制、限制或禁止，或(B)诽谤、辱骂、淫秽、威胁、恶意的或其他令人反感的或非法的，或违反任何适用的法律、法规、许可要求或行为准则的，或侵犯任何一方权利的，包括但不限于合同、道德和/或隐私权的数据、文件、材料或信息。如有违反，甲方应全额赔偿乙方可能遭受的任何损失、索赔、罚款或者损害。

5. In this Clause, unless the context otherwise requires:

除双方另有约定，本声明提及的相关定义如下：

名词	定义
“Party A” 甲方	is referring to the enterprise user/counterparty. 企业用户
“Party B” 乙方	is referring to Baidu Cloud. 百度智能云
“PDPA”	means the Singapore Personal Data Protection Act 2012. 新加坡个人数据保护法2012
“Personal Data”	means data, whether true or not, about an individual who can be identified either from that data or from that data when combined with other information to which an entity has access or is likely to have access. 指有关个人的数据，不论是否属实，可从该数据或从该数据与实体有权访问或可能有权访问的其他信息相结合时识别出该个人。

公共镜像发布记录

2023 EBC公共镜像发布记录

本文主要记录第三代BBC服务器的EBC公共镜像在2023年的更新动态。 **Linux公共镜像**

BaiduLinux

OS版本	镜像信息	发布时间	更新内容
BaiduLinux 3.0	UUID: 68e87510-ccb2-42b7-97e6-2e4c80ed5074 family: bbcv3#baidulinux#3.0#AMD64 OS version: 3.0 x86_64 (64bit)	2023-10-07	• 新镜像上线 • 内核版本: 5.10.0-3.0.1.6

CentOS

OS版本	镜像信息	发布时间	更新内容
CentOS 8	• **CentOS 8.4** UUID: 112e1ad7-6441-41cc-9f5b-645a372875e9 family: bbcv3#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit)	2023-12-26	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • kdump内存预留策略变更 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • roce网卡carrier-wait-timeout调整到最大 • 保留HandlePowerKey=halt配置 • 更新软件包版本 • 已知问题修复
			• 内核版本: 3.10.0-1160.83.1.el7.x86_64

CentOS 7	• **CentOS 7.9** UUID: e476774f-edd9-439b-ab21-f51493553f5e family: bbcv3#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit)	2023-12-26	• 内核命令行移除pci=realloc参数 • kdump内存预留策略变更 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复
CentOS 8	• **CentOS 8.4** UUID: 84d47fd5-e978-49bc-bd5a-b05592c0b7b7 family: bbcv3#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit)	2023-11-08	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 内核命令行移除pci=realloc参数 • has agent升级到1.1.3.73版本，支持自动升级功能 • 集成v1.0.2版本rdma无损配置服务 • dpu自研卡取消HandlePowerKey=halt配置 • 更新软件包版本 • 已知问题修复
CentOS 8	• **CentOS 8.4** UUID: da581f02-d3b5-404b-bbdc-34edba5f5e8a family: bbcv3#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: 343ad29c-fe3a-4117-93ce-3a1fef6dc3d4 family: bbcv3#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: b22a00c0-926b-4e17-9a79-8014f5156bd8 family: bbcv3#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: c6de52b0-3522-4ee1-a836-a0227636fb9e family: bbcv3#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: a64b3df9-f592-47c9-9182-6dc1ce0f5b3c family: bbcv3#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)	2023-06-14	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本 • 已知问题修复
	• **CentOS 8.4** UUID: 0a50266b-88a8-438e-8555-0f902e7f2fe1		

	UUID: 0a50266b-88a8-438c-8555-0f902e712fe1 family: bbcv3#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: 675c79e4-85c9-4912-ab59-9b42c8a4e75f family: bbcv3#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: d3a2b729-8e7b-4ad8-8e01-561a5887b5bd family: bbcv3#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: f7858215-28bc-4acd-99d5-07b04aecf52f family: bbcv3#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: 0d8ae9bd-32ce-453b-91a2-33d935145214 family: bbcv3#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)		
CentOS 8		2023-05-18	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本 • MLNX_OFED版本升级到5.8-1.1.2，支持ENI/ERI • cloud-init升级到18.5.4.3版本，支持密钥注入 • 网卡多队列绑核优化，支持按numa绑核
CentOS 7	• **CentOS 7.9** UUID: e95be19b-8e78-4b75-9eff-b4e250f51b6f family: bbcv3#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: e58e5bbb-6334-4a6c-878c-e0ef9bac23b3 family: bbcv3#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 62fca533-e7cf-4c53-8b23-0374aecd8822 family: bbcv3#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 785ed056-a6cf-457f-8fa6-907fa7c20ac8 family: bbcv3#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: cc0462f2-adfb-462a-baaf-ec61fd97bc65 family: bbcv3#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: edc8e976-52fe-46a0-b564-400180593652	2023-05-18	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 更新软件包版本 • MLNX_OFED版本升级到5.8-1.1.2，支持ENI/ERI • cloud-init升级到18.5.4.3版本，支持密钥注入 • 网卡多队列绑核优化，支持按numa绑核

	family: bbcv3#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: e46fec24-3aa8-4711-bbf0-e925dee9c2e1 family: bbcv3#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: d2547971-53ce-4255-a2c7-fbec8147136c family: bbcv3#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: aadc9a25-6f03-4dd0-8cd3-74ae73413f7c family: bbcv3#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)		
CentOS 8	• **CentOS 8.4** UUID: b1d6b04c-a129-46e0-ab49-2f63101cf29b family: bbcv3#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: 89115db2-1b21-4531-be13-746b81bee264 family: bbcv3#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: 9bfc5a34-765f-4da6-a964-f590fa97e4c2 family: bbcv3#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: 14547c02-e4c7-4d03-b13c-73ba6bb11971 family: bbcv3#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: 938cf1aa-a72e-4b9b-93d0-2984fae81635 family: bbcv3#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)	2023-03-14	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本 • 安装5.4-3.1.0版本MLNX_OFED • 新增RDMA网络自动配置功能 • 已知问题修复
	• **CentOS 7.9** UUID: b78a9644-d465-4888-8d60-c04ce21a8990 family: bbcv3#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8**		

CentOS 7	UUID: c98ea378-2b91-4b15-9eda-e8ae786261dc family: bbcv3#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 28d2e4c1-c50b-4dd8-a619-9a5f78c514fa family: bbcv3#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 9cacb749-ac33-49c7-905d-ff853201eb04 family: bbcv3#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: 35595f36-e9c8-454b-8669-fd3e77e69630 family: bbcv3#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: 20f2d14d-dcfe-4fa0-9b99-197fed355347 family: bbcv3#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: e7aaa3d8-43f2-4660-abc5-6b39a53f3d6f family: bbcv3#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: 5f1fbb03-9da4-45c5-b38e-d7d4e65896fc family: bbcv3#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: c11edd7a-1960-4ae3-86f2-b3c8722bbb27 family: bbcv3#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)	2023-03-14	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 更新软件包版本 • 安装5.4-3.1.0版本MLNX_OFED • 新增RDMA网络自动配置功能 • 已知问题修复

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 22.04	UUID: 9fea3197-0c4e-40cd-992d-0962f44064b4 family: bbcv3#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-09-07	• 新镜像上线 • 内核版本: 5.15.0-72-generic
Ubuntu 20.04	UUID: 156e5019-4281-4bb1-b6eb-eec76ac0898e family: bbcv3#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-05-23	• 内核版本: 5.4.0-139-generic • 更新软件包版本 • 已知问题修复
Ubuntu 20.04	UUID: 8c86fe42-5755-44a1-b9f7-cd8050d9d476 family: bbcv3#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-05-18	• 内核版本: 5.4.0-139-generic • 更新软件包版本 • MLNX_OFED版本升级到5.8-1.1.2，支持ENI/ERI • cloud-init升级到18.5.4.3版本，支持密钥注入 • 网卡多队列绑核优化，支持按numa绑核
Ubuntu 18.04	UUID: 137b9aad-09de-4246-9e2e-0f4db4fe8f53 family: bbcv3#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-05-18	• 内核版本: 4.15.0-209-generic • 更新软件包版本 • MLNX_OFED版本升级到5.8-1.1.2，支持ENI/ERI • cloud-init升级到18.5.4.3版本，支持密钥注入 • 网卡多队列绑核优化，支持按numa绑核
Ubuntu 20.04	UUID: 8c86fe42-5755-44a1-b9f7-cd8050d9d476 family: bbcv3#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-03-14	• 内核版本: 5.4.0-137-generic • 更新软件包版本 • 安装5.4-3.1.0版本MLNX_OFED • 新增RDMA网络自动配置功能 • 已知问题修复
Ubuntu 18.04	UUID: 137b9aad-09de-4246-9e2e-0f4db4fe8f53 family: bbcv3#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-03-14	• 内核版本: 4.15.0-204-generic • 更新软件包版本 • 安装5.4-3.1.0版本MLNX_OFED • 新增RDMA网络自动配置功能 • 已知问题修复

Rocky Linux

OS版本	镜像信息	发布时间	更新内容
Rocky Linux 8	• Rocky Linux 8.7 UUID: 9d0b0c12-2d7d-45d2-a607-6aa43672f8d4 family: bbcv3#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • Rocky Linux 8.6 UUID: af9299da-cc89-4a7b-8b10-c7644d3cd90d family: bbcv3#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit) • Rocky Linux 8.5 UUID: 3233c061-84de-4ee1-8a0c-c0cec28795e9 family: bbcv3#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)	2023-08-15	• 新增EBC公共镜像 • 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • 安装5.8-2.0.3版本MLNX_OFED

GPU专用镜像

CentOS

OS版本	镜像信息	发布时间	更新内容
CentOS 8.4	UUID: c4837dbf-a917-40d6-b360-6be94f0d2f88 family: bbcv3#gpu#CentOS#8.4#CUDA11.4#AMD64 OS version: 8.4-CUDA11.4 x86_64 (64bit)	2023-06-14	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • GPU Driver 470.161.03 CUDA 11.4.4 cuDNN 8.2.4 • 更新软件包版本
CentOS 7.9	UUID: bfc3bf1c-52db-47da-a039-31c7a4e00730 family: bbcv3#gpu#CentOS#7.9#CUDA11.4#AMD64 OS version: 7.9-CUDA11.4 x86_64 (64bit)	2023-06-14	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • GPU Driver 470.161.03 CUDA 11.4.4 cuDNN 8.2.4 • 更新软件包版本
CentOS 7.6	UUID: 59fbcfe3-c719-41f0-9ce7-1c1c5b65f1d8 family: bbcv3#gpu#CentOS#7.6#CUDA11.4#AMD64 OS version: 7.6-CUDA11.4 x86_64 (64bit)	2023-06-14	• 内核版本: 3.10.0-1160.76.1.el7.x86_64 • GPU Driver 470.161.03 CUDA 11.4.4 cuDNN 8.2.4 • 更新软件包版本

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 20.04	UUID: 129acb63-db02-4661-982b-d266a0634514 family: bbcv3#gpu#Ubuntu#20.04 LTS#CUDA11.4#AMD64 OS version: 20.04 LTS-CUDA11.4 x86_64 (64bit)	2023-06-14	• 内核版本: 5.4.0-135-generic • GPU Driver 470.161.03 CUDA 11.4.4 cuDNN 8.2.4 • 更新软件包版本

2023 BCC公共镜像发布记录

本文主要记录云服务器BCC公共镜像在2023年的更新动态。 **Linux公共镜像**

BaiduLinux

OS版本	镜像信息	发布时间	更新内容
BaiduLinux 3.0	UUID: 7183d3d0-3e24-464d-9c02-a73491c7dd2a family: bcc#baidulinux#3.0#AMD64 OS version: 3.0 x86_64 (64bit)	2023-12-07	• 内核版本: 5.10.0-3.0.1.7 • 优化nginx写回性能 • 修复uprobe引用计数问题，避免了使用perf或者ebpf触发的宕机 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
BaiduLinux 2.0	UUID: df8a3999-a04e-41fb-abf3-c392d90c378d family: bcc#baidulinux#2.0#AMD64 OS version: 2.0 x86_64 (64bit)	2023-12-07	• 内核版本: 4.19.0-4.0.0.11.bl7.x86_64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader 内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
BaiduLinux 3.0	UUID: bf82dfbe-33f6-4325-980f-95aff07363fe family: bcc#baidulinux#3.0#AMD64 OS version: 3.0 x86_64 (64bit)	2023-10-23	• 内核版本: 5.10.0-3.0.1.5 • rsyslog系统日志问题修复
BaiduLinux 2.0	UUID: d21b2b4a-a5ed-4383-9e08-b299ee3531a3 family: bcc#baidulinux#2.0#AMD64 OS version: 2.0 x86_64 (64bit)	2023-10-23	• 内核版本: 4.19.0-4.0.0.11.bl7.x86_64 • rsyslog系统日志问题修复
BaiduLinux 3.0	UUID: 4bf660c3-da0a-4c21-a2b7-8c3f7423e8d5 family: bcc#baidulinux#3.0#AMD64 OS version: 3.0 x86_64 (64bit)	2023-06-30	• 内核版本: 5.10.0-3.0.1.5 • 更新软件包版本
BaiduLinux 2.0	UUID: 4c533501-bb45-4ac9-a95c-75c0db6e65e4 family: bcc#baidulinux#2.0#AMD64 OS version: 2.0 x86_64 (64bit)	2023-06-30	• 内核版本: 4.19.0-4.0.0.11.bl7.x86_64 • 更新软件包版本
BaiduLinux 3.0	UUID: 7b498c5b-7543-42bb-9746-ea9add9e7d5b family: bcc#baidulinux#3.0#AMD64 OS version: 3.0 x86_64 (64bit)	2023-03-31	• 内核版本: 5.10.0-3.0.1.5 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
BaiduLinux 2.0	UUID: d9579f47-ada7-4ba3-9602-b12400f6eee8 family: bcc#baidulinux#2.0#AMD64 OS version: 2.0 x86_64 (64bit)	2023-03-13	• 内核版本: 4.19.0-4.0.0.11.bl7.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动

AlmaLinux

OS版本	镜像信息	发布时间	更新内容
AlmaLinux 9.2	UUID: ff32ba15-65e0-40a6-8b2b-09b14b4fb8c5 family: bcc#AlmaLinux#9.2#AMD64 OS version: 9.2 x86_64 (64bit)	2023-10-23	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • rsyslog系统日志问题修复
AlmaLinux 9.0	UUID: 3225debf-6b74-4512-8b3a-bfcf474be362 family: bcc#AlmaLinux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2023-10-23	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • rsyslog系统日志问题修复
AlmaLinux 8.8	UUID: bf72becd-6bab-45ca-8927-ddcbfbd10305 family: bcc#AlmaLinux#8.8#AMD64 OS version: 8.8 x86_64 (64bit)	2023-10-23	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • rsyslog系统日志问题修复
AlmaLinux 8.6	UUID: 5ae41dd8-a599-4fc4-851a-b2c76f43fd46 family: bcc#AlmaLinux#8.6#AMD64 OS version: 8.6 x86_64 (64bit)	2023-10-23	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • rsyslog系统日志问题修复
AlmaLinux 9.2	UUID: aef08f48-065a-446f-872a-81a191f80fec family: bcc#AlmaLinux#9.2#AMD64 OS version: 9.2 x86_64 (64bit)	2023-08-28	• 新镜像上线 • 内核版本: 5.14.0-284.11.1.el9_2.x86_64
AlmaLinux 8.8	UUID: b60aedcc-f2fe-4059-be63-c4e9dbe39a2f family: bcc#AlmaLinux#8.8#AMD64 OS version: 8.8 x86_64 (64bit)	2023-08-28	• 新镜像上线 • 内核版本: 4.18.0-477.13.1.el8_8.x86_64
AlmaLinux 9.0	UUID: 5ace87c4-45d9-4c51-975f-da9c493d1135 family: bcc#AlmaLinux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2023-06-30	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • 更新软件包版本
AlmaLinux 8.6	UUID: 1de9cbc0-ae70-4fbf-bb0b-718c3f6e8517 family: bcc#AlmaLinux#8.6#AMD64 OS version: 8.6 x86_64 (64bit)	2023-06-30	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • 更新软件包版本

CentOS

OS版本	镜像信息	发布时间	更新内容
CentOS 8	• **CentOS 8.4** UUID: dcf4b025-ac34-42e8-95a3-89eca33e1c43 family: bcc#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: df45f135-d64a-43fa-889b-b31a89d6eae5 family: bcc#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: 34afc0ce-3056-4f27-99ce-dcea61370977 family: bcc#CentOS#8.2#AMD64	2023-12-07	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志

	OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: 43e19b71-50a8-45c6-8a73-ff8afac15075 family: bcc#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: e99aad6e-1882-475b-8927-2db02d07fc92 family: bcc#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)		• yum repo源全部替换为百度源 • 更新软件包版本
CentOS 7	• **CentOS 7.9** UUID: 2be00f62-4a64-468c-a0e8-04aa1b7749bb family: bcc#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: b79f99ef-2419-4839-844d-b937b8a5f1e9 family: bcc#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: c00e3187-a6bb-4992-9055-63ef51601ff3 family: bcc#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: c5f01ddc-4490-4807-ad6e-a4211ece5f29 family: bcc#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: c3fc771a-b466-457f-aaf8-ae8240b3ccc1 family: bcc#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: e1d141d0-cea9-4594-b6c0-50455e07def7 family: bcc#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: 45d10bc5-67ce-41e8-871f-1d2821d6185e family: bcc#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: 6abbc0fc-305b-4728-a249-d3abc8dd2857 family: bcc#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: 8c5be4a3-275c-4d7c-a55b-95b97051ad56 family: bcc#CentOS#7.1#AMD64	2023-12-07	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • yum repo源全部替换为百度源 • 将时钟源改成tsc，优化时间性能 • 更新软件包版本

	OS version: 7.1 x86_64 (64bit)		
CentOS 8	• **CentOS 8.4** UUID: 3ad9b186-bd12-42f2-bc1d-e674fc902922 family: bcc#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: 48d76d3e-4264-4603-baee-7e7681d4c511 family: bcc#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: ea3fa161-03f5-48c8-87ce-881c70978a2d family: bcc#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: e94e7ae9-90e1-4373-a102-c1d31e60604b family: bcc#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: a3cf8519-f264-42cc-bfd9-478f84b5425f family: bcc#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)	2023-10-23	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • rsyslog系统日志问题修复
CentOS 7	• **CentOS 7.9** UUID: d75573a4-7cbb-4540-a3c4-2f64d085630c family: bcc#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: 74b6915b-949e-428a-89b0-91a01da41d61 family: bcc#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 8a865dc9-3bc3-4ce0-9df2-5be0d6bfba3c family: bcc#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 22d553f1-a223-4684-9d19-3b979a47d643 family: bcc#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: 28351f8c-e230-4cf8-953b-b5e60ebe21b7 family: bcc#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: 64305fb8-f17d-442c-a51c-c27541148a56	2023-10-20	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • rsyslog系统日志问题修复

	family: bcc#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: fc9fe4e4-9aa8-40f3-9d7f-27c24d077706 family: bcc#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: 4db9e368-faa9-4090-841a-f38ba803204e family: bcc#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: 6b8bcc85-5c5f-4230-a4fd-d8a9535e9b20 family: bcc#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)		
CentOS 8	• **CentOS 8.4** UUID: a3bd5cf9-b93e-44de-b1c9-a8fcb1f61c3c family: bcc#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: d8c79f4d-3f76-4a1b-ac61-1ff70293ed61 family: bcc#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: 0ca43079-b628-4ab0-9537-ee7f2cdeeaff family: bcc#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: e55ba3b5-38ae-4b74-9a03-047b5da602fd family: bcc#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: e95228b4-f486-417c-912a-87d2a524015f family: bcc#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)	2023-06-30	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本
	• **CentOS 7.9** UUID: fad3a70f-d6d2-4d90-ac43-d48eac42413e family: bcc#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: 092e5f34-73d5-45a9-87ff-59af24a6c9bd family: bcc#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7**		

CentOS 7	• **CentOS 7.7** UUID: c05a3a25-1bab-4d6a-b9f9-9afd040a08fa family: bcc#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: aca0ca0c-1905-44c7-be21-0db3470ea22f family: bcc#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: 6722db25-517c-448c-96df-f1df77a0a396 family: bcc#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: 305f77fd-80c3-4d29-a322-55b9cb44f3b9 family: bcc#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: fff13b42-afc8-4c30-a685-a2757acebc36 family: bcc#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: e57be923-9198-40e4-abb1-c2a5a785a790 family: bcc#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: 7ff1bd7d-bb39-48c5-8738-382d030685d1 family: bcc#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)	2023-06-30	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 更新软件包版本
CentOS 8	• **CentOS 8.4** UUID: d00f145a-9454-443f-af8b-f9e72614a229 family: bcc#CentOS#8.4#AMD64 OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: bbd47d91-b841-45d2-9726-3fc1a86f0017 family: bcc#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: a7180240-fdf7-49b0-9ec0-d7927754c295 family: bcc#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: 2556bfab-b0aa-45b2-b640-a4bdebef7716 family: bcc#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit)	2023-03-13	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动

	• **CentOS 8.0** UUID: 90042f53-0678-47d5-8428-5fb41baedf5d family: bcc#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)		
CentOS 7	• **CentOS 7.9** UUID: 0aea27c5-c383-4d12-9e38-94f64967e356 family: bcc#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: fc52cc91-9a35-408e-a53f-bd6ca0492775 family: bcc#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 91297bd6-d373-4abc-8fd9-dcad271fc009 family: bcc#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 2e3c550e-4306-4e99-bb6e-740fbd849953 family: bcc#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: cab7803f-5a8f-4440-8bd9-4a757b6afbb3 family: bcc#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: 9fe5179a-ed9d-4079-bdc9-72d9196592ea family: bcc#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: 4e4b6449-4d20-4a36-ae14-7995301b0807 family: bcc#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: 5f094953-2769-49c0-b80b-9666fe222fee family: bcc#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: cadcec11-c5a4-4ec6-980f-8e87af0c811d family: bcc#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)	2023-03-13	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
	• **CentOS 8.4** UUID: e4db12c7-6445-4109-bbb8-5161034dc153 family: bcc#CentOS#8.4#AMD64		

CentOS 8	OS version: 8.4 x86_64 (64bit) • **CentOS 8.3** UUID: 98083197-4fab-4aab-bc9a-477ef3b57584 family: bcc#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: 5c9dcdfb-e991-4f7a-a7bb-02d66099eed3 family: bcc#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: 62479b27-1270-4f93-bd30-4879206b38a6 family: bcc#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0** UUID: de35cf5b-5fd4-42bd-be25-dbd5cc84b706 family: bcc#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)	2023-02-02	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 更新软件包版本
CentOS 7	• **CentOS 7.9** UUID: 782f869c-5f3a-4a4b-b6db-f5bfb6f380f7 family: bcc#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: 4cbfd7c5-0250-4a50-b45f-ddcc26fd0f06 family: bcc#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 2896c329-a7f6-4fe3-a83b-45f8e7e596bc family: bcc#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 1b219e73-a27c-4ae5-a245-8d8d033c51d7 family: bcc#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: dd570915-8344-4da6-84dc-b9fc37e5b291 family: bcc#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: c8d3a943-cf21-411e-89ff-238ebaa45f19 family: bcc#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: 3b577edf-a280-4b32-9b1d-114f7f56adde family: bcc#CentOS#7.3#AMD64	2023-02-02	• 内核版本: 3.10.0-1160.81.1.el7.x86_64 • 更新软件包版本

	OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: ec791242-d33c-402e-b3e5-09517c03bf75 family: bcc#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: 31994f13-4db8-42f3-ba14-4583e8fbd083 family: bcc#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)		

CentOS Stream

OS版本	镜像信息	发布时间	更新内容
CentOS Stream 9	• **CentOS Stream 9** UUID: d80db72e-71e8-47cf-83e2-6d4f106269ce family: bcc#CentOS Stream#9#AMD64 OS version: 9 x86_64 (64bit)	2023-10-23	• 内核版本: 5.14.0-350.el9.x86_64 • rsyslog系统日志问题修复
CentOS Stream 8	• **CentOS Stream 8** UUID: 0d13b6e7-d98e-4c38-913f-e75edabcf010 family: bcc#CentOS Stream#8#AMD64 OS version: 8 x86_64 (64bit)	2023-10-23	• 内核版本: 4.18.0-500.el8.x86_64 • rsyslog系统日志问题修复
CentOS Stream 9	• **CentOS Stream 9** UUID: 01e1c6df-1341-4a20-a795-2758c403693f family: bcc#CentOS Stream#9#AMD64 OS version: 9 x86_64 (64bit)	2023-10-13	• 内核版本: 5.14.0-350.el9.x86_64 • 更新软件包版本
CentOS Stream 8	• **CentOS Stream 8** UUID: 154474a9-e2f4-4834-9e82-45309d7b1b33 family: bcc#CentOS Stream#8#AMD64 OS version: 8 x86_64 (64bit)	2023-08-28	• 内核版本: 4.18.0-500.el8.x86_64 • 更新软件包版本

Rocky Linux

OS版本	镜像信息	发布时间	更新内容
Rocky Linux 8	• **Rocky Linux 8.8** UUID: 640ea2de-22f2-47dd-9230-04cf6ac7fb2d family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.7** UUID: 64103a8f-713a-4507-9992-72d25ec70192 family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.6** UUID: 652ffdd2-7433-4924-9d23-3e90973dbd2c	2023-12-25	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志

	UUID: 692fdd2-7433-4924-9d23-3e9097... family: bcc#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit) • **Rocky Linux 8.5** UUID: 681af508-7700-45eb-a52b-7e19715a8c12 family: bcc#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)		• yum repo源全部替换为百度源 • 更新软件包版本
Rocky Linux 9	• **Rocky Linux 9.2** UUID: a8fa882b-edce-44b2-ae83-46ca0b2e4518 family: bcc#Rocky Linux#9.2#AMD64 OS version: 9.2 x86_64 (64bit) • **Rocky Linux 9.1** UUID: 1106be5f-d79c-4888-9755-7896207ad957 family: bcc#Rocky Linux#9.1#AMD64 OS version: 9.1 x86_64 (64bit) • **Rocky Linux 9.0** UUID: d7258e58-5087-439d-bce5-0ae1925bfe77 family: bcc#Rocky Linux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2023-10-23	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • rsyslog系统日志问题修复
Rocky Linux 8	• **Rocky Linux 8.8** UUID: 70a428bd-6760-42d5-be09-be5acea77401 family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.7** UUID: ddbef26e-0a77-412b-97df-30813cdd2dd9 family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.6** UUID: 7f9d4acd-9a35-4690-85c3-e0561fe7a766 family: bcc#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit) • **Rocky Linux 8.5** UUID: 3a685461-484c-4b8d-8fb9-61ba539d7873 family: bcc#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)	2023-10-23	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • rsyslog系统日志问题修复
Rocky Linux 9	• **Rocky Linux 9.2** UUID: 904553b8-ab26-4fb8-a532-a7e7be9e006c family: bcc#Rocky Linux#9.2#AMD64	2023-08-30	• 新镜像上线 • 内核版本: 5.14.0-284.11.1.el9_2.x86_64
Rocky Linux 8	• **Rocky Linux 8.8** UUID: c4f9fe62-5881-4f2f-a8b9-25f43a2a122f family: bcc#Rocky Linux#8.8#AMD64	2023-08-30	• 新镜像上线 • 内核版本: 4.18.0-477.13.1.el8_8.x86_64

Rocky Linux 9	• **Rocky Linux 9.1** UUID: 8d9bbfa3-ba2f-42b9-ba59-8fff54a8290e family: bcc#Rocky Linux#9.1#AMD64 OS version: 9.0 x86_64 (64bit) • **Rocky Linux 9.0** UUID: b061147f-2665-45e5-9d31-e9ad3fb3dc7b family: bcc#Rocky Linux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2023-06-30	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • 更新软件包版本 • 安装5.8-2.0.3版本MLNX_OFED驱动
Rocky Linux 8	• **Rocky Linux 8.7** UUID: de756551-416e-4615-981b-f36d5876dfa2 family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.6** UUID: 0f84aeea-aa4e-440b-ae97-5ba59022865f family: bcc#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit) • **Rocky Linux 8.5** UUID: 10d802cc-450d-4e50-867f-77ba43b0acf6 family: bcc#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)	2023-06-30	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • 更新软件包版本 • 安装5.8-2.0.3版本MLNX_OFED驱动
Rocky Linux 9	• **Rocky Linux 9.1** UUID: a5e9e57d-1285-433f-a7ea-e93e8f085126 family: bcc#Rocky Linux#9.1#AMD64 OS version: 9.0 x86_64 (64bit)	2023-03-17	• 新镜像上线 • 内核版本: 5.14.0-162.18.1.el9_1.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
Rocky Linux 8	• **Rocky Linux 8.7** UUID: 2a033167-8d24-42ff-8ca3-8b61a4999114 family: bcc#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.5** UUID: 5e87e7f4-3480-4537-b5f6-98aed617ec23 family: bcc#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)	2023-03-17	• 新镜像上线 • 内核版本: 4.18.0-425.13.1.el8_7.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
Rocky Linux 8	• **Rocky Linux 8.6** UUID: 873de26d-1355-4606-bca7-8a5d9bfebe30 family: bcc#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit)	2023-03-13	• 内核版本: 4.18.0-425.13.1.el8_7.x86_64 • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动

Ubuntu

OS版本	镜像信息	发布时间	更新内容

Ubuntu 22.04	UUID: 2a07338e-2deb-4078-9daa-ebcee1a2660e family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-12-13	• 内核版本: 5.15.0-72-generic • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 修复CVE-2023-38545漏洞 • 更新软件包版本
Ubuntu 20.04	UUID: b1cf41ef-8a58-41c9-915f-f90ca8735f3f family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-12-13	• 内核版本: 5.4.0-139-generic • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
Ubuntu 18.04	UUID: f9118543-e207-4843-897b-4d86257fc553 family: bcc#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-12-13	• 内核版本: 4.15.0-212-generic • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
Ubuntu 22.04	UUID: aab18ad3-1378-44ff-aa6a-ad67f98cf5f9 family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-10-24	• 内核版本: 5.15.0-72-generic • rsyslog系统日志问题修复
Ubuntu 20.04	UUID: 8d2b7c4c-d8c3-4fe2-bbfa-3bfc138e4ac5 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-10-24	• 内核版本: 5.4.0-139-generic • rsyslog系统日志问题修复
Ubuntu 18.04	UUID: a1132834-e766-4a85-a9b9-a00f24807da7 family: bcc#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-10-24	• 内核版本: 4.15.0-212-generic • rsyslog系统日志问题修复
Ubuntu 16.04	UUID: e7d34f9e-9889-4a95-af60-1d17674d16b9 family: bcc#Ubuntu#16.04 LTS#AMD64 OS version: 16.04 LTS amd64 (64bit)	2023-10-24	• 内核版本: 4.15.0-142-generic • rsyslog系统日志问题修复
Ubuntu 16.04	UUID: a3ab1d00-660c-4e73-aa56-fc0833c53807 family: bcc#Ubuntu#16.04 LTS#AMD64 OS version: 16.04 LTS amd64 (64bit)	2023-08-22	• 内核版本: 4.15.0-142-generic • CVE漏洞修复
Ubuntu 22.04	UUID: e50c4f0e-a1de-4174-840b-5e0a6a1aa184 family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-06-30	• 内核版本: 5.15.0-72-generic • 更新软件包版本
	UUID: f76e1217-fde2-41bf-9d1f-401764560864		

Ubuntu 20.04	UUID: f76c1317-fde3-41bf-9d1f-40176d569864 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-06-30	• 内核版本: 5.4.0-139-generic • 更新软件包版本
Ubuntu 18.04	UUID: 76dd923e-b43c-4a80-b19a-0795cbca8c67 family: bcc#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-06-30	• 内核版本: 4.15.0-212-generic • 更新软件包版本
Ubuntu 22.04	UUID: 3102dde1-27af-4097-8234-237cd0bfaf0a family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-03-13	• 内核版本: 5.15.0-60-generic • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
Ubuntu 20.04	UUID: 02f73649-98f3-4467-aefd-3a20c7cce757 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-03-13	• 内核版本: 5.4.0-139-generic • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
Ubuntu 18.04	UUID: 21dfd0e3-1e61-4634-9a0e-8c917e0f4b9a family: bcc#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-03-13	• 内核版本: 4.15.0-204-generic • 更新软件包版本 • 安装5.8-1.1.2版本MLNX_OFED驱动
Ubuntu 22.04	UUID: 02a60dbf-9b2a-4d9d-92f3-96771ee5f160 family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2023-02-02	• 内核版本: 5.15.0-57-generic • 更新软件包版本
Ubuntu 20.04	UUID: 251c0f23-2189-48eb-8f7d-e1d910a99cb1 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2023-02-02	• 内核版本: 5.4.0-136-generic • 更新软件包版本
Ubuntu 18.04	UUID: 88674981-3f26-41ec-a696-5c750bf74079 family: bcc#Ubuntu#18.04 LTS#AMD64 OS version: 18.04 LTS amd64 (64bit)	2023-02-02	• 内核版本: 4.15.0-202-generic • 更新软件包版本

Debian

OS版本	镜像信息	发布时间	更新内容
Debian 12	• **Debian 12.0.0** UUID: ec0fd07f-3134-48f0-b7a0-cf5a159207bf family: bcc#Debian#12.0.0#AMD64 OS version: Debian 12.0.0 amd64 (64bit)	2023-12-25	• 内核版本: 6.1.0-11-amd64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
	• **Debian 11.7.0** UUID: d5183f0b-f98f-4a37-bc03-59c3420b5ee7 family: bcc#Debian#11.7.0#AMD64		

Debian 11	OS version: Debian 11.7.0 amd64 (64bit) • **Debian 11.1.0** UUID: 21e3af25-181b-467f-98b7-b05af99f9d32 family: bcc#Debian#11.1.0#AMD64 OS version: Debian 11.1.0 amd64 (64bit) • **Debian 11.0.0** UUID: 49bdd4a8-ee9a-465d-b508-92fead7382f6 family: bcc#Debian#11.0.0#AMD64 OS version: Debian 11.0.0 amd64 (64bit)	2023-12-25	• 内核版本: 5.10.0-26-amd64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
Debian 10	• **Debian 10.10.0** UUID: e94f7941-3d25-4fda-bc93-860aa3c1304e family: bcc#Debian#10.10.0#AMD64 OS version: Debian 10.10.0 amd64 (64bit) • **Debian 10.9.0** UUID: 723ad424-c4a7-47bc-885a-c59b65f1d5ff family: bcc#Debian#10.9.0#AMD64 OS version: Debian 10.9.0 amd64 (64bit) • **Debian 10.8.0** UUID: df9fe02d-bd7a-4fb6-84de-93c947bcaf6a family: bcc#Debian#10.8.0#AMD64 OS version: Debian 10.8.0 amd64 (64bit) • **Debian 10.7.0** UUID: 60540dac-9d39-44b8-857d-bfa550238a37 family: bcc#Debian#10.7.0#AMD64 OS version: Debian 10.7.0 amd64 (64bit) • **Debian 10.6.0** UUID: 35b74a42-7ef2-4b91-8099-59fa3e9de6fc family: bcc#Debian#10.6.0#AMD64 OS version: Debian 10.6.0 amd64 (64bit) • **Debian 10.5.0** UUID: 426a8cc8-225b-4176-a4d1-58b7a6298ecc family: bcc#Debian#10.5.0#AMD64 OS version: Debian 10.5.0 amd64 (64bit) • **Debian 10.4.0** UUID: d5f9fc2f-dceb-4714-930c-66ec7d672835 family: bcc#Debian#10.4.0#AMD64 OS version: Debian 10.4.0 amd64 (64bit) • **Debian 10.3.0** UUID: 14979a5c-a7b9-4b5d-88f3-20e7ecf41659 family: bcc#Debian#10.3.0#AMD64 OS version: Debian 10.3.0 amd64 (64bit)	2023-12-25	• 内核版本: 4.19.0-25-amd64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • 更新软件包版本
Debian 12.0.0	UUID: 16f4e571-7f2f-45cc-9c56-42f7a0022432 family: bcc#Debian#12.0.0#AMD64	2023-10-24	• 内核版本: 6.1.0-11-amd64

	OS version: Debian 12.0.0 amd64 (64bit)		• rsyslog系统日志问题修复
Debian 11.7.0	UUID: 2f4abccd-66f7-450d-941d-ae2e22b1f2f7 family: bcc#Debian#11.7.0#AMD64 OS version: Debian 11.7.0 amd64 (64bit)	2023-10-24	• 内核版本: 5.10.0-23-amd64 • rsyslog系统日志问题修复
Debian 11.1.0	UUID: 19bcc937-a167-43e5-b674-9a1f0e88a8a8 family: bcc#Debian#11.1.0#AMD64 OS version: Debian 11.1.0 amd64 (64bit)	2023-10-24	• 内核版本: 5.10.0-19-amd64 • rsyslog系统日志问题修复
Debian 11.0.0	UUID: 1c2b5fa0-c867-49f3-8773-b41281d491f4 family: bcc#Debian#11.0.0#AMD64 OS version: Debian 11.0.0 amd64 (64bit)	2023-10-24	• 内核版本: 5.10.0-19-amd64 • rsyslog系统日志问题修复
Debian 10.10.0	UUID: a41acba1-30a3-49d0-a31b-ff51ad7add40 family: bcc#Debian#10.10.0#AMD64 OS version: Debian 10.10.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.9.0	UUID: 590854b8-590b-4b54-b30a-c04c46d5ee91 family: bcc#Debian#10.9.0#AMD64 OS version: Debian 10.9.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.8.0	UUID: f829e644-af43-4707-9f14-ee12d5f982de family: bcc#Debian#10.8.0#AMD64 OS version: Debian 10.8.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.7.0	UUID: 505c2b48-3c20-47f5-a856-96a35a67f049 family: bcc#Debian#10.7.0#AMD64 OS version: Debian 10.7.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.6.0	UUID: c01c0066-7698-4d05-a9fd-4d7d8b88fd33 family: bcc#Debian#10.6.0#AMD64 OS version: Debian 10.6.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.5.0	UUID: a16ae821-44e9-4bcb-b6de-d805dc5a38df family: bcc#Debian#10.5.0#AMD64 OS version: Debian 10.5.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.4.0	UUID: 2d65d646-f0a4-4dfe-aacb-f5c72d03ce09 family: bcc#Debian#10.4.0#AMD64 OS version: Debian 10.4.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 10.3.0	UUID: bb28bc12-903c-4041-a243-a3215126c93b family: bcc#Debian#10.3.0#AMD64 OS version: Debian 10.3.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.19.0-25-amd64 • rsyslog系统日志问题修复
Debian 9.13.0	UUID: cb902131-497c-4bd4-ab39-29dcb375aed1 family: bcc#Debian#9.13.0#AMD64 OS version: Debian 9.13.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.9.0-19-amd64 • rsyslog系统日志问题修复
Debian 9.9.0	UUID: 059dc21b-e0d0-46b1-88f5-f664ec3cea15 family: bcc#Debian#9.9.0#AMD64	2023-10-24	• 内核版本: 4.9.0-19-amd64

	OS version: Debian 9.9.0 amd64 (64bit)		• rsyslog系统日志问题修复
Debian 9.1.0	UUID: f52d16f7-4dd4-49e8-87e5-32621fc54765 family: bcc#Debian#9.1.0#AMD64 OS version: Debian 9.1.0 amd64 (64bit)	2023-10-24	• 内核版本: 4.9.0-19-amd64 • rsyslog系统日志问题修复
Debian 12.0.0	UUID: 1cf0eef8-44bd-44f4-999b-958f1a21c909 family: bcc#Debian#12.0.0#AMD64 OS version: Debian 12.0.0 amd64 (64bit)	2023-08-22	• 新镜像上线 • 内核版本: 6.1.0-11-amd64
Debian 11.7.0	UUID: d0bd448c-6d67-4a04-a5ec-f11abf3d0606 family: bcc#Debian#11.7.0#AMD64 OS version: Debian 11.7.0 amd64 (64bit)	2023-08-22	• 内核版本: 5.10.0-23-amd64 • 更新软件包版本
Debian 11.1.0	UUID: babc03c2-5a22-42ee-bda3-ef253a7b040d family: bcc#Debian#11.1.0#AMD64 OS version: Debian 11.1.0 amd64 (64bit)	2023-08-22	• 内核版本: 5.10.0-19-amd64 • 更新软件包版本
Debian 11.0.0	UUID: 6b4d90f8-0718-493b-8d3f-440e65963781 family: bcc#Debian#11.0.0#AMD64 OS version: Debian 11.0.0 amd64 (64bit)	2023-08-22	• 内核版本: 5.10.0-19-amd64 • 更新软件包版本
Debian 10.10.0	UUID: 71b7f0f3-9a8b-4543-b15f-e6ccb99a8ec0 family: bcc#Debian#10.10.0#AMD64 OS version: Debian 10.10.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.9.0	UUID: cc5c4da4-cea1-45ce-aed2-bfb368e69bca family: bcc#Debian#10.9.0#AMD64 OS version: Debian 10.9.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.8.0	UUID: 64dfc87b-3ac0-4f3b-8760-94459c61dec5 family: bcc#Debian#10.8.0#AMD64 OS version: Debian 10.8.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.7.0	UUID: e19a0892-d58b-4cdb-8828-37366478f263 family: bcc#Debian#10.7.0#AMD64 OS version: Debian 10.7.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.6.0	UUID: 5e7853d4-1ee5-45a5-b392-81567ceb5626 family: bcc#Debian#10.6.0#AMD64 OS version: Debian 10.6.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.5.0	UUID: c6b5c3a7-f299-42c6-9e17-99f970c6133d family: bcc#Debian#10.5.0#AMD64 OS version: Debian 10.5.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.4.0	UUID: cfa5d5f1-29ff-4b20-888b-49bedfa790dc family: bcc#Debian#10.4.0#AMD64 OS version: Debian 10.4.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.19.0-25-amd64 • 更新软件包版本
Debian 10.3.0	UUID: ab0d4ae0-b8e0-406c-b13e-3fdf509f2b0a family: bcc#Debian#10.3.0#AMD64	2023-08-22	• 内核版本: 4.19.0-25-amd64

Debian 10.3.0	family: bcc#Debian#10.3.0#AMD64 OS version: Debian 10.3.0 amd64 (64bit)	2023-08-22	• 更新软件包版本
Debian 9.13.0	UUID: 8f149771-5ca4-48f8-bfdb-724d07831ba3 family: bcc#Debian#9.13.0#AMD64 OS version: Debian 9.13.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.9.0-19-amd64 • 更新软件包版本 • 改用debian-archive源
Debian 9.9.0	UUID: b16adb5b-219f-4c4a-92fc-9b831d1cf8b0 family: bcc#Debian#9.9.0#AMD64 OS version: Debian 9.9.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.9.0-19-amd64 • 更新软件包版本 • 改用debian-archive源
Debian 9.1.0	UUID: 16902456-d549-4aa4-bd2e-809974b53ca4 family: bcc#Debian#9.1.0#AMD64 OS version: Debian 9.1.0 amd64 (64bit)	2023-08-22	• 内核版本: 4.9.0-19-amd64 • 更新软件包版本 • 改用debian-archive源
Debian 11.7.0	UUID: e17b07e5-c6a5-4205-842a-9a7114998a7e family: bcc#Debian#11.7.0#AMD64 OS version: Debian 11.7.0 amd64 (64bit)	2023-07-21	• 新镜像上线 • 内核版本: 5.10.0-23-amd64

Windows公共镜像

OS版本	镜像信息	发布时间	更新内容
Windows Server 2022	• **2022 x86_64 (64bit) 中文版** UUID: 4d5ef3e3-a2cc-4596-ab95-b3b616926d63 family: bcc#Windows Server#2022#chs#AMD64 OS version: 2022 x86_64 (64bit) 中文版 • **2022 x86_64 (64bit) 英文版** UUID: 5955bc0c-8e50-4fe0-b00d-636ddf2d08e4 family: bcc#Windows Server#2022#AMD64 OS version: 2022 x86_64 (64bit) 英文版	2023-11-02	• 更新系统补丁
Windows Server 2019	• **2019 x86_64 (64bit) 中文版** UUID: e0f6b5b7-ba26-4623-b201-83a7c354230c family: bcc#Windows Server#2019#chs#AMD64 OS version: 2019 x86_64 (64bit) 中文版 • **2019 x86_64 (64bit) 英文版** UUID: b3e75205-e0f6-4396-bffd-2c389f329340 family: bcc#Windows Server#2019#AMD64 OS version: 2019 x86_64 (64bit) 英文版	2023-11-02	• 更新系统补丁
Windows Server 2016	• **2016 x86_64 (64bit) 中文版** UUID: c92493cb-2ade-4cb4-9b21-0379682ed0a2 family: bcc#Windows Server#2016#chs#AMD64 OS version: 2016 x86_64 (64bit) 中文版	2023-11-02	• 更新系统补丁

	• **2016 x86_64 (64bit) 英文版** UUID: fc578c6b-6212-4729-91a1-974f775f8941 family: bcc#Windows Server#2016#AMD64 OS version: 2016 x86_64 (64bit) 英文版		
Windows Server 2012	• **2012 x86_64 (64bit) 中文版** UUID: 271d0651-08e8-4b6a-94fe-ff2a06b9f8bc family: bcc#Windows Server#2012#chs#AMD64 OS version: 2012 x86_64 (64bit) 中文版 • **2012 x86_64 (64bit) 英文版** UUID: 21d05834-acf5-4490-b8c6-94aff54e7b67 family: bcc#Windows Server#2012#AMD64 OS version: 2012 x86_64 (64bit) 英文版	2023-11-02	• 更新系统补丁
Windows Server 2008	• **2008 x86_64 (64bit) 中文版** UUID: cd76992f-ba26-460f-9326-c6850ada4cef family: bcc#Windows Server#2008#chs#AMD64 OS version: 2008 x86_64 (64bit) 中文版 • **2008 x86_64 (64bit) 英文版** UUID: 21bc3e79-bce5-472a-82ce-b6ae25eb4ec0 family: bcc#Windows Server#2008#AMD64 OS version: 2008 x86_64 (64bit) 英文版	2023-11-02	• 更新系统补丁
Windows Server 1909	• **1909 x86_64 (64bit) 中文版** UUID: 3ae2c048-fa87-4766-aac4-576003fe5b5b family: bcc#Windows Server#1909#chs#AMD64 OS version: 1909 x86_64 (64bit) 中文版 • **1909 x86_64 (64bit) 英文版** UUID: 972a4a80-ceff-4828-bd3d-42eba43ff36a family: bcc#Windows Server#2016#AMD64 OS version: 1909 x86_64 (64bit) 英文版	2023-11-02	• 更新系统补丁
Windows Server 2022	• **2022 x86_64 (64bit) 中文版** UUID: a3a56ed4-6233-426d-b0f1-cf836fdc1f28 family: bcc#Windows Server#2022#chs#AMD64 OS version: 2022 x86_64 (64bit) 中文版 • **2022 x86_64 (64bit) 英文版** UUID: f715445d-d4ba-428e-ae10-579280060cfe family: bcc#Windows Server#2022#AMD64 OS version: 2022 x86_64 (64bit) 英文版	2023-01-04	• 更新系统补丁
Windows Server 2019	• **2019 x86_64 (64bit) 中文版** UUID: b6825957-20ea-4e88-9c33-ce5993ff2124 family: bcc#Windows Server#2019#chs#AMD64 OS version: 2019 x86_64 (64bit) 中文版 • **2019 x86_64 (64bit) 英文版**	2023-01-04	• 更新系统补丁

	UUID: 02ef43d5-167b-4c1d-b55e-0597d6ce56b3 family: bcc#Windows Server#2019#AMD64 OS version: 2019 x86_64 (64bit) 英文版		
Windows Server 2016	• **2016 x86_64 (64bit) 中文版** UUID: 534ae74b-bcdc-4c07-95ce-869a8ff32ced family: bcc#Windows Server#2016#chs#AMD64 OS version: 2016 x86_64 (64bit) 中文版 • **2016 x86_64 (64bit) 英文版** UUID: 224eb780-7e08-407c-8ee7-2c8e40c581ae family: bcc#Windows Server#2016#AMD64 OS version: 2016 x86_64 (64bit) 英文版	2023-01-04	• 更新系统补丁

ARM公共镜像 | OS版本 | 镜像信息 | 发布时间 | 更新内容 | | :----- | :-------| :-------| :-------| | centos 8.5 | UUID: f29471aa-430a-4a59-bee6-f815906097bc
family: bcc#CentOS#8.5#AARCH64
OS version: 8.5 aarch64 (64bit) | 2023-01-06 |

- 内核版本: 4.18.0-348.7.1.el8_5.aarch64
- 更新软件包版本 | | centos 7.9 | UUID: 76c7e35c-2712-4a22-afbe-879ca6da24a5
family: bcc#CentOS#7.9#AARCH64
OS version: 7.9 aarch64 (64bit) | 2023-01-06 |
- 内核版本: 4.18.0-348.20.1.el7.aarch64
- 更新软件包版本 | | centos 7.6 | UUID: b025d9f1-1f2f-4dc4-a8ff-1c9501b42aec
family: bcc#CentOS#7.6#AARCH64
OS version: 7.6 aarch64 (64bit) | 2023-01-06 |
- 内核版本: 4.18.0-348.20.1.el7.aarch64
- 更新软件包版本 | | ubuntu 20.04 | UUID: 2a5f9275-d2a6-4941-a6d8-ff216132a5c7
family: bcc#Ubuntu#20.04 LTS#AARCH64
OS version: 20.04 LTS aarch64 (64bit) | 2023-01-06 |
- 内核版本: 5.4.0-100-generic
- 更新软件包版本 |

2024 BCC公共镜像发布记录

本文主要记录云服务器BCC公共镜像在2024年的更新动态。

Linux公共镜像

BaiduLinux | OS版本 | 镜像信息 | 发布时间 | 更新内容 | | :----- | :-------| :-------| :-------| | BaiduLinux 3.0 | UUID: b7fb3b9d-c3f6-4828-a749-827cc4057241
family: bcc#baidulinux#3.0#AMD64
OS version: 3.0 x86_64 (64bit) | 2024-09-19 |

- 内核版本: 5.10.0-3.0.1.8
- MLNX OFED升级到5.8-4.1.5版本
- 配置多个时间服务器提高时钟同步精度和可靠性

- 集成中文字符集
- 已知问题修复
- 更新软件包版本 |

AlmaLinux

OS版本	镜像信息	发布时间	更新内容
AlmaLinux 9	• **AlmaLinux 9.2** UUID: a3597ef4-f51d-4c20-8cd5-c9de77fc60f1 family: bcc#AlmaLinux#9.2#AMD64 OS version: 9.2 x86_64 (64bit) • **AlmaLinux 9.0** UUID: d1cfa1a2-1767-47e9-b6b4-7441e039144a family: bcc#AlmaLinux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2024-02-05	• 内核版本: 5.14.0-362.8.1.el9_3.x86_64 • 修复CVE-2023-38545漏洞 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • yum repo源全部替换为百度源 • 更新软件包版本
AlmaLinux 8	• **AlmaLinux 8.8** UUID: fa7aaca7-fc27-433e-a368-c0f4635fc10b family: bcc#AlmaLinux#8.8#AMD64 OS version: 8.8 x86_64 (64bit) • **AlmaLinux 8.6** UUID: 9e44fc8e-2a00-4379-ba96-43cf67ec6c2e family: bcc#AlmaLinux#8.6#AMD64 OS version: 8.6 x86_64 (64bit)	2024-02-05	• 内核版本: 4.18.0-513.5.1.el8_9.x86_64 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • yum repo源全部替换为百度源 • 更新软件包版本

Debian

OS版本	镜像信息	发布时间	更新内容
Debian 9.8.0	UUID: ce189516-d98b-4e37-a66b-7eee16466f71 family: bcc#Debian#9.8.0#AMD64 OS version: Debian 9.8.0 amd64 (64bit)	2024-02-06	• 新镜像上线 • 内核版本: 4.9.0-19-amd64 • 集成5.8-1.1.2版本MLNX_OFED驱动

Rocky Linux

OS版本	镜像信息	发布时间	更新内容
Rocky Linux 9.4	UUID: 43ab3a12-d7db-4b91-b56c-4c368cb29dc3 family: bcc#Rocky Linux#9.4#AMD64 OS version: 9.4 x86_64 (64bit)	2024-11-27	• 内核版本: 5.14.0-427.42.1.el9_4.x86_64 • MLNX OFED升级到5.8-5.1.1版本 • 集成RoCE网卡自动配置脚本及RoCE网卡carrier-wait-timeout配置脚本 • 集成RDMA无损配置服务v1.0.3版本 • 集成has-agent 1.1.3.121版本 • 集成中文字符集 • 已知问题修复 • 更新软件包版本
Rocky Linux 9	• **Rocky Linux 9.2** UUID: 27cac026-180f-45a4-89f7-f8794c52d206 family: bcc#Rocky Linux#9.2#AMD64 OS version: 9.2 x86_64 (64bit) • **Rocky Linux 9.1** UUID: 976ebf28-cb9a-4563-a9b9-72c2b32879da family: bcc#Rocky Linux#9.1#AMD64 OS version: 9.1 x86_64 (64bit) • **Rocky Linux 9.0** UUID: cac5f95d-a443-4061-ab5b-5a64c4083288 family: bcc#Rocky Linux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2024-11-27	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • MLNX OFED升级到5.8-5.1.1版本 • 集成RoCE网卡自动配置脚本及RoCE网卡carrier-wait-timeout配置脚本 • 集成RDMA无损配置服务v1.0.3版本 • 集成has-agent 1.1.3.121版本 • 集成中文字符集 • 已知问题修复 • 更新软件包版本
Rocky Linux 9	• **Rocky Linux 9.2** UUID: 0a6a843c-bee8-461f-804a-89ace741b9fe family: bcc#Rocky Linux#9.2#AMD64 OS version: 9.2 x86_64 (64bit) • **Rocky Linux 9.1** UUID: fd5aa961-eb64-42d7-827a-ef823f1fdab4 family: bcc#Rocky Linux#9.1#AMD64 OS version: 9.1 x86_64 (64bit) • **Rocky Linux 9.0** UUID: f83a8e52-6a45-4ffd-baa5-b9f9d6666684 family: bcc#Rocky Linux#9.0#AMD64 OS version: 9.0 x86_64 (64bit)	2024-01-09	• 内核版本: 5.14.0-284.11.1.el9_2.x86_64 • 修复CVE-2023-38545漏洞 • 网卡多队列绑核脚本自适应CPU拓扑 • bootloader内核选择超时时间调整到1s • 保存半年安全日志 • yum repo源全部替换为百度源 • 更新软件包版本

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 20.04	UUID: 5a38b14e-d619-4770-85b7-12bf01dbe622 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2024-12-04	• 内核版本: 5.4.0-198-generic • MLNX OFED版本：5.8-4.1.5 • 集成has-agent 1.1.3.121版本 • cloud-init升级到18.5.5.1，支持fqdn功能 • 更新软件包版本
Ubuntu 22.04	UUID: f66126c0-6ce7-4d9f-9c58-e0e08a9425ba family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2024-08-07	• 内核版本: 5.15.0-72-generic • MLNX OFED版本升级到5.8-4.1.5 • 集成RoCE网卡自动配置脚本 • 配置多个时间服务器提高时钟同步精度和可靠性 • 修复CVE-2024-6387漏洞 • RDMA无损配置服务升级到1.0.4版本 • 已知问题修复 • 更新软件包版本
Ubuntu 20.04	UUID: ce17932e-c649-4293-8a1a-0d5642008b59 family: bcc#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS amd64 (64bit)	2024-08-07	• 内核版本: 5.4.0-139-generic • MLNX OFED版本升级到5.8-4.1.5 • 集成RoCE网卡自动配置脚本 • 配置多个时间服务器提高时钟同步精度和可靠性 • 已知问题修复 • 更新软件包版本

2024 EBC公共镜像发布记录

本文主要记录第三代BBC服务器的EBC公共镜像在2024年的更新动态。

Linux公共镜像

CentOS

OS版本	镜像信息	发布时间	更新内容
	• **CentOS 7.9** UUID: 4b32e89b-0dc5-4cfb-a3ff-cd3843a29d4c family: bbcv3#CentOS#7.9#AMD64 OS version: 7.9 x86_64 (64bit) • **CentOS 7.8** UUID: 463b2286-2657-4395-8920-		

CentOS 7	8837d12d941c family: bbcv3#CentOS#7.8#AMD64 OS version: 7.8 x86_64 (64bit) • **CentOS 7.7** UUID: 5945df21-0c33-4318-818a-f1a1f621ee49 family: bbcv3#CentOS#7.7#AMD64 OS version: 7.7 x86_64 (64bit) • **CentOS 7.6** UUID: 0b538700-9def-4cfd-8e92-53acc32d7412 family: bbcv3#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit) • **CentOS 7.5** UUID: 13a41fdb-0bd6-48ac-8dd0-9e6485b73860 family: bbcv3#CentOS#7.5#AMD64 OS version: 7.5 x86_64 (64bit) • **CentOS 7.4** UUID: 9a136fb9-ee72-41f8-af83-6e5a3aa5d9d8 family: bbcv3#CentOS#7.4#AMD64 OS version: 7.4 x86_64 (64bit) • **CentOS 7.3** UUID: 64854052-5f51-42bf-8c73-16bc1243a784 family: bbcv3#CentOS#7.3#AMD64 OS version: 7.3 x86_64 (64bit) • **CentOS 7.2** UUID: c599b9eb-fbd6-4b26-8b4b-b2dc89cc792d family: bbcv3#CentOS#7.2#AMD64 OS version: 7.2 x86_64 (64bit) • **CentOS 7.1** UUID: 8f8be7f8-1d58-4cb9-82ea-0fa77e688e00 family: bbcv3#CentOS#7.1#AMD64 OS version: 7.1 x86_64 (64bit)	2024-06-24	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 内核命令行移除pci=realloc参数 • kdump内存预留策略变更 • 配置多个时间服务器提高时钟同步精度和可靠性 • has agent升级到1.1.3.79版本 • 网卡多队列绑核功能优化 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复
CentOS 8	• **CentOS 8.3** UUID: 004805d0-fd72-4ec5-95fc-3bdf5b1e80a7 family: bbcv3#CentOS#8.3#AMD64 OS version: 8.3 x86_64 (64bit) • **CentOS 8.2** UUID: 7c03fa77-486c-4022-8379-93a3c3c7237c family: bbcv3#CentOS#8.2#AMD64 OS version: 8.2 x86_64 (64bit) • **CentOS 8.1** UUID: f554cae7-1a53-4867-ab90-7fb58edfe25c family: bbcv3#CentOS#8.1#AMD64 OS version: 8.1 x86_64 (64bit) • **CentOS 8.0**	2024-03-28	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • 内核命令行移除pci=realloc参数 • kdump内存预留策略变更 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复

	UUID: 89681535-a970-4a59-aef4-57efa516b267 family: bbcv3#CentOS#8.0#AMD64 OS version: 8.0 x86_64 (64bit)		
CentOS 7	• **CentOS 7.6** UUID: 8275e196-4f8f-441f-8d05-7f69c823d407 family: bbcv3#CentOS#7.6#AMD64 OS version: 7.6 x86_64 (64bit)	已下线	• 内核版本: 3.10.0-1160.83.1.el7.x86_64 • 内核命令行移除pci=realloc参数 • kdump内存预留策略变更 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 18.04	UUID: d1400972-c05c-4cde-9c70-e427098b84e0 family: bbcv3#Ubuntu#18.04 LTS#AMD64 OS Version: 18.04 LTS amd64 (64bit)	2024-07-12	• 4.15.0-212-generic • kdump内存预留策略变更 • 使用chrony服务进行时间同步，配置多个时间服务器提高时钟同步精度和可靠性 • 内核参数去掉pci=realloc • 网卡多队列绑核功能优化 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复
Ubuntu 22.04	UUID: 1f227a5b-0dd3-4b81-805a-d29feabd3131 family: bbcv3#Ubuntu#22.04 LTS#AMD64 OS Version: 22.04 LTS amd64 (64bit)	2024-06-26	• 5.15.0-72-generic • kdump内存预留策略变更 • 配置多个时间服务器提高时钟同步精度和可靠性 • 网卡多队列绑核功能优化 • has agent升级到1.1.3.79版本 • 集成v1.0.4版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复
Ubuntu 20.04	UUID: 527e15b5-6702-492b-9646-7deb285243ab family: bbcv3#Ubuntu#20.04 LTS#AMD64 OS Version: 20.04 LTS amd64 (64bit)	2024-02-01	• 5.4.0-139-generic • 内核命令行移除pci=realloc参数 • kdump内存预留策略变更 • has agent升级到1.1.3.79版本 • 集成v1.0.3版本rdma无损配置服务 • 更新软件包版本 • 已知问题修复

Rocky Linux

OS版本	镜像信息	发布时间	更新内容
Rocky Linux 8	• **Rocky Linux 8.7** UUID: bc3bfdcc-079a-4706-8fba-cc079d5fb32e family: bbcv3#Rocky Linux#8.7#AMD64 OS version: 8.7 x86_64 (64bit) • **Rocky Linux 8.6** UUID: 864889bc-aa92-4f00-a725-51a57085f067 family: bbcv3#Rocky Linux#8.6#AMD64 OS version: 8.6 x86_64 (64bit) • **Rocky Linux 8.5** UUID: 6585dbdc-f103-4bf9-b6a4-865216214ce5 family: bbcv3#Rocky Linux#8.5#AMD64 OS version: 8.5 x86_64 (64bit)	2024-08-02	• 内核版本: 4.18.0-477.13.1.el8_8.x86_64 • 安装5.8-2.0.3版本MLNX_OFED • has agent升级到1.1.3.79版本 • 网卡多队列绑核功能优化 • 集成v1.0.3版本rdma无损配置服务 • 内核参数去掉pci=realloc • 配置多个时间服务器提高时钟同步精度和可靠性 • kdump内存预留策略变更 • 更新软件包版本 • 已知问题修复

GPU专用镜像

CentOS

OS版本	镜像信息	发布时间	更新内容
CentOS 8.4	UUID: 04bef08c-fb63-44de-a016-5c422b8aea2b family: bbcv3#gpu#CentOS#8.4#CUDA11.4#AMD64 OS version: 8.4-CUDA11.4 x86_64 (64bit)	2024-01-09	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • GPU Driver 470.223.02 CUDA 11.4.4 cuDNN 8.9.5 dcgm 3.3.0 • has agent升级到1.1.3.79版本 • 更新软件包版本
CentOS 8.4	UUID: 9f3956f7-7c81-45c4-9c7e-469dcfdebae5 family: bbcv3#gpu#CentOS#8.4#CUDA12.2#AMD64 OS version: 8.4-CUDA12.2 x86_64 (64bit)	2024-01-02	• 内核版本: 4.18.0-348.7.1.el8_5.x86_64 • GPU Driver 535.129.03 CUDA 12.2.2 cuDNN 8.9.5 dcgm 3.3.0 • has agent升级到1.1.3.79版本 • 更新软件包版本

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 20.04	UUID: 1cec744d-352f-482c-a2f1-011fff3aecf7 family: bbcv3#gpu#Ubuntu#20.04 LTS#AMD64 OS version: 20.04 LTS-CUDA12.2 amd64 (64bit)	2024-02-05	• 内核版本: 5.4.0-139-generic • GPU Driver 535.129.03 CUDA 12.2.2 cuDNN 8.9.5 dcgm 3.3.0 • has agent升级到1.1.3.79版本 • 更新软件包版本

NPU专用镜像

Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 22.04	UUID: 07c7128c-803d-4040-abdc-64d303d9440e family: bbcv3#npu#Ubuntu#22.04 LTS#XRE5.0.5#AMD64 OS version: 22.04 LTS-XRE5.0.5 amd64 (64bit)	2024-06-12	• 内核版本: 5.15.0-72-generic • 集成5.0.5版本XRE驱动 • 集成1.1.0.2版本XDR • 集成5.8-1.1.2版本MLNX_OFED • 集成v1.0.4版本rdma无损配置服务 • 集成1.1.3.79版本has agent • 更新软件包版本
Ubuntu 20.04	UUID: 96c4c7fa-c723-4d1e-8391-4012ef6044b5 family: bbcv3#npu#Ubuntu#20.04 LTS#kunlun#AMD64 OS version: 20.04 LTS-kunlun amd64 (64bit)	2024-03-20	• 内核版本: 5.4.0-139-generic • 集成4.31.0版本XRE驱动 • 集成1.1.3.79版本has agent • 更新软件包版本

2025 BCC公共镜像发布记录

本文主要记录云服务器BCC公共镜像在2025年的更新动态。

Linux公共镜像 Ubuntu

OS版本	镜像信息	发布时间	更新内容
Ubuntu 24.04	UUID: 91478caa-a413-4702-8367-46c57c1d55b0 family: bcc#Ubuntu#24.04 LTS#AMD64 OS version: 24.04 LTS amd64 (64bit)	2025-03-17	• 内核版本: 6.8.0-54-generic • MLNX OFED版本：24.10-1.1.4.0 • 添加内核启动参数：nvme_core.io_timeout=4294967295 nvme_core.admin_timeout=4294967295 • 集成has-agent 1.1.3.125版本 • cloud-init版本：18.5.6.1 • 禁用nouveau模块 • 修复高危风险漏洞，更新软件包版本
Ubuntu 22.04	UUID: 87501502-cf93-4af1-bcbe-7c2812abaec7 family: bcc#Ubuntu#22.04 LTS#AMD64 OS version: 22.04 LTS amd64 (64bit)	2025-02-20	• 内核版本: 5.15.0-124-generic • MLNX OFED版本：5.8-4.1.5 • 添加内核启动参数：nvme_core.io_timeout=4294967295 nvme_core.admin_timeout=4294967295 • 集成has-agent 1.1.3.125版本 • cloud-init升级到18.5.6.1，支持FQDN功能 • 禁用nouveau模块 • 已知问题修复 • 修复高危风险漏洞，更新软件包版本

操作系统官方停止维护计划

百度智能云提供的公共镜像维护周期将与镜像平台官方公布的维护周期保持一致。您可在本文档查看不同操作系统平台对于各版本镜像的停止维护计划。

操作系统停止维护EOL（End of Life）后，百度智能云会同时停止对使用相应操作系统版本的BCC实例提供技术支持服务。建议您及时更新或选用更加稳定的镜像版本。

BaiduLinux

版本	停止支持日期
BaiduLinux 3	暂无明确时间
BaiduLinux 2	2024-12-31

第三方操作系统EOL计划

Linux CentOS

版本	停止维护日期
CentOS Stream 9	2027-05-31
CentOS Stream 8	2024-05-31
CentOS 8	2021-12-31
CentOS 7	2024-06-30
CentOS 6	2020-11-30

> 您也可以访问Centos官网查看详细的EOL计划。

Ubuntu

版本	停止维护日期
Ubuntu 24.04 LTS	2029-06-01
Ubuntu 22.04 LTS	2027-06-01
Ubuntu 20.04 LTS	2025-04-01
Ubuntu 18.04 LTS	2023-04-01
Ubuntu 16.04 LTS	2021-04-01
Ubuntu 14.04 LTS	2016-08-01
Ubuntu 12.04 LTS	2014-08-01

> 您也可以访问Ubuntu官网查看详细的EOL计划。

RockyLinux |版本|停止维护时间| |:----|:----| |RockyLinux 9.2|2032-05-31| |RockyLinux 9.1|2032-05-31| |RockyLinux 9.0|2032-05-31| |RockyLinux 8.8|2029-05-31| |RockyLinux 8.7|2029-05-31| |RockyLinux 8.6|2029-05-31| |RockyLinux 8.5|2029-05-31|

> 您也可以访问RockyLinux官网查看详细的EOL计划。

Debian |版本|停止支持时间|EOL-长期支持（LTS）|EOL-扩展长期支持（ELTS）| |:----|:----|:----|:----| |Debian 12|暂无|暂无|暂无| |Debian 11|暂无|暂无|暂无| |Debian 10|2022-08-01|暂无|暂无| |Debian 9|2020-07-06|2022-06-30|2027-06-30| |Debian 8|2018-06-17|2022-06-30|2025-06-30| |Debian 7|2016-04-25|2018-05-31|2020-06-30|

> 您也可以访问Debian官网查看详细的EOL计划。

AlmaLinux |版本|停止维护时间| |:----|:----| |AlmaLinux 9.2|2032-03-31| |AlmaLinux 9.0|2032-03-31| |AlmaLinux 8.8|2029-03-01| |AlmaLinux8.6|2029-03-01|

> 您也可以访问AlmaLinux官网查看详细的EOL计划。

OpenSUSE |版本|停止支持时间| |:----|:----| |OpenSUSE 42.3|2019-07-01| |OpenSUSE 15.2|2022-01-04|

> 您也可以访问OpenSUSE官网查看详细的EOL计划。

Fedora |版本|停止支持时间| |:----|:----| |Fedora 32|2021-05-25|

> 您也可以访问Fedora官网查看详细的EOL计划。

FreeBSD |版本|停止支持时间| |:----|:----| |FreeBSD 11.4|2020-06-23|

> 您也可以访问FreeBSD官网查看详细的EOL计划。

Windows |版本|停止支持时间| |:----|:----| |Windows 2022 数据中心版|2031-10-14| |Windows 2019 数据中心版|2029-01-09| |Windows 2016 数据中心版|2027-01-12| |Windows 2012 数据中心版|2023-10-10| |Windows2008 R2 数据中心版|2020-01-14|

> 您也可以访问Windows官网查看详细的EOL计划。

存储

概述

云服务器BCC使用云磁盘CDS做为自己的系统盘和数据盘。同时，您可以通过创建快照的方式对磁盘数据进行备份。

什么是云磁盘

云磁盘CDS（Cloud Disk Service）是一种安全可靠的高弹性存储服务，可作为云服务器的扩展块存储部件，为云服务器数据存储提供高可用和高容量支持。

云磁盘CDS支持数据的随机读写，在吞吐量，IOPS以及异常恢复时间等方面，都有不错的性能，能够满足大部分客户在各类场景下的使用需求。 云磁盘CDS具有独立于云服务器的生命周期，并支持在线扩容，在线转换云磁盘类型，通过自动快照进行容灾备份和磁盘回滚等众多弹性管理功能。 相比本地盘，云磁盘CDS采用三副本技术对磁盘数据进行存储。若部分存储节点，存储介质或网络交换机发生意外故障，数据会立即迁移到其它节点，数据访问不会受到任何影响，更不会感知到故障的发生。

云磁盘类型

CDS云磁盘提供增强型SSD、通用型SSD、高性能云磁盘及通用型HDD四种磁盘类型，不同磁盘类型的使用场景不同，价格也不同，请根据您的需求选择合适的云磁盘类型。更多关于云磁盘类型的介绍，请查看云磁盘CDS产品介绍。

云磁盘性能

CDS云磁盘提供增强型SSD、通用型SSD、高性能云磁盘及通用型HDD四种磁盘类型，不同磁盘类型的性能不同，主要体现在单盘IOPS、单盘吞吐量、延迟、购买容量等方面。更多关于云磁盘性能的介绍，请查看云磁盘CDS性能介绍。

云磁盘操作

CDS云磁盘目前在计费管理、磁盘基础能力、弹性能力等方面都有丰富的产品功能，如云磁盘服务端加密、在线扩容、在线变更磁盘类型、性能突发等。更多关于云磁盘功能的介绍，请查看云磁盘CDS功能操作介绍。

快照介绍

快照是某一时间节点一块CDS云磁盘的一种数据状态文件。您可以通过在某一时刻为CDS云磁盘创建快照的方式，对该时刻CDS云磁盘的数据进行保存。通过快照，您可以实现磁盘数据备份，磁盘数据恢复以及磁盘镜像的制作。更多关于快照的使用介绍、创建原理、快照额度等介绍，请查看快照使用说明。

磁盘快照操作介绍

您可以为每一块云磁盘CDS创建自动快照、手动快照，并可以对老旧快照进行定期删除。同时，您也可以根据快照创建新的CDS云磁盘，并对快照做跨区域的复制，以实现数据的跨区域备份。更多关于磁盘快照的使用介绍，请查看磁盘快照操作介绍。

实例快照操作介绍

您可以为同一台BCC实例的全部CDS云磁盘在同一时刻创建实例快照，即整机快照，以达到整机数据备份。更多关于实例快照

的使用介绍，请查看实例快照操作介绍。

系统盘扩容

本文介绍百度智能云“云服务器BCC扩容系统盘”的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《云服务器BCC扩容系统盘》

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

概述

随着业务的增长，您的磁盘容量可能无法满足数据存储的需要，这时您可以在原有磁盘容量基础上，扩充到您想要的目标容量。扩容磁盘后，之前磁盘上的数据仍会保留。

扩容须知

项目	说明
影响	• 扩容不需要重启实例。但为避免风险，请在扩容前创建快照，用于数据备份。
扩容前	• 实例状态必须是**运行中**或**正常**状态，如异常或欠费则不能进行扩容操作。 • 目前仅支持**单分区系统盘**。 • 扩容时可以同时变更云磁盘类型，如将40GB的高性能云磁盘变更为100GB的通用型SSD。 • 由于系统盘存储业务重要数据，若缩小容量会带来删除数据的风险。因此，您只能增加云磁盘容量，而不能缩小容量。
扩容后	• 扩容系统盘不会更改您实例的IP地址和MAC地址。

扩容上限和计费

扩容新值必须大于现有容量，且不能超过该类磁盘的容量上限，系统盘扩容上限如下表所示。

系统盘类型	扩容磁盘上限（GB）
增强型SSD	2048
通用型SSD	2048
高性能云磁盘	2048

扩容增加的相应费用，请参考价格云磁盘定价。

系统盘扩容

1. 登录 云服务器BCC控制台。
2. 在左侧导航栏中选择**实例**，进入实例列表页面。
3. 选择需要进行系统盘扩容的实例的名称，进入实例详情页面，然后点击**本实例磁盘**进入磁盘详情页面。
4. 选中磁盘，在更多操作中选择**扩容及变配**，在弹出的菜单栏中点击**确认**，进入扩容页面。
5. 在**变更配置**选择您需要扩容的容量（必须大于现有容量）

   例如，您一块预付费系统盘的原容量为40G，需要加到100GB，只需为60GB的容量付费。查看配置费用和已选配置信息正确后，点击**确认变更**，进行订单付费。
6. 在**确认支付**之后，**重启**实例，扩容容量生效。

通过API扩容

您可以使用**磁盘扩容及类型变更接口**为指定磁盘变配扩容，API接口请参考磁盘相关接口。

快照

快照概述

概述

快照是某一时间节点一块 CDS 云磁盘的一种数据状态文件。您可以通过在某一时刻为 CDS 云磁盘创建快照的方式，对该时刻 CDS 云磁盘的数据进行保存。通过快照，您可以实现磁盘数据备份，磁盘数据恢复以及磁盘镜像的制作。

> **说明：**
>
> - 目前百度智能云快照不会占据用户已购买的存储空间;
> - CDS 快照服务已于 2020 年 3 月 13 日开始计费，详情请查看快照计费帮助文档；
> - 百度智能云快照采用多副本冗余存储的方式对快照进行保存，快照数据具有极高的可靠性。

使用场景

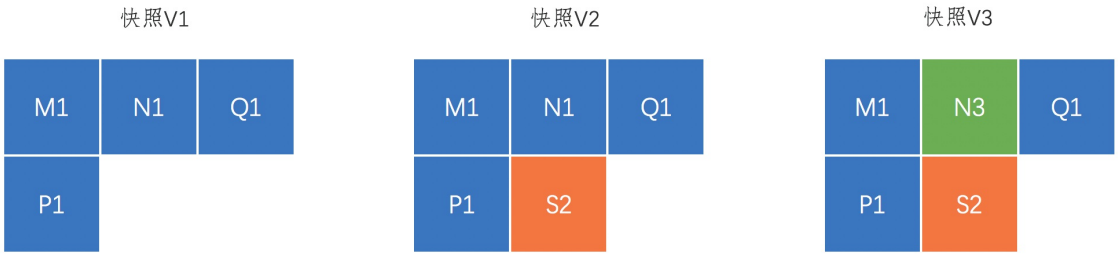
快照是一种高可靠性的数据备份方式，您可以在多种场景中对 CDS 云磁盘使用快照。

- **磁盘备份**：您可以通过快照定期对 CDS 云磁盘进行备份，以防丢失磁盘中的重要数据；
- **容灾备份**：若您要对 BCC 实例进行操作系统切换，或者要对服务器进行重大更新，强烈建议对磁盘进行快照操作，以免误删重要数据；
- **磁盘变配**：若您要进行磁盘扩容，磁盘类型变更等操作时，强烈建议先对磁盘进行快照操作，以防磁盘中的重要数据造成误删；
- **磁盘卸载**：若您要对磁盘进行卸载或释放，建议先对磁盘进行快照操作，以防丢失重要数据；
- **磁盘回滚**：若您想将磁盘中数据回滚到过去某一时刻的状态，可以使用快照进行磁盘回滚操作；
- **自定义镜像**：若您想搭建和已有系统盘相同的系统盘环境，可以通过使用快照创建新的系统盘来实现自定义镜像。

快照原理

云磁盘 CDS 采用分布式存储技术。当您创建云磁盘并格式化之后，逻辑磁盘会划分为若干数据块。当数据写入云磁盘时，本质是写入到相应的数据块。当某数据块中的数据发生变化时（如数据写入，数据删除），只有该数据块中的数据会发生变化。

相应的，快照服务采用了业界先进的增量快照技术，即您在磁盘数据变化前后分别为该磁盘创建的快照，只有已变数据块对应的快照数据会发生变化，其他未变数据块对应的快照数据保持不变。增量快照技术可以大幅度减少快照数据的相似冗余，使您的快照费用降到最低。



如上图中所示：快照 V1、V2 和 V3 分别是为磁盘创建的第一份、第二份和第三份快照。

创建首份快照

快照 V1 为该磁盘创建的第一份快照。该快照数据分为 4 个数据块，分别是：M1、N1、Q1、P1。

> **说明：**
>
> - 2017 年10 月之前创建的大部分磁盘，其首份快照为全量快照，即不检测磁盘中的有效写入数据，会对磁盘中全部容量

> 创建快照。比如，购买的云磁盘大小为 50 GB，其中用户写入到该磁盘的数据只有 10 GB，那么首份快照的容量大小为 50 GB，而不是 10 GB；
>
> - 2017 年10 月之后创建的磁盘，其首份快照便为增量快照。如上面的例子中，该磁盘创建的首份快照的容量大小为 10 GB。

创建新的快照

所有云磁盘的从第二份快照开始，均为增量快照。上图中，快照 V2 是该磁盘创建的第二份快照，在原有 4 个数据块的基础上增加了一个数据块：S2，其他数据块全部指向 V1 中的四个数据块，并不会新增数据相同的重复数据块；

快照 V3 是该磁盘创建的第三份快照。由于 V3 相较 V2 而言，只有 N3 这一数据块中的数据发生了变化，因此，V3 中包含的 M1、Q1 和 P1 数据块依旧指向 V1 中的 M1、Q1 和 P1，S2 指向 V2 中的数据块 S2，N3 为本次新增的数据块。

删除快照

当用户将快照 V3 删除时，只删除数据块 N3 ，其他存在指向关系的数据块会被保留。因此，只删除快照 V3 不会导致 V1 和 V2 的不可用。

回滚快照

当用户希望将磁盘回滚到快照V3的状态时，数据块 M1、N3、Q1、P1 和 S2 将全部恢复到磁盘上，保证数据的精确回滚。

快照链中数据量

该磁盘对应快照链中包含的块总量为：4 + 1 + 1 = 6，而非 4 + 5 +5 = 14。

快照方式

您可以通过手动快照或者设置自动快照的方式对磁盘进行快照操作。

- **手动快照**：手动快照需要您主动发起，详细操作请查看创建磁盘快照；
- **自动快照**：自动快照您可以创建时添加自动快照策略，详细操作请查看自动快照。

快照上限

每个云账号最大可创建 512 份快照数据，包括 256 份手动快照和 256 份自动快照。

- 若磁盘的自动快照额度达到上限，创建时间最早的自动快照会被自动删除，新的自动快照不受影响。
- 若磁盘的手动快照额度达到上限，创建新的手动快照操作被禁止，请及时清理不需要的手动快照数据。当磁盘手动快照数量达到上限后，在 CDS 控制台的磁盘列表中，快照已达上限的 CDS 磁盘会在名称后方出现一个红色闹钟的小图标，请您留意。

快照生效时间

CDS的手动快照或自动快照，从创建到成功的生效时间会根据快照大小不同而存在差异。平均时间会在秒级到分钟级不等。

创建磁盘快照

背景信息

您可以通过创建快照作为某一时间云磁盘数据状态的备份，用于日常数据备份、服务器重大更新备份、磁盘数据回滚、制作自定义镜像等场景。本文介绍如何在BCC控制台上创建磁盘快照。关于实例快照的说明，可以点击实例快照进行查看。

前提条件

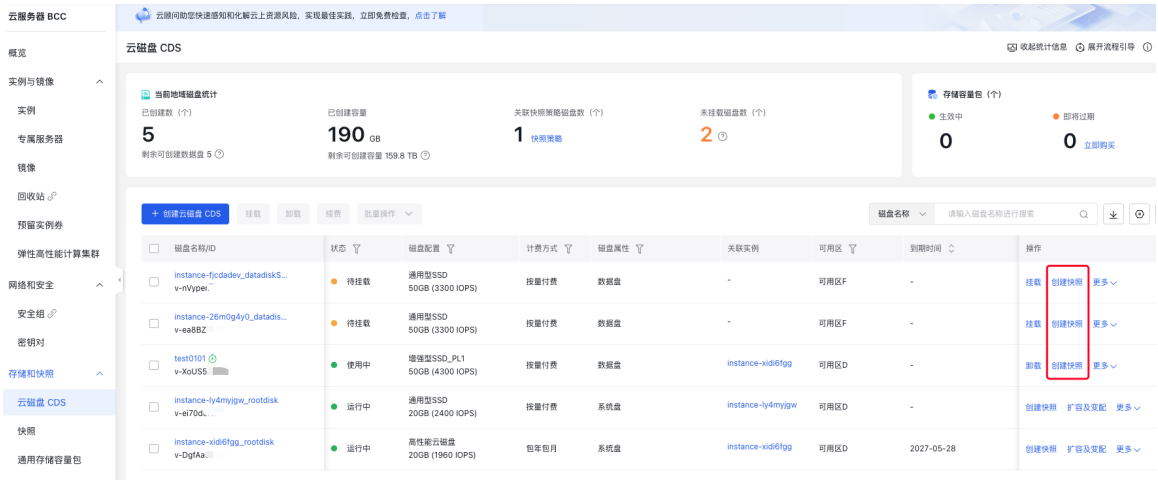
- 若您为 **数据盘** 进行快照操作，必须当数据盘在 **已挂载** 或者 **未挂载** 状态下进行。为了您的数据安全性，其他状态下快照操作将不被允许；
- 若您为 **系统盘** 进行快照操作，必须当系统盘在 **运行中** 或者 **正常** 状态下进行操作。

注意事项

- 每块云磁盘 CDS 共可创建 512 份快照数据，包括 256 份手动快照和 256 份自动快照。
  - 若磁盘的自动快照额度达到上限，创建时间最早的自动快照会被自动删除，新的自动快照不受影响。
  - 若磁盘的手动快照额度达到上限，创建新的手动快照操作被禁止，请及时清理不需要的手动快照数据。
- 创建到成功的生效时间会根据快照大小不同而存在差异。平均时间会在秒级到分钟级不等。
- 快照内容只包含开始创建时已经写入磁盘的内容，不包括还在缓存中的数据。为了保证数据的完整性，建议您先卸载磁盘，或者确认已停止所有的文件写操作。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 点击磁盘页，选择需要创建快照的磁盘，点击右侧的"创建快照"按钮。

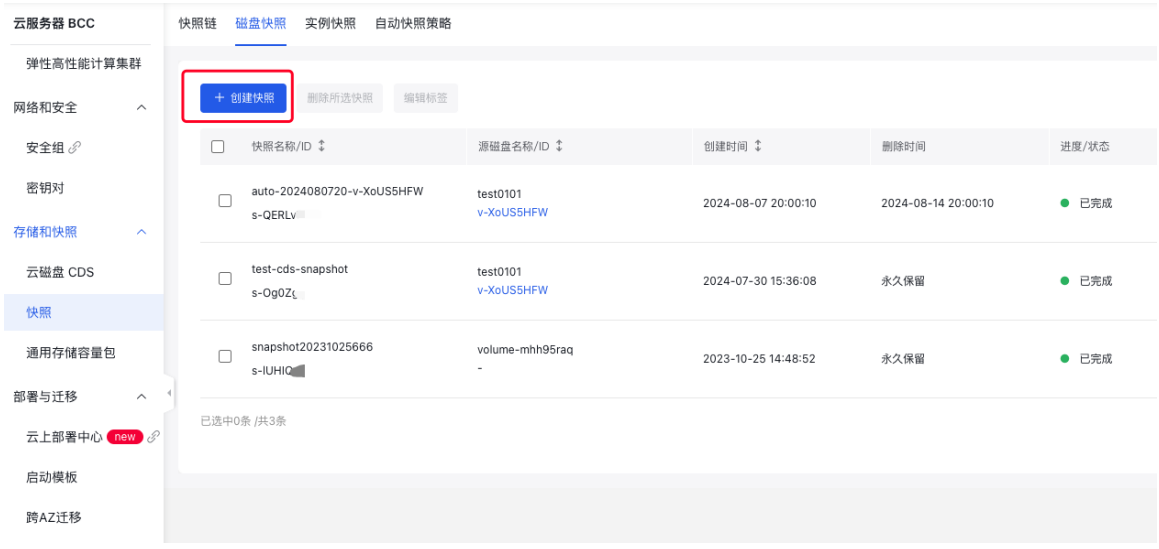


3. 在弹出的 **创建快照** 对话框，输入快照名称，选择是否需要创建"实例快照"（实例快照为系统盘和数据盘快照，无需单独为单块磁盘创建快照），如果只是针对单块磁盘创建快照，可以选择"不创建"并单击 **确定**。

4. 实例的状态将变成 **创建快照中**，当状态重新变为 **运行中**，证明已经完成了快照的创建。

其他方式

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航选择 “快照 - 快照列表”，进入快照列表页面。
3. 单击 **创建快照**。



4. 在弹出的 **创建快照** 对话框中，完成快照类型、磁盘ID、快照名称的设置后，单击 **确定**。

5. 此时快照列表会增加一条状态为 **创建中** 的快照，当状态变为 **已完成** 时证明快照创建成功。

常见问题

- **创建快照时BCC会关闭吗？**
  - 创建快照时，BCC不需要关闭。
- **BCC释放后，自定义镜像和快照是否还存在？**
  - 自定义镜像：创建的自定义镜像只要不删除，将会一直保留。
  - 快照：会随着BCC释放而被清除。

通过API创建快照

您可以通过 创建快照接口 从指定的磁盘创建快照。

回滚快照

背景信息

回滚快照是指将磁盘的状态恢复到创建快照时的系统状态。例如您在2015年5月20日17点整在磁盘A创建了快照A，如果您在2015年5月21日17点整选择回滚快照A，则磁盘A的系统信息将恢复到2015年5月20日17点整的状态，2015年5月20日17点到2015年5月21日17点之间的数据存储将丢失。

前提条件

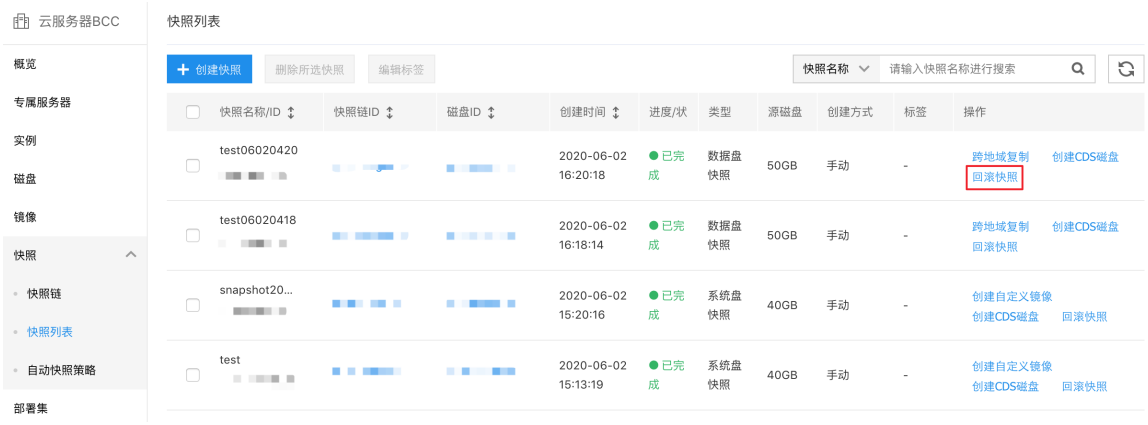
- 云磁盘已创建快照，而且要回滚的云磁盘当前没有正在创建的快照。
- 对于由数据盘创建的快照，回滚之前请先将数据盘从挂载的BCC实例中卸载。
  - Windows操作系统的BCC实例，需要先在Windows磁盘管理中删除此磁盘后，才能在控制台中卸载此CDS磁盘。否则，控制台将提示报错信息。
  - Linux操作系统的BCC实例，为了保证安全卸载，建议先在实例操作系统中执行卸载（umount命令）操作后，再在控制台执行卸载操作。
  - 具体操作请参见卸载CDS磁盘。

注意事项

回滚快照 操作不可逆，请谨慎使用。使用前请备份好相关存储数据，以避免对系统造成影响。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航选择 “快照 - 快照列表”，进入快照列表页面。
3. 选择需要操作的快照，在最右侧操作栏单击 **回滚快照**，如下图所示。



4. 在弹出的 **回滚前确认** 对话框中，单击 **确认**。
5. 在安全验证弹窗中，输入短信验证码，单击 **确认**，完成回滚快照操作。

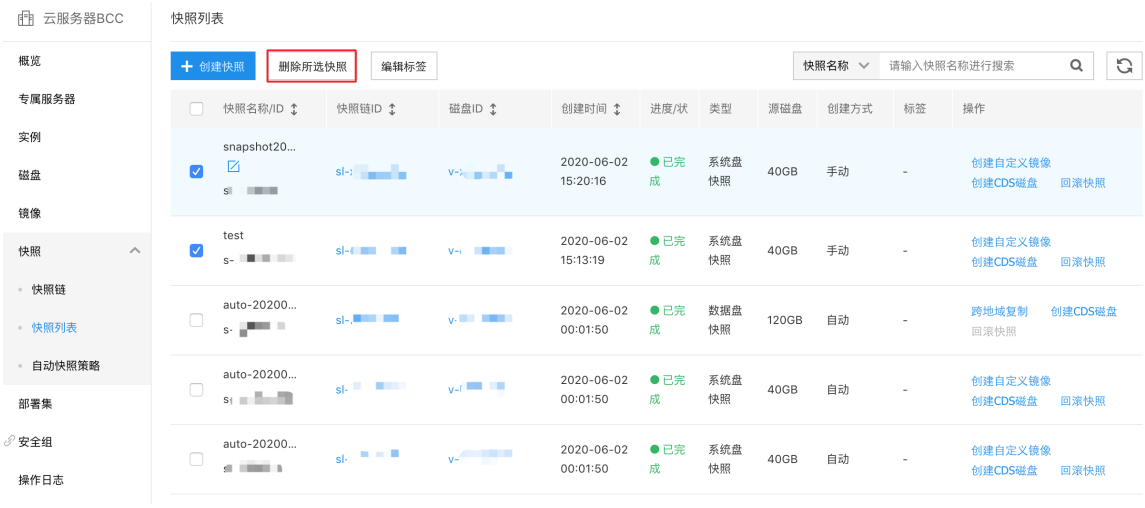
删除快照

背景信息

删除快照将删除您在某磁盘上创建的某时刻的快照备份信息，删除后您无法通过回滚操作回到以前的系统状态，且删除快照无法恢复，请谨慎操作。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航选择 “快照 - 快照列表”，进入快照列表页面。
3. 选择一个或者多个需要删除的快照，单击顶部 **删除所选快照**，如下图所示。



4. 在弹出的 **删除前确认** 对话框，单击 **确定**。

> 注意：快照删除后不可恢复，请谨慎操作。

5. 在安全验证弹窗中，输入短信验证码，单击 **确定** 完成删除快照操作。

通过API删除快照

您可以通过 删除快照接口 删除指定ID的快照。

自动快照

自动快照策略可以帮助您周期性的为云磁盘创建快照。在一些核心业务中，磁盘中数据重要程度高、数据变更速度快。为了保证数据的安全性，开发者需要周期性创建磁盘快照。基于此类场景，百度智能云为您提供了自动快照功能，您只需提前定义好创建快照的磁盘和时间周期，系统将会自动为您实现快照操作。

使用限制

- 您在在同一个区域内最多可创建100条快照策略；
- 快照策略和要绑定的 CDS 云磁盘必须在同一个区域内；
- 磁盘只有挂载到一个实例后才可关联快照策略，且一个磁盘只能关联到一个快照策略。

相关操作

- 创建自动快照策略
- 修改自动快照策略
- 将快照策略与磁盘绑定
- 将磁盘与快照策略关联

网络

弹性网卡

弹性网卡概述

弹性网卡介绍

百度智能云弹性网卡可在多个云服务器间自由迁移。通过在云服务器上绑定多个弹性网卡，实现高可用网络方案。也可以在弹性网卡上绑定多个私网 IP，每个私网IP都可以绑定一个EIP，用户在配置好IP地址后通过配置应用策略即可实现不同业务流量以不同源IP从网卡流出，实现单主机多IP部署。

使用场景

- 部署高可用集群 弹性网卡是系统架构中的关键组件，满足系统高可用架构对于单实例多网卡的需求。

- 快速故障迁移 通过将弹性网卡从云主机实例卸载后再挂载到另外一台云主机实例，将故障实例上的业务流量快速迁移到备用实例，实现服务的容灾部署。
- 精细化网络管理 实例可以挂载多个弹性网卡，弹性网卡可分别位于不同的子网，可独立配置安全组、ACL安全访问规则，从而对每个弹性网卡的流量进行安全访问控制，同时可实现不同业务间数据隔离。

云主机的网卡类型

类型	说明
主网卡	虚机实例上的原生网卡（eth0），在虚机实例创建时自动生成并配置。主网卡不可进行插拔、迁移等操作，与云主机的生命周期保持一致。 BCC主网卡不属于弹性网卡，不在弹性网卡列表中，主网卡在BCC详情页面中配置。
弹性网卡	为用户单独创建并挂载在虚机实例上，用户可以在云服务器创建、停机或运行时挂载弹性网卡。用户挂载并配置好网卡后通过配置系统路由即可实现不同流量走不同网卡。

产品限制

- BCC实例与弹性网卡必须在同一可用区、同一VPC，可以在不同子网，挂载不同的安全组。
- 在一台实例上挂载多个弹性网卡不能提高实例内网带宽性能。
- 每个VPC未挂载主机的弹性网卡最大数量500个，已挂载主机的弹性网卡不占用配额。
- 单网卡上IP数量最少1个，最多40个。
- 云主机可挂载的弹性网卡数量与实例规格相关，6系及以前的机型实例最多可挂载的弹性网卡数量为min{cpu核数，8}，即当云主机cpu核数小于等于8时，最大可挂载的弹性网卡数量等于云主机cpu核数；当云主机cpu核数大于8时，最大可挂载8个弹性网卡。7系及以后随实例规格变化，详情可见实例规格。

云服务器绑定的网卡上可配置IP数量

内存	IP数量
1G	2
(1-8]G	8
(8-32]G	16
(32-64]G	30
大于64G	40

创建弹性网卡

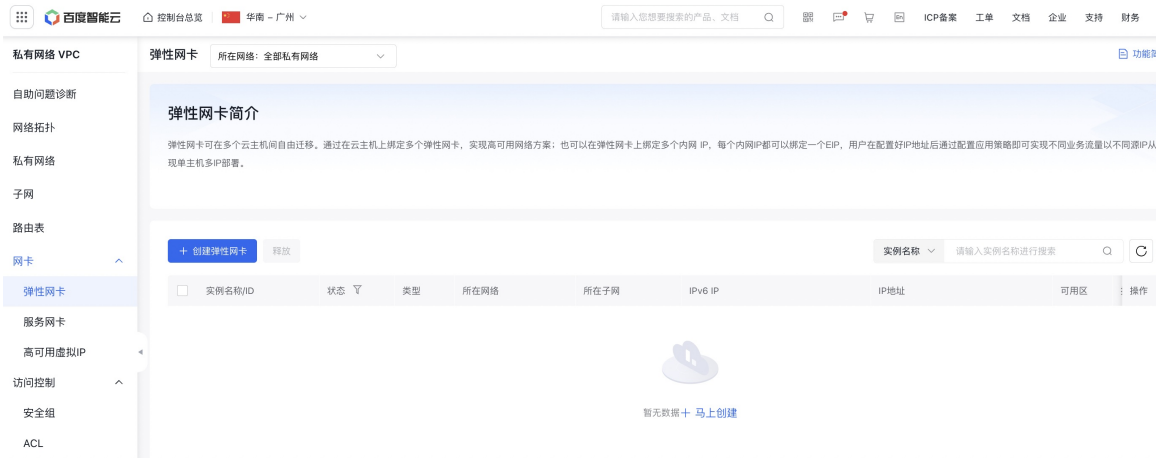
本章节为您介绍如何在控制台上创建弹性网卡。

前提条件

在相应地域已经创建私有网络VPC。

创建弹性网卡

1. 登录 私有网络VPC控制台。
2. 在左导航栏选择"弹性网卡"，点击"创建弹性网卡"。



3. 填写下列配置信息

配置项	说明
当前地域	支持北京、保定、广州、苏州、成都、武汉、香港，通过左上角地域进行切换
网卡名称	用户自定义弹性网卡名称。 命名规则符合：大小写字母、数字以及-_/.特殊字符，必须以字母开头，长度1-65
类型	弹性网卡类型，支持普通型，RDMA型。RDMA型网卡仅能挂载支持RDMA网络的BCC实例
所在子网	弹性网卡所属的子网
描述	编辑弹性网卡相关描述信息
安全组	弹性网卡绑定的安全组，最少1个，最多10个
IP地址	默认自动分配一个主IP，可自行添加辅助IP
标签	编辑弹性网卡绑定的标签信息

4. 点击“确定”，弹性网卡创建完成。

后续步骤

弹性网卡创建成功后，您可以执行挂载弹性网卡。

挂载弹性网卡

前提条件

1. 云服务器挂载弹性网卡前，已在VPC中创建弹性网卡实例。
2. 弹性网卡处于可用状态。

相关影响

- 云服务器镜像暂时无法自动识别弹性网卡，在挂载主机后，您需要配置弹性网卡使其能被识别。
- 挂载主机后，如需卸载，在页面中点击“卸载网卡”即可完成卸载。

- 云服务器释放时，可选择弹性网卡是否关联释放或解绑。

挂载弹性网卡

1. 在BCC实例列表页，选择已创建的BCC实例，点击进入详情页面。
2. 导航栏选择 **网卡**，进入网卡页面。
3. 在弹性网卡中，点击 **挂载网卡**，出现挂载网卡弹框，选择需要挂载的弹性网卡。
4. 点击 **确定**，完成挂载弹性网卡。

配置弹性网卡

根据您的实例使用的镜像不同，部分镜像需要配置弹性网卡，才能正常使用弹性网卡。

前提条件

您已经将弹性网卡绑定到BCC实例上。

准备工作

弹性网卡配置工具

- 将弹性网卡绑定到BCC/EBC实例、且实例内识别到弹性网卡后，根据镜像版本分别执行以下脚本配置ip和路由表。
  CentOS/CentOS Stream/RockyLinux/BaiduLinux/AlmaLinux 请使用CentOS 版本的脚本。

  CentOS:bcc_elastic_net_centos_20241121.sh

  Ubuntu:bcc_elastic_net_ubuntu_20241120.sh

  Debian:bcc_elastic_net_debian_20241211.sh

- Windows系统需要手动配置。

配置工具会创建静态路由，重启网络或者实例后，无需再次进行路由的设置。

运行配置工具

- 进入创建的BCC服务器，并为配置工具添加可执行权限。
- 完成挂载弹性网卡操作后，执行配置工具。
- 对网卡进行修改IP，解绑及重新绑定等操作时，需要再次运行配置工具。

如下所示：

```
----------------------------------
Disable NetworkManager.
Begin to config network...
  create/recreate network config file [OK]
Begin to create route table...
  add 192.168.32.212 to route table: rt_ens3 [OK]
  add 192.168.32.228 to route table: rt_ens3 [OK]
  add 192.168.32.209 to route table: rt_ens6 [OK]
  add 192.168.32.210 to route table: rt_ens6 [OK]
  add 192.168.32.211 to route table: rt_ens6 [OK]
  add 192.168.32.226 to route table: rt_ens7 [OK]
  add 192.168.32.227 to route table: rt_ens7 [OK]
  add 192.168.32.229 to route table: rt_ens7 [OK]
Restart network...  [OK]
----------------------------------
```

配置完成后，可通过ip addr命令查看IP是否已启用（以Centos为例）

```
 [root@instance-h1q92lvr ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UP qlen 1000
    link/ether fa:16:3e:7d:2d:ad brd ff:ff:ff:ff:ff:ff
    inet 192.168.32.212/20 brd 192.168.47.255 scope global eth0
    inet 192.168.32.228/20 brd 192.168.47.255 scope global secondary eth0:1
    inet6 fe80::f816:3eff:fe7d:2dad/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UP qlen 1000
    link/ether fa:16:3e:a6:69:8f brd ff:ff:ff:ff:ff:ff
    inet 192.168.32.209/20 brd 192.168.47.255 scope global eth1
    inet 192.168.32.210/20 brd 192.168.47.255 scope global secondary eth1:1
    inet 192.168.32.211/20 brd 192.168.47.255 scope global secondary eth1:2
    inet6 fe80::f816:3eff:fea6:698f/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UP qlen 1000
    link/ether fa:16:3e:93:e5:44 brd ff:ff:ff:ff:ff:ff
    inet 192.168.32.226/20 brd 192.168.47.255 scope global eth2
    inet 192.168.32.227/20 brd 192.168.47.255 scope global secondary eth2:1
    inet 192.168.32.229/20 brd 192.168.47.255 scope global secondary eth2:2
    inet6 fe80::f816:3eff:fe93:e544/64 scope link
       valid_lft forever preferred_lft forever
```

配置弹性网卡

根据BCC实例的操作系统，选择配置弹性网卡的方式：

- Windows系统配置弹性网卡
- CentOS系统配置弹性网卡
- Ubuntu/Debian系统配置弹性网卡

Windows系统配置弹性网卡

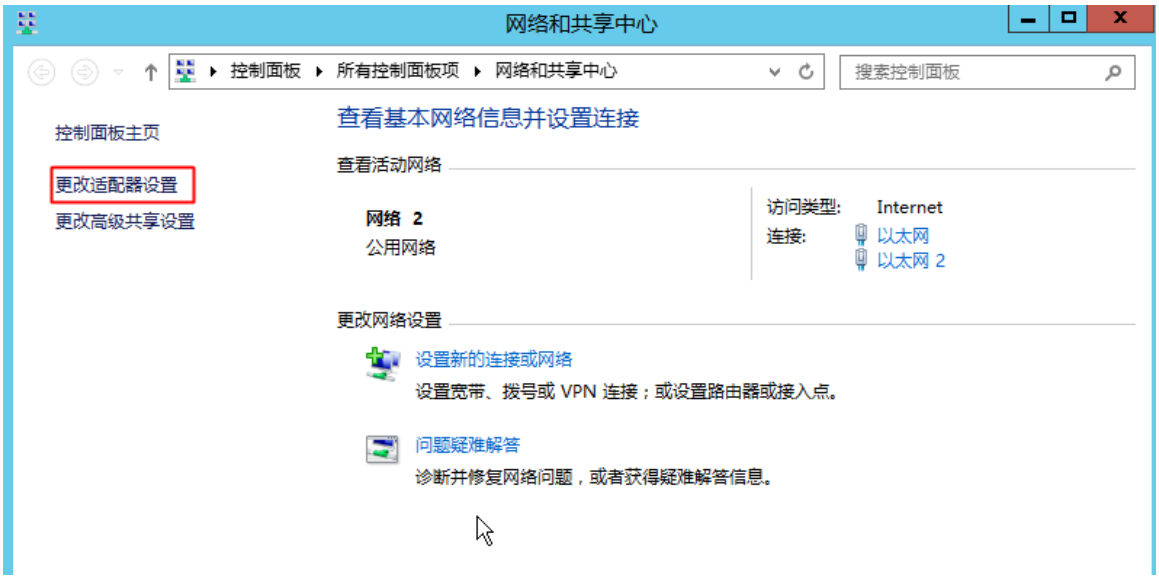
准备工作

在给您的Windows系统实例配置弹性网卡前，需要先将弹性网卡绑定在实例上。您可以根据需求在一个弹性网卡上分配一个或多个辅助私网IP地址。通过使用多个私网IP地址，能提高Windows实例高利用率和实现负载故障时的流量转移。

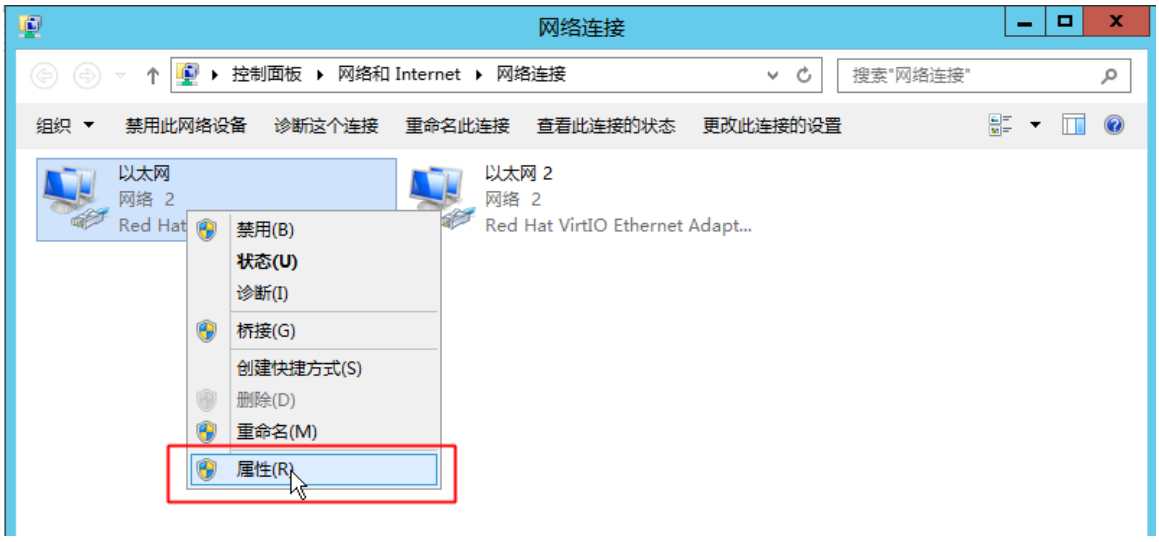
配置方法

以Windows Server 2012R2为例说明Windows系统配置辅助IP的过程。

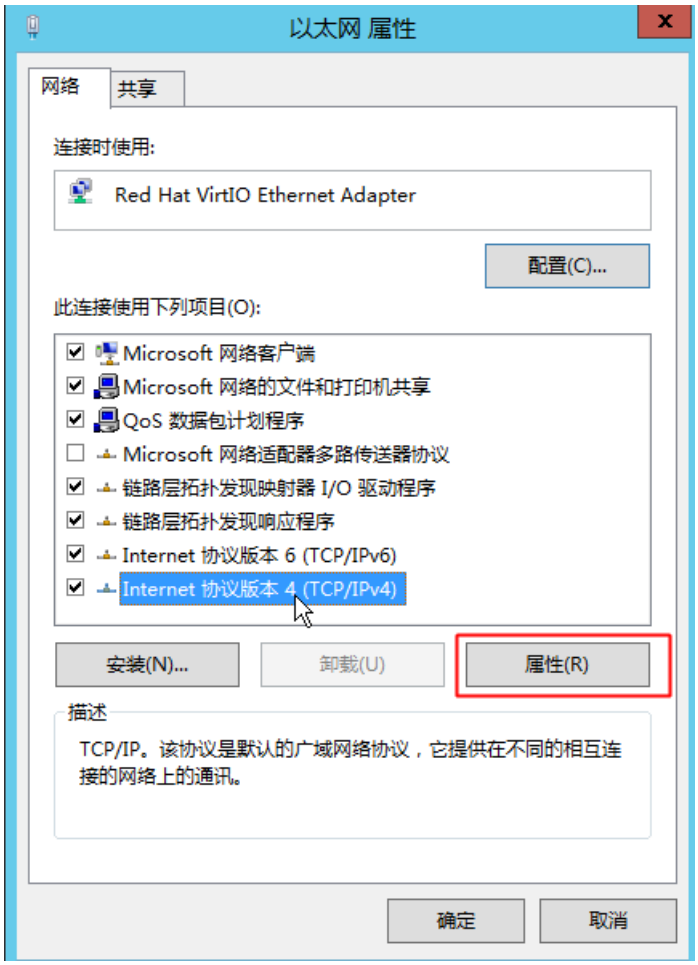
1.打开网络与共享中心，选择更改适配器设置。



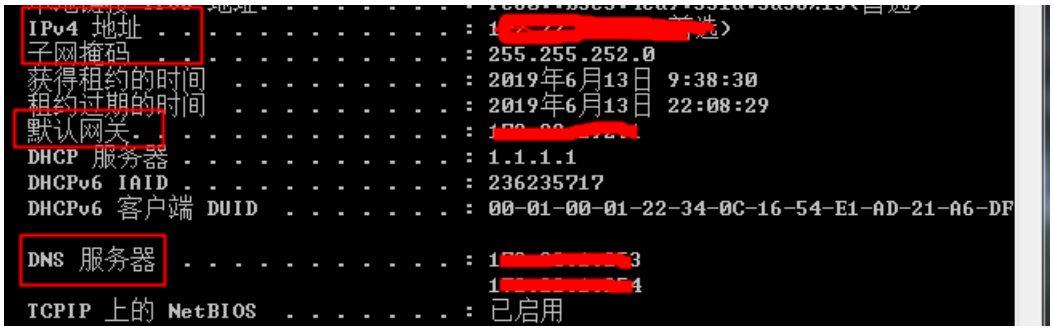
2.在弹出的页面中选择一个网卡，右键属性。



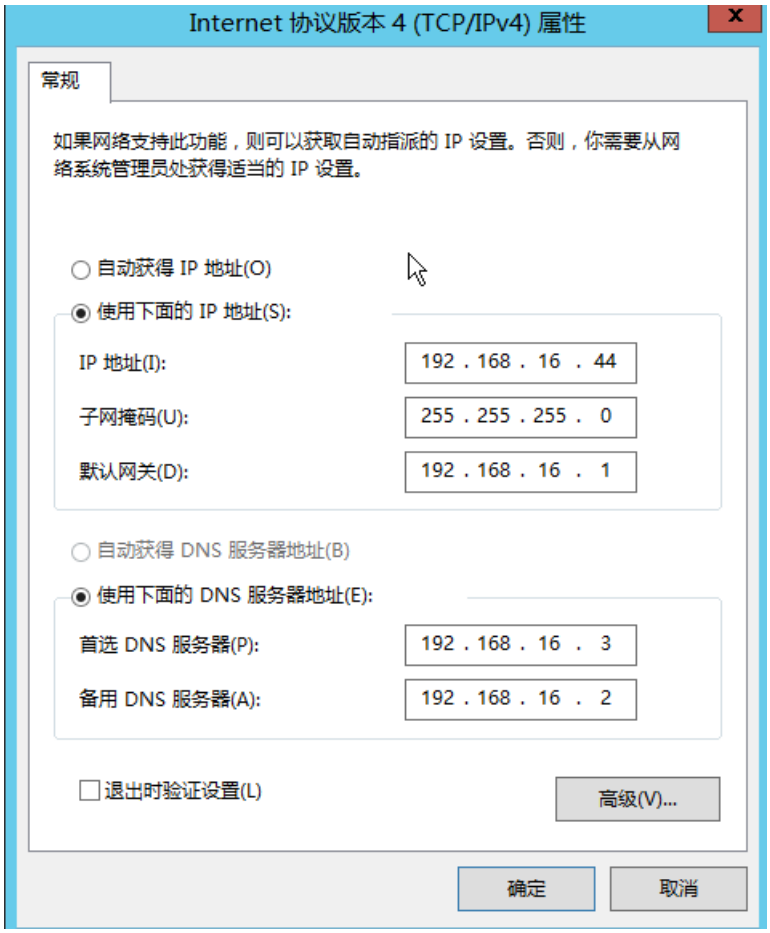
3.选择TCP/IPV4，点击属性。



4.在弹出的对话框中选择“使用下面的ip地址”，这时需要填写ip地址，子网掩码，DNS服务器等信息，这些信息可以通过在命令行下输入命令 ipconfig /all 获取，输完命令或显示很多信息，记下其中的IPv4 Address (IPv4 地址)、Subnet Mask (子网掩码)、Default Gateway (默认网关) 和 DNS Servers (DNS 服务器)这些值。

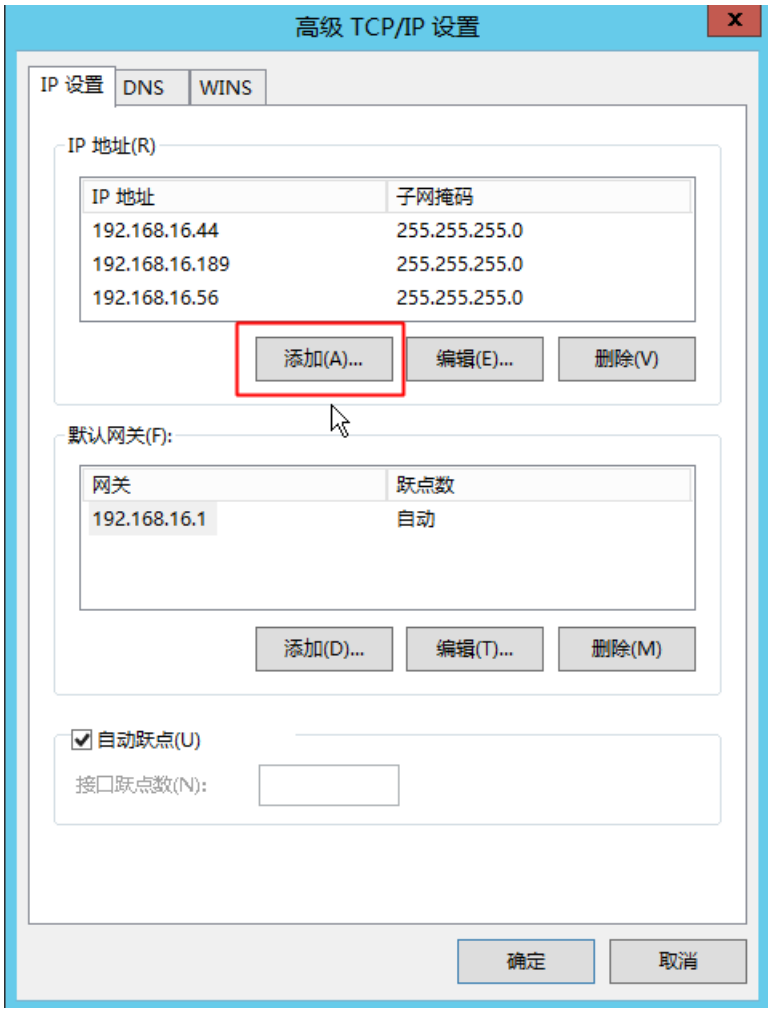


5.接下来回到刚才弹出的对话框配置静态ip地址。依次填入获取到的IP地址，子网掩码，默认网关，DNS服务器地址等信息，如下图所示，点击确定，设置过程中会有间断数秒钟的断网，这个是正常现象。



6.配置好静态IP后，如果您在控制台添加了辅助ip，还需要继续添加辅助ip。

7.在刚才的“Internet协议版本 4 TCP/IP属性”对话框界面点击高级，在弹出的对话框中点击添加，进行添加IP的操作，将在控制台添加的辅助IP依次填写就完成了。



8.要验证辅助ip确实已添加至操作系统中，可以在命令行下输入ipconfig /all 查看刚才添加的ip是否已经存在。

CentOS系统配置弹性网卡

准备工作

1. 为防止 cloud-init 重置您的自定义网络配置，请禁用cloud-init中的网络配置，编辑 /etc/cloud/cloud.cfg，添加以下内容。

   network: {config: disabled}

2. BCC实例中安装了NetworkManager，NetworkManager可能会修改路由设置，因此需要禁用 NetworkManager。

- CentOS 7 禁用NetworkManager

```
 systemctl stop NetworkManager
systemctl disable NetworkManager
```

- CentOS 6 禁用NetworkManager

```
 service NetworkManager stop
chkconfig NetworkManager off
```

配置网卡

1. 获取网络接口名称（网卡名称）

- BCC实例添加弹性网卡后，可通过以下命令获取弹性网卡的名称。

```
ls /sys/class/net
```

- 输出结果（示例）

```
eth0 eth1 eth2 lo
```

- BCC实例默认的主网卡名称为eth0或ens3（部分BCC实例主网卡名为ens3）。以上结果中，添加的弹性网卡为eth1，eth2。

2. 为弹性网卡创建配置文件，以eth1为例。

```
vim /etc/sysconfig/network-scripts/ifcfg-eth1
```

添加以下内容：

```
 DEVICE=eth1
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
PEERDNS=no
IPV6INIT=no
PERSISTENT_DHCLIENT=yes
HWADDR=fa:16:3e:6a:14:b5
DEFROUTE=no
```

> **注意：**
>
> - 需要将弹性网卡的MAC地址写入到配置文件中。（可通过cat /sys/class/net/[网卡名称]/address查看弹性网卡的MAC地址）。

- 为了防止更改默认路由，需要在弹性网卡的配置文件中添加DEFROUTE=no选项。

如果网卡上有多个IP，需要将网卡上的辅助IP添加到配置文件中。

```
 DEVICE=eth1
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
PEERDNS=no
IPV6INIT=no
PERSISTENT_DHCLIENT=yes
HWADDR=fa:16:3e:6a:14:b5
DEFROUTE=no
IPADDR1=192.168.32.210
PREFIX1=24
IPADDR2=192.168.32.211
PREFIX2=24
```

3. 重启网络

```
systemctl restart network / service network restart
```

配置路由

为了保证数据能够通过正确的网络接口（网卡），您必须为弹性网卡创建新的路由表及规则。

1. 创建路由表

编辑 /etc/iproute2/rt_tables，分别为每一个弹性网卡创建一个路由表。

```
 10 rt_eth1
20 rt_eth2
```

2. 设置路由规则（以eth1为例）

在终端中运行以下命令：

```
 ip route add default via 192.168.32.1 dev eth1 table rt_eth1

ip route add 192.168.32.209 dev eth1 table rt_eth1
ip route add 192.168.32.210 dev eth1 table rt_eth1
ip route add 192.168.32.211 dev eth1 table rt_eth1

ip rule add from 192.168.32.209 lookup rt_eth1
ip rule add from 192.168.32.210 lookup rt_eth1
ip rule add from 192.168.32.211 lookup rt_eth1
```

注：

- 192.168.32.209 / 210 / 211为弹性网卡的IP。
- 192.168.32.1为弹性网卡的默认网关。

设置完成后，可以通过 ip route show table rt_eth1 查看弹性网卡上每个 IP 的路由。

3. 创建弹性网卡的静态路由文件（以eth1为例）

为弹性网卡创建静态路由和规则文件，当BCC实例启动或重启网络时都会保留路由和规则，无需再次手动添加。

创建 route-eth1 静态路由文件，vim /etc/sysconfig/network-scripts/route-eth1，添加以下内容：

```
 default via 192.168.32.1 dev eth1 table rt_eth1

192.168.32.209 dev eth1 table rt_eth1
192.168.32.210 dev eth1 table rt_eth1
192.168.32.211 dev eth1 table rt_eth1
```

创建或编辑 rule-eth1 的规则文件，vim /etc/sysconfig/network-scripts/rule-eth1，添加以下内容：

```
 from 192.168.32.209 lookup rt_eth1
from 192.168.32.210 lookup rt_eth1
from 192.168.32.211 lookup rt_eth1
```

Ubuntu/Debian系统配置弹性网卡

配置前准备工作

1. 为防止 cloud-init 重置您的自定义网络配置，请禁用cloud-init中的网络配置，编辑 /etc/cloud/cloud.cfg ，添加以下内容:

   ```
   network: {config: disabled}
   ```

2. BCC实例中安装了NetworkManager，NetworkManager可能会修改路由设置，因此需要禁用 NetworkManager，并保证系统重启后，NetworkManager仍处于禁用状态。

- Ubuntu14 需要卸载NetworkManager

  ```
   service network-manager stop
  apt-get --purge remove -y network-manager
  ```

- CentOS 6 禁用NetworkManager

  ```
   systemctl stop NetworkManager
  systemctl disable NetworkManager
  ```

配置网卡

1. 获取网络接口名称（网卡名称）

BCC实例添加弹性网卡后，可通过以下命令获取弹性网卡的名称。

```
ls /sys/class/net
```

输出结果（示例）

```
eth0 eth1 eht2 lo
```

BCC实例默认的主网卡名称为eth0或ens3（部分BCC实例主网卡名为ens3）。以上结果中，添加的弹性网卡为eth1，eth2。

2. 编辑 /etc/network/interfaces

编辑/etc/network/interfaces网络接口文件，修改为以下内容：

```
 auto lo
iface lo inet loopback
source /etc/network/interfaces.d/*
```

对于BCC Ubuntu16实例，编辑 /etc/network/interfaces.d/50-cloud-init.cfg，将其修改为以下内容：

```
 auto ens3
iface ens3 inet dhcp
```

同时修改文件名为ens3.cfg

```
mv /etc/network/interfaces.d/50-cloud-init.cfg  /etc/network/interfaces.d/ens3.cfg
```

对于BCC Ubuntu20实例，编辑 /etc/netplan/01-netcfg.yaml，将其修改为以下内容：

```
network:
  version: 2
  renderer: networkd
  ethernets:
     eth0:
        dhcp4: yes
     eth1:
        dhcp4: yes
     eth2:
        dhcp4: yes
```

3. 为弹性网卡创建配置文件，以eth1为例。

在/etc/network/interfaces.d/目录下新建eth1.cfg文件

```
vim /etc/network/interfaces.d/eth1.cfg
```

添加以下内容：

```
 auto eth1
iface eth1 inet dhcp
hwaddress=fa:16:3e:6a:14:b5
```

> **注意：**
>
> - 需要将弹性网卡的MAC地址写入到配置文件中。（可通过cat /sys/class/net/[网卡名称]/address查看弹性网卡的MAC地址）。

如果网卡上有多个IP，需要将网卡上的辅助IP添加到配置文件中。

```
 auto eth1
iface eth1 inet dhcp
hwaddress=fa:16:3e:6a:14:b5

iface eth1 inet static
address 192.168.32.210
netmask 255.255.240.0

iface eth1 inet static
address 192.168.32.211
netmask 255.255.240.0
```

4. 创建 restrict-default-route 文件，以防止默认路由被覆盖。

vim /etc/dhcp/dhclient-enter-hooks.d/restrict-default-route，添加以下内容：

```
  case ${interface} in
  eth0)          #  ubuntu16/debian9为ens3
    ;;
  *)
    unset new_routers
    ;;
esac
```

5. 重启网络。

- Ubuntu14

  ```
  ifdown eth1 && ifup eth1
  ```

- Ubuntu16

  ```
  systemctl restart networking
  ```

- Ubuntu20

  ```
  netplan apply
  ```

配置路由

为了保证数据能够通过正确的网络接口（网卡），您必须为弹性网卡创建新的路由表及规则。

1. 创建路由表

编辑 /etc/iproute2/rt_tables，分别为每一个弹性网卡创建一个路由表。

```
  10 rt_eth1
20 rt_eth2
```

2. 设置路由规则（以eth1为例）。

在终端中运行以下命令：

```
  ip route add default via 192.168.32.1 dev eth1 table rt_eth1

ip route add 192.168.32.209 dev eth1 table rt_eth1
ip route add 192.168.32.210 dev eth1 table rt_eth1
ip route add 192.168.32.211 dev eth1 table rt_eth1

ip rule add from 192.168.32.209 lookup rt_eth1
ip rule add from 192.168.32.210 lookup rt_eth1
ip rule add from 192.168.32.211 lookup rt_eth1
```

注：

- 192.168.32.209 / 210 / 211为弹性网卡的IP
- 192.168.32.1为弹性网卡的默认网关

设置完成后，可以通过 ip route show table rt_eth1 查看弹性网卡上每个 IP 的路由。

3. 创建弹性网卡的静态路由文件（以eth1为例）。

编辑/etc/network/interfaces.d/eth1.cfg

```
 auto eth1
iface eth1 inet dhcp
hwaddress=fa:16:3e:6a:14:b5

iface eth1 inet static
address 192.168.32.210
netmask 255.255.240.0

iface eth1 inet static
address 192.168.32.211
netmask 255.255.240.0

# default gateway for eth1
up ip route add default via 192.168.32.1 dev eth1 table rt_eth0

# route for every IP
up ip route add 192.168.32.210 dev eth1 table rt_eth0
up ip route add 192.168.32.211 dev eth1 table rt_eth0

# policy rule for every IP
up ip rule add from 192.168.32.210  lookup rt_eth0
up ip rule add from 192.168.32.210  lookup rt_eth0
```

配置ipv6地址

要使用IPv6，您可以使用IPv6双栈的方式。

IPv6双栈

IPv6双栈技术就是让主机同时支持IPv6和IPv4协议，使得主机可以和其它仅支持IPv4或IPv6协议的主机进行通信。目前IPv6双栈向所有地域（Region）的BCC/DCC实例开放使用。

第一步：先配置IPv6VPC

第二步：根据操作系统判断是否需要进行进一步的操作。具体规则如下：

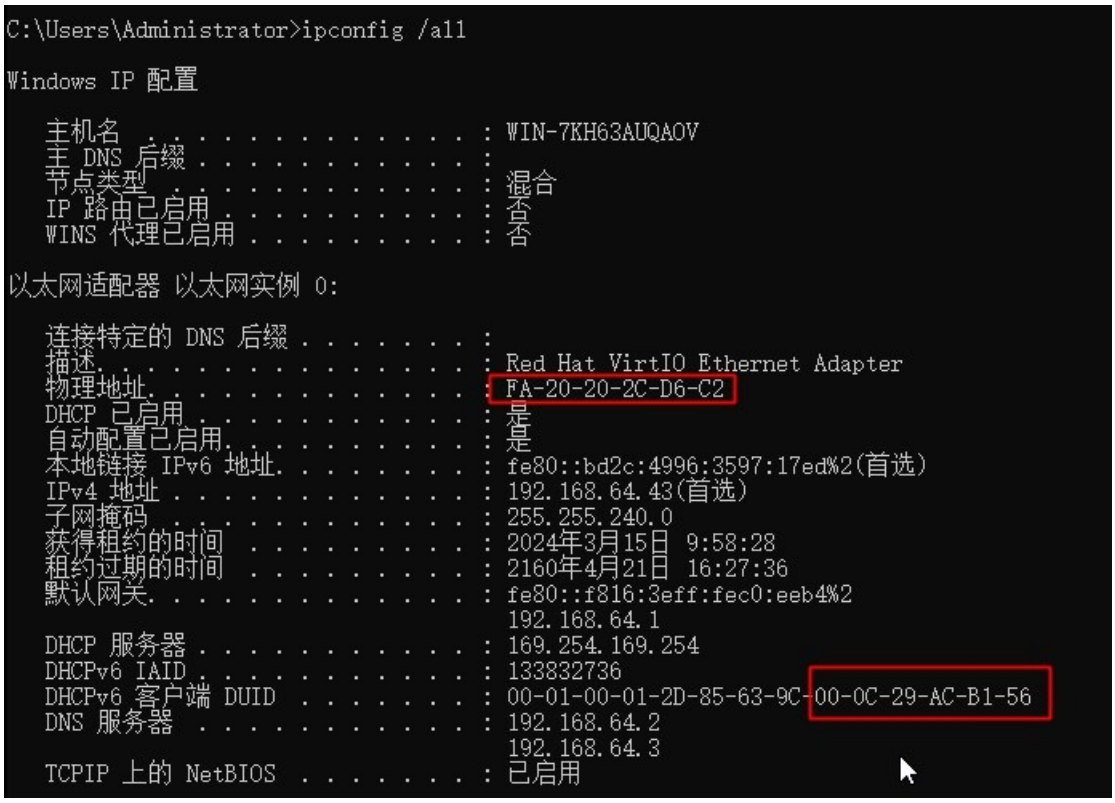
1. centos6 、centos7、centos8 系列镜像自动支持IPv6。
2. opensuse 镜像暂不支持IPv6。
3. 其他系类的公共镜像（windows/debian/ubuntu）需要执行脚本才能支持IPv6。具体脚本配置方法如下：

Windows镜像配置IPv6

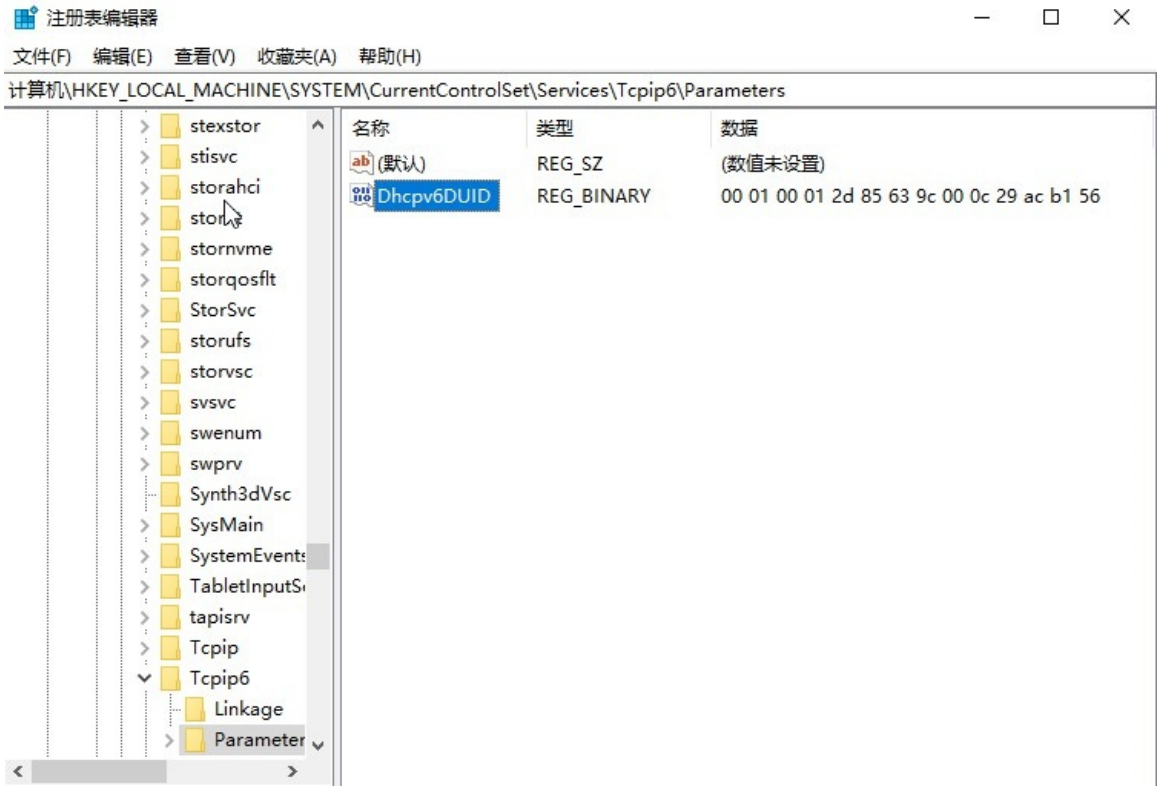
1. 在云服务器上以管理员身份启动 cmd 命令提示符窗口，执行以下命令，禁用 IPv6 随机标识。

   netsh interface ipv6 set global randomizeidentifiers=disabled

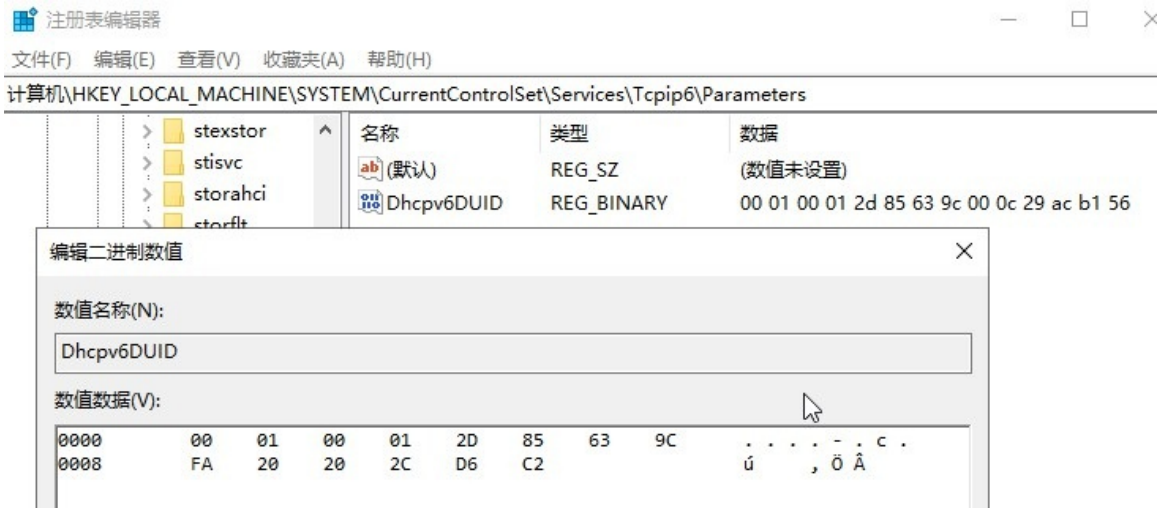
2. 检查Dhcpv6DUID的值的最后12个字符是否与MAC地址一致，如果一致，则调过此步骤。如果不一致，如下：



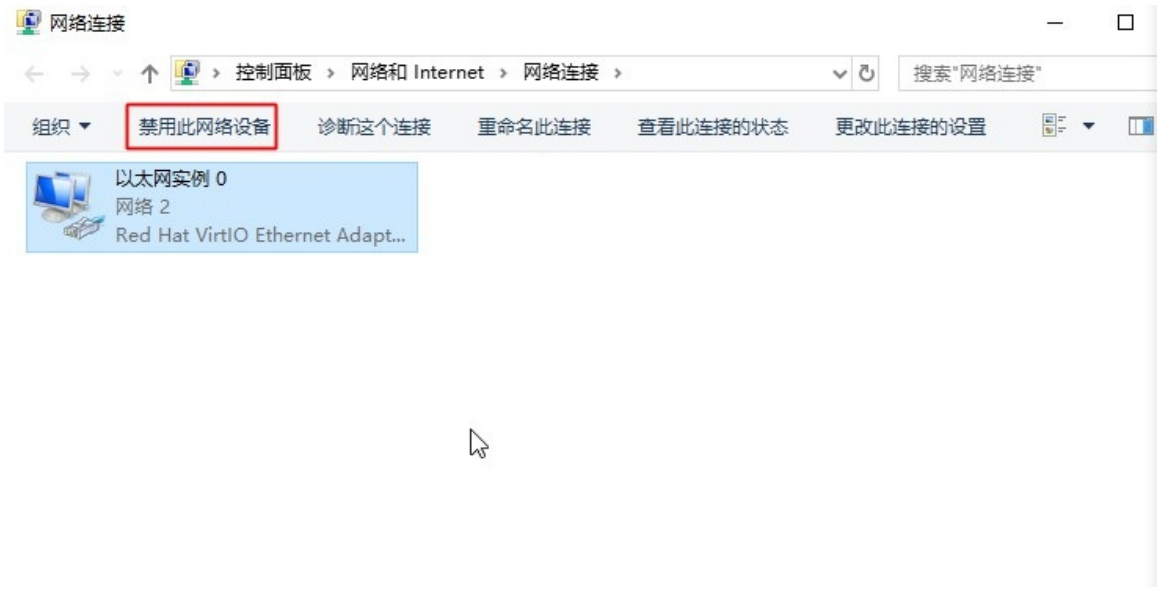
则需要修改注册表中的Dhcpv6DUID值。在 cmd 命令行窗口中，执行 regedit 命令打开注册表编辑器，并打开注册表目录 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\TCPIP6\Parameters。



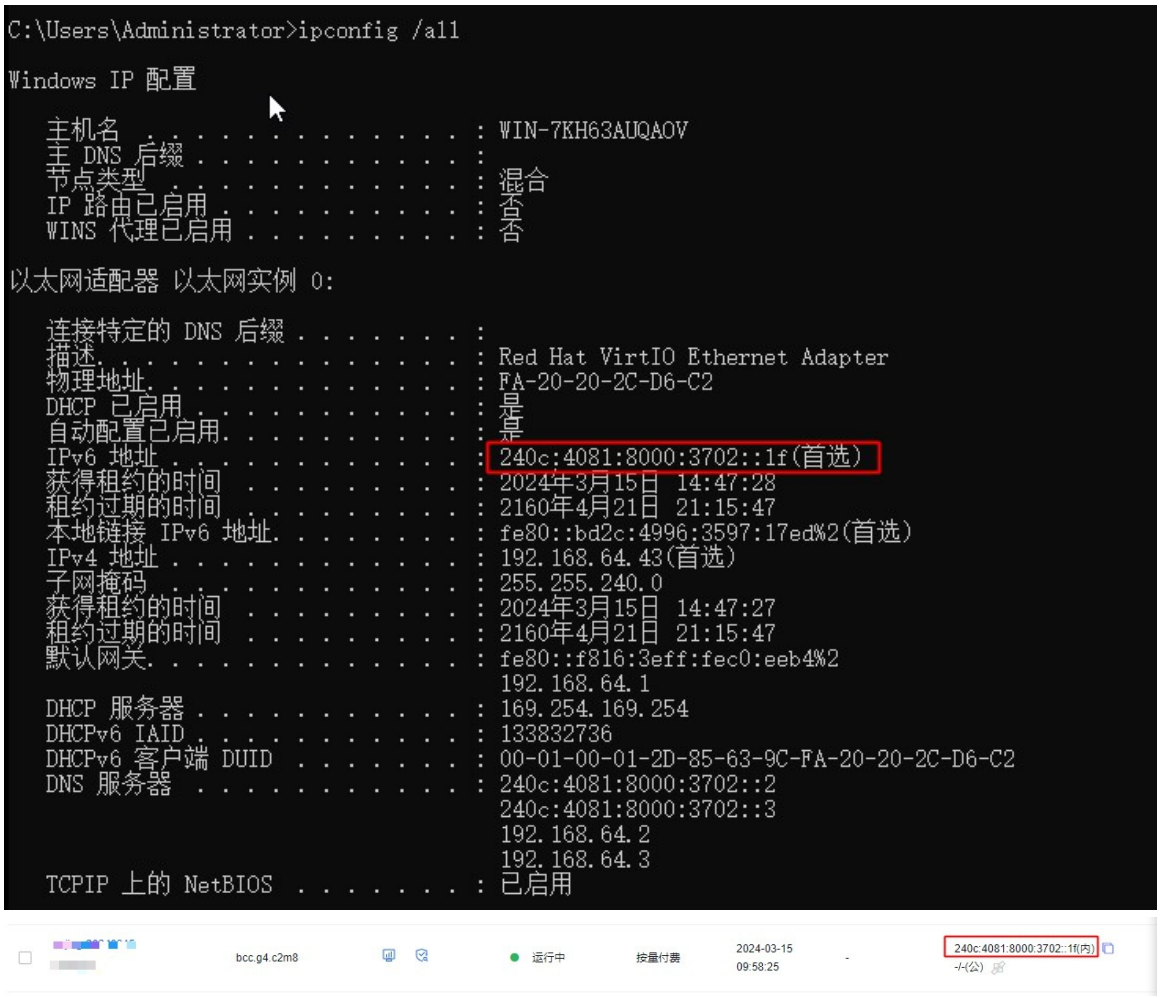
右键点击 Dhcpv6DUID，修改 Dhcpv6DUID 最后的 12 个字符修改为接口的 MAC 地址。



3. 进入网络连接配置页面，点击网络接口，然后点击禁用此网络设备，禁用成功后再点击启用此网络设备。启用成功后，网络接口便可成功获取到 IPv6 地址。



在命令行下输入 ipconfig /all，可以看到IPv6地址，这个IPv6地址和控制台上显示的一致才能说明正确。



debian/ubuntu镜像配置IPv6

1. 下载附件脚本，并复制进您的云服务器中。
2. 为该bcc-ipv6脚本文件添加可执行权限。
   - chmod +x bcc-ipv6-debian
3. 在终端运行bcc-ipv6-debian
   - ./bcc-ipv6-debian --add 为云服务器配置ipv6
   - ./bcc-ipv6-debain --del 删除已配置的ipv6

> 注：如果BCC创建后才将对应的VPC的IPv6开启，那该BCC仍然获取不到IPv6地址，需要手动执行dhclient -6 eth0来获取IPv6地址。

Windows云服务器配置双网卡外网访问

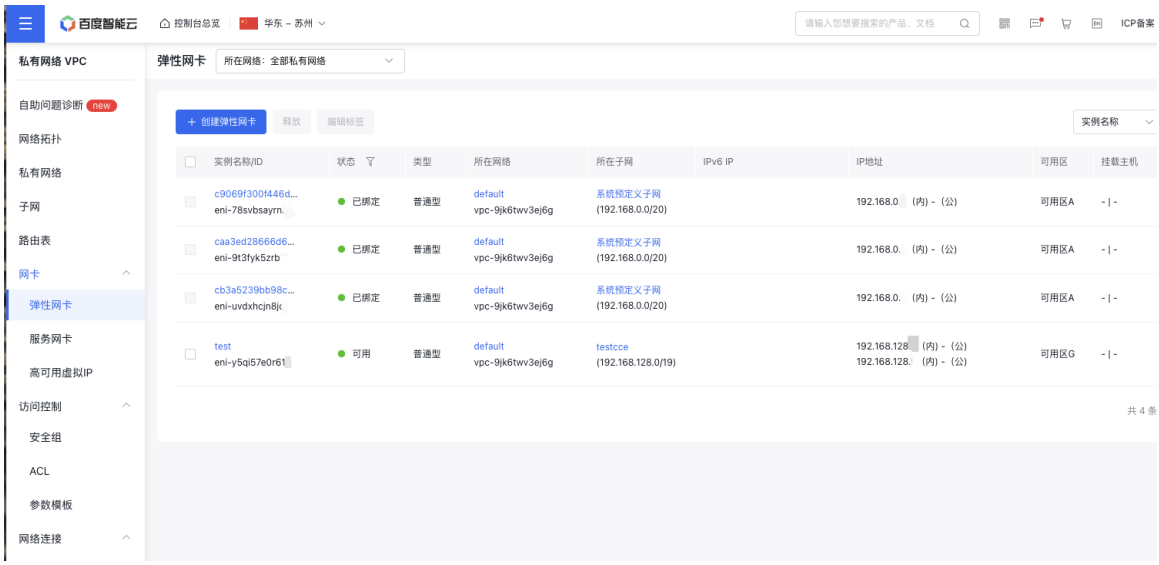
前提条件

主网卡在购买实例过程已自动分配弹性公网IP，且已正常访问外网。

操作步骤

购买弹性网卡

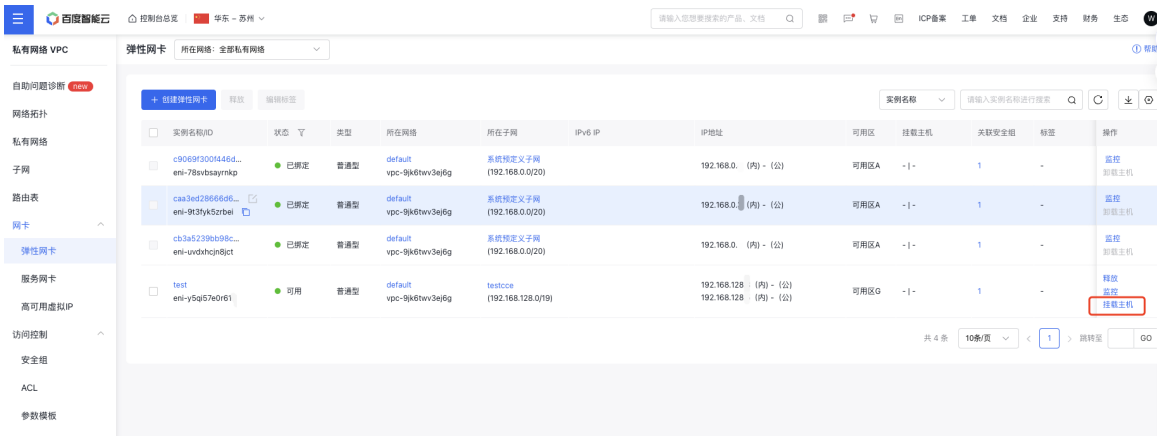
1. 在“私有网络VPC”界面，选择“弹性网卡”



2. 点击“创建弹性网卡”，填写相关信息

> 所在网络、所在子网、安全组 必须与主网卡相同

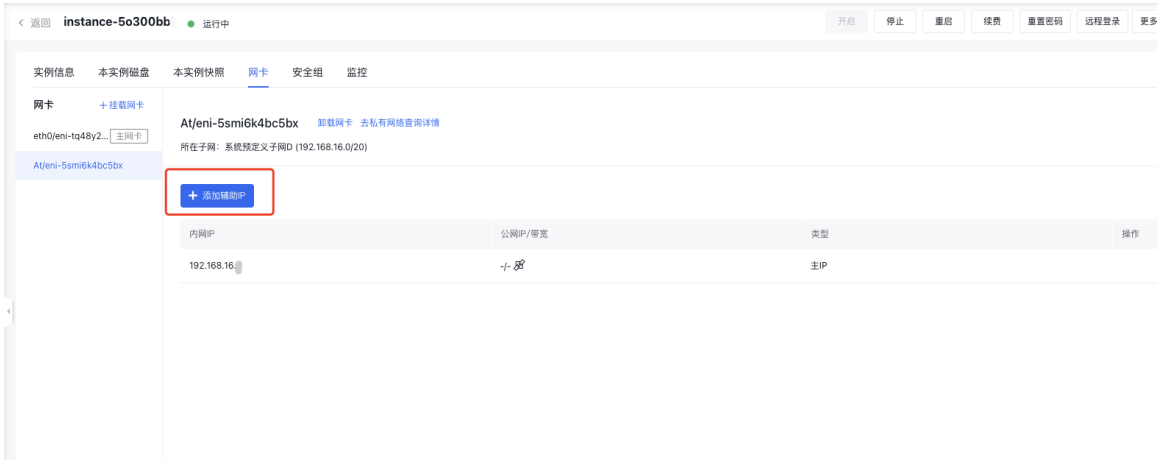
3. 弹性网卡创建完成后，可以在弹性网卡列表界面挂载到对应虚机，或在虚机信息界面挂载弹性网卡



购买弹性公网IP

1. 在“弹性公网IP”界面，可以创建EIP实例

2. 点击“创建实例”，在创建实例界面，选择“付费方式”、“公网带宽”、“购买时长”、“购买个数”等信息，即可完成购买
3. 在云服务器信息界面，给弹性网卡绑定弹性公网IP



4. 至此，扩展弹性网卡可以访问公网

弹性RDMA网卡

弹性RDMA网卡概述

弹性RDMA网卡（ERI）是一种能支持云服务器间在私有网络以RDMA通信的弹性网卡类型，具有低时延、高吞吐、低负载的特点，能显著提升云服务器之间的通信效率。同时，ERI支持弹性网卡的产品特性，可实现高可用网络方案。

使用场景 使用弹性RDMA网卡具有以下优势：

- 低时延访问 可实现5us级通信时延并极大降低网络传输在CPU的负载。
- 弹性高可用 支持在线挂载/解绑网卡，以及网卡在服务器之间的灵活迁移，实现高可用网络。
- 低成本 相比专用网络设备，ERI基于私有网络实现，降低使用成本。

适用场景：

- 高性能计算集群
- 人工智能分布式训练
- 数据库

使用说明

创建RDMA网络增强型实例 参考创建实例导航创建RDMA网络增强型实例，RDMA网络增强型实例可参考实例规格。

创建并挂载弹性RDMA网卡 参考创建弹性网卡，需要注意选择RDMA型的网卡类型以创建ERI，如下图。同时在安全组中不可限制UDP 4791端口。

参考挂载弹性网卡的使用说明来挂载ERI到实例，弹性RDMA网卡支持挂载RDMA网络增强型实例。

配置弹性RDMA网卡

参考配置弹性网卡来配置IP地址和路由。

测试弹性RDMA网卡

测试2个RDMA网络增强型实例之间的ERI性能

1. 安装OFED驱动

访问OFED下载链接下载系统环境对应的OFED版本，建议您下载LTS 5.4-3.1.0版本。 如您使用CentOS7u8或者CentOS 8u4，则需要额外下载依赖包

CentOS 7u8：

```
 yum install createrepo
yum install elfutils-libelf-devel python-devel redhat-rpm-config rpm-build libtool
```

CentOS 8u4：

```
 yum install createrepo
yum install kernel-rpm-macros python36 elfutils-libelf-devel python36-devel gdb-headless rpm-build libtool gcc
```

解压驱动安装包后，在安装目录执行以下命令

```
./mlnxofedinstall --without-fw-update  --add-kernel-support --skip-distro-check
```

2. 查看网卡设备信息

通过show_gids查看网卡的GID表，在VER为v2的选项中，查找目标ERI设备及IP地址对应的INDEX。

3. perftest测试ERI的RDMA带宽

在第一个实例中通过ib_write_bw -d mlx5_1 -x 3 -q 16 --report_gbits 作为server端

在第二个实例中通过ib_write_bw -d mlx5_1 -x 3 -q 16 --report_gbits 192.168.0.3 作为client端，然后可观察到带宽测试结果。

参数解释：

-d: 填写步骤2中DEV对应字段

-x：填写步骤2中INDEX字段

-Client端需要填写server端ERI的IPV4地址

监控弹性RDMA网卡 支持针对已绑定RDMA的ERI粒度的监控，支持以下指标：

指标名	MetricName
端口接收数据总数	RdmaPortRcvData
端口传输数据总数	RdmaPortXmitData
端口接收包总和	RdmaPortRcvPackets
端口传输包总和	RdmaPortXmitPackets
端口链接关闭次数	RdmaPortLinkDowned
端口被丢弃出站数据包总数(由于端口关闭或拥塞)	RdmaPortXmitDiscards

变更VPC

本文介绍BCC实例如何更换到其他VPC。如果您在创建BCC实例时选错VPC或者需要重新规划网络，可以通过此功能更换BCC实例的VPC。

注意事项

- 若实例有弹性网卡，请先将弹性网卡释放。
- 若实例主网卡有辅助IP，请先将辅助IP释放。
- 若实例有IPv6 IP，请先将IPv6 IP释放。
- 目前仅支持迁移到同AZ下不同VPC的不同子网，不支持跨AZ变更VPC。
- 仅支持对单个实例变更VPC，不支持批量操作。

操作步骤

1.登录云服务器BCC控制台进入实例列表页面。

2.选择您要变更VPC的实例，在右侧的操作栏，单击“更多>网络和安全>变更VPC”。

3.在操作页面选择相应的配置，点击**确认**，实例状态变为**变更VPC中**。

通过API变更VPC

您可以使用变更VPC接口为指定实例变更VPC。

变更内网IP

您可以直接修改私有网络中BCC实例的内网IP地址，具体操作请参见本节内容。

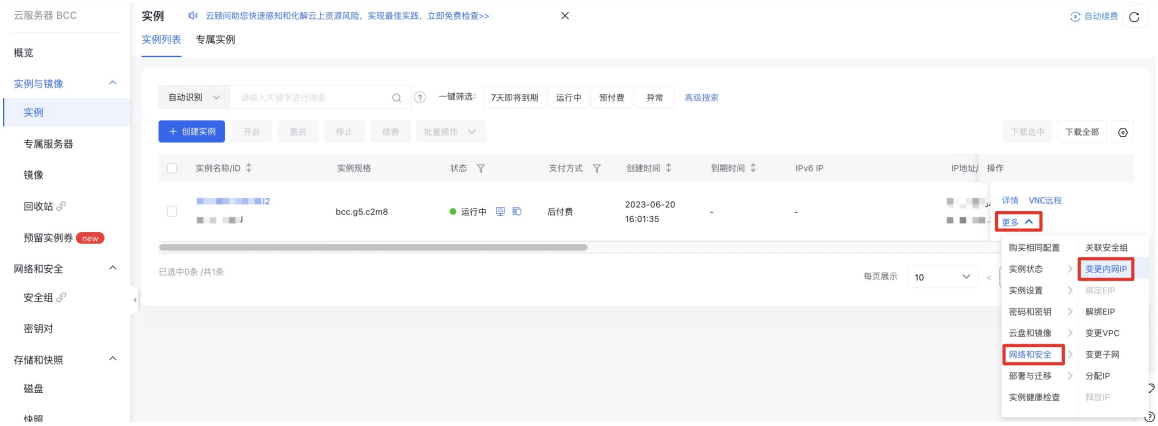
注意事项

- 变更内网IP会重启云服务器。
- 实例状态需为**运行中**或**已关机**状态。

操作步骤

1.登录云服务器BCC控制台进入实例列表页面。

2.选择您要变更内网IP的实例，在右侧的操作栏，单击“更多>网络和安全>变更内网IP”。



3.在操作页面选择相应的配置，点击确认，实例状态变为**变更内网IP中**。

变更子网

本文介绍BCC实例如何更换到其他子网。如果您在创建BCC实例时选错子网或者需要重新规划网络，可以通过此功能更换BCC实例的子网。

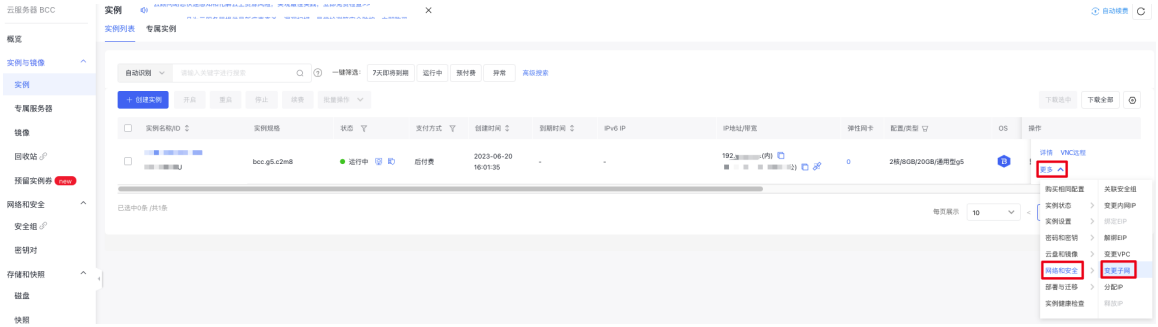
注意事项

- 若实例有弹性网卡，请先将弹性网卡释放。
- 若实例主网卡有辅助IP，请先将辅助IP释放。
- 若实例有IPv6 IP，请先将IPv6 IP释放。
- 目前仅支持迁移到同AZ下变更子网，不支持跨AZ或跨VPC变更子网，如果从普通子网变更至NAT专属子网，需要先手动解绑EIP。
- 子网变更过程中机器会重启。

操作步骤

1.登录云服务器BCC控制台进入实例列表页面。

2.选择您要变更子网的实例，在右侧的操作栏，单击“更多>网络和安全>变更子网”。



3.在操作页面选择相应的配置，点击确认，实例状态变为**变更子网中**。

绑定和解绑弹性公网IP

弹性公网IP（EIP）作为一个独立的产品为用户提供公网带宽服务，可以与任意BCC或BLB实例绑定或解绑，匹配用户的不同业务场景。

前提条件

如果您需要为BCC实例绑定EIP，请确保BCC实例满足以下条件：

- 地域必须和EIP的地域相同。
- 实例必须处于 **已停止**、 **运行中** 状态。
- 实例的主网卡未绑定弹性公网IP。

绑定EIP

1.登录云服务器BCC控制台进入实例列表页面。

2.选择待绑定EIP的实例，在右侧的操作栏，单击“更多>网络和安全>绑定EIP”。

3.在弹性公网IP绑定对话框中，选择待绑定的EIP。

4.单击**确定**。绑定成功后，在BCC实例页面的**IP地址/带宽**列会显示已绑定的EIP及带宽值，例如：10.1.XX.XX/5Mbps（公）。

解绑EIP

1.登录云服务器BCC控制台进入实例列表页面。

2.选择待解绑EIP的实例，在右侧的操作栏，单击“更多>网络和安全>解绑EIP”。

3.在**解绑弹性公网IP**对话框中，确认提示信息。

4.单击**确定**。解绑成功后，在BCC实例页面的**IP地址/带宽**列不再显示EIP。

> 说明：解绑后的EIP仍会继续收费，如果不再使用请自行释放EIP。更多信息，请参见EIP操作。
>
> 解绑成功后，BCC实例无法再通过EIP访问公网，若需访问公网，可再重新绑定EIP。

安全

安全组

安全组概述

安全组是在VPC网络内为BCC实例、DCC实例、负载均衡实例和云数据库实例创建的安全防火墙，定义IP＋端口的入站和出站访问策略，从而提高云服务器、负载均衡、云数据库等实例的安全性。

什么是安全组

安全组是一种虚拟防火墙，能够控制BCC实例的出入站流量。您可以通过配置安全组规则，允许或禁止安全组内实例的出流量和入流量。

安全组分为普通安全组和企业安全组，功能概述如下：

功能	普通安全组	企业安全组
安全组无任何规则	入方向：拒绝所有访问请求 出方向：拒绝所有访问请求	入方向：拒绝所有访问请求 出方向：拒绝所有访问请求
新建安全组默认规则	入方向：拒绝所有访问请求 出方向：允许所有访问请求	入方向：拒绝所有访问请求 出方向：允许所有访问请求
安全组规则策略	仅支持允许策略	支持允许、拒绝策略
设置规则优先级	纯白名单机制，无优先级	取值范围1-1000，可以与已有条目重复。数值越小，优先级越高，规则匹配顺序为按优先级由高到低匹配。若拒绝与允许的优先级相同，则拒绝优先
Source（源）或Target（目的）选择安全组	支持	不支持
关联实例数量	无限制	无限制
实例关联多个安全组	所有规则逐一匹配，有一条规则允许则允许	将关联的多个企业安全组中的所有规则按优先级重新排序，按优先级大小匹配规则，若允许拒绝策略的优先级相同，则拒绝优先
使用场景	适用于运维成本更低的场景	适用于网络控制更精细化的场景

使用限制

- 每个VPC会自动创建一个默认普通安全组，默认普通安全组无法删除，可以增、删、改规则。仅默认安全组提供一键恢复初始设置按钮。
- 每个BCC实例最多只能与10个安全组关联，如果某个BCC实例关联了多个安全组，则此BCC实例生效的规则是已关联安全组所有规则的合集。
- 每个安全组最多允许添加50个入站规则和50个出站规则。
- 当一个BCC实例仅关联有一个安全组时，该安全组不能取消关联。
- 同一个实例不支持同时关联普通安全组和企业安全组两种类型。

安全组规则

组成部分

安全组规则包括以下组成部分：

- 规则方向：出方向或入方向。
- 类型：IPv4或IPv6。
- 协议类型和端口：TCP、UDP、ICMP等。
- 来源/目的：普通安全组支持选择IP或安全组，企业安全组仅支持选择IP。
- 策略：企业安全组支持选择策略：允许或拒绝。普通安全组仅支持允许策略。

- 优先级：企业安全组支持设置规则的优先级1-1000，数值越小，优先级越高。

常用操作

若您想要新创建安全组，可参见创建安全组。

若您创建安全组后想关联至BCC实例上，可参见关联安全组。

若您想将BCC实例移出安全组，可参见取消关联安全组。

创建安全组

背景信息

安全组是BCC实例的虚拟防火墙，每台云服务器实例必须至少属于一个安全组。如果创建实例的时候您未创建安全组，可选择BCC的默认安全组。默认安全组允许所有入站和出站访问全部通过。详细介绍请见安全组概述。

如果您想建立一个BCC实例的入站与出站规则白名单，需要创建一个新的安全组，在新建安全组中定义自己想要的入站与出站规则，然后将BCC实例关联到此安全组，并取消云服务器与默认安全组的关联。本文将介绍如何使用控制台创建一个安全组。

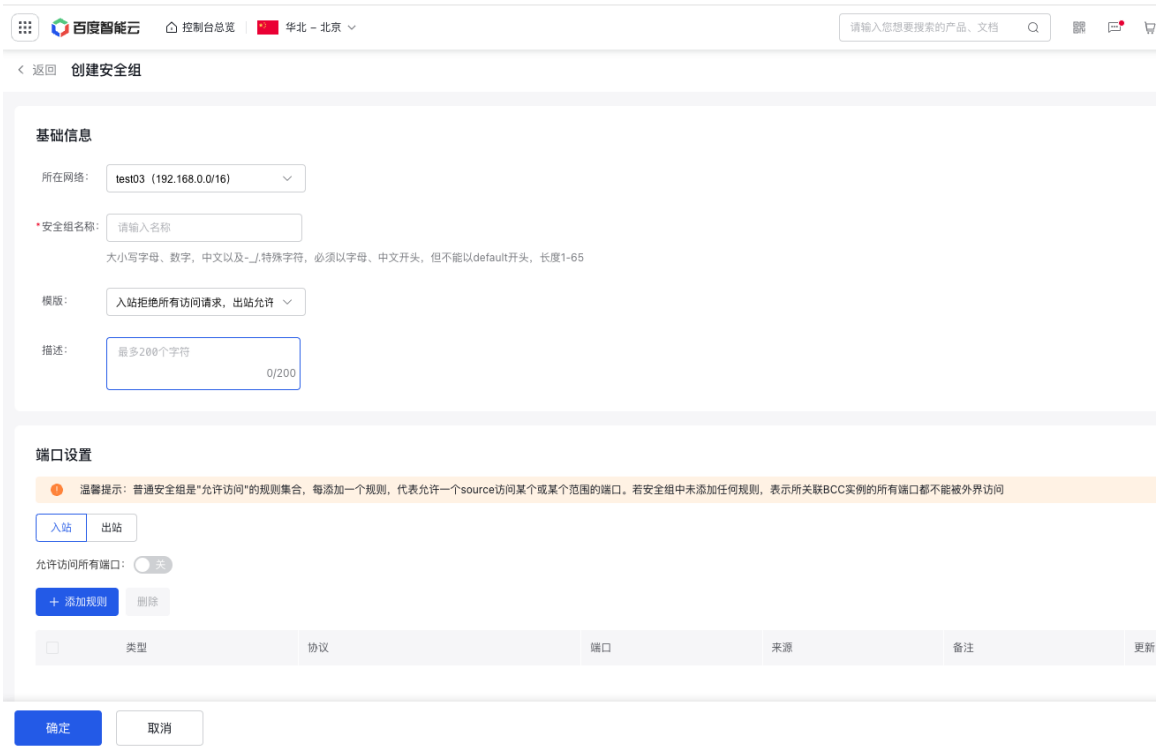
相关规则

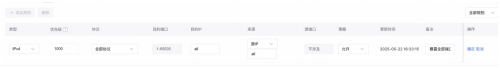
用户可通过控制台对安全组进行创建操作。创建安全组的规则如下：

- 每个账户最多能创建100个安全组；
- 当安全组配额已用完时，按钮置灰，不可创建；
- 创建安全组时，可通过“端口设置”的“入站”和“出站”页签设置详细规则；
- 每个安全组最多允许添加50个入站规则和50个出站规则。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **安全组**，进入安全组管理页面。
3. 单击 **创建安全组**，您也可以在顶部状态栏左上角处，选择地域。
4. 在 **创建安全组** 页面，配置安全组相关属性。



配置	描述
安全组名称	设置安全组的名称。
描述	自定义简短描述安全组，便于后期管理。
端口设置	选择端口设置方式，入站和出站的规则设置分为两个独立页签，规则设置有如下方式。 • **允许访问所有端口** 为 OFF 状态，可选择 **添加规则** 进行入站和出站的规则设置，设置完成后点击**确认**，可以进行下一条规则的添加。  • **允许访问所有端口** 为 ON 状态，将使云服务器的所有服务端口完全暴露在网络环境下，此时可能会面临一定的安全风险，因此 **请慎重选择该选项**。 • **允许访问所有端口** 为 OFF 状态且没有添加规则，将导致云服务器无法与外界通信。此时只能通过远程VNC登录访问云服务器，因此 **请慎重选择该选项**。
标签	您可对创建的安全组添加标签，便于资源分类及管理。

5. 单击 **确定**，完成安全组的创建。

相关操作

- 如果您需要修改安全组规则，可以通过 编辑安全组 修改安全组信息。
- 如果您想将实例加入一个或多个安全组，可以通过 关联安全组 操作。

通过API创建安全组

您可以通过 创建安全组接口 创建安全组。

编辑安全组

背景信息

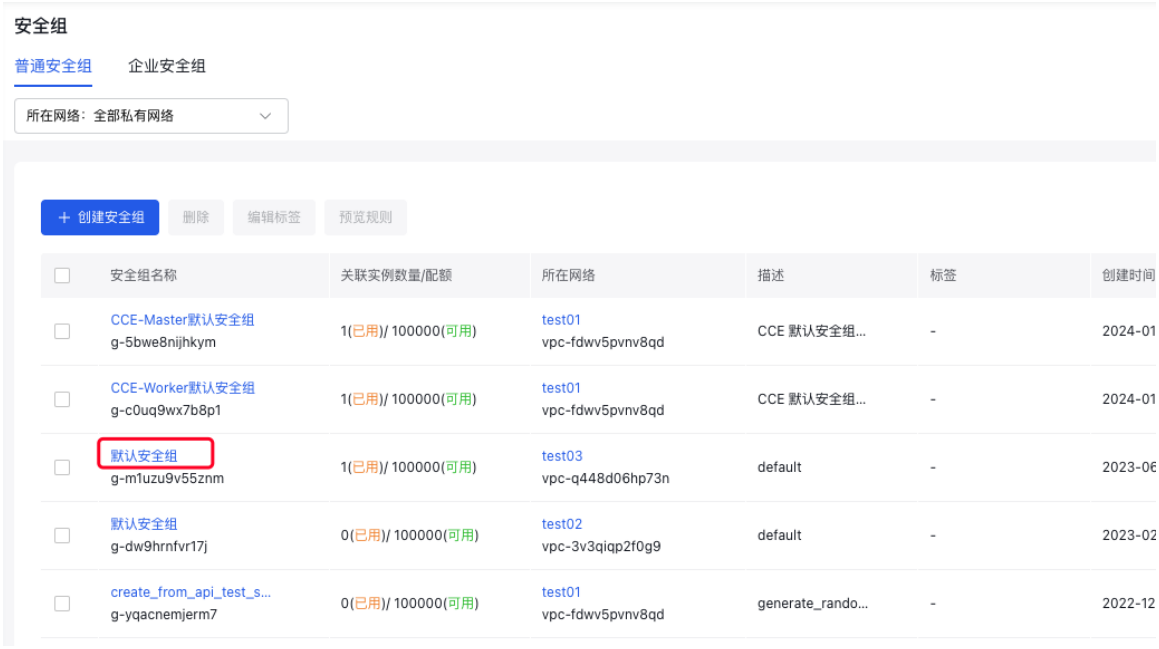
当需要修改安全组规则，或变更安全组名称、描述信息时，可使用此功能。

注意事项

- 为了保障系统安全，建议用户不要更改 **默认安全组** 的相关配置，如果需要其他权限的安全组机制，可以 创建安全组 并与BCC实例进行绑定。
- 用户可对除安全组ID外的所有信息进行编辑。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **安全组**，进入安全组管理页面。
3. 选择需要编辑的安全组，单击安全组名称链接，如下图所示。



4. 针对需要修改规则的 **协议**，单击对应操作列的 **编辑** 进入编辑操作。

5. 修改完成后，单击 **确定**，完成安全组相应信息的修改。

通过API编辑安全组规则

- 您可通过 授权安全组规则接口 在安全组中授权新的安全组规则。
- 您可通过 撤销安全组规则接口 撤销安全组中的规则。

复制安全组

背景信息

您可通过复制安全组功能，快速创建一组同样规则的安全组。

注意事项

- 当达到20个安全组配额时，不能复制安全组。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **安全组**，进入安全组管理页面。
3. 选择需要复制的安全组，单击右侧操作列 **复制**，进入复制安全组页面。
4. 您需要填写新安全组的名称及描述信息，确认信息后单击 **确定**，完成安全组信息的复制。

关联安全组

背景信息

创建安全组后，您需要将该安全组关联绑定到对应的云服务器实例上，这样该云服务器实例才能按安全组规则的设定实现网络访问控制功能。

您可以根据业务需求，将云服务器实例关联一个或多个安全组。

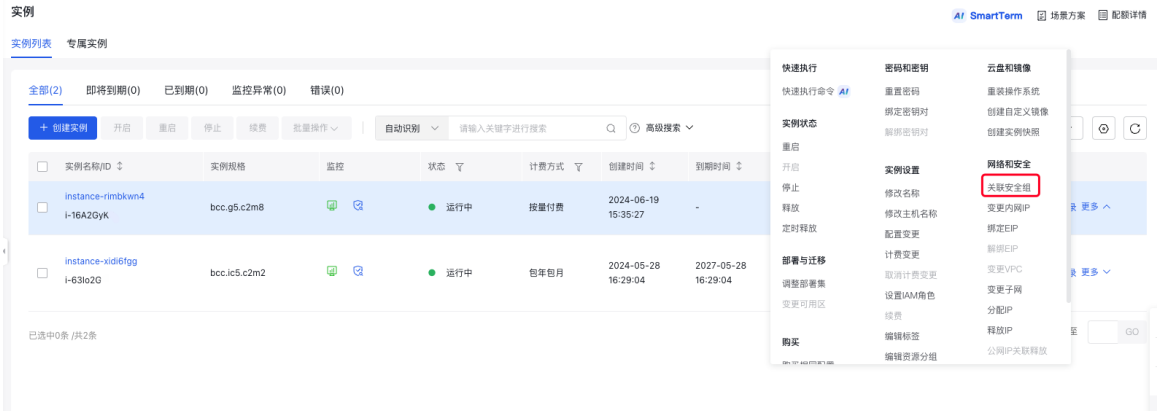
前提条件

将云服务器实例关联到安全组之前，请确认以下信息：

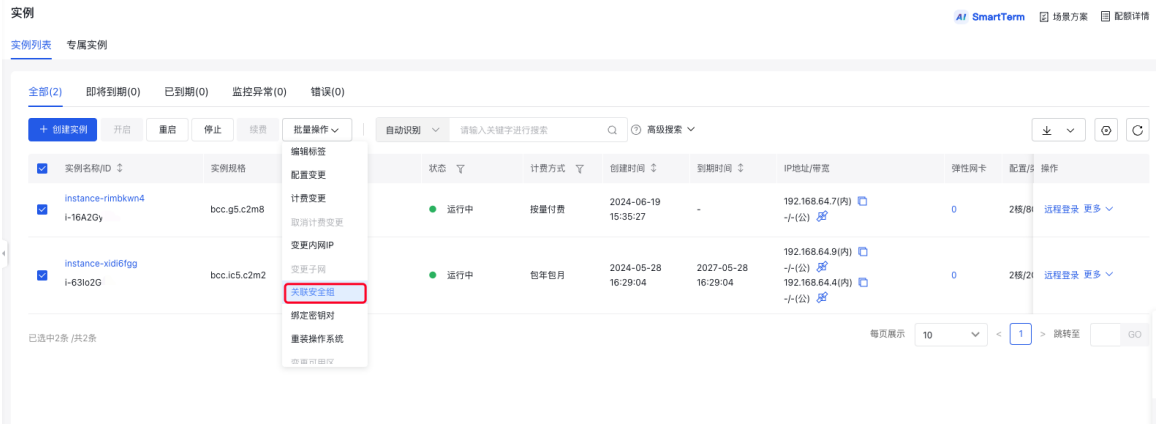
- 您必须已经成功创建云服务器实例。具体操作，请参见 通过购买页创建实例。
- 云服务器实例加入安全组时，目标安全组的网络类型与云服务器实例的网络类型必须一致。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 根据您的需求，选择不同的关联安全组方式。
   - 单个实例关联安全组：选择您要关联安全组的实例，在右侧操作列单击 “更多 - 实例设置 - 关联安全组”，如下图所示。



- 多个实例关联安全组：选中您要关联安全组的实例，在顶部单击 “批量操作 - 关联安全组”，如下图所示。



3. 在弹出的 **关联安全组** 对话框，勾选需要关联的安全组。
4. 单击 **确定**，完成云服务器实例和安全组的关联。

> **说明：**
>
> - 普通IV及以上机型，后端服务器绑定的安全组不会阻断BLB的流量，BLB和BCC之间默认放通。
> - 普通III及以下机型，来自BLB的流量将受到BCC安全组的管控。

后续步骤

- 如果您不希望您的云服务器实例属于某个或某几个安全组，您可以将云服务器实例移出安全组。具体操作，请参见 取消关联安全组。
- 如果您的业务已经不再需要一个或多个安全组，您可以删除安全组。安全组删除后，组内所有安全组规则同时被删除。具体操作，请参见 删除安全组。

取消关联安全组

背景信息

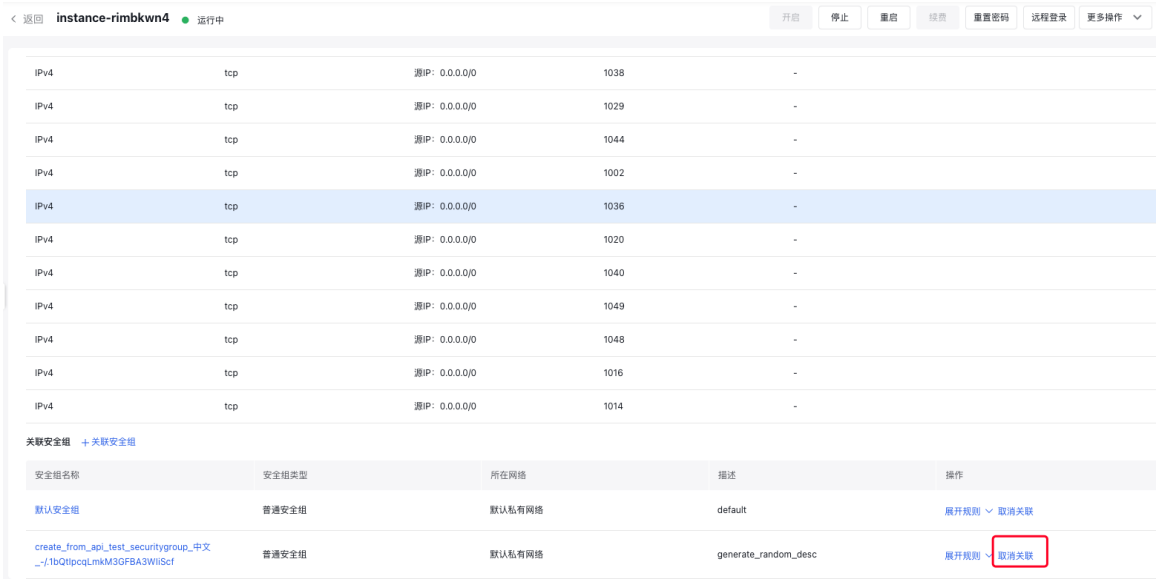
您可以根据业务需要，将云服务器实例移出安全组。

前提条件

云服务器实例已关联两个或两个以上安全组。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在 **实例名称** 列下，选择需要取消关联安全组的实例，单击 **实例名称链接**，进入 **实例详情** 页签。
3. 选择 **安全组** 页签，在 **关联的安全组列表** 选择要取消关联的安全组，单击 **取消关联**，如图所示。



4. 实例将取消关联相应安全组。

删除安全组

背景信息

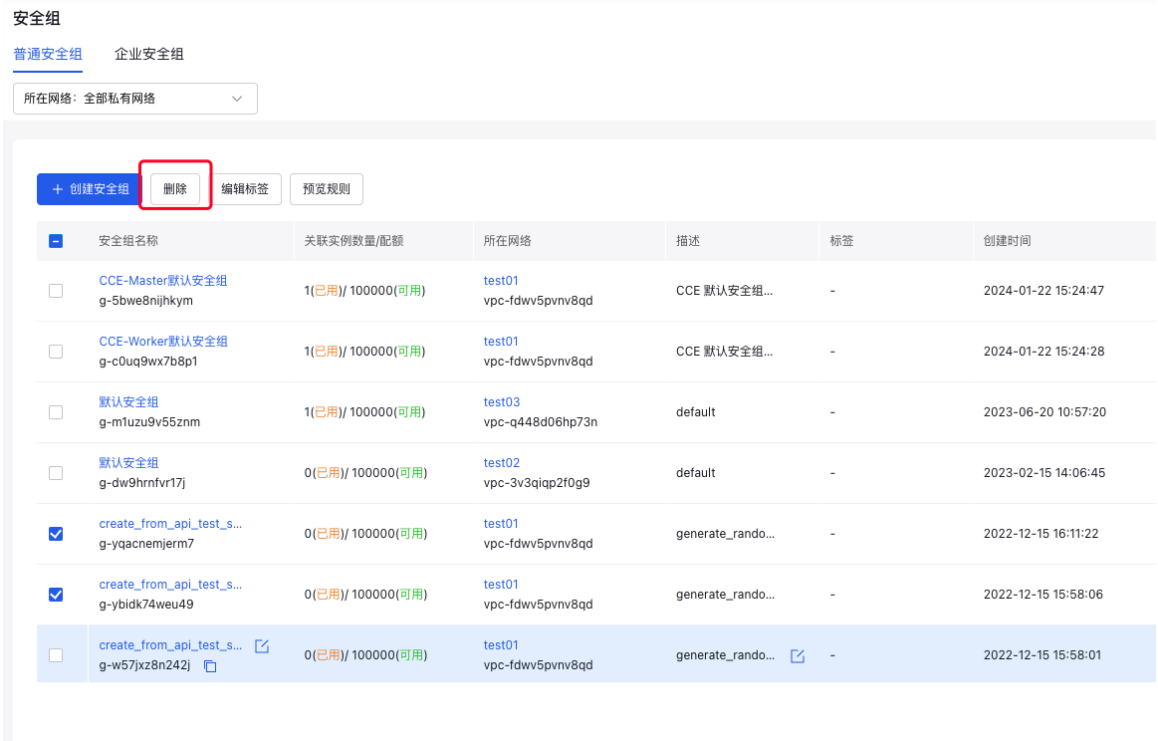
如果您的业务已经不再需要一个或多个安全组，您可以删除安全组。安全组删除后，该安全组内所有安全组规则同时被删除。

前提条件

- 已关联了实例的安全组，不可删除；如需删除，请先将实例移出安全组，具体操作请参见 取消关联安全组。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **安全组**，进入安全组管理页面。
3. 在安全组列表页面中，选中一个或多个安全组，单击顶部 **删除**按钮。



4. 在 **删除安全组** 对话框里，确认信息后，单击 **确定**。

通过API删除安全组

您可以通过 删除安全组接口 删除指定的安全组。

密钥对

SSH密钥对概述

SSH密钥对是一种安全便捷的登录方式，由公钥和私钥组成，仅支持Linux实例。

什么是密钥对

SSH密钥对是通过加密算法生成的一对密钥，默认采用RSA 2048位的加密方式。使用密钥对登录Linux实例，您需要先创建一个密钥对。密钥对创建成功后，会生成公有密钥（公钥）和私有密钥（私钥）：

公钥：百度智能云仅存储公钥。对于Linux实例，公钥内容存储在**~/.ssh/authorized_keys**文件中。

私钥：您需要下载并妥善保管私钥，私钥仅有一次下载机会，百度智能云不会保存您的私钥。拥有您私钥的任何人都可以解密您的登录信息，因此您需将私钥保存在一个安全的位置。

功能与优势

相较于传统的用户名和密码认证方式，使用SSH密钥对有以下优势：

- 安全性更高：通过公钥无法推导出私钥，SSH密钥对的安全强度远高于常规用户口令，可以杜绝暴力破解威胁。
- 操作更便捷：只需在控制台和本地客户端做简单配置即可远程登录实例，再次登录时无需再输入密码。同时也方便管理大量Linux实例。

使用限制

- 仅支持Linux实例。
- 一个云账号在一个地域最多可以拥有500个密钥对。
- 一个密钥对可以绑定多个实例，但是同一个实例只能绑定一个密钥对，如果您的实例已绑定密钥对，绑定新的密钥对会替换原来的密钥对。
- 可以在开机、关机状态下批量绑定实例。**开机时绑定**，虚机会自动重启生效；**关机时绑定**，您需要再开机才会生效。
- 密钥对绑定成功后，虚机的原密码仍可用。

常用操作

如果您还未创建SSH密钥对，可参见创建SSH密钥对。

如果您创建BCC实例后需要绑定SSH密钥对，可参见绑定SSH密钥对。

如果您不想使用SSH密钥对登录BCC实例，可参见解绑SSH密钥对。

如果您想删除SSH密钥对，可参见删除SSH密钥对。

创建SSH密钥对

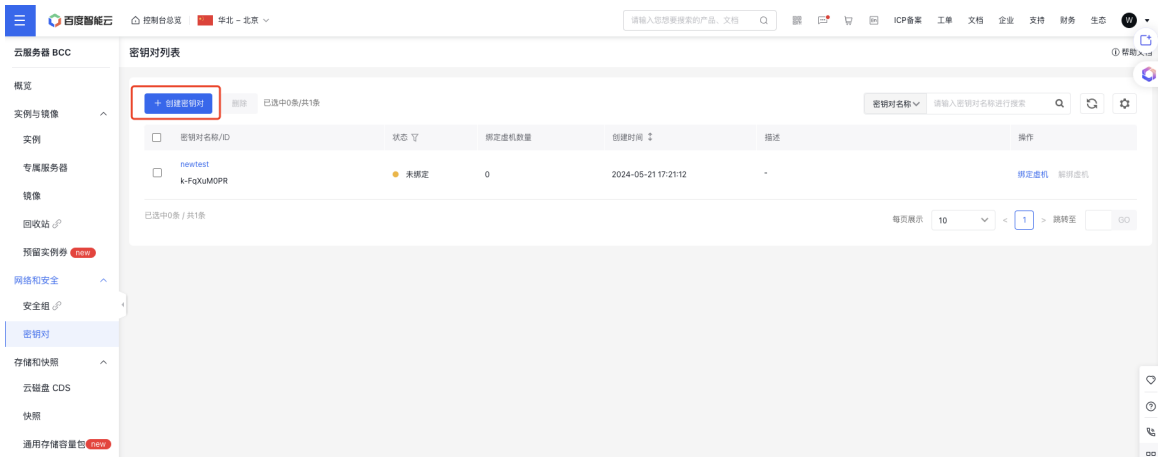
百度智能云平台不会保留您的私钥信息，创建后系统自动下载私钥到本地，且只有一次下载私钥的机会，请妥善保存。本文介绍如何在BCC控制台上创建SSH密钥对。

注意事项

- 使用密钥对绑定BCC实例后，如果没有私钥，您将无法登录该BCC实例。
- 您在一个地域最多可以拥有500个密钥对。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **密钥对**，进入密钥对列表页面。
3. 在密钥对列表页面上，选择所需地域后，单击 **创建密钥对**。



4. 在 **创建密钥对** 页面，选择创建密钥对方式、设置密钥对名称并填写密钥对描述信息。
   - 若创建方式选择 **创建新的密钥对**，输入密钥对名称以及密钥对描述后，单击 **确认**；
   - 若创建方式选择 **使用已有公钥** ，输入密钥对名称、密钥对描述、原有的公钥信息后，单击 **确认**；

> **注意：**百度智能云平台不会保留您的私钥信息，点击确定后系统自动下载私钥到本地。**特别注意，如果取消下载，则无法重新获取私钥。**

5. 密钥对创建成功后，您可以在密钥对列表里看到新创建的密钥对信息，包括密钥对名称、状态、绑定虚机数量等。

查看密钥对详情

您可单击密钥对的名称链接查看密钥对的详情信息。

绑定SSH密钥对

您可以在创建实例时指定SSH密钥对，也可以在创建实例后绑定SSH密钥对。本文介绍如何在创建实例后绑定SSH密钥对。

注意事项

- 仅支持Linux实例。
- 一个密钥对可以绑定多个实例，但是同一个实例只能绑定一个密钥对，如果您的实例已绑定密钥对，绑定新的密钥对会替换原来的密钥对。
- 可以在开机、关机状态下批量绑定实例。**开机时绑定**，虚机会自动重启生效；**关机时绑定**，您需要再开机才会生效。
- 密钥对绑定成功后，虚机的原密码仍可用。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 **密钥对**，进入密钥对列表页面。
3. 在顶部状态栏左上角处，选择地域。
4. 选择需要操作的密钥对，单击最右侧操作列 **绑定虚机**。
5. 在弹出的 **绑定虚机** 对话框，勾选一个或多个实例，单击 **确定**。

6. 绑定成功后，在密钥对列表页面的 **绑定虚机数量** 下可查看当前已绑定虚机数量。

后续步骤

如何使用SSH密钥对登录Linux实例，请参见 使用SSH密钥登录Linux实例。

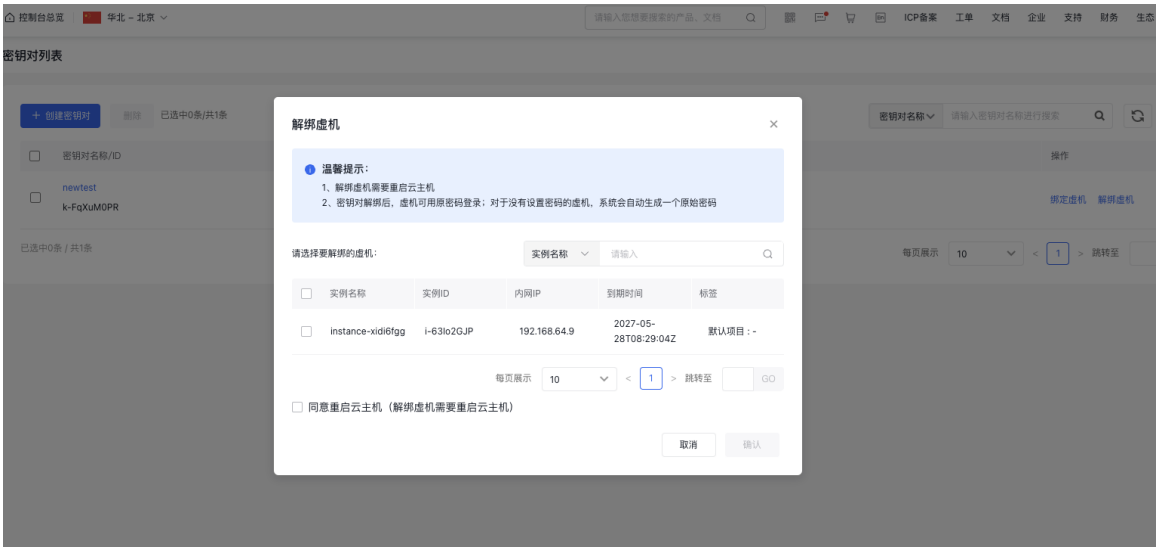
解绑SSH密钥对

前提条件

SSH密钥对已绑定了BCC实例。具体操作，请参见 绑定SSH密钥对。

操作步骤

1. 登录 云服务器BCC控制台 进入实例列表页面。
2. 在左侧导航栏单击 密钥对，进入密钥对列表页面。
3. 选择需要进行解绑操作的SSH密钥对所处的地域。
4. 在密钥对列表中找到需要操作的密钥对，在 **操作** 列中，单击 **解绑虚机**。
5. 在 **解绑虚机** 对话框里，选中需要解绑密钥对的 BCC 实例名称后，单击 **确认**。



> **注意：**密钥对解绑后，虚机可用原密码登录；对于没有设置密码的虚机，必须重置实例密码才能继续使用，具体操作请参见 重置实例密码。

删除SSH密钥对

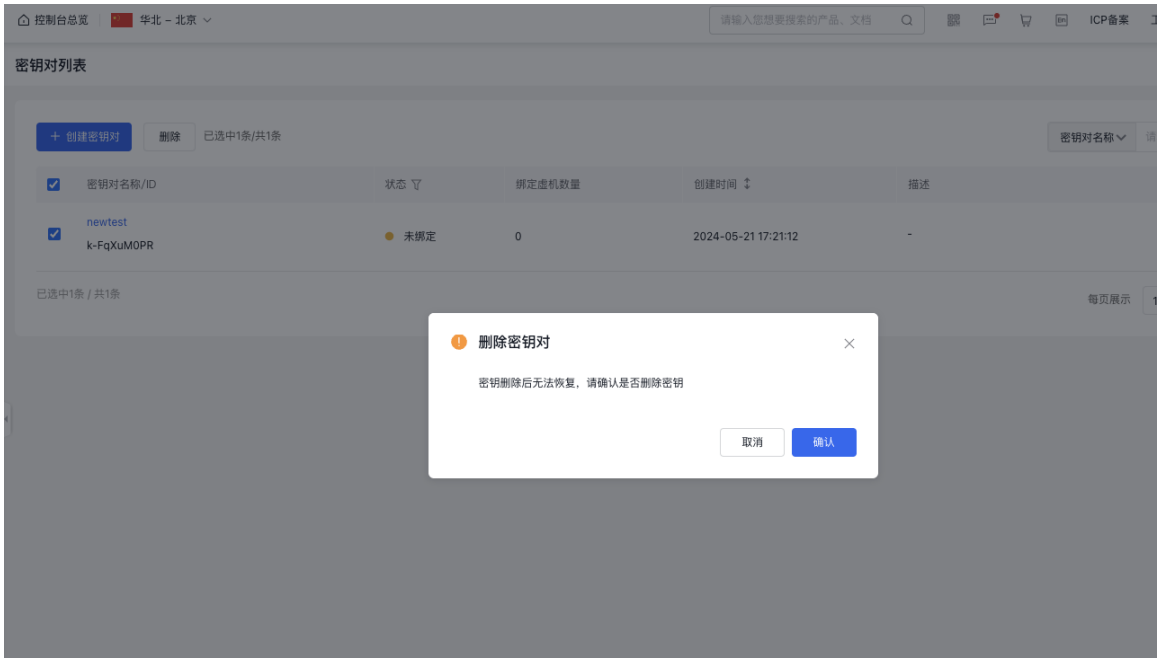
如果您不再需要某个SSH密钥对，可以使用删除密钥对操作。

前提条件

已绑定在实例上的SSH密钥对，必须在 **解绑后** 才能进行删除。

操作步骤

1. 登录 **云服务器BCC控制台** 进入实例列表页面。
2. 在左侧导航栏单击 **密钥对**，进入密钥对列表页面。
3. 选择需要进行删除操作的SSH密钥对所处的地域。
4. 选中需要删除的一个或多个密钥对后，单击 **删除**，完成SSH密钥对的删除操作。



多用户访问控制

多用户访问控制

介绍

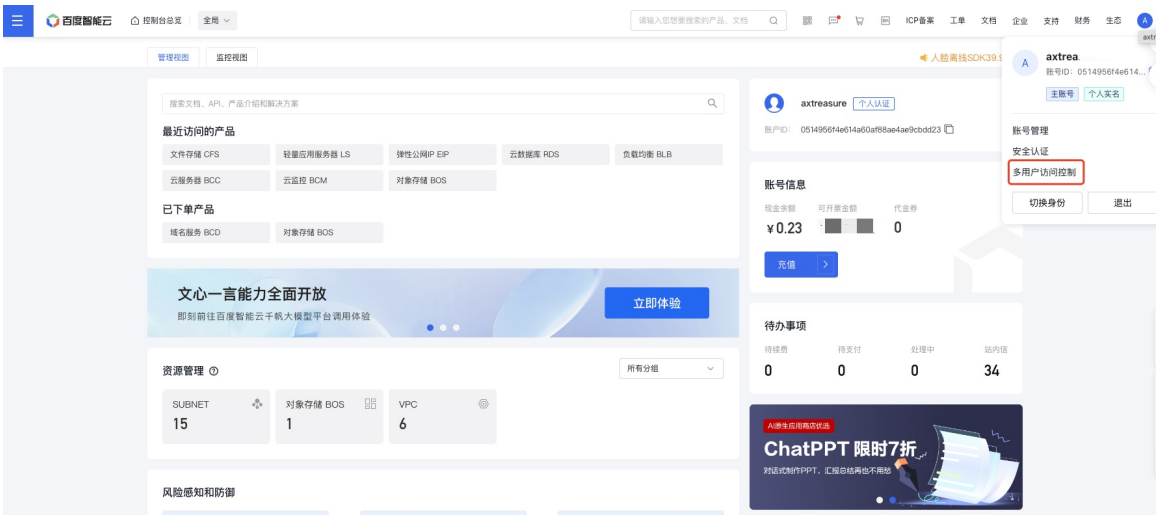
多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限，适用于企业内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。

适用于下列使用场景：

- 中大型企业客户：对公司内多个员工授权管理
- 偏技术型vendor或SAAS的平台商：对代理客户进行资源和权限管理
- 中小开发者或小企业：添加项目成员或协作者，进行资源管理

创建用户

1. 主账号用户登录后在控制台选择“多用户访问控制”进入用户管理页面。



2. 在左侧导航栏点击“用户管理”，在“子用户管理列表”页，点击“新建用户”。
3. 在弹出的“新建用户”对话框中，完成填写“用户名”和确认，返回“子用户管理列表”区可以查看到刚刚创建的子用户。

配置策略

BCC支持系统策略和用户自定义策略两种，分别实现BCC的产品级权限和实例级权限控制。

- 系统策略：百度智能云系统为管理资源而预定义的权限集，这类策略可直接为子用户授权，用户只能使用而不能修改。
- 自定义策略：由用户自己创建，更细化的管理资源的权限集，可以针对单个实例配置权限，更加灵活的满足账户对不同用户的差异化权限管理。

系统策略

系统策略包含完全控制管理权限、运维权限和只读权限2种策略，权限范围详细如下：

操作类型	权限范围
只读操作	仅可以查看BCC实例，及所挂载的CDS磁盘、快照、安全组列表

运维操作	• BCC实例：
	  • 开启、停止、重启
	  • 修改名称
	  • 关联安全组
	  • 创建快照
	  • 创建自定义镜像
	  • 重装操作系统
	  • VNC远程
	  • 重置密码
	  • 绑定/解绑密钥对等
	• 在有权限BCC实例上挂载的CDS磁盘：
	  • 系统盘：创建镜像、创建快照
	  • 数据盘：创建快照
	• 通过挂载的CDS磁盘创建的快照，可以创建自定义镜像。
	• 安全组、镜像列表仅支持只读。

完全控制管理	• BCC实例：
	  • 创建(若无财务权限，支持创建按量付费实例，创建包年包月实例会生成订单，需管理员确认支付后继续创建)
	  • 释放
	  • 计费变更、配置变更
	  • 开启、停止、重启
	  • 修改名称
	  • 关联安全组
	  • 创建快照
	  • 创建自定义镜像
	  • 重装操作系统
	  • VNC远程
	• 在有权限BCC实例上挂载的CDS磁盘：
	  • 系统盘：创建镜像，创建快照；
	  • 高性能云磁盘：创建快照；
	  • 普通云磁盘：创建快照。
	• 通过挂载的CDS磁盘创建的快照，可以创建自定义镜像。
	• 安全组、镜像列表仅支持只读。

自定义策略

自定义策略是从实例维度进行授权，与系统策略不同，只对选定的实例生效。

用户先通过左侧导航栏进入【策略管理】，然后点击“创建策略”，用户填写策略名称并选择服务类型为云服务器BCC，其中策略生成方式默认为策略生成器，不需要修改。

用户授权

在“用户管理->子用户管理列表页”的对应子用户的“操作”列选择“添加权限”，并为用户选择系统权限或自定义策略进行授权。

> **说明：**如果在不修改已有策略规则的情况下修改某子用户的权限，只能通过删除已有的策略并添加新的策略来实现，不能取消勾选已经添加过的策略权限。

子用户登录

主账号完成对子用户的授权后，可以将链接发送给子用户；子用户可以通过IAM用户登录链接登录主账号的管理控制台，根据被授权的策略对主账户资源进行操作和查看。

其他详细操作参考：多用户访问控制。

价格权限管控

概述

子用户被赋予财务查询、管理等权限时，子用户登录控制台购买BCC实例能看见账户的优惠信息。

若需要赋予子用户财务和购买BCC实例权限，但不想子用户登录控制台购买实例能查看到优惠信息时，多用户访问控制IAM支持设置子用户权限禁止查看优惠价，设置生效后，子用户登录控制台购买实例时只会看到目录价。

注意事项

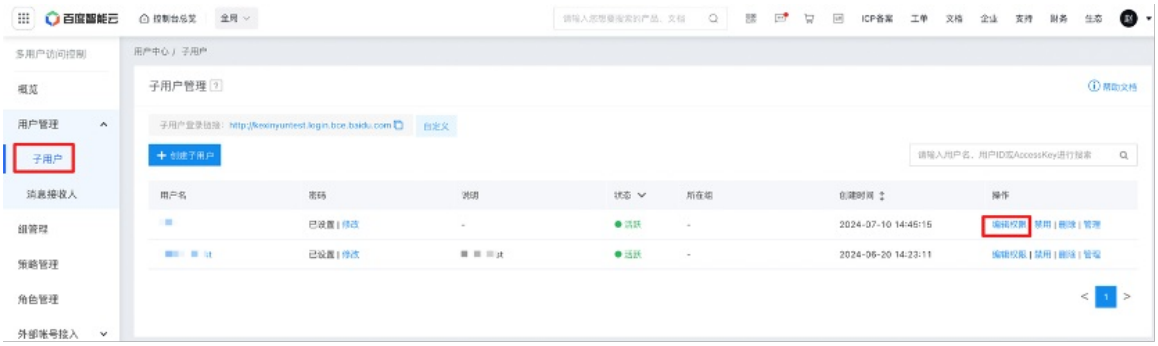
- 仅对子用户开放BCC产品运维/管理权限，未设置财务权限时，子用户看不到优惠信息。
- 支持对单个子用户或者组粒度设置子用户禁止查看优惠权限。
- 若需要操作的子用户数量较多，可以新建分组，将目标子用户加入并设置权限。

- 仅支持主账户或者具备系统管理员权限的子用户操作。

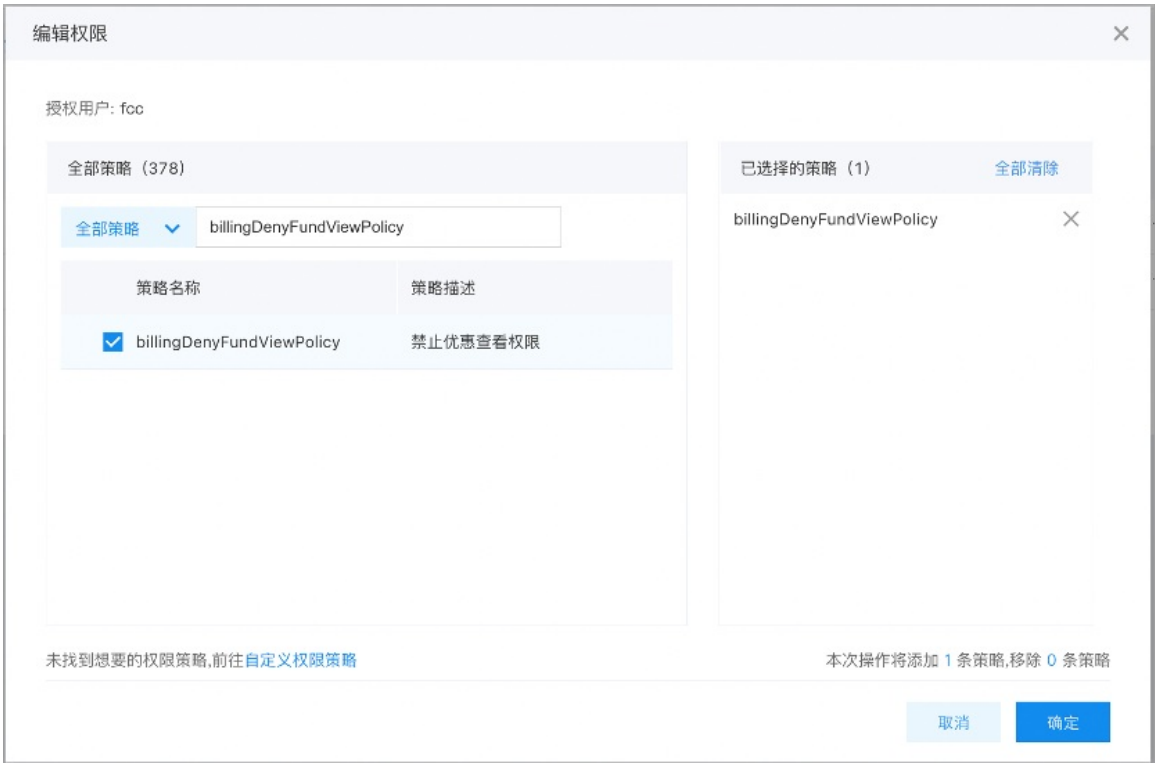
操作步骤

单子用户设置 1.登录控制台，鼠标移入右上角 **用户中心** ，选择 **多用户访问控制** 。

2.选择 **用户管理-子用户** ，在对应操作列点击 **编辑权限** 。



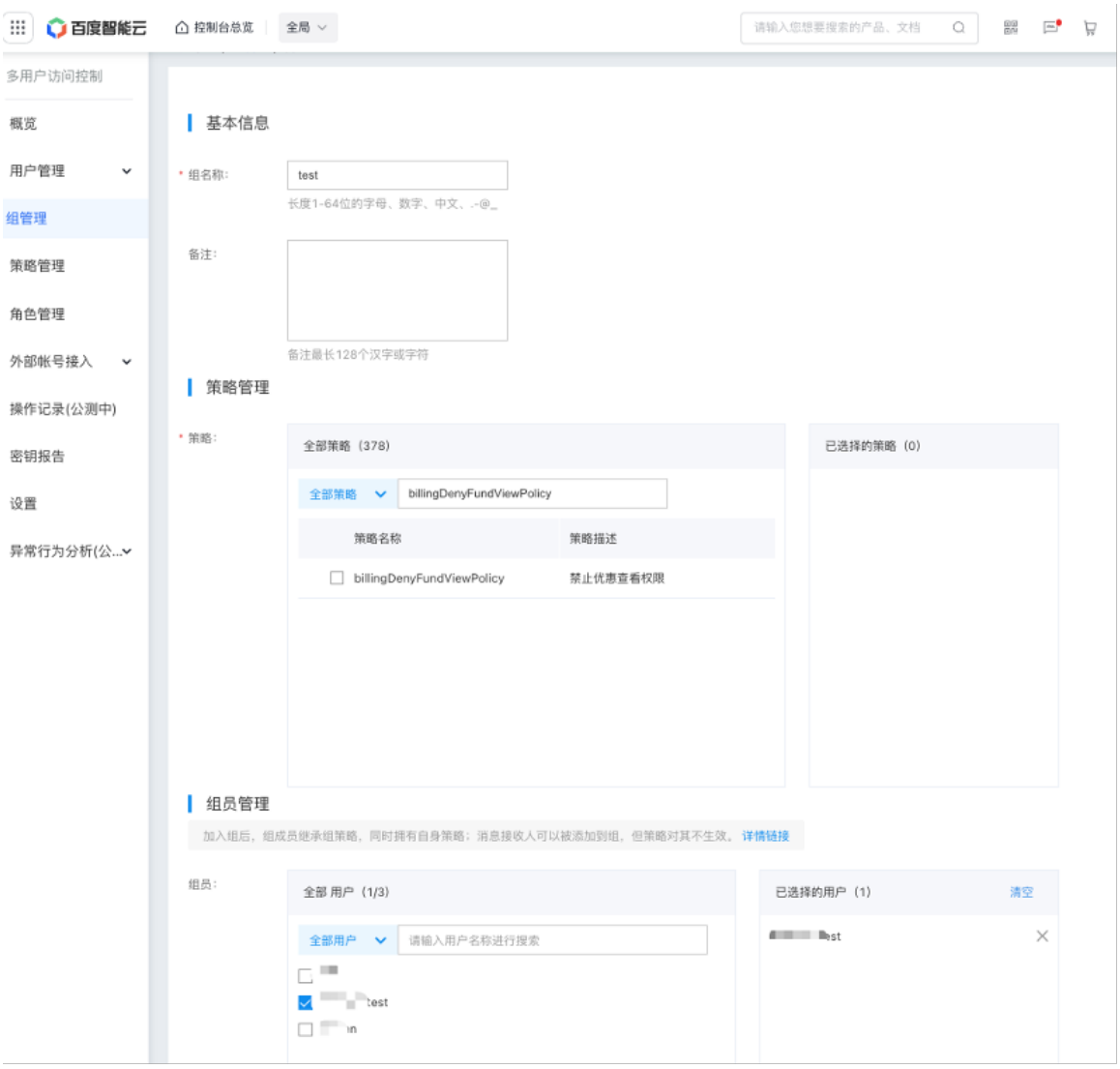
3.在策略搜索框里勾选billingDenyFundViewPolicy（禁止优惠查看权限），点击确定，即完成操作。



组设置 1.登录控制台，鼠标移入右上角 **用户中心** ，选择 **多用户访问控制** 。

2.选择 **组管理** ，在对应操作列点击 **编辑** 。

3.在策略搜索框里勾选billingDenyFundViewPolicy（禁止优惠查看权限），点击确定，即完成操作。



启动模板

概述

启动模板是一项持久化BCC实例配置的功能。您可以通过使用启动模板快速创建实例，提升使用效率。启动模板中包含了用于创建实例的配置信息，可以存储除密码以外的任意配置信息。

> **说明：** 已有实例启动模板不支持修改，但支持创建多个版本，每个版本支持设置不同的配置信息。

使用限制

项目	说明
模板	单个地域可创建30个模板
单个模板的版本	单个模板最多可支持20个版本

操作步骤

创建启动模板

- **直接创建模板**

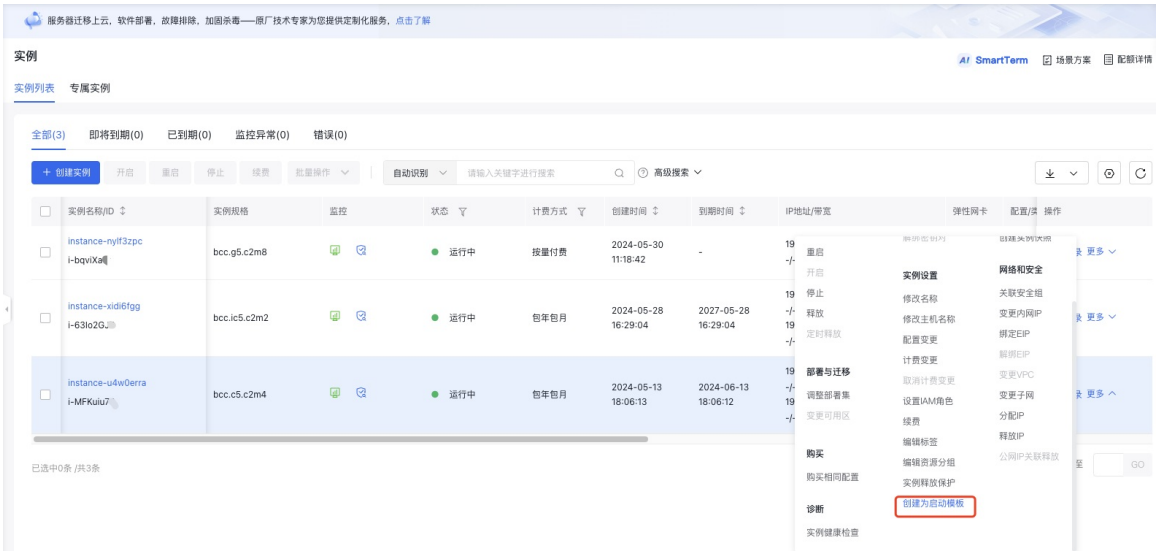
1. 登录BCC管理控制台。
2. 点击左侧导航栏**启动模板**，进入启动模板列表页。
3. 点击**创建模板**，填写基本配置信息，详情请参考通过购买页创建实例进行配置。
4. 在**确认配置**页面，输入以下配置信息，并单击**创建启动模板**。

配置项	必选/可选	配置说明
模板名称	必选	用户自定义启动模板名称。 支持大小写字母、数字、中文以及-_ /.特殊字符，必须以中文或字母开头，长度1-65个字符。
模板描述	可选	用户自定义启动模板描述。 支持任何字符，长度为2～256个字符。
模板标签	可选	每个启动模板支持绑定一个或多个标签，具体可参考标签管理进行配置。

5. 创建成功后，可在控制台查看该启动模板。

- **通过已有实例创建模板**

1. 登录BCC管理控制台，进入实例列表页面。
2. 选择对应的实例，在操作栏点击**更多**，点击**创建为启动模板**。



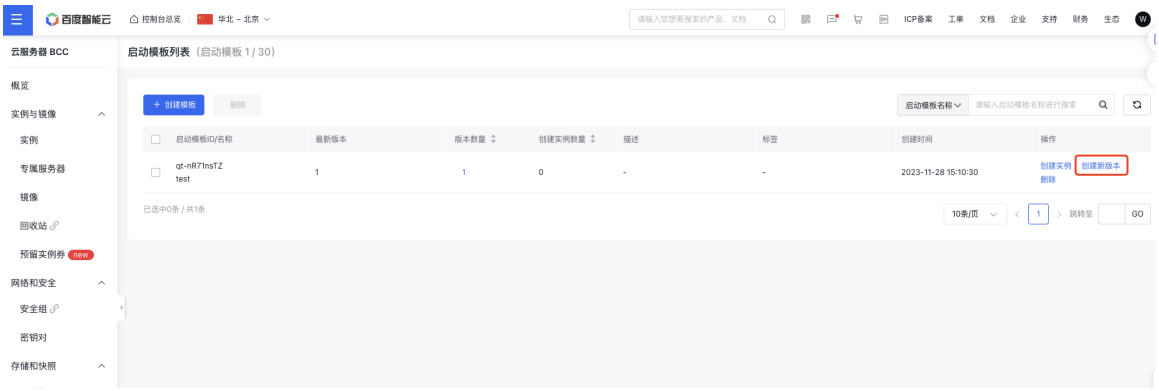
3. 在**创建启动模板**页面编辑模板信息，点击**确认模板配置**。
4. 在**确认配置**页面，输入以下配置信息，并单击**创建启动模板**。

配置项	必选/可选	配置说明
模板	必选	用户根据需要选择模板的保存方式。创建新模板是指新建一个全新模板将当前配置保存为新模板的默认版本。使用现有模板需选择一个已有模板将当前配置保存为该模板的最新版本。
模板名称	必选	用户自定义启动模板名称或直接选择已有模板。 支持大小写字母、数字、中文以及-_ /.特殊字符，必须以中文或字母开头，长度1-65个字符。
模板描述	可选	用户自定义启动模板描述。 支持任何字符，长度为2～256个字符。

5. 创建成功后，可在控制台查看该启动模板。

创建启动模板新版本

1. 登录BCC管理控制台。
2. 点击左侧导航栏**启动模板**，进入启动模板列表页。
3. 在操作栏点击对应模板的**创建新版本**，进入创建启动模板页面。



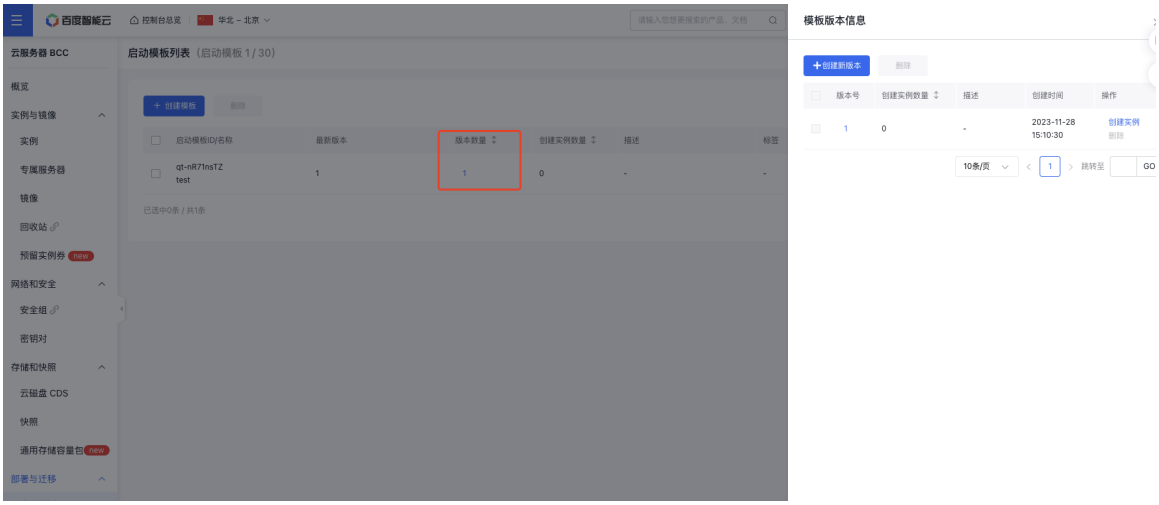
4. 在**创建启动模板**页面编辑模板信息，点击**确认模板配置**。
5. 在**确认配置**页面，输入以下配置信息，并单击**创建启动模板**。

配置项	必选/可选	配置说明
模板名称	必选	用户选择的当前启动模板的名称，不可修改。
模板描述	可选	用户自定义启动模板描述。 支持任何字符，长度为2～256个字符。
模板标签	可选	每个启动模板支持绑定一个或多个标签，具体可参考标签管理进行配置。

6. 创建成功后，可在控制台查看该启动模板。

查看启动模板版本信息

1. 登录BCC管理控制台。
2. 点击左侧导航栏**启动模板**，进入启动模板列表页。
3. 点击对应模板的**版本数量**，在右侧弹窗中点击**版本号**即可查看模板配置信息。



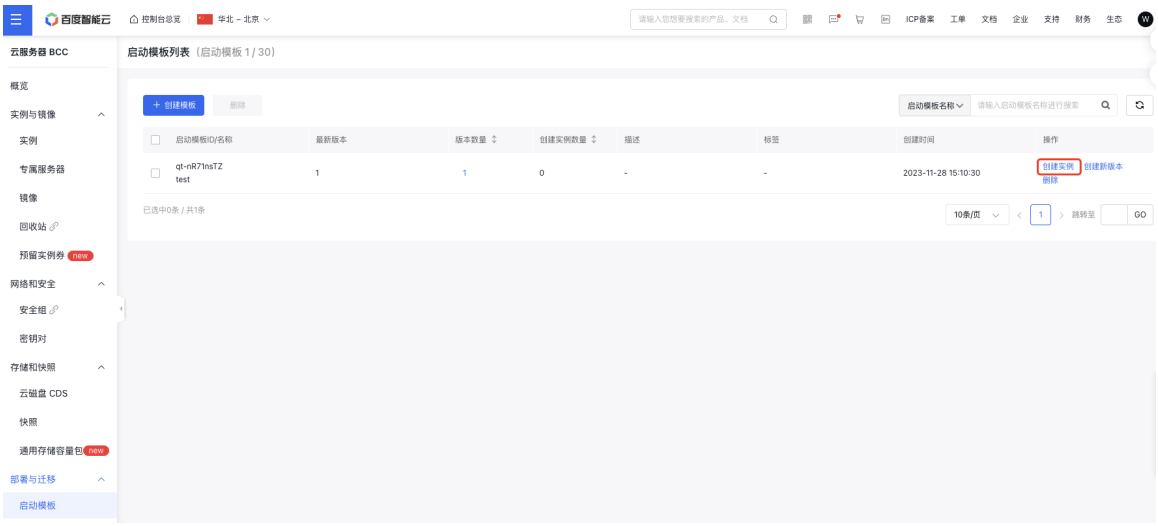
使用启动模板创建实例

- **创建最新版本实例**

1. 登录BCC管理控制台。

2. 点击左侧导航栏**启动模板**，进入启动模板列表页。

3. 在操作栏点击对应模板的**创建实例**，进入创建实例页面。



4. 点击**提交订单**即可创建成功。

> 说明：默认以当前启动模板的最新版本为模板创建实例。

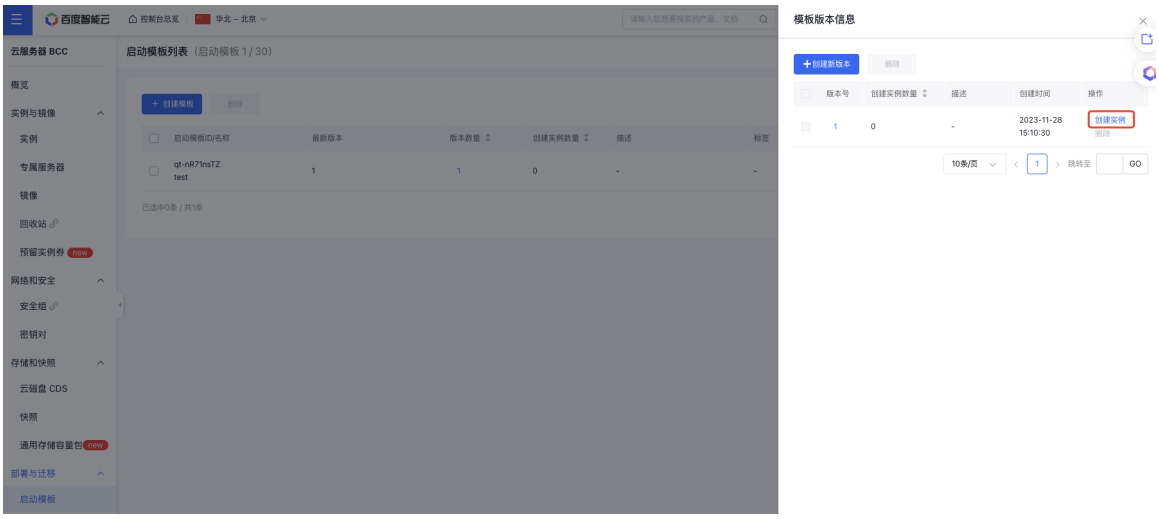
- **创建历史版本实例**

1. 登录BCC管理控制台。

2. 点击左侧导航栏**启动模板**，进入启动模板列表页。

3. 点击对应启动模板的**版本数量**。

4. 在模板版本信息列表中，操作栏点击对应版本的**创建实例**按钮，进入创建实例页面。



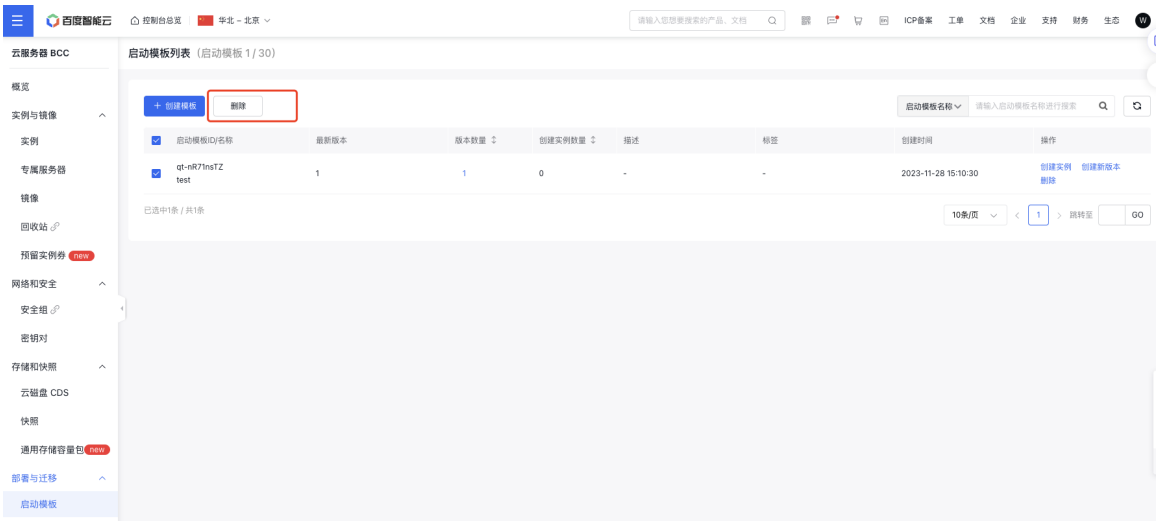
5. 点击**提交订单**即可创建成功。

删除启动模板

1. 登录BCC管理控制台。

2. 点击左侧导航栏**启动模板**，进入启动模板列表页。

3. 在启动模板列表中，选中一个或多个需要删除的启动模板，点击列表上方的**删除**按钮即可完成批量删除操作。您也可以通过操作列的**删除**按钮，对单个模板进行删除。

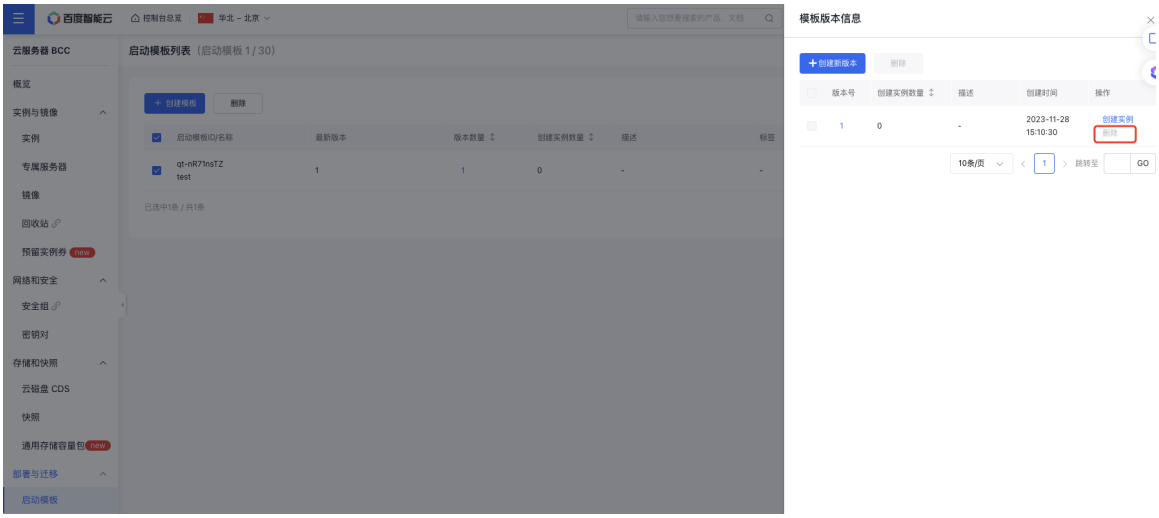


4. 在弹出的**删除启动模板**确认框中，单击**确定**即可删除成功。

> 说明：
>
> - 删除启动模板后，模板下的所有版本都将被删除，请慎重操作。
> - 删除启动模板不影响已经创建成功的实例。

删除启动模板版本

1. 登录BCC管理控制台。
2. 点击左侧导航栏**启动模板**，进入启动模板列表页。
3. 点击对应启动模板的**版本数量**。
4. 在模板版本信息列表中，选中一个或多个需要删除的版本，点击列表上方的**删除**按钮即可完成批量删除操作。您也可以通过操作列的**删除**按钮，对单个模板进行删除。



5. 在弹出的**删除版本**确认框中，单击**确定**即可删除成功。

> 说明：初始版本不支持删除。如果您不再需要某个模板的所有版本，可以删除模板。

部署集

部署集概述

部署集概述

部署集是控制BCC实例分布的策略，使您能在创建BCC实例时就设计容灾能力和高可用性。

部署策略

部署集支持高可用部署策略。高可用部署策略是将一组BCC实例放在不同底层硬件上，从而保证服务的高可用性。

- 采用高可用部署策略后，部署集内所有BCC实例会在指定地域内严格分散在不同的物理机上。
- 高可用部署策略下，遇到地域内资源紧张，可能无法创建BCC实例。
- 部署集之间不支持相互合并。

策略	说明
宿主机	同一部署集内的所有BCC实例会在指定地域内严格分散部署在不同的物理机上，主要用于BCC虚机实例。
机架	同一部署集内的所有BCC实例会在指定地域内严格分散部署在不同机架的物理机上，多用于EBC裸金属实例。
交换机	同一部署集内的所有BCC实例会在指定地域内严格分散部署在不同交换机下联的物理机上，多用于EBC裸金属实例。
集群	同一部署集内的所有BCC实例会在指定地域内严格集中部署在同一网络POD中，降低实例间通信时延。当前仅支持ERI的实例可以加入该类型的部署集。

使用限制

在使用部署集之前，您需要注意以下几点：

项目	说明
配额	部署集数量： • 部署集以地域为维度，单地域可创建10个部署集。 部署集与实例数量限制: • 每个可用区下，每个部署集中最多可有10台BCC实例。 • 一个百度智能云地域下单个部署集中的BCC实例数量 = 10 * 当前地域下的可用区数量。 • 一个实例最多加入2个部署集。
实例规格	• 搭载本地盘的实例（如本地SSD型、大数据型）、GPU实例和EBC裸金属实例，仅支持在创建实例时加入部署集，后续可以移出，但不支持再次加入或更换部署集。

相关操作

- 创建部署集

- 在部署集内创建BCC实例
- 修改部署集信息
- 删除部署集
- 为已有实例调整部署集
- 部署集内移出BCC实例

创建部署集

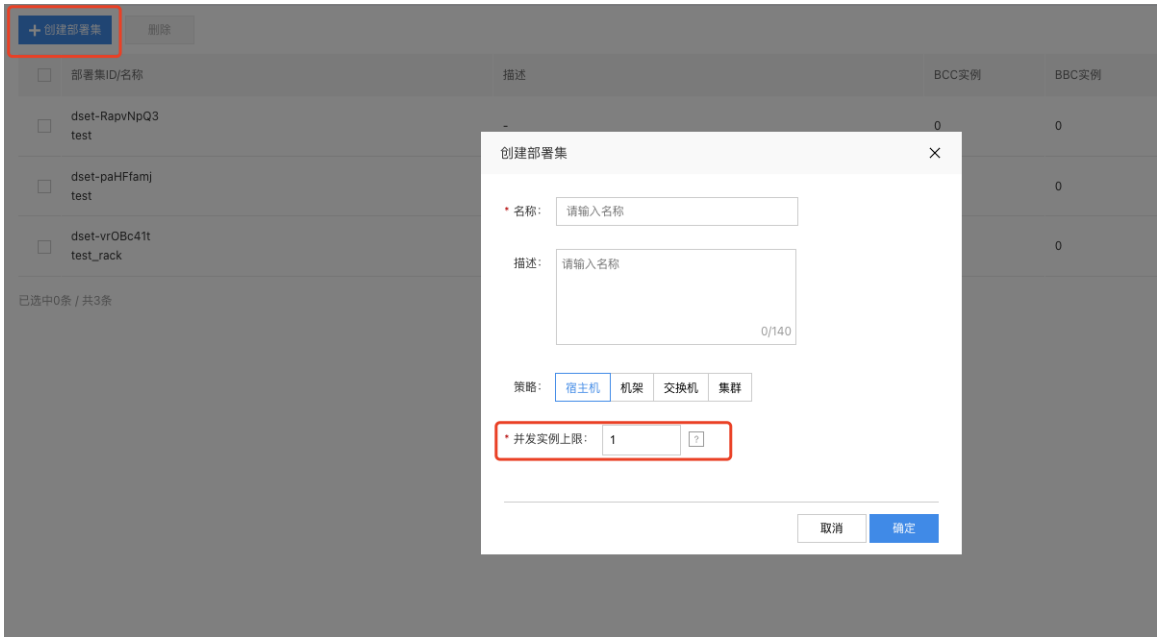
本节介绍如何在BCC管理控制台上创建一个部署集。

操作步骤

1. 登录 BCC管理控制台。
2. 在顶部状态栏左上角处，选择 **地域**。
3. 在左侧导航栏，单击 **部署集**。
4. 在创建部署集页面上，点击创建 **部署集**
5. 在弹窗中输入 **部署集名称**、**描述**，选择**部署集策略**以及**并发实例上限**

> 部署集策略说明可见部署集概述。
>
> **并发实例上限**：加入该部署集的实例允许在单宿主机/机架/交换机/集群中并发数量的上限，编辑时只能调大。
>
> 例：用户某宿主机策略的部署集，并发实例上限设置为3，则该部署集下的实例会优先分散在不同的宿主机上，若底层资源不足时，则单宿主机上最多调度上3台该部署集内的实例。

6. 填写之后点击 **确认**按钮，完成部署集的创建。



在部署集内创建BCC实例

前提条件

您在指定地域中已经创建了部署集。详细步骤请参见创建部署集。

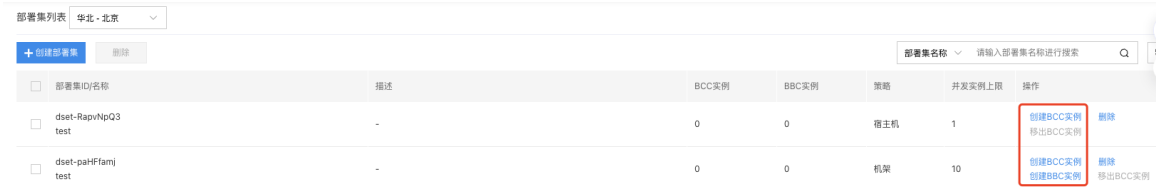
使用限制

- 一个部署集在单地域单可用区内只能创建最多10台BCC实例。

- 一个地域内可以创建（10*可用区个数）数量的BCC实例。

操作步骤

1. 登录 BCC管理控制台。
2. 在左侧导航栏，单击 **部署集**。
3. 选择已创建好的部署集，并在右侧操作栏中点击 **创建BCC实例**，跳转到实例创建页面。



4. 在跳转后的**自定义购买页面**完成实例配置选项，详情见通过购买页创建实例。

> **创建过程的注意项：**
>
> - 地域：BCC实例必须和目标部署集在同一个地域；
> - 可用区：部署集内不能超过10台BCC实例，如需添加更多实例，请提交工单；
> - 实例：带本地盘的BCC规格族（本地SSD型l1/l2/l3，大数据型d1/d1s以及vGPU vgn2/vgn3）均不支持加入至部署集；
> - 部署集：默认选中前面选中的部署集，也可以在此处调整。

5. 确认无误后，创建实例。
6. 在左侧导航栏，单击 **部署集**，您可以查看在部署集内成功创建的实例信息。

更多操作

在BCC创建实例过程中，可以选择已经存在的部署集创建BCC实例。

修改部署集信息

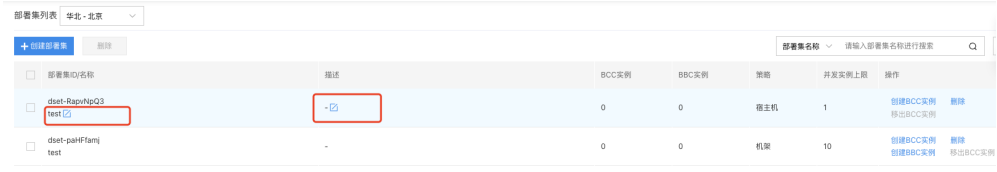
创建了部署集后，您可以重新修改部署集名称和描述信息。

前提条件

请确保您已经创建了部署集，具体操作请参见 创建部署集。

操作步骤

1. 登录 BCC管理控制台。
2. 在左侧导航栏，单击 **部署集**。
3. 找到需要编辑的部署集。
4. 分别将鼠标悬浮至部署集名称和描述处，单击编辑图标，在弹出的对话框中重新填写部署集名称和描述。



删除部署集

您可以删除不再需要的部署集以确保部署集数量不超过使用限制。

前提条件

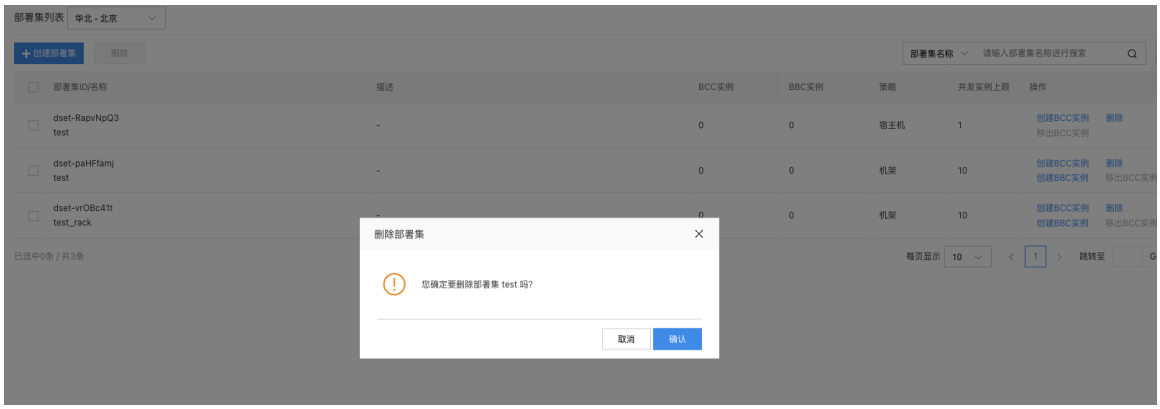
请确保部署集内没有实例。如果存在实例，必须 释放实例 后才能删除部署集。

操作步骤

1. 登录 BCC管理控制台。
2. 在左侧导航栏，单击 **部署集**。
3. 对于要删除的部署集，在操作列下单击 **删除**。

> **说明：** 支持批量删除部署集。

4. 在弹出的对话框中，单击 **确定**，完成删除部署集的操作。



为已有实例调整部署集

前提条件

- 您至少已创建了一台BCC实例。更多详情，请参加创建实例。
- 您至少已创建了一个部署集。更多详情，请参见创建部署集。
- 要进行调整部署集的目标实例必须处于**运行中**的状态。

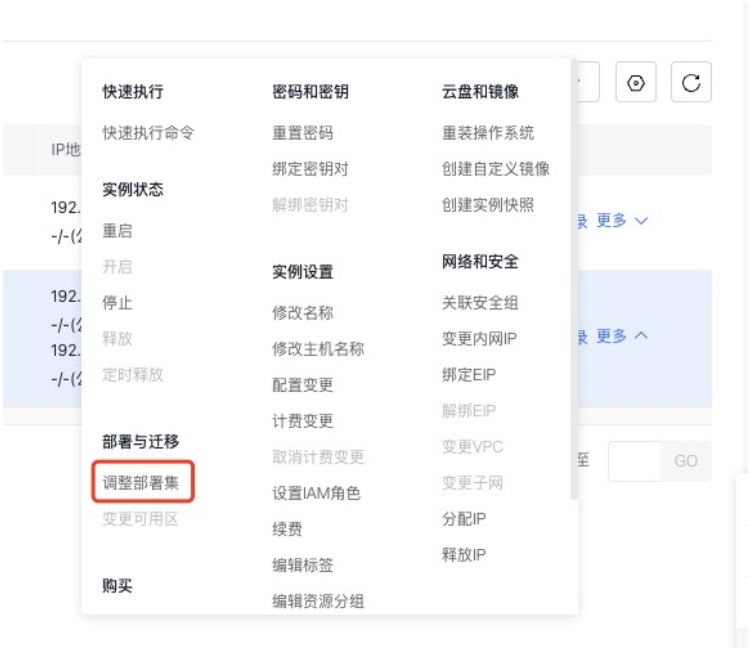
注意

为已有实例调整部署集，会进行实例重启，请您谨慎操作。

带本地盘的BCC规格族（例：本地SSD型l3d、l5等；大数据d1、d1s等）、GPU实例和EBC裸金属实例不支持调整部署集。

操作步骤

1. 登录BCC控制台；
2. 在左侧导航栏选择实例，在左上角选择目标实例所在地域；
3. 在实例列表点击目标实例的【操作】->【更多】->【实例设置】->【调整部署集】；



4. 在弹出的【调整部署集】弹出框中，选择要添加/要去掉的部署集；

5. 点击确认，完成操作。

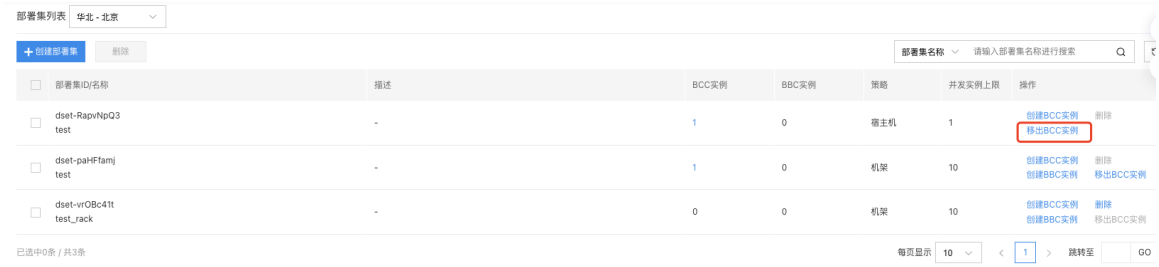
部署集内移出BCC实例

前提条件

- 您至少已创建了一台BCC实例。更多详情，请参加创建实例。
- 您至少已创建了一个部署集。更多详情，请参见创建部署集。
- 实例已加入了至少一个部署集。更多详情，请参见为已有实例调整部署集。

操作步骤

1. 登录 BCC控制台；
2. 在左侧导航栏选择部署集，在左上角选择目标部署集所在地域；
3. 在部署集列表点击目标部署集的【操作】->【移出BCC实例】项；



4. 在弹出的【移出实例】弹出框中，选择要移出的实例；

5. 点击确认，完成操作。

服务器迁移

服务器迁移概述

概述

服务器迁移支持用户将单台或多台服务器迁移至百度智能云，通过此功能可实现将源服务器上的系统、应用程序等从自建机房（IDC）或云平台等源环境迁移至百度智能云，方便实现企业上云、跨云平台迁移等业务需求。

基本概念

- **迁移源**：迁移源（源服务器）指的是用户待迁移的IDC服务器、云服务器、其他云平台的云服务器。
- **迁移任务**：可以查看具体服务器的迁移源信息和目前迁移服务器的信息以及迁移进度。
- **迁移中心**：帮助实现用户的内网间、外网间、百度智能云账号间、百度智能云资源不同地域间的资源迁移。 当前文档仅涉及用户的外网资源的迁入。
- **迁移agent**：需要在源服务器下载迁移agent并启动，上报源服务器信息，配合将服务器数据迁移至百度智能云。

使用场景

在线迁移适用于以下场景（包括但不限于）：

- IT架构上云
- 混合云架构部署
- 跨云迁移
- 跨账号或跨地域迁移

使用限制

- 当前迁移中心支持的迁移源端操作系统版本包括：

类型	操作系统版本
Linux	CentOS 6/7/8
	Ubuntu 16.04/18.04/20.04/22.04/24.04
	Rocky Linux 8/9
	Debian 9/10/11/12
	AlmaLinux 8/9
	Oracle Linux 6/7/8
	OpenSUSE 15/SUSE 15
	Kylin V10
	Alibaba Cloud Linux 2.1903/3.2104
	TencentOS 2.4/3.1
	openEuler 20.03/22.03
	火山OS veLinux 1.0/1.0 Centos兼容版 火山OS veLinux 2.0 | veLinux 2.0 Centos兼容版
	Redhat 6/7/8/9
Windows	Windows Server 2008R2/2012R2/2016/2019/2022

- 您的源端服务器和目标云服务器都需要具有**公网访问**能力。

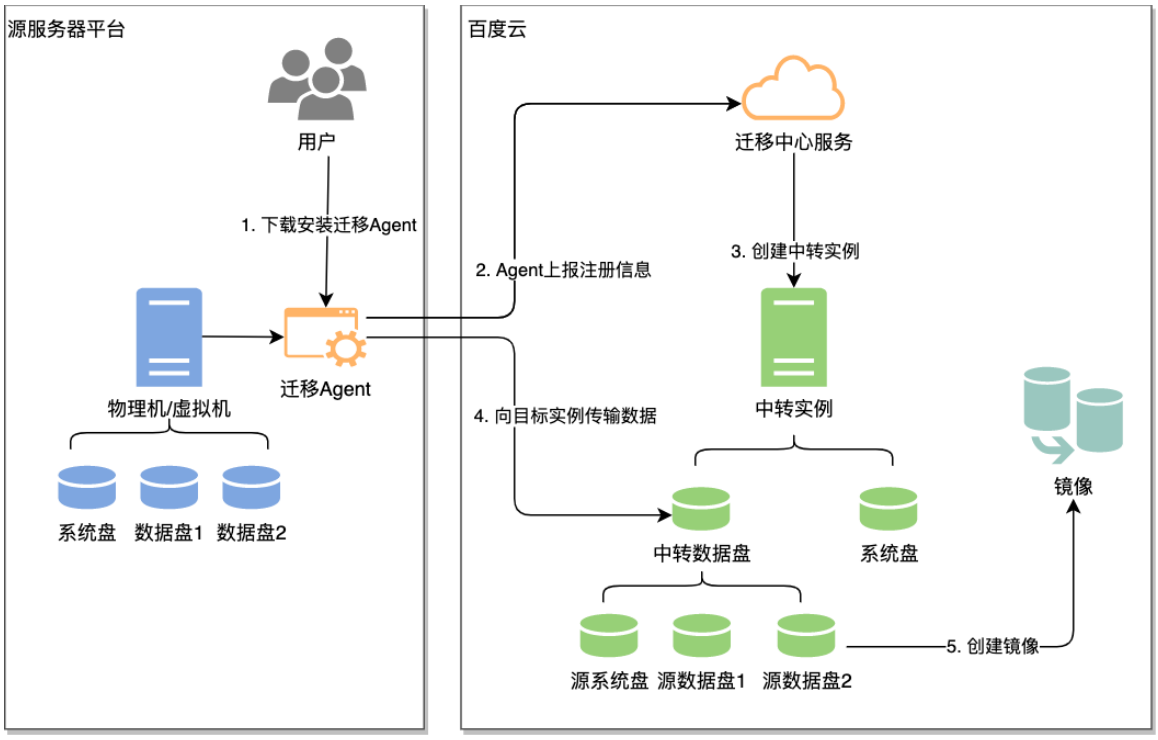
Linux系统使用限制：

- 为了保证迁移后 BCC 云服务器的正常运行，建议用户在迁移之前在迁移源上安装cloud-init，详情请参见 安装cloudbase-init。
- 请在使用之前确保迁移源端安装有版本大于等于 3.1.0 的 rsync。以上支持操作系统中仅CentOS 6不支持。若您待迁移的源服务器为CentOS 6，需升级rsync。
- 当前迁移目标侧只支持ext4和xfs型的系统盘和数据盘，如果迁移源使用其他文件系统类型则迁移后会自动转换成ext4型的文件系统。
- Centos 6的迁移源不支持保持磁盘分区结构，迁移后每个分区会变成独立的磁盘。
- Centos 6不支持UEFI方式启动的迁移源，其他UEFI方式启动的OS的源迁移后会变成Legacy方式启动。

Windows系统使用限制：

- 只支持基本盘的迁移。
- 只支持NTFS格式的文件系统的迁移。
- 源端系统必须以管理组的用户登录。

迁移原理



迁移原理如下：

1. 在源服务器中下载并安装迁移agent。
2. 在源服务器中启动迁移agent，迁移agent会通过公网将源服务器信息注册上报到服务器迁移中心。
3. 创建迁移任务并启动后，服务器迁移中心会创建临时的中转实例（若是Linux，中转实例所在安全组入向需放开22端口；若是Windows，中转实例所在安全组入向需放开12121端口）。
4. 服务器迁移中心通过连接中转实例的22端口来建立通信，将源服务器的磁盘数据传输到中转实例的数据盘。
5. 服务器迁移中心将中转实例的系统盘数据创建镜像，数据盘数据创建快照。

迁移计费

计费项	说明	计费方式
BCC实例（中转实例）	在迁移的过程中，系统会自动创建一个中转实例（配置：2核2G），迁移完成后自动释放。 **说明：** 迁移任务未完成（出错或暂停），中转实例会随任务继续保留一段时间并产生一定的费用。如果您已确认不再进行该任务，可通过删除任务来自动清理中转实例，避免不必要的扣费。	按量付费 说明 为避免中转实例创建失败，您的账号余额不能小于100元。
云盘	在迁移的过程中，系统需要将中转云盘挂载到中转实例上，迁移完成后中转云盘随中转实例同时释放。 **说明：** 中转云盘随中转实例释放，不会对源系统产生影响。	按量付费
快照	在数据迁移完成后，系统会对数据盘做快照，保存数据盘数据。	按量付费 **说明：** 快照按快照容量和存储时长收取存储费用，默认采用按量付费。
公网带宽	创建迁移任务时，会创建一个EIP。通过公网带宽将源服务器的数据迁移到中转实例。数据传输过程中会产生网络费用。 **说明：** EIP带宽根据用户填写大小创建，如果填0或者不填，则按照最大带宽200创建。	按流量计费
自定义镜像	在数据迁移完成后，系统盘数据会生成自定义镜像。	免费

服务器迁移操作流程

本文介绍云服务器迁移的基本使用流程，具体如下图所示。

详细操作步骤请见迁移操作步骤

迁移操作步骤

注意事项

- 若源端服务器是Linux系统，迁移前需要设置源服务器出向开放22端口；
- 若源端服务器是Windows系统，迁移前需要设置源服务器出向开放443、12121端口；
- 因迁移过程会创建中转实例和EIP用于数据传输，数据传输完成后自动释放。中转实例和EIP会按实际使用时长进行计费，故需要账户余额不少于100元。计费情况详见迁移计费。

迁移步骤

源服务器安装并启动迁移工具 Linux 1.下载迁移agent到您的源服务器中。 您可以点击下载安装包至本地后，再上传至源端服务器，或者在源端服务器执行对应下载命令。

操作系统及版本	安装包	下载命令
CentOS 6 Redhat 6 Oracle 6	MigrateToBaiduClientV2.5.0_centos6_x86-64.tgz	wget https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClientV2.5.0_centos6_x86-64.tgz
Centos 7/8 AlmaLinux 8/9 Rocky Linux 8/9 Oracle Linux 7/8 Alibaba Cloud Linux 2.1903/3.2104 TencentOS 2.4/3.1 openEuler 20.03/22.03 火山OS veLinux 1.0 Centos兼容版 火山OS veLinux 2.0 Centos兼容版	MigrateToBaiduClientV2.5.0_centos78-rocky89_x86-64.tgz	wget https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClientV2.5.0_centos78-rocky89_x86-64.tgz
Ubuntu 16.04/18.04/20.04/22.04/24.04 Debian 9/10/11/12 火山OS veLinux 1.0 火山OS veLinux 2.0	MigrateToBaiduClientV2.5.0_ubuntu-debian_x86-64.tgz	wget https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClientV2.5.0_ubuntu-debian_x86-64.tgz
openSUSE 15/SUSE 15	MigrateToBaiduClientV2.5.0_suse15_x86-64.tgz	wget https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClientV2.5.0_suse15_x86-64.tgz
Kylin V10	MigrateToBaiduClientV2.5.0_kylin_x86-64.tgz	wget https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClientV2.5.0_kylin_x86-64.tgz

2. 确保迁移源内存在/root/.ssh/id_rsa和/root/.ssh/id_rsa.pub文件，如果不存在请使用ssh-keygen生成。

3. 运行以下命令进行解压缩。

```
tar -xzvf migagent_xxx.tgz
```

例若源端服务器是CentOS 7，则运行：

```
tar -xzvf MigrateToBaiduClientV2.5.0_centos78-rocky89_x86-64.tgz
```

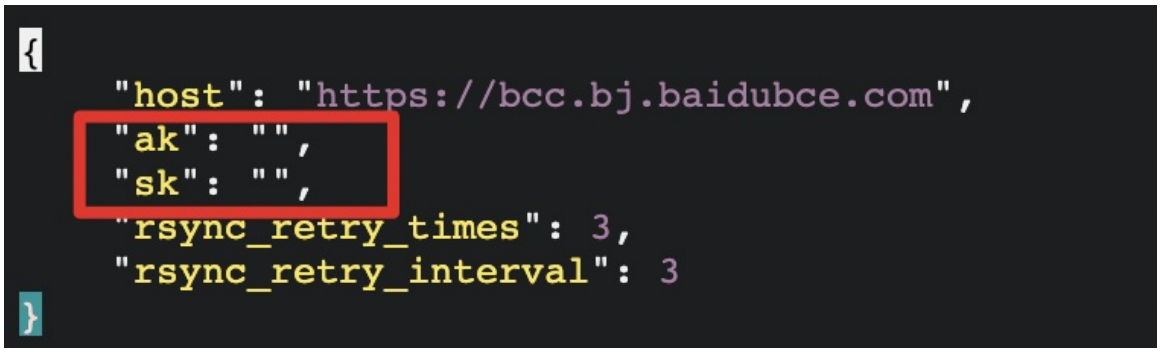
4.运行以下命令进入agent目录。

```
cd migrate-center-agent_pyinst
```

5.运行以下命令，编辑**user_cfg.json**，填入百度智能云账户的AK、SK。

```
vim user_cfg.json
```

移动光标，键盘按`i`，填写AK、SK。填写完毕，`esc`键退出，`:wq`保存退出。



> 若您有专线接入，通过私网传输进行迁移，host也需要调整为迁移目标地域的访问域名。各地域域名见服务域名。
>
> 若您通过公网传输迁移，则host无需变动。

6.运行以下命令，启动迁移源。

```
./migrate-center start
```

Windows 1.下载迁移agent到您的源服务器中并解压。

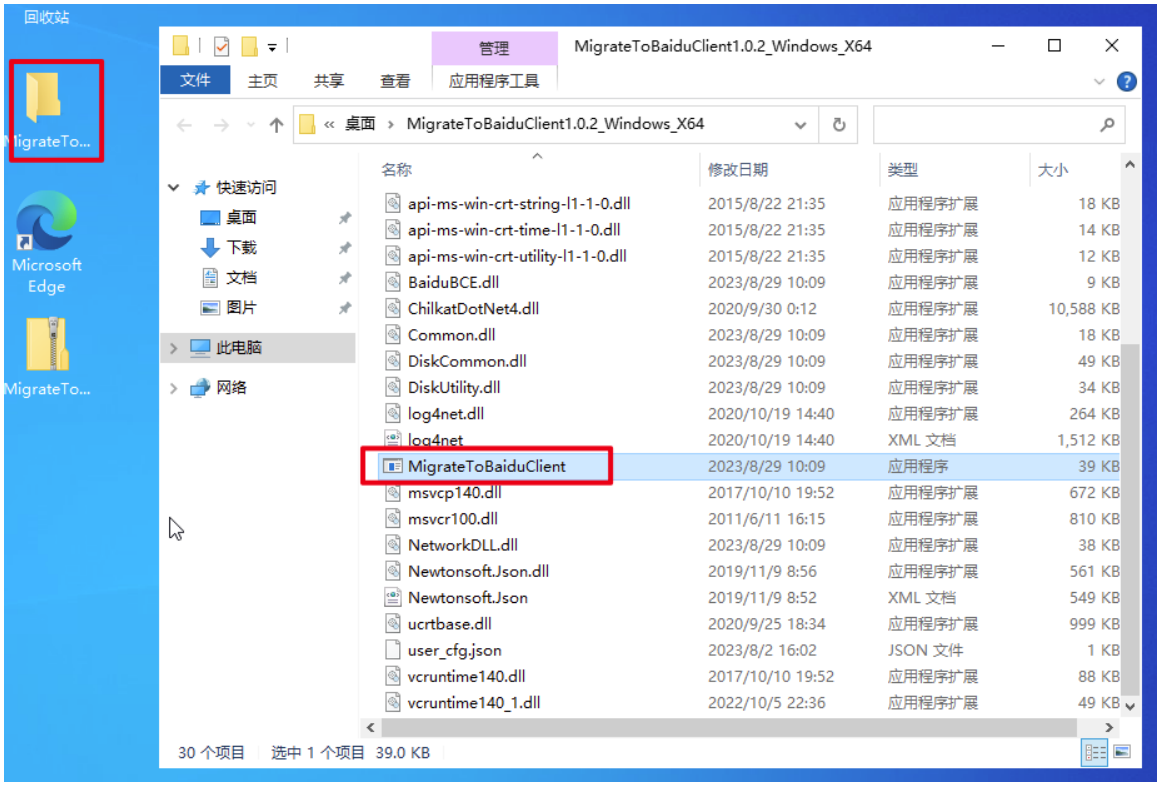
点击下载：MigrateToBaiduClient1.0.2_Windows_X64.zip，

或者在源服务器，打开Windows PowerShell，输入以下命令：

```
Start-BitsTransfer -Source https://wanjuan-docs.bj.bcebos.com/MigrateToBaiduClient1.0.2_Windows_X64.zip -Destination C:\Users\Administrator\Desktop\MigrateToBaiduClient1.0.2_Windows_X64.zip
```

下载完成后即在桌面上生成一个名为MigrateToBaiduClient1.0.2_Windows_X64.zip的压缩包，右键选择“全部解压”。

2.解压后，在解压后的文件夹（MigrateToBaiduClient1.0.2_Windows_X64文件夹）找到并双击运行“MigrateToBaiduClient”应用程序。



3.填入迁移创建出的镜像所属用户的 AK、SK，并点击启动。

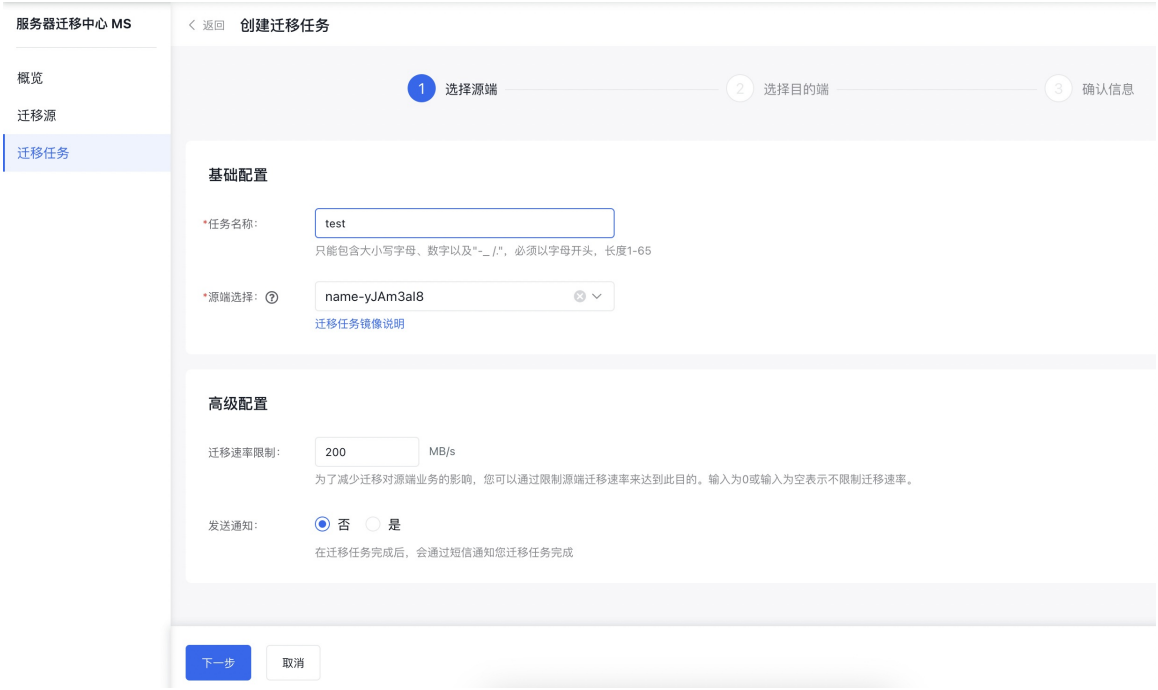
管理迁移源 前提条件

- 迁移agent在源服务器安装成功并启动完成

操作步骤

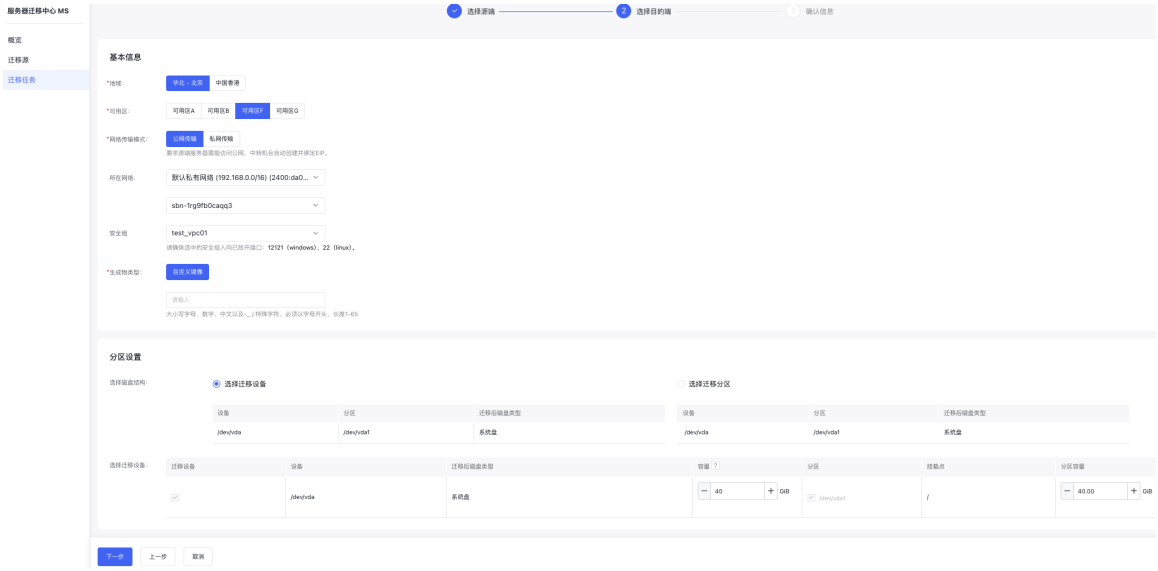
1. 待迁移agent安装成功并启动后，前往百度智能云控制台进入云服务器实例列表。
2. 在控制台左侧导航点击 服务器迁移中心，进入服务器迁移中心。
3. 当迁移agent成功启动，迁移源自动创建。迁移源信息会自动展示在列表中。
4. 仅当状态为 在线 的迁移源可以新建迁移任务。
5. 只有无迁移任务的迁移源才可以被删除，若需要删除迁移源，需要先删除迁移任务。

新建迁移任务 迁移源状态为在线时，可以新建迁移任务。 **1、选择源端**



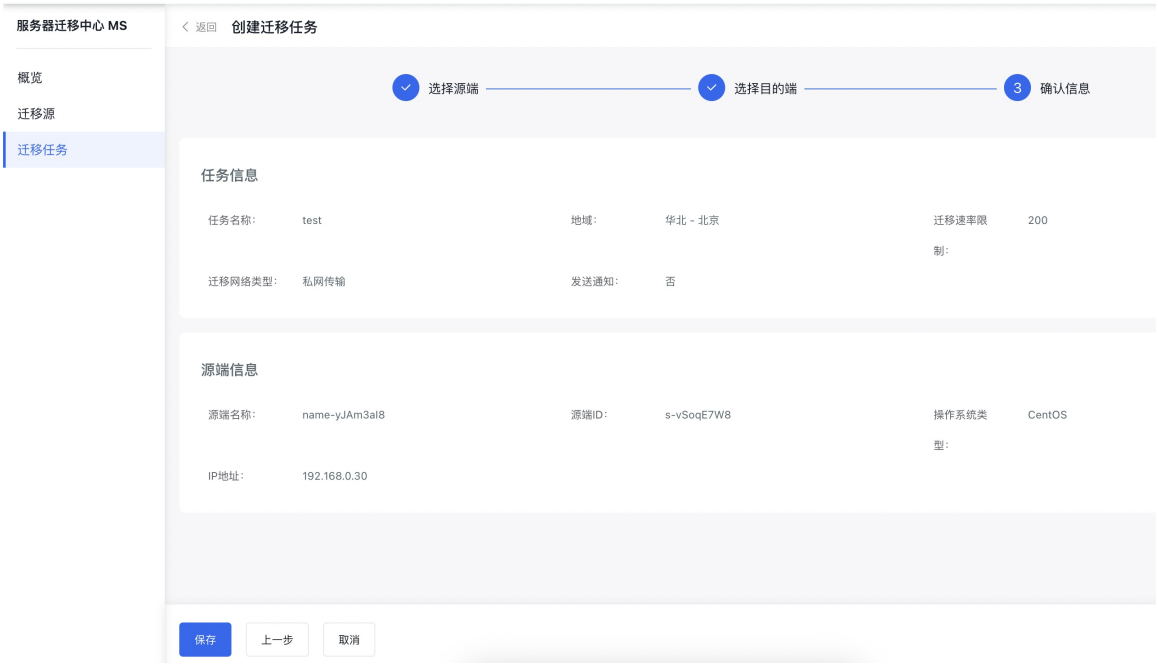
配置项	说明
任务名称	用于标识您创建的迁移任务。
源端选择	选择您要进行迁移的迁移源。
迁移速率配置	为了减少迁移对源端业务的影响，您可以通过限制源端迁移速率来达到此目的。输入为0或输入为空表示不限制迁移速率。
发送通知	选择迁移任务完成后是否发送通知。若选择是，则会通过短信通知您迁移任务完成。

2、选择目的端



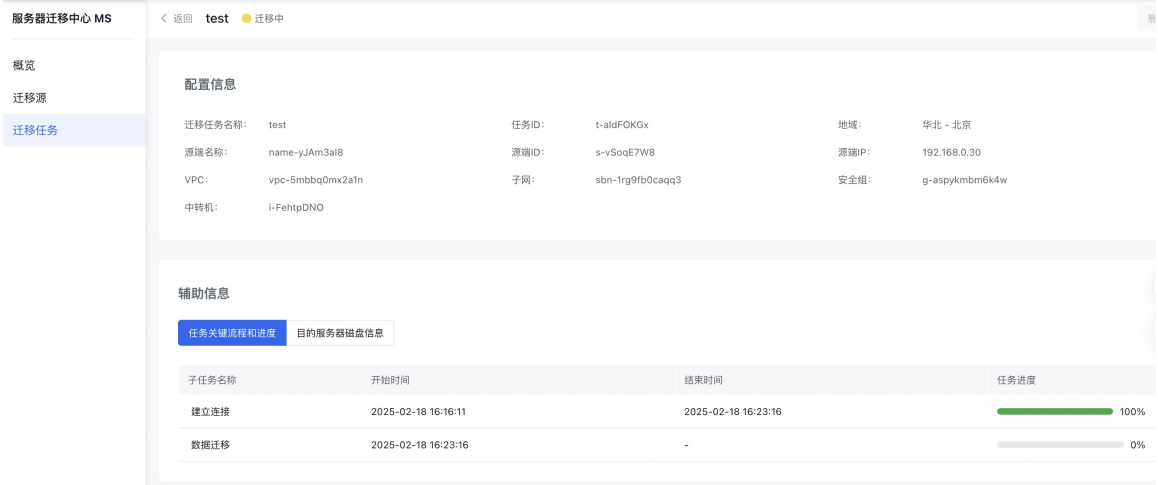
配置项	说明
地域	选择迁移到百度智能云后的服务器地域。
可用区	选择迁移到百度智能云后的服务器可用区。
迁移网络类型	公网传输：中转机会自动创建并绑定EIP。 私网传输：要求源服务器能够与百度智能云指定私有网络打通（参见专线接入），同时必须设置并指定VPC/子网的私有网络参数。
所在网络	选择中转实例加入的VPC和子网。若有专线接入，则选择专线接入的VPC和子网。
安全组	中转实例加入的安全组，请确保选中的安全组入向已放开端口：12121（windows）、22（linux）。
生成物类型	当前仅支持自定义镜像，可自定义该镜像的名称。
分区设置	默认选择全部磁盘进行迁移，可以按需取消勾选，取消勾选的磁盘或者分区则不会迁移。 注意：低于5G的数据盘不显示，不支持迁移。

3、确认信息 信息确认无误后点击**保存**，迁移任务列表生成一条迁移任务。任务状态为**未开始**。



4、启动迁移任务 在迁移任务列表在对应迁移任务操作栏，点击**启动**，任务状态变更为**迁移中**。

可点击迁移任务名称，进入详情查看任务进度。



其他说明

启动/暂停/删除迁移任务限制和说明：

- 启动：会校验账户是否欠费，如果欠费则提示无法启动任务。如果用户选择的是走自动创建代理服务器流程，启动的校验就是创建后付费资源的校验。
- 暂停：中止迁移任务，任务状态变为已暂停。总耗时时间记录正常进行。
- 删除：如果迁移任务状态为『迁移中』，迁移任务不可删除，需要先暂停，才可删除； 删除迁移任务，删除后迁移任务从列表里消失，且创建的自动代理服务器也随着任务删除而释放。

异常处理

若您的迁移任务状态异常，可从以下几种情况排查。迁移任务状态异常的几种情况：

- 迁移任务异常，可能由于网络不稳定。若失去心跳超过3分钟，迁移任务将会异常。
- 迁移任务异常，可查询迁移源状态是否异常。
- 迁移失败，可能是因为迁移过程中有对中转实例进行了关机等操作，或者没有提前设置中转实例所在安全组中是否放开入向ssh访问22端口。
- 迁移任务一直在迁移中，可能是因为中转实例未创建出来，请确认中转实例是否创建成功。
- 迁移任务失败后，需要删除失败的迁移任务，才能对迁移源重新新建迁移任务。

常见问题

如果迁移源是IDC物理机，并且网卡名不是ethX，则迁移后的网卡名可能会变成ethX而导致网络不通。这种情况下可以用SmartTerm/VNC登录服务器，删除旧网卡的配置，并按如下建议配置eth0，然后重启网络服务。

Centos 6/7/8:

删除/etc/sysconfig/network-scripts/下的旧网卡配置，并按如下配置创建ifcfg-eth0

```
 TYPE=Ethernet
DEVICE="eth0"
ONBOOT=yes
BOOTPROTO=dhcp
RES_OPTIONS="rotate timeout:1"
IPV6INIT="yes"
IPV6_PEERDNS="no"
DHCPV6C="yes"
DHCPV6C_OPTIONS=-nw
IPV6_DEFROUTE="yes"
IPV6_PEERROUTES="yes"
```

Ubuntu 16.04:

编辑/etc/network/interfaces添加如下配置

```
 auto eth0
iface eth0 inet dhcp
```

Ubuntu 18.04/20.04:

编辑/etc/netplan/01-netcfg.yaml

```
network:
version: 2
renderer: networkd
ethernets:
  eth0:
    dhcp4: yes
```

Ubuntu 22.04:

编辑/etc/netplan/01-netcfg.yaml

```
network:
version: 2
renderer: networkd
ethernets:
  eth0:
    dhcp4: yes
    dhcp-identifier: mac
```

标签管理

标签管理概述

概述

百度智能云提供标签管理功能，支持按照各种标准（如用途、所有者或项目）提供跨服务、跨地域对资源进行添加标签，从而快速分类和管理资源。

应用场景

场景一

随着用户云资源的种类和数量逐渐增多，可以通过标签管理服务将资源基于不同维度进行分组管理，以便对资源快速检索和操作。

用户可对不同的应用、服务模块、或者环境等绑定不同的标签，如对某OA系统生产环境和测试环境下的资源分别绑定标签，进而可在不同的资源类型下快速筛选出某标签下的资源进行操作；如对于云服务器BCC下筛选出标签为测试环境的实例进行批量操作。

场景二

在部门或者项目管理中，可以根据不同视角如部门、团队或项目去分类资源，从而进一步划分成本和权限控制。

如某游戏客户有多个大型游戏项目，基于标签可以帮助用户按照不同的游戏项目进行资源分类管理，通过查看不同标签的资源账单可进一步控制及汇报项目成本，还可以根据不同标签对项目赋予权限控制。

核心能力

- 支持批量创建标签及快速检索标签
- 支持查看标签下各产品及各region下所属资源
- 支持跨服务、跨地域批量绑定、解绑资源
- 支持收藏标签，便捷查看及管理标签资源
- 支持查看并导出不同标签下资源账单
- 支持根据标签赋予子账号不同权限范围

使用说明

使用标签前，您需要了解：

- 标签：每个标签由键和值两部分组成，标签（键+值）唯一。
- 单个和批量：支持为单个资源设置标签，也可以批量为云资源创建标签。
- 排序和搜索：支持按照标签（键和值）进行资源搜索和排序。
- 跨产品标记：不同产品可以添加同一个标签，例如给BCC 添加标签时可以将关联的CDS、EIP统一加标签。

标签规格

每个用户最多可以创建200个标签。

配置标签

添加标签

拥有多台云服务器的用户可以根据项目和场景需求，为BCC实例添加标签，便于对实例进行分类和识别管理。

> **说明：** 批量创建实例标签时，无法查看已添加的单个实例标签。

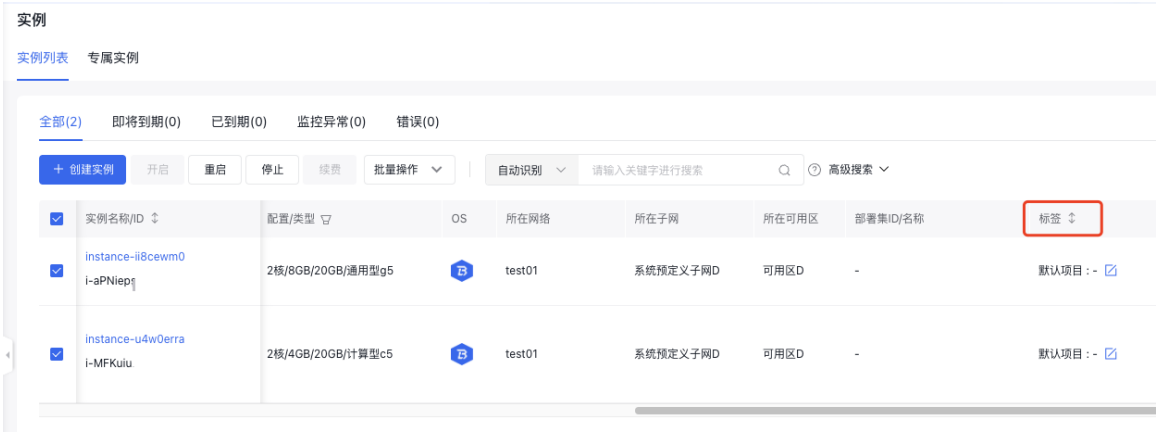
操作步骤

1. 登录 BCC管理控制台。
2. 选择页面顶部 **地域**，点击左侧导航栏 **实例**，进入实例列表页。
3. 勾选一个或多个实例，操作栏点击 **批量操作 > 编辑标签**。
4. 在弹出的对话框中，输入自定义标签键和标签值，注意"键"必须唯一，值可以为空，不填写。
5. 点击 **确定**，完成标签创建。

标签排序和搜索

标签排序

用户可以在控制台对云资源的标签进行排序，点击列名 **标签**，进行排序。

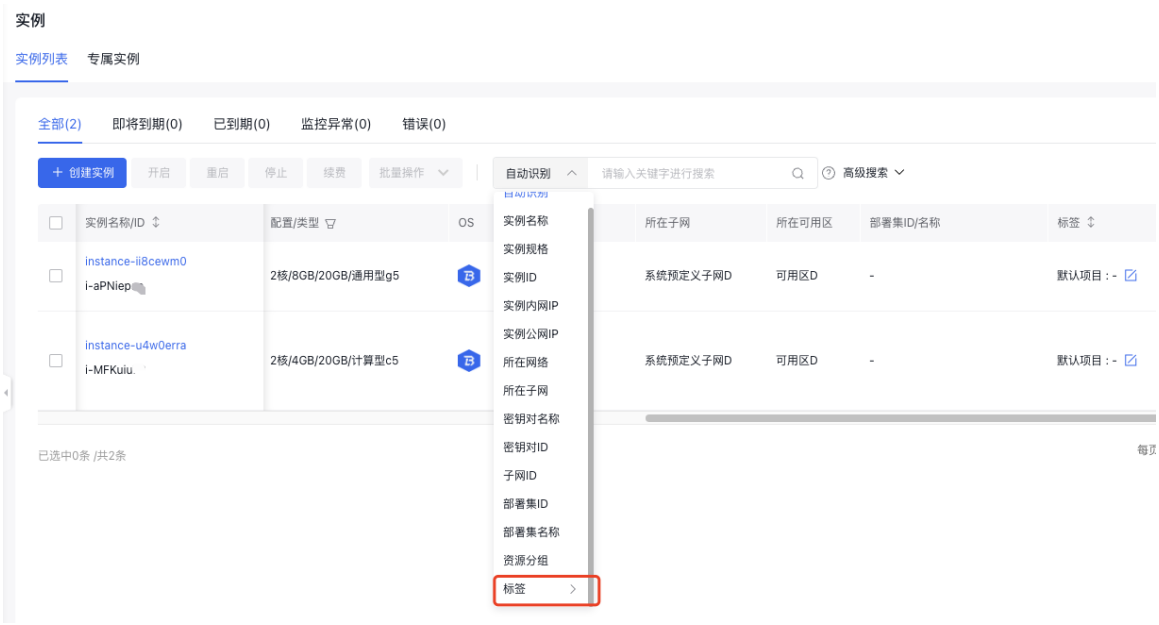


标签搜索

通过组合筛选条件执行标签搜索，从而获得精确的云服务器结果，您可以按标签键和值进行筛选。

> **说明：**
>
> - 所有值，表示当前键下所有自定义的值。

- 空值，表示当前键下未设定的值。



解绑标签

如果实例不再需要标签，您可以将标签解绑。

操作步骤

1. 登录 BCC管理控制台。
2. 点击左侧导航栏 **实例**，进入实例列表页。
3. 勾选一个实例，操作栏点击 **更多 > 实例设置 > 编辑标签**。
4. 弹出对话框中，点击标签名称旁的**删除**，即可解绑该标签，解绑实例标签后，与此实例关联的CDS、EIP的标签也自动解绑。

删除标签

如果已创建的标签不再使用，您可以将标签删除。

注意事项

标签删除后无法恢复，系统自动解绑之前的资源。

操作步骤

1. 登录 BCC管理控制台。

2. 点击左侧导航栏 **标签管理**，进入标签列表页。
3. 展开一个或多个标签键的下拉列，选择要删除的便签值，点击 **批量删除便签值**。
4. 弹出对话框中，点击确定即可删除该标签。

基于标签查看资源账单

资源账单是记录每个资源的生命周期与消费相关的数据的功能，方便用户以资源为维度查看资源及产品的消费数据。

操作步骤

1. 登录 BCC管理控制台。
2. 点击左侧导航栏 **标签管理**，进入标签列表页。
3. 勾选一个标签，操作栏点击 **查看账单**。
4. 进入资源账单页面，选择需要查看账单的月份，会显示当前标签资源的消费信息。

> **说明：**
>
> - 可以通过组合筛选条件执行标签搜索，按标签键和值进行选择其他标签的消费详情，并下载相关信息。
> - 支持以实例ID进行检索，可以查看实例的消费信息。

5. 点击查看详情，可以查看实例ID、产品名称、账单金额。

> **说明：**
>
> - 实例有可能会有预付费和后付费都存在的情况，所以，点击查看实例详情，会分tab显示后付费的账单和预付费的账单。
> - 因为预付费的是一次性扣费，消费信息只会在创建当月显示，而后付费的是一直持续扣费，消费信息会在创建时间到使用时间段内显示。

预留实例券

预留实例券概述

预留实例券介绍

预留实例券（Reserved instance，简称RI）是一种抵扣券，可以抵扣相关配置的按需付费BCC实例账单，也能够预留实例资源，支持多种购买周期，对比包年包月预付费，预留实例券与按量付费实例这种组合模式灵活性更好。

预留实例券在生效期间会预留对应资源，**购买预留实例券意味着您承诺了对应的购买时长，预留实例券在购买后无法退订。**

适用范围

RI 按适用范围可分为：可用区（AZ）级别的券与区域（Region）级别的券。

适用范围	概念	优势
可用区级预留实例券（当前已支持）	实例券生效后将自动匹配并抵扣相同AZ、实例规格以及EHC集群（仅适用EHC类型实例）的后付费BCC实例账单	支持资源预留，购买预留实例券后，百度云将保证您账号下在对应的AZ中至少能有与RI相同的实例规格*台数的后付费BCC资源。
区域级预留实例券（即将支持，敬请期待）	支持抵扣该Region所有可用AZ下同等算力的后付费BCC实例账单	灵活自动抵扣。

付费方式

RI 按付费方式可分为：全预付、部分预付、0预付。

> 0预付和部份预付会根据您在百度智能云的信控等级开放，如您在控制台无法选择这两种计费请提交工单。

付费方式	概念	总支付金额
全预付	购买券时支付购买周期内对应算力的总额，价格一般与预付费一致。总价最低。	总价最低
部分预付	购买券时支付购买周期内对应算力50%的费用，其他费用将于有效期内每小时扣除（类似后付费）。总价居中。	总价适中
0预付	购买券时无需支付任何费用，费用将于有效期内每小时扣除（类似后付费）。总价最高。	总价最高

预留实例券及后付费实例计费对比后付费实例计费

	预留实例券+后付费实例	后付费实例
定义	通过预留实例券来抵扣后付费BCC实例账单。	购买BCC时的一种付费方式（先使用后付费），分钟计费，小时出账
特点	1. **价格优惠**：总成本比后付费实例低； 2. **资源保障**：AZ级实例券支持资源预留； 3. **灵活抵扣**：实例券不与某个实例绑定，一张实例券可抵扣同规格族的不同后付费BCC账单 4. **灵活支付**：支持按每小时支付（部分预付或0预付），避免一次性支付带来的资金链压力。 5. **承诺时长**：购买实例券意味着您承诺了相应的资源购买时长，购买后**实例券无法退订**，但您可以转将实例券转移给其他账号使用，	1. 即用即买，随时释放 2. 单价较高

资源抵扣

实例券生效后将自动匹配相同AZ、规格族的按需付费-后付费BCC实例进行抵扣，如您购买的是EHC实例规格，则实例券会自动匹配同EHC集群的实例。您无需再为被抵扣的实例额外付费。在预留实例券到期后，按需付费-后付费BCC实例将按照实例费用出账并恢复计费。例如：

> 举例：2c4g实例，后付费：1元/小时； 4c8g实例，后付费：2元/小时；一张 2c4g Region级预留实例券0预付：预留费用0.7元/小时；一张 4c8g AZ级预留实例券0预付：预留费用1.4元/小时。
>
> 用户买了2台4c8g实例、1张2c4gRegion级预留实例券和1张4c8gAZ级预留实例券。因1张4c8gAZ级预留实例券匹配一台4c8g实例，这台4c8g实例费用为0元/小时；1张2c4gRegion级预留实例券抵扣另1台4c8g实例的2c4g算力，这台费用为(2-

> 1)=1元/小时，总费用为2张预留实例券与1台2c4g算力的费用相加，即：(0.7+1.4+1)=3.1元/小时。

预留实例券创建

功能说明

本文介绍如何在云服务器控制台购买预留实例券。

前提条件

- 创建预留实例券之前必须完成实名认证。请根据自身情况，进行“企业认证”或者“个人认证”，具体请参考 个人认证和企业认证。
- 付费类型0预付和部分预付会根据您在百度智能云使用云产品的情况开放。如在控制台没有显示对应的付费类型，请提交工单。
- 创建部分预付或者0预付预留实例券时，需要您的账户余额大于或等于100元整。

操作步骤

1. 登录 云服务器控制台，左侧栏选择**预留实例券**，进入**预留实例券**列表页。
2. 在顶部菜单栏左上角处，选择**目标地域**。
3. 单击**创建预留实例券**。

4. 选择**地域**、**可用区**，预留实例券与要抵扣的按需付费-后付费BCC实例在同一个可用区下才可抵扣BCC实例账单。
5. 选择**实例配置**，与需要抵扣的后付费BCC实例规格一致。
6. 选择**弹性高性能计算集群**，与需要抵扣的后付费实例一致，**实例券仅支持抵扣同EHC集群的实例。**

7. 设置**实例券名称**，可以根据需要选择**用户自定义**或者**随机生成**。
8. 设置**实例数量**，当前1个券支持抵扣1个BCC实例。
9. 选择**付款类型**，按需选择全预付、半预付、0预付。
10. 设置**预留实例券数量**，缺省值为1，如您需要抵扣多个BCC实例，请填写对应的券数量。默认配额为50，如需更多配额，请联系工单。
11. 设置是否开启**自动续费**，实例券到期后是否开启自动续费快到期的券，默认关闭。

12. 设置自动续费的**续费周期**，当前支持按照购买时长的周期执行自动续费。
13. 设置**生效时间**，默认是**立即生效**，您也可**自定义生效时间**，最长可设置24小时，支持设置到分钟级别。
14. 设置**绑定标签**，可以选择已有的**标签键**和**值**，也可以添加新标签。
15. 确认费用价格，查看控制台显示的价格是否符合预期，预付费用代表券生效前预先支付的费用，分期费用即在券的有效期内每小时支付的费用，月预估表示折算到每个月需要支付的分期费用。

16. 点击**下一步**，在订单确认页勾选**同意《百度智能云线上订购协议》**，并点击**确认订单**即可。

最终结果

在**预留实例券**列表页可看到已创建成功的预留实例券，并可抵扣相同实例规格的后付费BCC实例账单。

查看抵扣明细

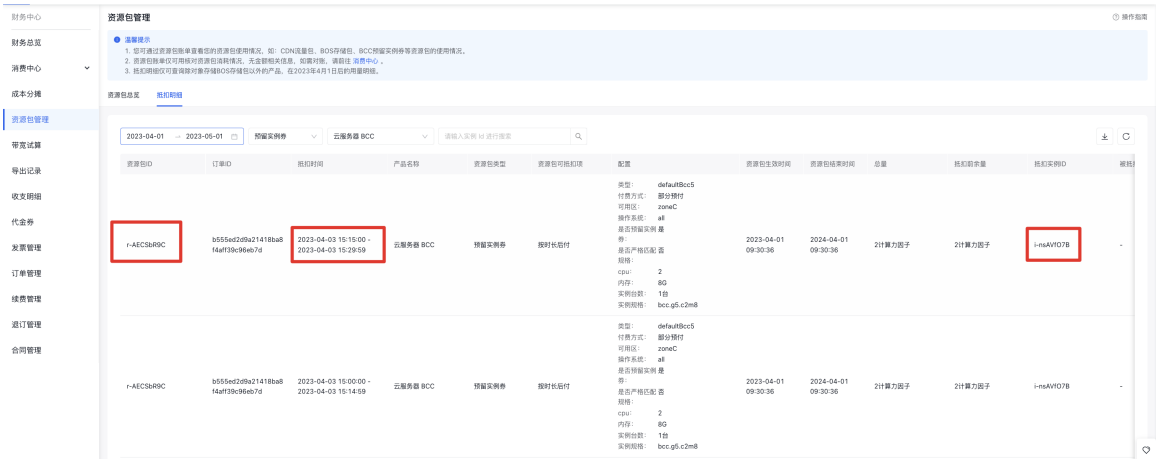
可在抵扣明细页面中查看预留实例券抵扣BCC实例的抵扣明细。

入口1：在预留实例券列表页**，选择要查询的实例券，在**操作**中点击**查看抵扣明细****

入口2：

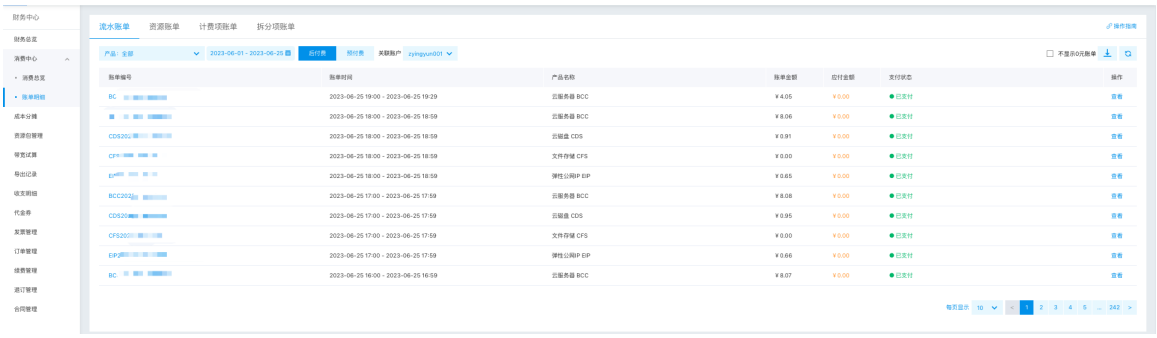
1. 在控制台右上角点击**财务**，进入**财务中心**

2. 在财务中心页面中的左侧栏点击**资源包管理**，在头部标签页中点击**抵扣明细**

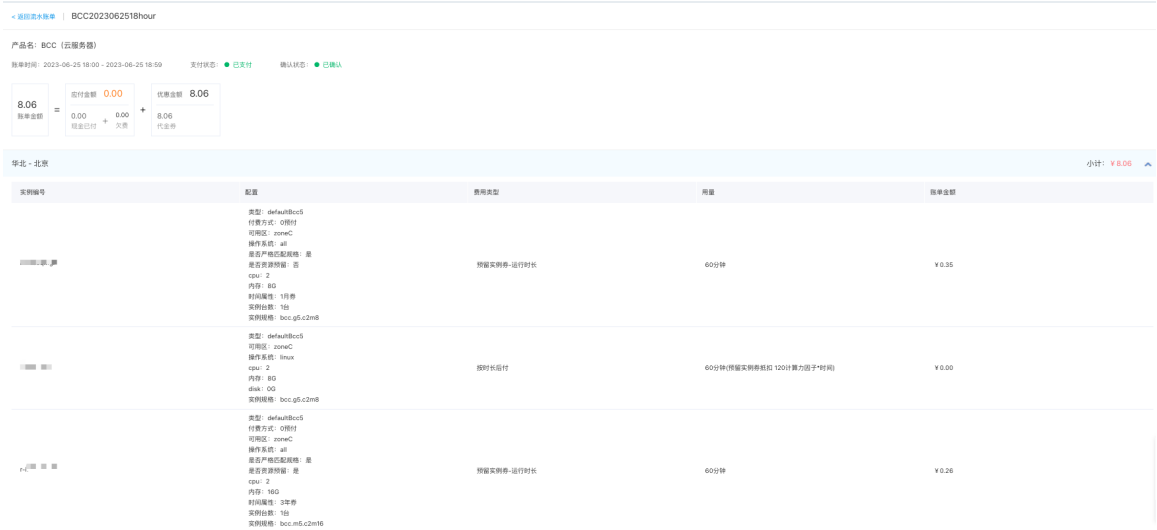


查看预留实例券流水账单

1. 进入流水账单



2. 点击近期的账单编号，进入账单详情



注：预留实例券和后付费的BCC均为15min扣费，每小时出账

预留实例券续费

功能说明

本文介绍如何对已创建的预留实例券做续费操作，续费分为手动续费和自动续费。执行手动续费会直接为该预留实例券延长到期时间，开启自动续费的预留实例券会在到期前的7天自动执行续费。

前提条件

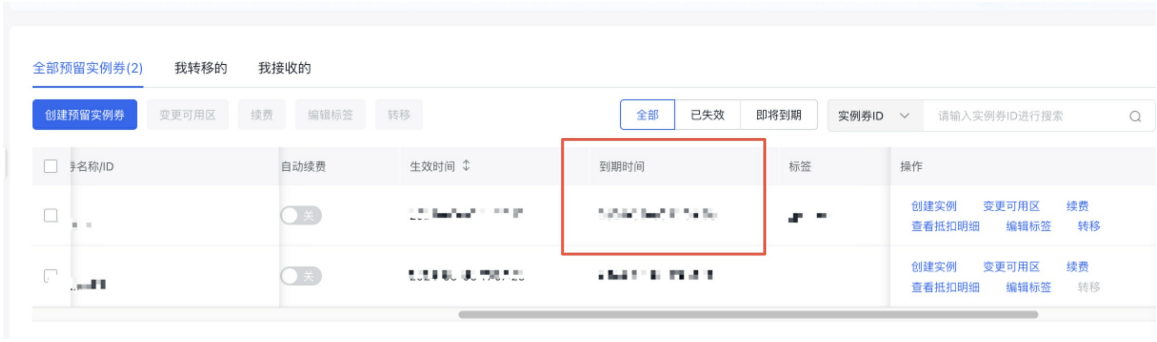
- 实例券处于生效中状态。

操作步骤

手动续费

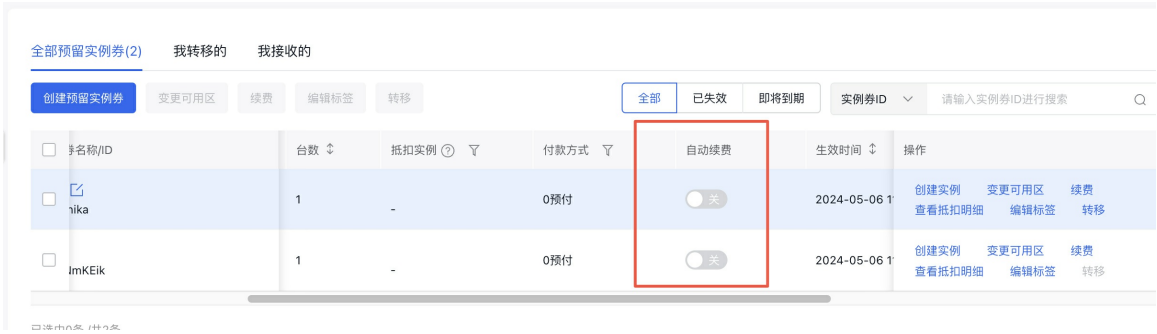
1. 登录 云服务器控制台，左侧栏选择**预留实例券**标签页，进入**预留实例券**列表页。
2. 在顶部菜单栏左上角处，选择要续费的实例券所在的**目标地域**。
3. 在预留实例券列表中找到要续费的实例券，在其**操作**中，单击续费。

4. 选择续费时长后，进入**订单确认页**，完成下单后，可在实例券列表查看到期时间确认续费后的到期时间。



自动续费

1. 登录 云服务器控制台，左侧栏选择**预留实例券**标签页，进入**预留实例券**列表页。
2. 在顶部菜单栏左上角处，选择要续费的实例券所在的**目标地域**。
3. 在预留实例券列表中找到要开启自动续费的实例券，在列表中找到自动续费开关，可按需开启关闭自动续费。



预留实例券跨账号转移

功能说明

本文介绍如何转移预留实例券的所属权到其他账号，当前预留实例券所属的账号可以作为发起方申请转移给作为接受方的其他账号，在成功转移后，转移的预留实例券不再对发起方计费、预留资源以及抵扣对应实例计费，而是对接受方账号继续计费以及预留资源。转移前后的预留实例券的关键属性不发生变化，包括以下：

- 预留实例券名称/ID。
- 生效时间、到期时间。
- 预留实例券的付款方式。
- 支持抵扣的实例规格。
- 区域及可用区。

前提条件

- 发起转移的券需要为0预付费券，暂不支持转移其他付款类型。
- 发起转移的券需要为生效中状态。
- 接收方账号有创建0预付费预留实例券的权限。

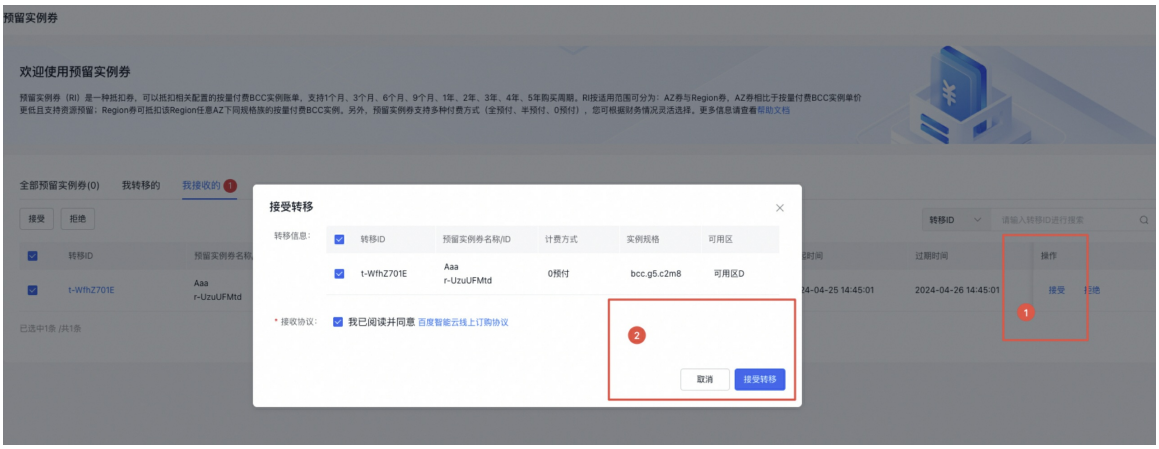
操作步骤

1、登陆券的控制台，在图中编号1处勾选需要转移的券，并点击图中编号2申请转移。

2、填写接收方账号id，点击确认后发起转移。

3、发起后，在发起方账号的我转移的页面可看到刚才转移的券，同时，在接收方账号的我接收的可看到待处理的券。注意转移的有效期为24小时，如接收方没有对转移在24小时内做任何操作则转移会变为超时状态且无法处理。

4、点击图中编号1的接受后出现确认弹窗，确认转移信息后点击接受转移。



5、点击接受后，转移券进入转移中状态，预计持续几秒后显示转移成功，接收方账号可在实例券列表中看到转移成功的实例券，且该实例券开始在接收方按照接收方账号的计费规则执行计费及抵扣实例。

预留实例券变更可用区

功能说明

本文介绍如何修改预留实例券属性。您可以修改预留实例券的名称。如果业务区域发生变化，还可以修改预留实例券的可用区属性。为便于描述，待修改的预留实例券称为原券，修改后得到的预留实例券称为目标券。

前提条件

- 原券处于生效中状态。
- 原券不存在其他变更可用区请求。

操作步骤

1. 登录 云服务器控制台，左侧栏选择**预留实例券**标签页，进入**预留实例券**列表页。
2. 在顶部菜单栏左上角处，选择**目标地域**。
3. 在预留实例券列表中找到要变更可用区的实例券，在其**操作**中，单击**变更可用区**。

4. 进入**订单确认页**，在目标可用区列为每个目标券选择变更后的可用区，可通过批量变更为所有目目标券统一选择变更后的可用区。确认变更后费用是否符合预期。

5. 点击**下一步**，在订单确认页勾选**同意《百度智能云线上订购协议》**，并点击**确认订单**

最终结果

在**预留实例券**列表页可看到已变更可用区成功的预留实例券，并可抵扣当前可用区相同实例规格的按需付费-后付费BCC实例账单。

资源预留

1. 购买预留实例券后，百度智能云将保证您账号下在对应的可用区中至少能有与RI相同的实例规格*台数的按需付费-后付费BCC资源。
2. 购买预留实例券后，在购买按需付费-后付费BCC时若加入了**部署集**，资源预留将有一定风险。如果您对资源保障有强要求，请谨慎使用部署集。

预留实例券变更EHC集群

概述

本文介绍如何变更预留实例券的EHC集群。

注意事项

- 预留实例券仅支持抵扣同EHC集群的、同规格的、后付费实例，调整后则不再抵扣原EHC集群的实例。
- 如您发现目标EHC集群因资源不足置灰，请提交工单处理。

操作步骤

1. 登录 云管理控制台，选择“产品服务 > 计算 > 云服务器 BCC”。
2. 选择云服务器所在的区域，侧边栏选择弹性高性能计算集群。
3. 在EHC集群列表页中，选择需要变更的实例券，点击变更集群操作，并选择目标的EHC集群

运维与监控

监控与报警

BCC实例监控信息

监控项	描述及单位
CPU使用率	展示实例的CPU使用量与总量的占比，单位：%
内存已用量	展示实例已使用的内存量，单位：GB
用户实际内存使用量	展示实例用户实际内存使用量，单位：GB 用户实际内存使用量小于内存已用量，因为实例运行时系统会占用部分内存
内存可用量	展示实例剩余可用内存量，单位：GB
内存使用率	展示实例的内存使用量与总量的占比，单位：%
网络流量	展示实例的主网卡和主IP流入流出流量，单位：MB
网络带宽	展示实例主网卡和主IP流入流出带宽，单位：bps
网络包速率	展示实例主网卡和主IP流入流出速率，单位：pps
磁盘空间使用量	展示实实例的磁盘已使用量，单位：GB
磁盘空间使用率	展示实例的磁盘的使用量与总量的占比，单位：%
磁盘读写量	展示实例的磁盘（系统盘/数据盘）的每秒读写的字节数，单位：MB
磁盘读写次数	展示实例的磁盘（系统盘/数据盘）的每秒读写请求次数，单位：次
磁盘错误IO	展示实例的磁盘的错误IO的次数，单位：次

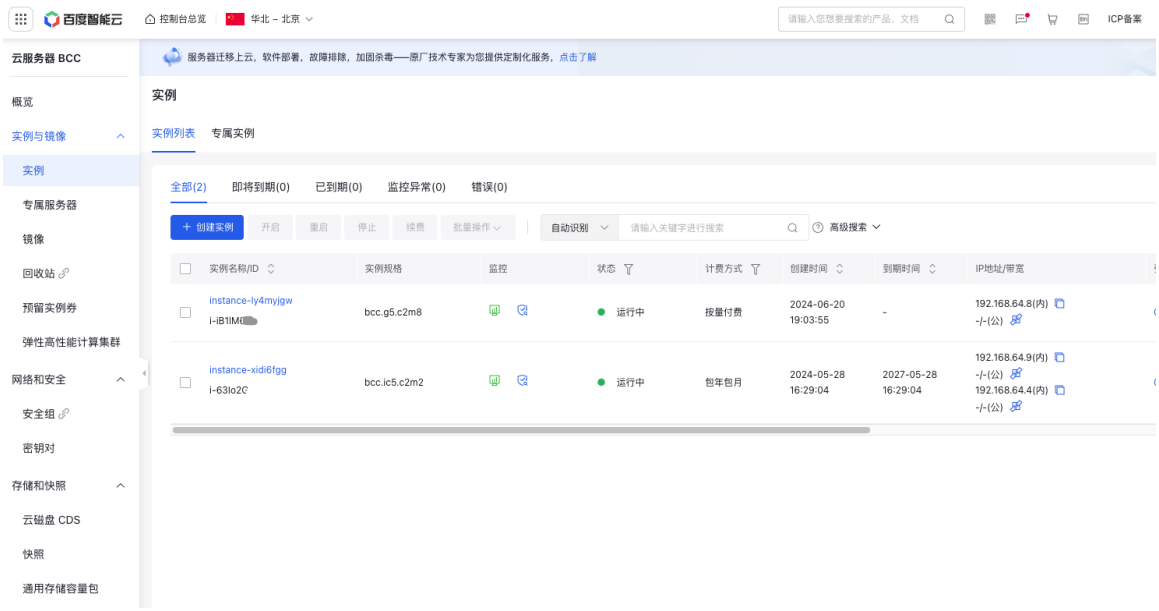
更多监控信息

BCC实例安装BCM-agent后可查看更多监控指标项，BCC的公共镜像/GPU镜像中已默认安装BCM-agent，如您使用其他镜像或者删除了BCM-Agent，可通过重装实例或者参考BCM-Agnet安装说明手动安装。

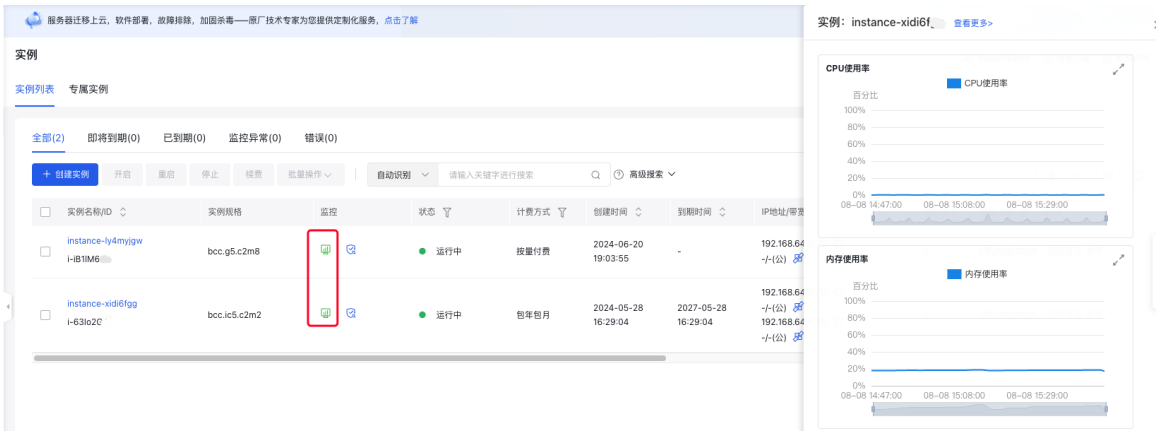
更多指标详见云产品指标列表。

展示监控

1. 选择“产品服务>云服务器 BCC”，进入“实例列表”界面。



2. 在“实例名称”列，选择某一实例链接，在“监控”页签查看相关监控数据。



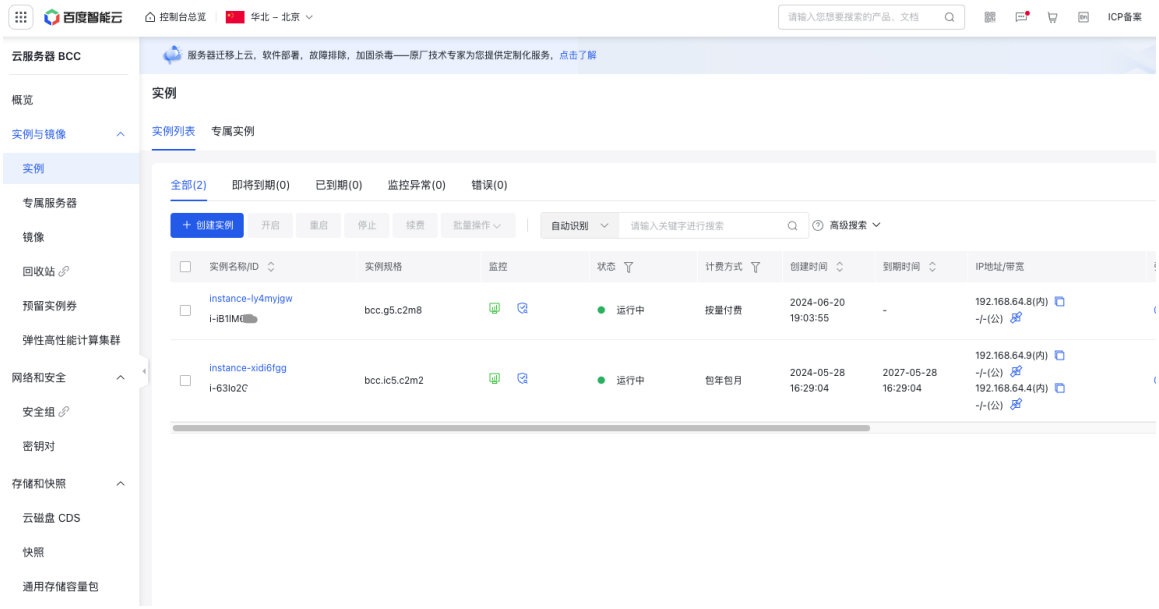
配置报警

应用场景

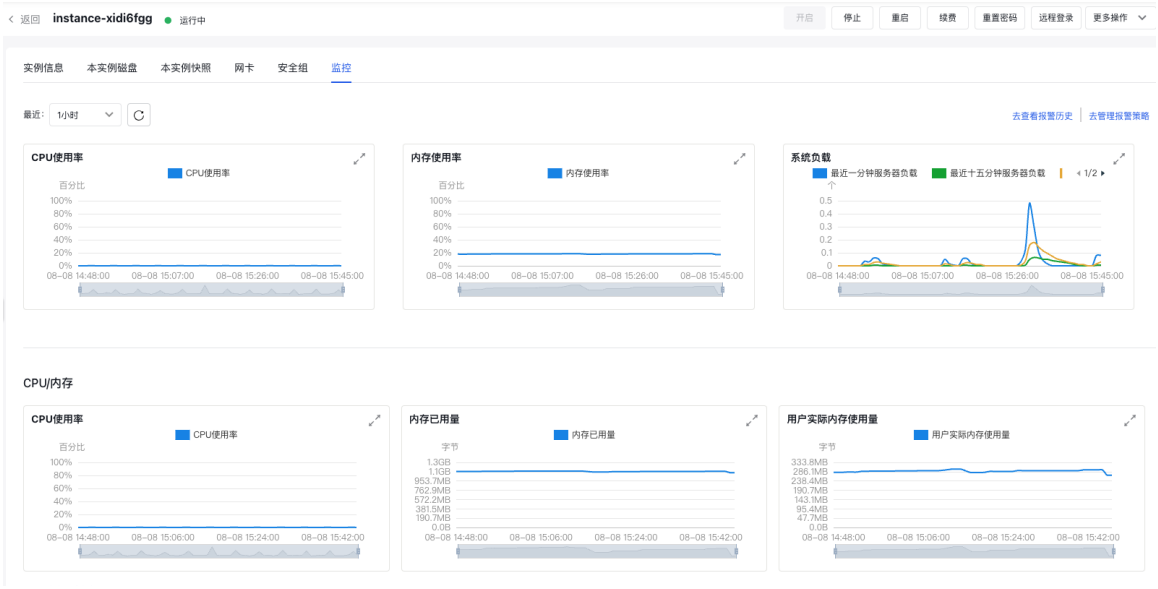
可对BCC云服务器监控的数据进行报警设定，如监控值达到设定的报警阈值，将短信或邮件通知报警接收人。

操作步骤

1. 选择“产品服务>云服务器 BCC”，进入“实例列表”界面。



2. 在“实例名称”列，选择ID链接，进入“实例详情”页面。



3. 在"监控"页面，点击“报警详情”，进入报警管理子页面。

4. 设定相应监控项的报警阈值和通知方式。

5. 点击“配置报警”，新的报警策略出现在子页面列表中。

管理报警策略

具体操作请参考管理报警。

云助手

云助手概述

概述

百度智能云提供的云助手是云服务器BCC、裸金属服务器BBC的原生运维部署工具。无需远程连接实例，云助手便能帮您自动批量执行PowerShell或者Shell等命令，完成运行自动化运维脚本、安装或者卸载软件、更新应用以及安装补丁等任务。

应用场景

- 上传并运行自动化运维脚本
- 运行实例上已有的脚本
- 批量部署代码
- 批量安装软件或应用
- 管理软件生命周期
- 安装补丁或安全更新
- 修改主机名或用户登录密码

产品优势

- 免费：云助手BCC、BBC实例提供的免费运维工具，无需额外付费就可使用
- 最佳实践：百度智能云沉淀了多种常用云助手命令，适用于一键执行命令或在此基础上改造形成自定义命令
- 权限统一：对于云助手操作均支持鉴权，具备云助手权限的用户才能使用

名词解释

使用云助手前，您需要了解一下核心概念：

名词	含义
云助手客户端（BSM-Agent）	安装在BCC实例中的轻量级插件，所有在实例中完成的命令都会通过客户端完成。Windows操作系统中任务进程名称为baiduyun.service。Linux操作系统中任务进程名称为baiduyun.service。
命令	需要在实例中执行的具体命令操作，如一份Shell脚本或者PowerShell脚本。
自定义参数	在命令中设置的变量值，以{{key}}的形式表示，可以在执行命令时以{{"key":"value"}}的形式设置自定义参数的值。由于您在一个地域下能保有的云助手命令有配额限制，建议您通过设置自定义参数提高命令的灵活性以及多场景适用性。您也可以指定内置环境参数作为自定义参数，执行命令时也可以手动对参数赋值。
执行计划	支持立即执行：在一台或者多台实例中执行某个命令，命令即时执行。

任务状态说明

针对BCC实例执行的任务，都会有任务状态展示任务的执行情况，任务状态及对应的状态如下：

状态	说明
执行中	命令正在被执行。即所选部分或全部实例级别执行状态为执行中。
执行失败	命令无法被执行，或者进程过了超时时间（Timeout）后仍然未执行完成。
执行完成	命令顺利执行完成。执行完成并不代表执行成功，您需要根据命令进程的实际输出信息（Output）及退出码（ExitCode）查看命令是否按预期执行成功。
部分失败	命令在部分实例执行失败。

云助手使用限制

限制项目	限制
支持region	华北-北京、华北-保定、华南-广州、华东-苏州、金融华中-武汉
脚本命令大小上限	18KB
单条命令自定义参数个数上限	20个
单次执行命令实例数量上限	1000台
上传文件大小上限	36KB
云助手客户端支持操作系统	Linux操作系统（包含以下版本及更高版本centos 7.1-7.9，centos 8.0-8.4，ubuntu 16.04, 18.04, 20.04，debian 9.1, 9.9, 9.13, 10.3-10.10）和Windows系统

实例列表

实例列表

为了提升选择实例和执行命令的效率，云助手提供的实例列表按照产品类型区分展示了云服务器BCC实例和裸金属服务器BBC实例，您可以快速选择实例并执行命令。

快速执行命令

您可以在云助手控制台、云服务器BCC控制台、裸金属服务器BBC控制台选中要操作的实例，并点击**快速执行命令**按钮，即可打开快速执行命令对话框。

> 实例需要为运行中的状态，并且实例的BSM-Agent需要为在线状态。

调整选择的实例

您可以在实例名称/ID再次调整已选择的实例，以便在部分执行成功的场景下针对执行失败的实例再次执行命令。

由于不同操作系统的命令不同，您同时选择Linux和Window实例时，会分为两个页签展示两类实例，可以点击对话框顶部页签切换查看对应操作系统的实例。

更多设置

在更多设置中，您可以调整执行用户、命令超时时间等参数。

选择和输入命令

您可以通过直接输入命令的方式单次执行一次命令，也可以选择**公共命令**或者**我的命令**直接执行。

> 请务必注意执行用户、超时时间与您的命令执行需求是否一致，如果命令需要执行较长时间但超时时间却保持为默认的60秒，命令会因为执行超时而显示为执行失败。

AI助手-智能生成命令

快速执行命令能力集成了基于大模型的AI助手能力，可以通过在命令内容输入框内输入“/”唤起AI助手。 您可以提问关于实例的各种命令。

> 目前云助手侧AI助手的问答次数为20次/天，次数与云服务器BCC的SmartTerm控制台的AI助手问答次数共享，已使用次数会在每日0点重置。 命令内容仅供参考，请结合实际需求酌情选用。

执行结果

命令执行后，会在右侧命令输出区域展示每台实例的执行结果，您可以点击名称切换查看具体的执行结果。还可以点击**历史记录**查看全部执行记录。

AI助手-智能分析命令结果

命令执行完成后，可一键智能分析执行结果，AI助手会根据失败信息给出处理建议，分析结果仅供参考。

我的命令

我的命令

用户自行创建的命令均会展示在**我的命令**中，仅当前用户可以查看、使用和维护这些命令。命令创建后还能再次执行命令，也可以编辑、克隆、删除和查看该命令。

操作步骤

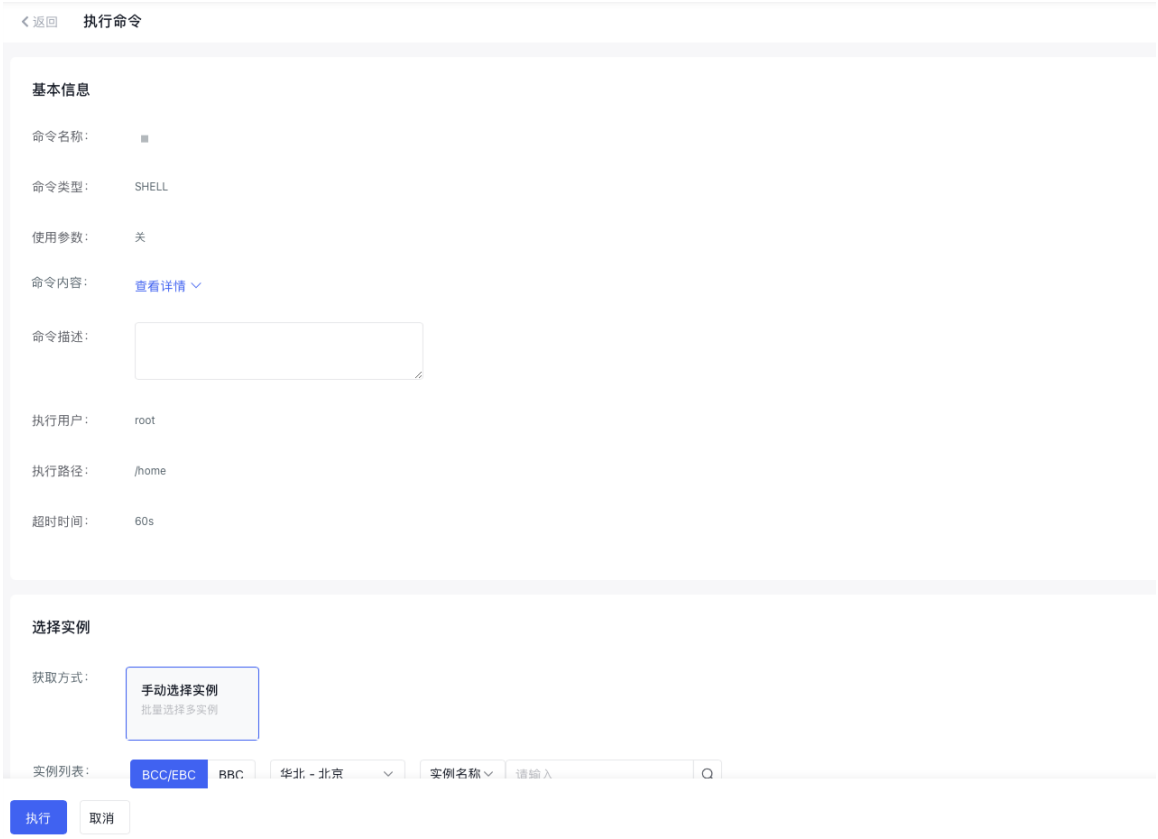
1. 登录 BCC控制台。
2. 点击左侧菜单栏 **云助手**，点击**我的命令**，即可查看到用户所有自行创建的命令。
3. 点击**命令ID**，即可在打开的页面中查看命令内容、执行用户、命令参数等详细信息。

> **说明：**
>
> - 命令与区域紧密相关，命令不能跨区域使用，创建命令前请注意当前所处的区域。
> - 可通过点击**命令类型**筛选Shell或PowerShell类型的命令。
> - 可通过点击命令内容后**复制**按钮，复制命令内容。

执行命令 需要再次执行已经创建的命令时，可以通过执行命令快速执行。

操作步骤

1. 在**我的命令**列表中，点击想要执行命令所在行的操作列中**执行**按钮。
2. 在打开的**执行命令**页面中根据需要配置调整参数，点击**执行**按钮，即可执行命令。



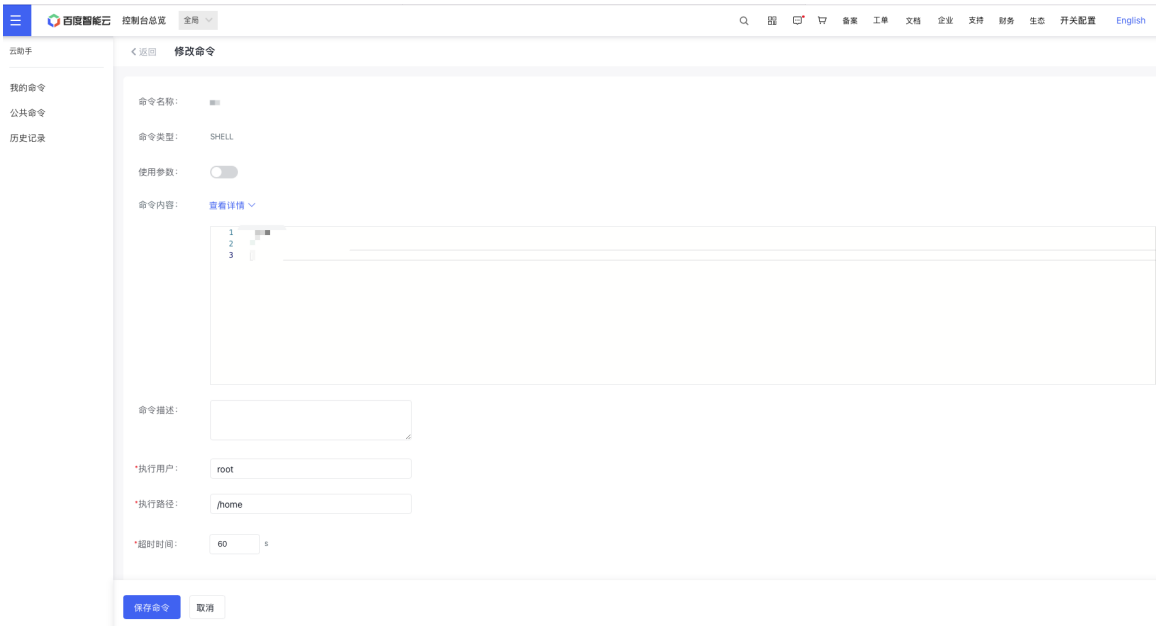
> **说明：**
>
> - 执行命令时，不允许修改命令内容，如需修改，需通过在我的命令列表中点击**修改**按钮进行操作。
> - 命令具有区域属性，区域A的命令无法在区域B中的实例执行，请先切换至实例所在的区域创建命令后再执行。

修改命令

如果需要修改命令内容和相关配置，可以通过修改命令操作。

操作步骤

1. 在**我的命令**列表中，点击想要修改命令所在行的操作列中**修改**按钮。
2. 在打开的**修改命令**页面中根据需要配置调整参数，点击**保存**按钮，即可修改命令。



> **说明：**
>
> - 您可以修改名称与类型以外的内容。

执行历史

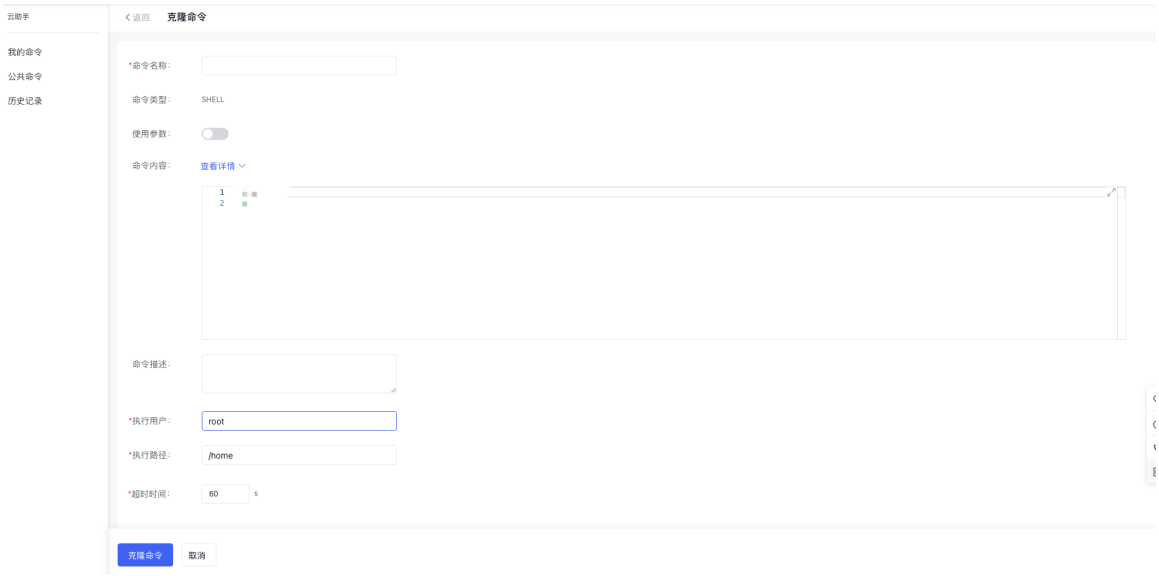
如果需要查看命令执行历史，可以点击命令所在行的**执行历史**按钮，将直接进入历史记录页面，进一步内容请参见**历史记录**页面。

克隆命令

克隆命令可快速创建一份与现有命令完全一致的副本，用户可以完全保留被克隆的命令的所有参数和配置，也可以根据需求灵活修改修改命令的信息。

操作步骤

1. 在**我的命令**列表中，点击想要克隆命令所在行的操作列中**克隆**按钮。
2. 在打开的**克隆命令**页面中根据需要调整命令的信息，点击**克隆**按钮，即可克隆命令。



删除命令

如果不再需要某一条命令，可以将其删除。

操作步骤

1. 在**我的命令**列表中，点击想要删除命令所在行的操作列中**删除**按钮。
2. 在打开的**删除命令**弹窗中确认删除，即可删除命令。

> **说明：**
>
> - 命令删除后无法恢复，请谨慎操作。

创建命令

创建命令

用户可以创建重置实例密码、安装/维护/卸载实例中的应用程序和软件、安装补丁等多类命令。 根据实例操作系统的不同，用户可以创建Linux Shell脚本和Windows PowerShell脚本，并自行指定命令中的自定义参数及其变量值。

操作步骤

1. 登录 BCC控制台。
2. 点击左侧导航栏 **云助手**，在云助手列表页面的**我的命令**菜单中点击**创建命令**。

3. 在弹出的对话框中，配置以下信息。

配置项	必选/可选	配置说明
命令名称	必选	符合大小写字母、数字以及-_ /.特殊字符，必须以字母开头，长度1-65。
输入命令	必选	选择创建命令的方式：输入命令内容：手动输入命令或把复制的命令内容粘贴入命令输入框；选择我的命令：从已创建的命令中选择命令。
命令类型	必选	根据要操作的BCC实例的操作系统类型选择对应命令类型：Shell：面向Linux系统；Power Shell：面向Windows系统。
命令内容	必选	输入或者粘贴已复制到剪贴板的命令。 可以选择开启**使用参数**，开启后，当输入新参数时，同步生成新参数，界面将增加对应的命令参数值输入框和描述输入框。
命令描述	可选	填入命令的描述信息，合理的描述信息有助于后续快速辨识此命令的用途。
执行用户	必选	指定在BCC实例中执行命令的用户名称。Linux系统BCC实例的命令默认用户为root，用户可根据实际情况自定义Linux操作系统的执行用户，但无法自定义Windows操作系统的执行用户。
执行路径	必选	指定命令的执行路径。Linux系统BCC实例的默认命令执行路径为/home，Windows系统BCC实例的默认命令执行路径为C:\Windows\system32 ，用户可以根据实际情况自定义执行路径。
超时时间	必选	设置命令在实例中执行的超时时间，超过设定的时长后，云助手将强制终止命令进程。 单位为秒，默认为60秒，最小时间为10秒，最长时间为86400秒（即24小时）
获取方式	必选	支持手动选择实例的方式，可批量选择多实例，一次最多支持100台实例。
实例列表	必选	建议您选择目前正常运行且安装了云助手客户端的BCC实例，如果实例的BSM-Agent状态为离线，命令很可能会执行失败，请务必保证BSM-Agent在线，管理BSM-Agent可参考安装BSM-Agent

4. 点击 **执行**，即可完成创建并立刻执行一次当前命令。
5. 如果您不需要现在立刻执行命令，可以选择**保存**命令，后续再手动执行。

> 注意:直接执行不会保存当前配置的命令，如果后续您需要继续使用这一命令，建议您通过**执行并保存**的方式执行命令。

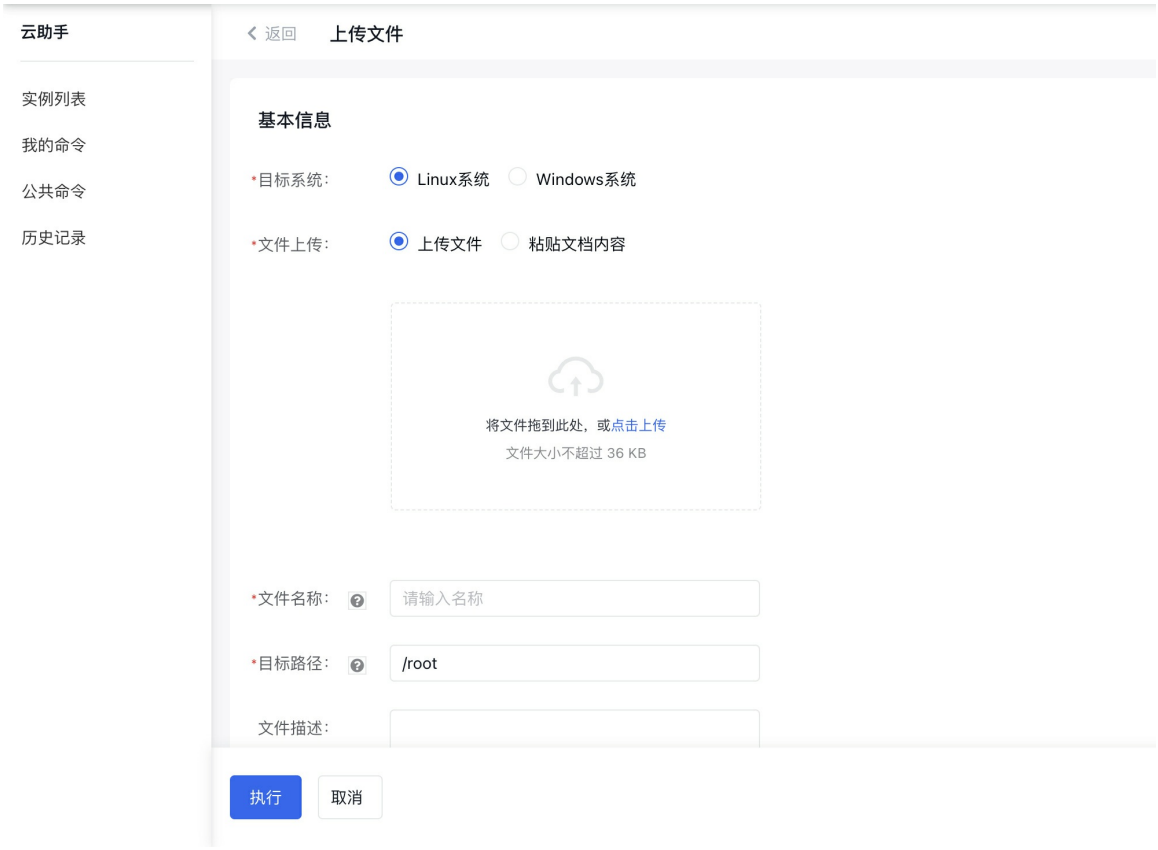
上传文件

上传文件

用户可以通过云助手把本地文件上传至对应的BCC或者BBC实例。

操作步骤

1. 登录 BCC控制台。
2. 点击左侧菜单栏 **云助手**，在云助手**我的命令**列表页面点击**上传文件**。
3. 在弹出的对话框中，配置以下信息。



配置项	必选/可选	配置说明
目标系统	必选	支持Linux系统和Windows系统。
文件上传	必选	选择上传文件的方式：上传文件：把文件拖拽到文件上传区域或点击上传按钮并选择文件即可实现上传；粘贴文档内容：手动输入或把复制的文档内容粘贴入文档输入框。
文件名称	必选	上传文件后将自动解析文件名作为文件名称，如未选择覆盖文件，则文件名称不能相同。
目标路径	必选	默认采用绝对路径，默认路径：Linux系统:/root、Windows系统:C:\Users\Administrator\Documents。
文件描述	可选	填入命令的描述信息，合理的描述信息有助于后续快速辨识此命令的用途。
用户	可选	指定在BCC实例中执行命令的用户名称。Linux系统BCC实例的命令默认用户为root，您可根据实际情况自定义Linux系列操作系统的执行用户，Windows系统的用户名无法修改。
用户组	可选	联动目标系统，仅Linux系统需要设置此参数，默认为root。
文件权限	可选	仅Linux系统可以配置此项。默认填入的0644代表文件拥有者有读取、写入的权限，同组用户和其他用户具有读取的权限。
是否覆盖	可选	设置是否覆盖目标路径下的同名文件。
超时时间	可选	设置命令在实例中执行的超时时间，超过设定的时长后，云助手将强制终止命令进程。 单位为秒，默认为60秒，最小时间为10秒，最长时间为86400秒（即24小时）
获取方式	可选	支持手动选择实例的方式，可批量选择多实例。
实例列表	可选	建议您选择目前正常运行且安装了云助手客户端的BCC实例，如果BSM-Agent状态为离线，可能导致文件上传失败。

> **说明：**
>
> - 上传文件的大小限制为36KB

4. 点击 **执行**，即可完成创建并立刻上传文件。

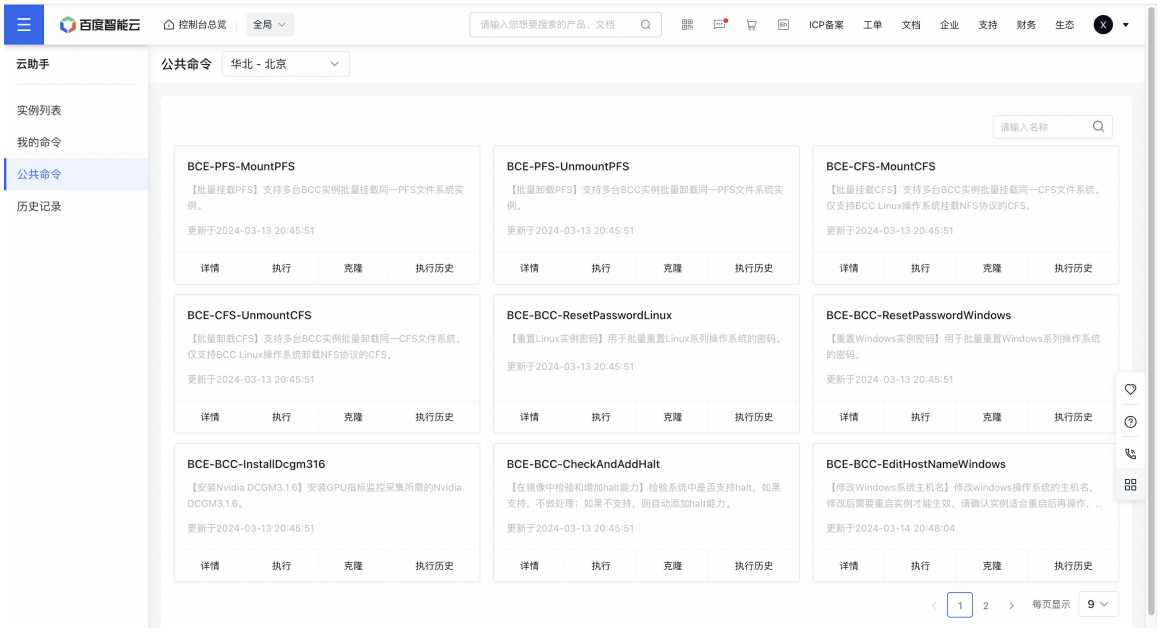
公共命令

公共命令

为帮助用户快速使用云助手，提升运维工作效率，我们整理提供了一些常用命令作为公共命令，所有用户都可以查看和使用公共命令，但无法删除公共命令，公共命令是由百度智能云直接维护。

操作步骤

1. 登录 BCC控制台。
2. 选择页面顶部 **地域**，点击左侧菜单栏 **云助手**，点击**公共命令**，即可查看到全部公共命令。



> 说明：
>
> - 公共命令持续发布中。

查看命令详情

用户可以点击命令卡片中的**详情**按钮，在弹出的命令详情页面查看到命令的详细信息。

执行命令

用户可以点击命令卡片中的**执行**按钮，在弹出的**执行命令**页面进行执行相关的配置，点击**执行**即可执行。具体使用方法可参见我的命令-执行章节。

克隆命令

用户可以点击命令卡片中的**克隆**按钮，在弹出的**克隆命令**页面进行查看和修改命令的信息，点击**克隆**按钮，即可完成克隆。具体使用方法可参见我的命令-克隆章节。

执行历史

用户可以点击命令卡片中的**执行历史**按钮，将跳转到**执行记录**页面，即可查看命令历史的执行情况。具体使用方法可参见历史记录章节。

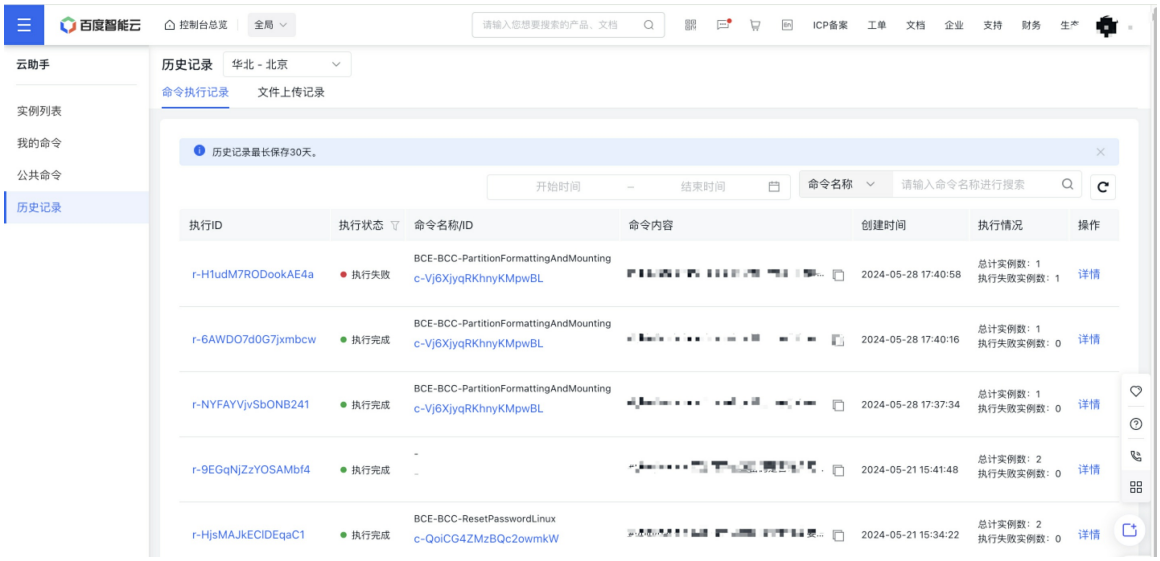
历史记录

历史记录

历史记录展示了所有创建命令与上传文件的记录，用户可以在命令执行记录页面中查看该命令的历史执行情况，可以在上传文件记录页面查看到文件上传的记录。历史记录最长可保存30天。

操作步骤

1. 登录 BCC控制台。
2. 点击左侧菜单栏 **云助手**，点击**历史记录**后，切换命令执行记录或者文件上传记录，从而查看对应的历史记录信息。



3. 单击要查看的命令所在行的**命令ID**，即可在打开的页面中查看到命令详情。
4. 单击执行ID或者操作列中的**详情**，即可在打开的页面中查看执行记录的详细信息，获取任务的执行状态和结果。如果命令执行时间较久，您可以在执行记录详情中开启自动刷新。

常见问题

为什么提示没有云助手权限，应怎么处理？ 为提升安全性，云助手服务升级了权限控制。用户必须开通云助手具备云助手使用权限并且具备BCC的只读及以上权限时，才能为云服务器执行命令。请在云助手控制台开通服务，并确认已经具备了上述权限。如果依旧无法使用，请通过工单联系技术支持。

为什么云助手执行命令时，查看到实例列表有些实例的BSM-Agent状态为离线？ 百度智能云提供的公共镜像安装了BSM-Agent，如果您使用了其他未安装BSM-Agent的自定义镜像、市场镜像等创建了实例，需要自行安装BSM-Agent，安装的方式请前往百度智能云文档中心的运维编排OOS中查看安装BSM-Agent的介绍。

云助手支持哪些类型的实例？ 目前支持BCC（包含EBC）实例和BBC实例，更多实例类型持续支持中。

云助手快速执行命令和创建命令使用场景有什么区别吗？ 云助手快速执行命令是为了便于以BCC或BBC实例为入口多次执行命令，执行结果与命令输入框在一个页面展示，便于快速使用，但支持配置的参数比创建命令略少，且命令不会被保存。如果您需要更丰富的命令参数配置并需要保存命令以备后续多次执行，可以前往云助手控制台的我的命令创建。

云助手执行命令时可以取消吗？ 无论是快速执行命令还是执行命令，目前都不允许取消执行。快速执行命令时，即使您关闭了执行窗口，命令仍会在后台执行，执行结果可以在云助手控制台的历史记录中查看。

多用户访问控制

介绍

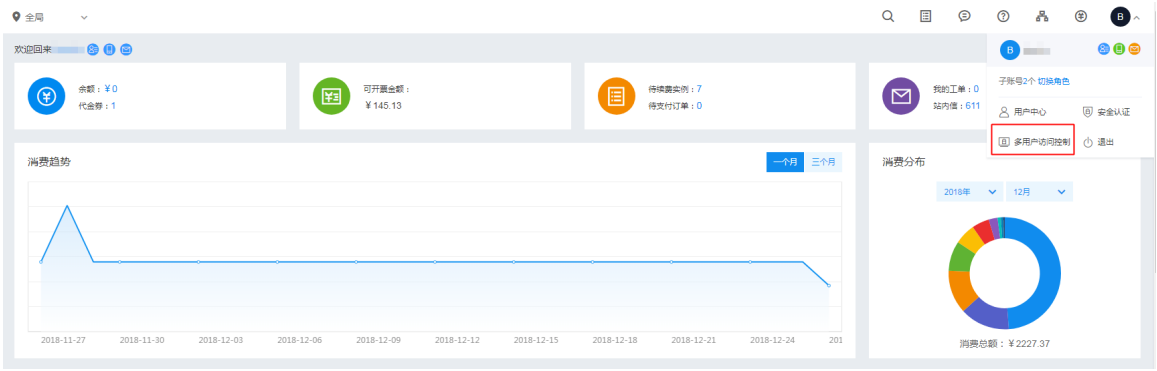
多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限，适用于企业内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。

适用于下列使用场景：

- 中大型企业客户：对公司内多个员工授权管理；
- 偏技术型vendor或SAAS的平台商：对代理客户进行资源和权限管理；
- 中小开发者或小企业：添加项目成员或协作者，进行资源管理。

创建用户

1. 主账号用户登录后在控制台选择“多用户访问控制”进入用户管理页面。



2. 在左侧导航栏点击“用户管理”，在“子用户管理列表”页，点击“新建用户”。

3. 在弹出的“新建用户”对话框中，完成填写“用户名”和确认，返回“子用户管理列表”区可以查看到刚刚创建的子用户。

配置策略

云助手支持系统策略，实现云助手产品级管理权限控制。

- 系统策略：百度智能云系统为管理资源而预定义的权限集，这类策略可直接为子用户授权，用户只能使用而不能修改。

系统策略

系统策略包含管理权限、只读权限2种策略，权限范围详细如下：

策略名称	权限说明	权限范围
CAFullControlAccessPolicy	完全控制管理云助手的权限	创建执行命令、上传文件、查看命令列表、查看命令详情、修改命令、删除命令、查看历史记录列表、查看历史记录详情
CAReadAccessPolicy	只读访问云助手的权限	查看命令列表、查看命令详情、查看历史记录列表、查看历史记录详情

用户授权

在“用户管理->子用户管理列表页”的对应子用户的“操作”列选择“添加权限”，并为用户选择系统权限进行授权。

> **说明：**如果在不修改已有策略规则的情况下修改某子用户的权限，只能通过删除已有的策略并添加新的策略来实现，不能取消勾选已经添加过的策略权限。
>
> 用户还需要同时具备云助手的管理权限和云服务器资源的查看权限，才能针对服务器执行命令和上传文件。

子用户登录

主账号完成对子用户的授权后，可以将链接发送给子用户；子用户可以通过IAM用户登录链接登录主账号的管理控制台，根据被授权的策略对主账户资源进行操作和查看。

其他详细操作参考：多用户访问控制。

云助手使用须知

云助手服务近期完善了安全性设计，需要用您手动开通才能继续使用云助手服务，支持主子用户鉴权，子用户应具备云助手和云服务器的必要权限才能成功执行命令。此次调整不会清除您存量的自定义云助手命令和执行数据。过于宽松的权限可能带来安全风险，为保障云服务器安全，请您务必在开通云助手后，严格控制子用户的权限。

云助手具备免登录执行命令和上传、下载文件的能力，开通和使用云助手代表您认可本服务的下述行为：

1. 免登录下发命令：为了提升实力运维效率，您可以通过云助手免输入用户名和密码执行命令，实际执行的命令内容由您按需自行填写，请勿随意执行未知来源的命令。
2. 上传文件或从指定路径下载文件到操作系统：您可以从本地选择文件或从BOS对象存储或其他我们支持的途径把文件传输到云服务器BCC实例和弹性裸金属服务器BBC的操作系统中。

自助诊断工具

自助诊断工具概述

概述

百度智能云提供的自助诊断工具是面向云服务器的问题诊断服务，支持对实例的系统状态、硬件健康状态、应用程序等进行全面诊断。自助诊断工具是您云服务器的专属医生，帮助您了解实例的健康状态，并及时解决常见的问题。

应用场景

- 实例运行的应用程序出现中断。
- 实例运行的应用程序性能不符合预期。
- 实例操作系统卡顿。
- 无法远程连接实例。
- 定期的实例健康检测。

产品优势

- 易用性：不需要具备专业运维知识或背景，您可通过控制台一键对实例进行问题排查，排查后系统将自动提供实例遇到的问题及解决方案。
- 提高运维效率：依赖人工的实例故障排查通常需要天级别，通过自助诊断工具可实现分钟级问题定位。
- 零成本：自助诊断工具是购买云服务器提供的增值服务，无需支付额外费用。

名词解释

使用云助手前，您需要了解一下核心概念：

名词	含义
诊断	通过自助诊断工具在实例中执行的诊断任务，针对某一场景问题现象完成系统性的问题根因定位，并提供解决问题的解决方案。
诊断项	在一次诊断中所执行的最小任务粒度，例如在诊断GPU硬件过程中执行的GPU掉卡诊断即为1个诊断项。
诊断项分类	对多个诊断项进行分类，方便统计。

诊断结果说明 每个诊断项都有诊断结果，结果定义如下：

- 正常：没有发现异常。
- 中断 ：此诊断项没有被完整执行完成，例如在诊断项执行时实例关机等原因导致。
- 低风险：诊断项中识别到了一些指标异常，通常这些异常指标对实例没有使用影响。
- 中风险：诊断项中识别到了一些指标异常，通常这些异常会导致实例的性能受到损失。
- 高风险：通常这些异常会导致实例部分硬件资源不可用或性能严重下降。

诊断状态说明 执行一次诊断通常需要1-2分钟时长，根据您选择的诊断场景可能会有所浮动，以下为诊断状态详情：

状态	说明
诊断中	诊断正在运行中。
诊断完成	已完成该次诊断所包含的的所有诊断项
诊断失败	未完成该次诊断所包含的所有诊断项，例如执行过程中操作关机，未完成的诊断项会标识为中断

自助诊断工具使用依赖

使用自助诊断工具依赖您的实例已安装云助手，您可参考云助手文档了解并安装云助手。

应用性能诊断工具Btune

Btune概述

Btune是一款百度智能云自研的性能分析和调优工具，沉淀百度集团内部多年的应用优化经验，从底层到上层覆盖CPU微架构、内核、runtime和基础库、业务应用等，对业务进行全栈性能分析和优化，快速定位并解决性能瓶颈，实现业务性能最优。

Btune可支持以下如下功能：

1. 指标采集和分析展示：采集整机配置、整机资源、进程热点、进程资源等多维度指标，进行分析展示，初步确定异常点
2. 性能瓶颈定位和优化建议：Btune提供的诊断报告中包含两部分：分析摘要和详细报告。其中，分析摘要展示了业务性能瓶颈点和相应的优化建议，可以满足绝大部分的场景的需求。详细报告提供了更详细的性能分析数据，从系统配置、系统性能、进程线程模型、函数指令热点等多个维度呈现负载的资源分布、耗时分布、线程关系等运行特性，满足用户做更细粒度度性能优化。

注意事项

- Btune检测的实例必须处于 **运行中** 状态。
- 待检测BCC实例需要安装Btune-agent

Btune-agent安装方式 使用Btune自助诊断需要您在待检测实例中安装Btune-agent，请参考如下步骤安装或卸载Btune-agent。

- 安装Btune-agent：

登陆待检测BCC实例，下载安装脚本并执行：

```
 **下载安装脚本**
wget http://download.bcm.baidubce.com/btune-agent/install.sh
**执行安装**
bash install.sh install
```

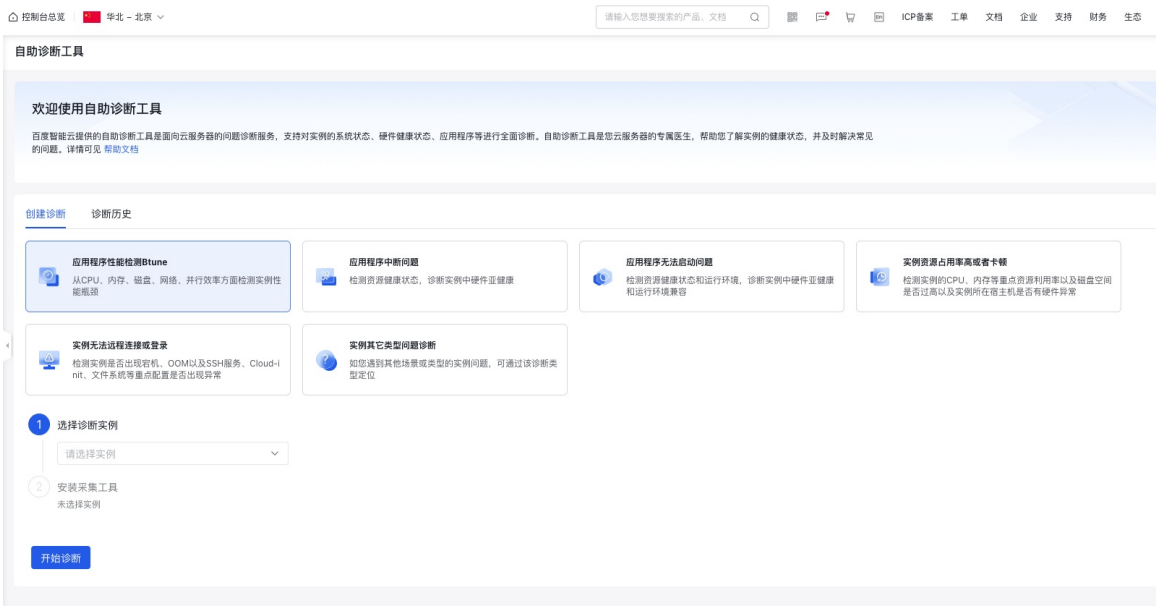
- 卸载Btune-agent：

登陆BCC实例，执行如下命令：

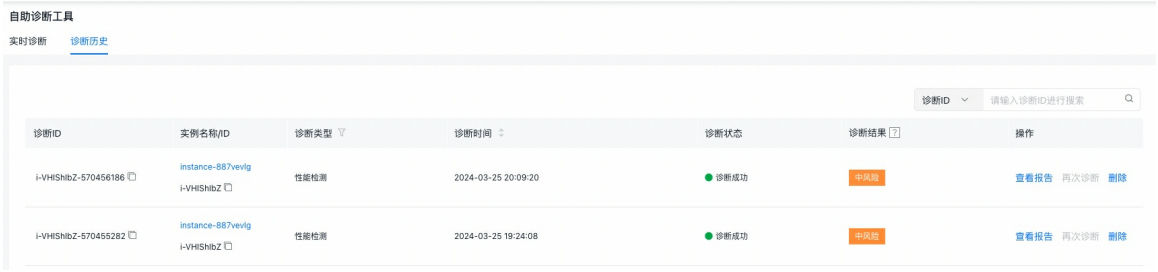
```
 **停止Btune agent**
bash -x /home/work/btune/btune-agent/control stop
**清除目录文件**
rm -rf /home/work/btune
```

创建Btune诊断 1、远程连接BCC实例，并运行需要诊断的应用程序，此处以后台运行test应用程序为例。

2、登录BCC云服务器控制台，在左侧侧边栏选择自助诊断工具，并选择性能检测，按照向导选择实例以及需要诊断的应用程序，可根据进程的PID或者名称执行搜索，选择采集时间后即可创建诊断。



查看Btune诊断 1、在自助诊断工具诊断历史中可查看已创建的诊断。

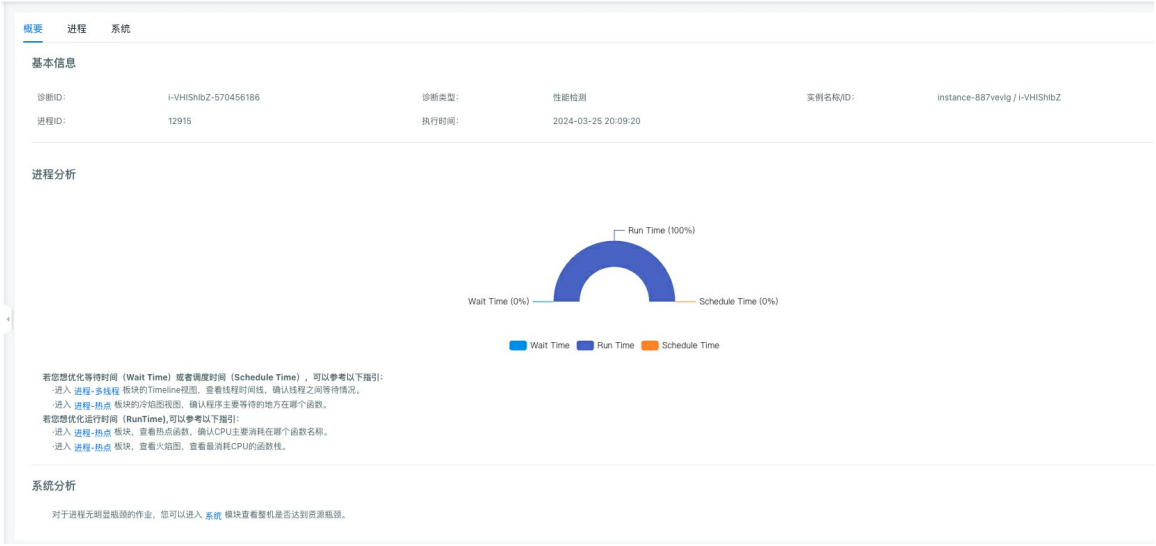


2、选择查看报告即可查看诊断结果及Btune分析摘要。

3、点击页尾的查看详细报告可查看更详细的分析报告，您可以在此报告中了解以下内容辅助实现性能优化方案。

- 系统配置分析：包含机器硬件、系统、runtime配置参数分析。
- 系统性能分析：系统CPU/MEM/存储/网络/调度各个维度USE分析。
- 进程线程模型分析：USE、CPU时间线拆分、进程线程并发效率、TMA维度分析。

- 函数指令热点分析：函数热点、指令热点、函数时间线、函数TMA维度分析。



维修平台

维修平台概述

实例在实际使用过程中会有极小概率出现硬件级别故障。百度智能云推荐对这些问题进行完整的维修处理，从而充分保证实例的可用性和业务的稳定运行。为给用户提供一套标准化的云服务器故障处理、授权维修方案，百度智能云提供用户维修平台功能。

维修平台是一个能够提供相关能力支持的工具，通过此平台，用户能够直接获取到涵盖故障感知、维修任务处理、维修进度确认等信息，并可以进行维修任务交互（如授权、验收）等一系列操作的能力。

功能特点 免费开通 维修平台功能已面向所有用户开放，无需额外购买即可免费使用。您可在“云服务器控制台“左侧导航栏中，找到并进入“维修平台”中，查看实例涉及的所有维修任务，并进行授权维修等操作。 **故障全面覆盖** 维修平台通过HAS组件，对硬件资源进行故障、功耗、资源、性能等全方面感知，包含70+的事件类型，对cpu、内存、硬盘、网卡等部件进行全方位的监控并在故障时自动开启维修流程。 **策略灵活设置** 维修平台提供灵活的维修策略，可对故障实例授权维修，也可选择暂不处理，待数据备份完成后，再行授权。维修平台也支持预授权功能，可根据实例tag灵活设置不同的预授权策略。 **功能优势 硬件自动维修** 维修平台会自动检测实例故障，并在发生故障时，立即发起维修流程。维修流程自动化，减少线下交互造成的延迟，提升运维效率。 **极简交互** 用户可在控制台查看故障实例的详细信息，并对故障实例进行维修授权或维修后确认等操作。也可以通过API进行相关操作，线上交互，方便快捷。 **全流程监控** 用户可根据实例的状态或维修任务的相关状态，实时监控故障维修进展。可在维修记录里回溯故障实例的授权时间、维修完成时间等，做到实例维修的全流程监控。 **产品覆盖**

维修任务平台现已完成云服务器以及弹性裸金属服务器的接入，其他产品暂不支持。

维修平台接入说明

维修平台已针对百度智能云所有用户开放，使用前请您确认已完成**HAS组件的安装或升级**，并在BCM里设置了相关的**云产品事件的报警策略**，保证故障的及时检测与通知。 **HAS组件检查** 硬件感知服务（Hardware-Aware Service，或称HAS）是一款线上硬件故障监控修复、功耗和资源管理工具，主要功能包括故障感知、功耗感知、资源感知、性能感知四大部分，提供了一系列的线上硬件的配置/状态监控，故障检测与修复，健康预警，功耗、擦除及相关硬件自动化管理功能，并提供统一的硬件资源状态查询、推送和管理接口。 **存活状态查看** 您可在实例列表页面查询HAS组件的存活状态，及时安装或更新HAS组件，保证硬件故障的及时检测与维修。

注：HAS Agent安装在宿主机上，BCC暂不涉及该组件，百度智能云会保证BCC宿主机上的HAS Agent保持在最新版本。

安装与升级

当您需要安装或升级HAS组件时，我们建议您按照下述指引重新安装，以获取到完整的、全方位的硬件感知能力，帮助您提高系统可用性，并能够顺利使用维修平台的完整能力。 **环境确认**

1. 在安装或升级之前，您需要确认实例当前的操作系统是否符合要求。当前Has-agent可适配的操作系统主要面向linux os，包括Redhat、Centos、Ubuntu、Fedora、Debian、Slackware、欧拉。
2. 检查域名has-master-a.sdns.baidu.com是否通，使用方法：ping has-master-a.sdns.baidu.com

升级组件

- 在**/tmp**目录下执行：

```
curl -sm10 http://has-master-a.sdns.baidu.com/download/qa_packages/bbc/has-agent-installer-first.sh
```

执行输出如下：有个" ERROR: BIO_new_file ..... " ，告警可忽略，不影响升级

```
[root@zitao-baidu-10001 tmp]# curl -sm10 http://has-master-a.sdns.baidu.com/download/qa_packages/bbc/has-agent-installer-first.sh | bash
[root@zitao-baidu-10001 tmp]# [/home/scmpf/compiler_src/liugao02_1420201_linux64/0/op/oped/cloudwatch/upgrader/src/main.c:274 verify_signature()] ERROR: BIO_new_file() data bio
[/home/scmpf/compiler_src/liugao02_1420201_linux64/0/op/oped/cloudwatch/upgrader/src/main.c:393 downloader_process()] verify signature failed, downloading new executable.
[/home/scmpf/compiler_src/liugao02_1420201_linux64/0/op/oped/cloudwatch/upgrader/src/main.c:427 downloader_process()] download newest executable success!
[/home/scmpf/compiler_src/liugao02_1420201_linux64/0/op/oped/cloudwatch/upgrader/src/main.c:568 dispose_zombies()] downloader process exit with status 1
[/home/scmpf/compiler_src/liugao02_1420201_linux64/0/op/oped/cloudwatch/upgrader/src/main.c:473 bear_executor()] run command /home/opt/has-agent/has-agent-installer.sh

[root@zitao-baidu-10001 tmp]#
```

结果检测

- 部署后等待10min，has会部署成功，目前是1.1.3.92版本。
- 后续has升级后会自动升级，版本号的第一位或最后一位比现在的版本高都是新版，均符合预期。

1. 检查自升级进程，单机上执行如下命令：

```
ps -ef|grep -v grep |grep "/opt/avalokita/bin/avalokita --update-url=http://has-master-a.sdns.baidu.com/download/qa"
```

出现保活、自升级进程：

/opt/avalokita/bin/avalokita --update-url=http://has-master-a.sdns.baidu.com/download/qa_packages/bbc//has-agent-installer.sh --signature-url=http://has-master-a.sdns.baidu.com/download/qa_packages/bbc//has-agent-installer.sh.sig --certificate=/home/opt/has-agent/cert.pem --update-interval=3600 --max-executable-size=1000000000 /home/opt/has-agent/has-agent-installer.sh

```
[root@h800-1-14 has]# ps -ef|grep -v grep |grep "/opt/avalokita/bin/avalokita --update-url=http://has-master-a.sdns.baidu.com/download/qa"
root      578661       1  0 18:56 ?        00:00:00 /opt/avalokita/bin/avalokita --update-url=http://has-master-a.sdns.baidu.com/download/qa_packages/bbc//has-agent-installer.sh --signature-url=http://has-master-a.sdns.baidu.com/download/qa_packages/bbc//has-agent-installer.sh.sig --certificate=/home/opt/has-agent/cert.pem --update-interval=3600 --max-executable-size=1000000000 /home/opt/has-agent/has-agent-installer.sh
[root@h800-1-14 has]#
```

2. 检查has版本和主进程：

```
curl -s 127.0.0.1:428/self/basic
```

```
[                                                    ]$ curl -s 127.0.0.1:428/self/basic
[
   {
      "1709801940" : {
         "has_version" : "1.1.3.92",
         "uptime" : "2024-01-30-17-42-31"
      }
   }
][                                                    ]$
```

ps -ef|grep has_client

```
][                                                    ]$ ps -ef|grep has_client
zhangyu+  4036  1997  0 19:52 pts/0    00:00:00 grep --color=auto has_client
root     17641     1  0 16:59 ?        00:00:04 /home/opt/has/has_client/has_agent
[                                                     ]$
```

3. 查看资源版本：

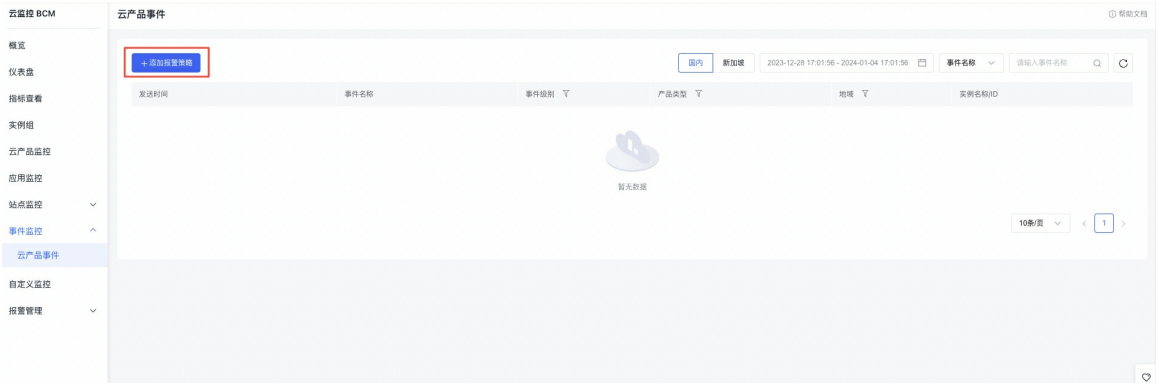
- 部署完成在/home/opt下会出现：has和has-agent。
- 查看部署后包的版本如下:

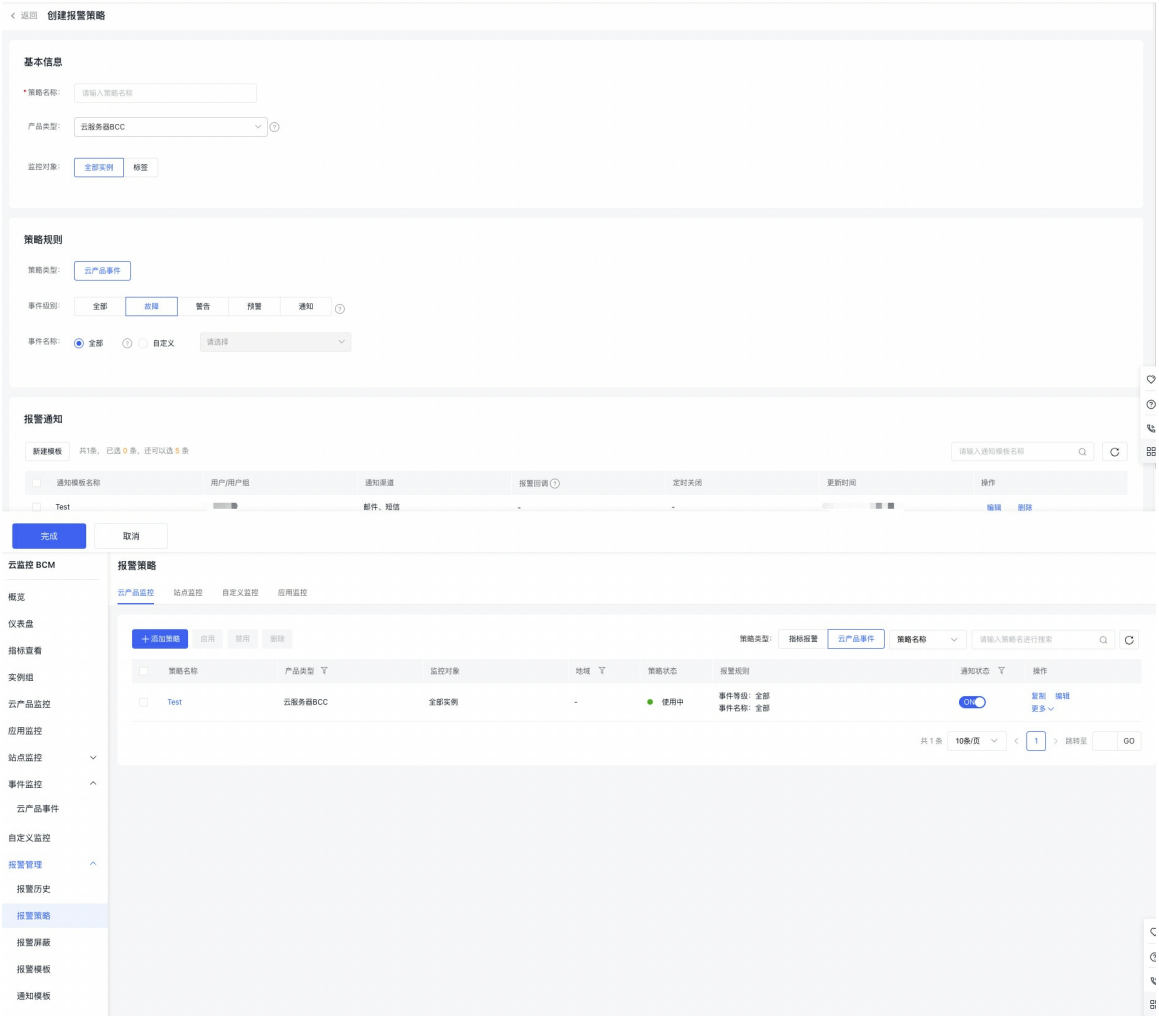
使用cat /home/opt/has/VERSION |head -1

```
][                                                    ]$ ps -ef|grep has_client
zhangyu+  4036  1997  0 19:52 pts/0    00:00:00 grep --color=auto has_client
root     17641     1  0 16:59 ?        00:00:04 /home/opt/has/has_client/has_agent
[                                                     ]$
```

报警策略设置 在维修任务生成以及维修完成时，百度智能云会通过BCM给您推送相关信息，提醒您对故障实例“授权”维修，或者确认维修完成后，实例是否恢复正常。您可在BCM中设置云服务器与弹性裸金属服务器的“云产品事件”的告警策略，以便您能够成功接收到相应实例的告警信息。

- 如无特殊要求，建议您将告警策略设置为对所有实例的所有故障事件的监控。
- 您可以在“报警策略”的“云产品事件”中，查看已设置的告警策略，可以针对实际告警需求，设置多条不同的告警策略；也可随时对策略进行修改、删除等操作。





维修任务

维修平台检测到实例故障时，会立即生成一条维修任务，并且在实例列表页面将对应实例显示为“待维修”状态。

您可以前往维修任务页面查看相关信息，包括维修任务ID、关联实例信息、故障名称等，并在维修任务界面进行授权维修等操作。在相关的维修任务生成后，百度智能云也会根据您在BCM（云监控产品）中设置的报警策略，向您推送相关信息，并告知相关解决建议。

注：为保证故障告警信息正常推送，请确保故障监控组件HAS Agent已安装并处于在线状态，且已在BCM中设置了相应的告警策略。

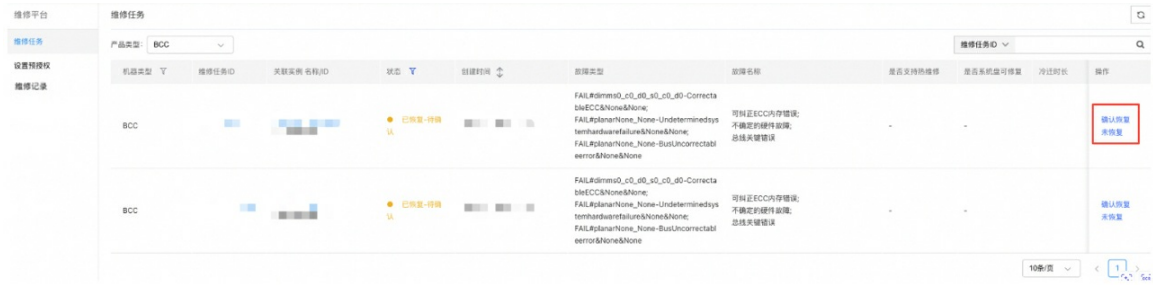
授权维修 您可以在维修平台中，筛选出“未授权”状态的维修任务，选择 “授权” 维修或 “暂不处理” 操作。

- “授权”：授权开始维修故障操作，请确保业务已离线；授权后将对实例停机维修，此时实例变为”维修中”状态，维修任务变为“处理中”状态。
- “暂不处理”： 在未授权的情况下，如果无法停机或是有冗余硬件故障，可选择”暂不处理”，后续完成数据备份或者业务迁移后，安排时间再进行”授权”维修。

维修结果确认 实例维修完成后，在操作中可选择”确认恢复”、”未恢复”操作。

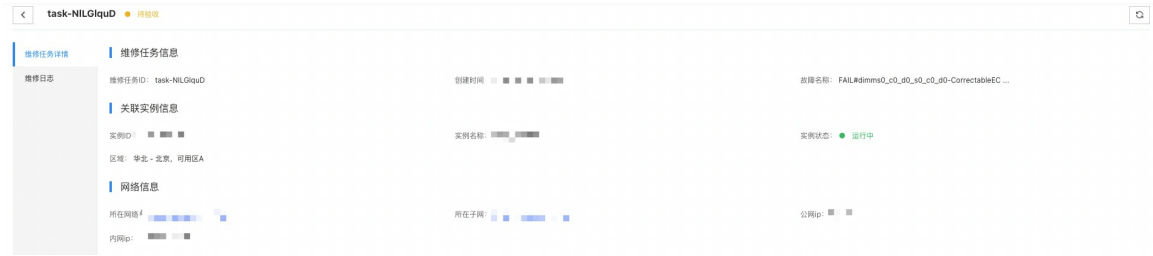
- “确认恢复”：确认机器已恢复，维修任务结束，可在”维修记录”查看历史记录。

- “未恢复”：机器未恢复，继续处理维修。



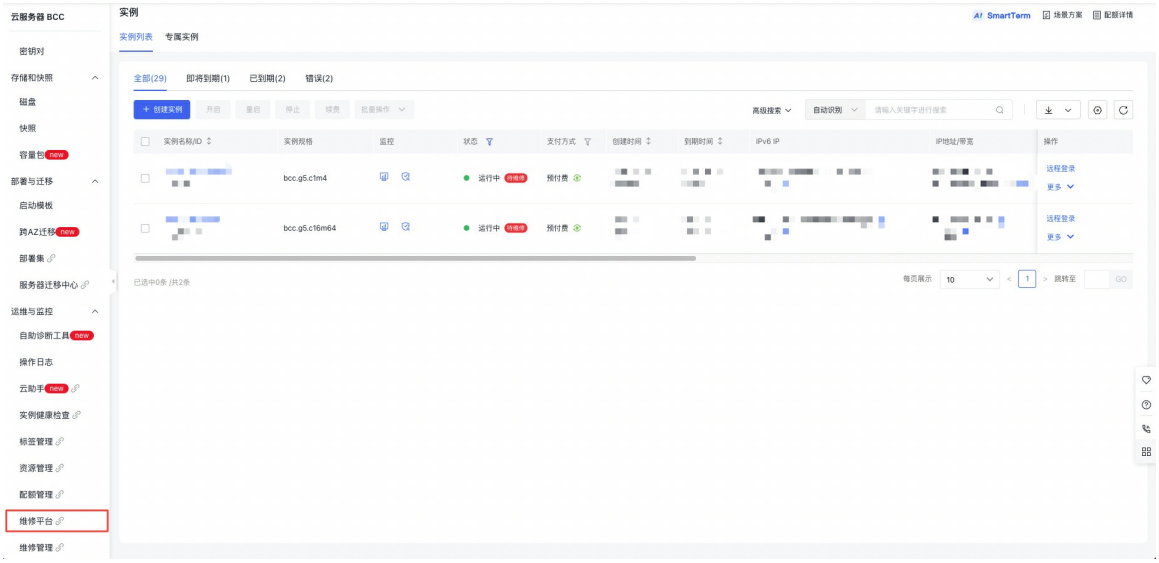
查看任务详情

- 点击维修任务ID，可查看维修任务详情信息。

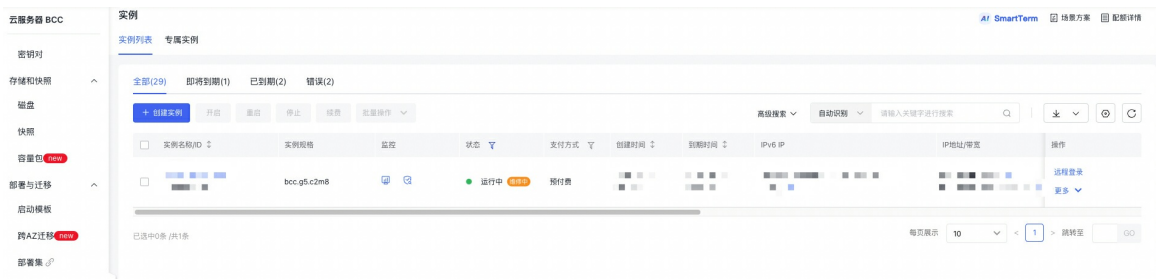


待维修实例查询

- 您可以在控制台**“实例“**界面的**”实例列表“**处，筛选出“待维修”状态的实例，并到”维修平台“中查询具体的维修信息。

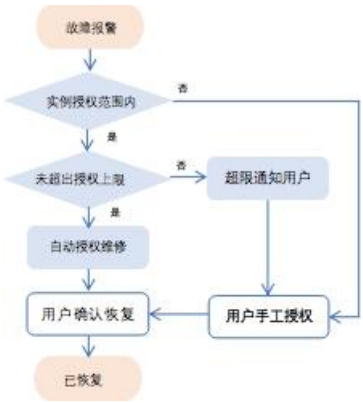


- 故障实例授权维修后，实例的维修状态将变为“维修中”，您可以在控制台**“实例“**界面的**”实例列表“**处，筛选出“维修中”的实例，并到维修平台中查询进一步的维修进展。



预授权规则设置

维修平台提供实例维修预授权功能，您可根据实例标签，设置维修的预授权规则，该规则对标签下的所有实例生效。当维修任务产生时，将按照正在启用中的预授权策略进行处理，立即开始维修，无需再单独操作授权。



新建预授权规则 进入维修平台的"设置预授权"页面，单击【新建预授权规则】，即可进入规则设置页面。

1. 授权规则必须关联标签键，标签范围内的实例才可以触发预授权。
2. 可根据实例标签设置多个不同的维修预授权规则。
3. 新增规则时勾选【创建规则后默认生效】，规则会立即生效；如果新建时没有勾选该选项，也可以在新建完成后选择【启动】操作，生效规则。
4. 授权上限可基于业务容量考虑，如果在规则中设置授权上限（默认不填的话，表示无上限）

- 如果关联标签范围内的故障实例数量未超出授权上限，则自动触发授权维修
- 如果关联标签范围内的故障实例数量超出授权上限，超出范围的机器将不会触发自动授权维修，同时会发送通知，通知用户到控制台操作手工授权，后续流程同未授权维修流程。

预授权规则启用与停止

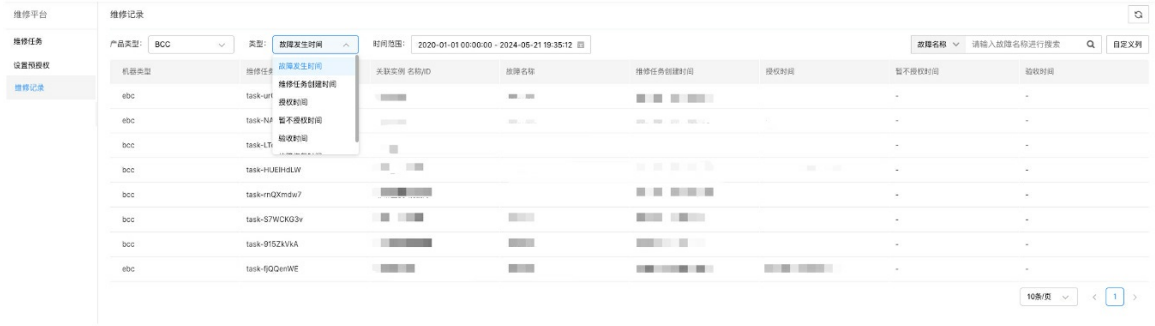
- 建立的预授权规则在启用状态下正常生效，停止状态下的预授权规处于无效状态，维修任务产生时，用户仍需进行授权，触发后续维修。

预授权规则删除 处于停止状态的预授权规则可进行删除，规则删除后，所涉及标签下的实例发生故障时需单独授权。

维修记录查询

维修平台提供维修记录功能，记录所有已完成的维修任务。

- 每条记录包含维故障发生时间、修任务创建时间、授权时间等维修全流程的各个时间点。
- 用户可根据故障名称、实例ID等搜索历史维修任务，查看发生相同故障的实例列表。
- 用户可根据故障发生时间、授权时间等类型，搜索一定时间段内的所有维修记录，回溯故障原因。



实例健康检查

实例健康检查是一款为BCC实例提供资源占用、网络连通性、EIP状态、漏洞扫描等检查的工具。将为用户提供实例相关检查并依据检查结果提供建议解决方案。

操作场景

当您遇到以下几种场景时，可以使用实例健康检查操作：

1. 服务器无法远程连接；
2. 服务器使用较为卡顿；
3. 登录服务器后无法执行任何操作；
4. Ping服务器出现丢包问题；
5. 检查服务器漏洞信息。

操作步骤

入口一：实例健康检查页

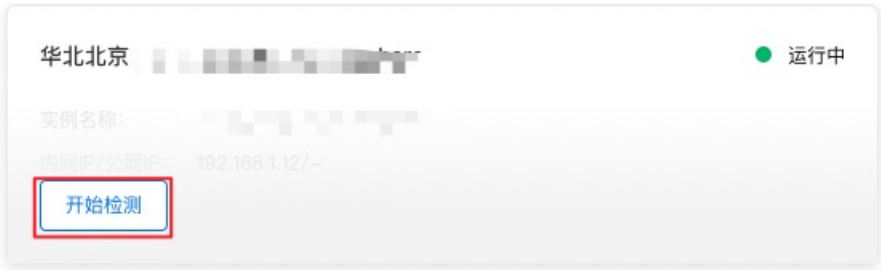
1. 进入实例健康检查页面。

- 登录云服务器BCC控制台，进入实例列表页面。在左侧的操作栏，单击“运维与监控”>“实例健康检查”。

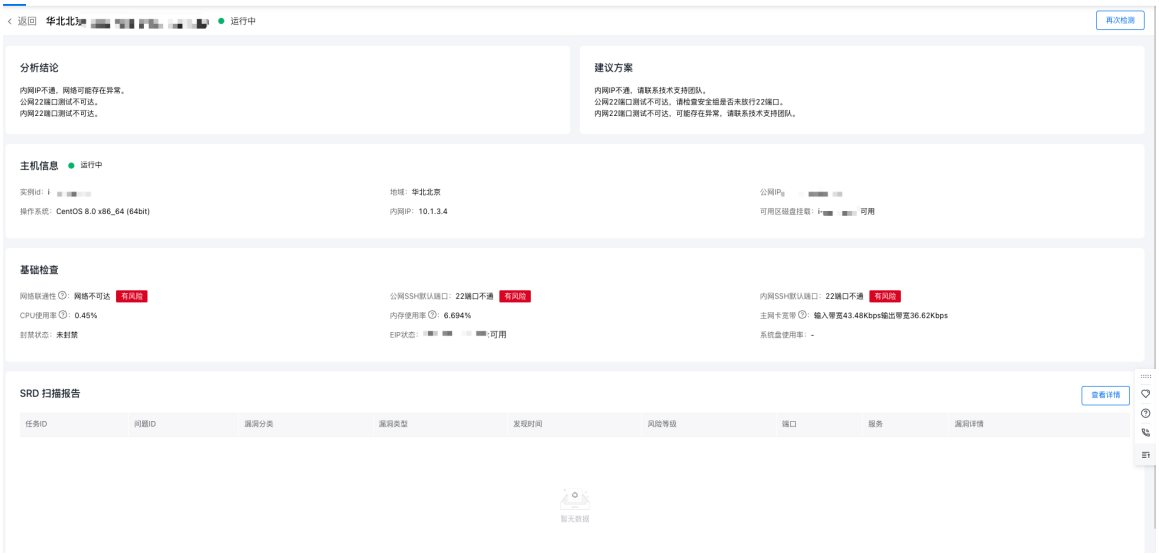
2. 选定地域，在页面上选定需要进行检测的服务器。

也可利用服务器的实例ID、实例名称、服务器IP地址等信息进行搜索，查找到对应实例。

3. 鼠标移至对应实例，点击“开始检测”，并等待检测结束。



4. 查看检测结果。



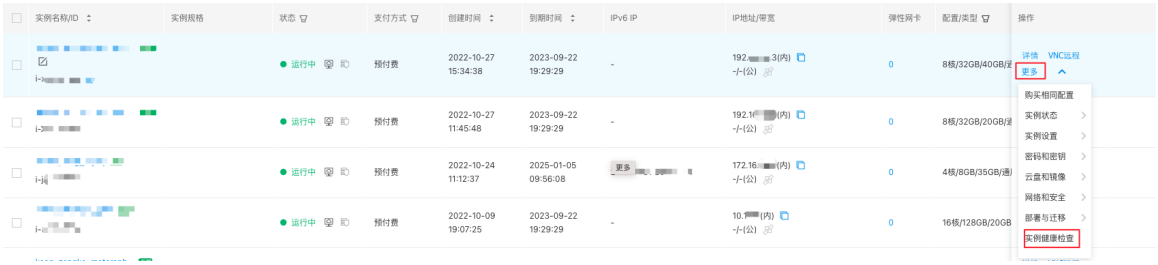
5. （可选）查看SRD扫描报告的详细信息。

6. （可选）对该实例进行“再次检测”。

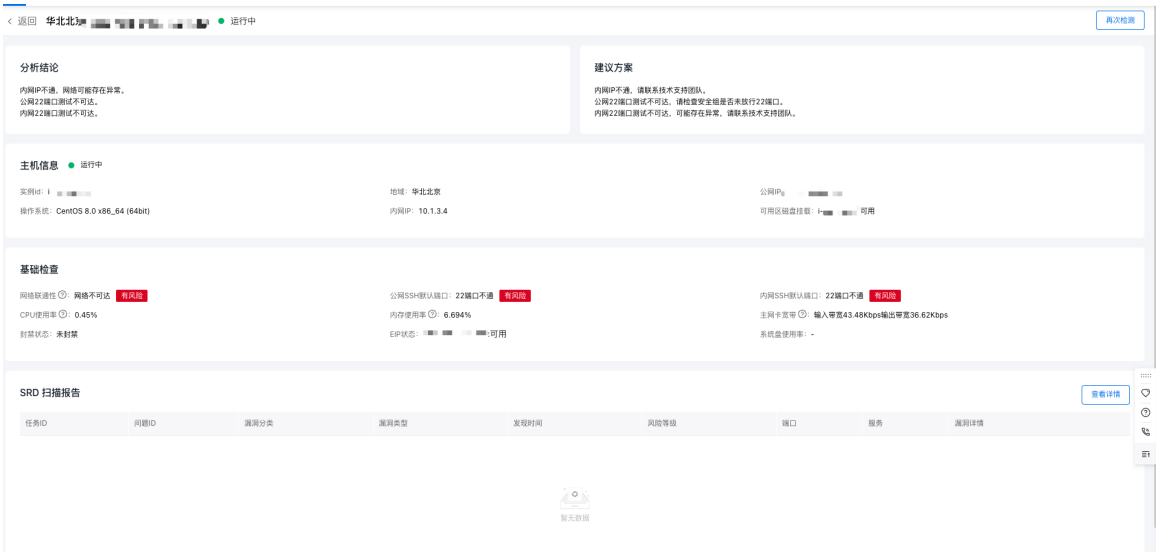
入口二：实例列表页

1. 登录云服务器BCC控制台进入实例列表页面。

2. 选择您要进行健康检查的实例，在右侧的操作栏，单击“更多>实例健康检查”。



3.查看检测结果。



4.（可选）查看SRD扫描报告的详细信息。

5.（可选）对该实例进行“再次检测”。

疏散故障实例

本文档主要说明如何对故障的实例通过疏散进行重新部署。

功能说明

疏散能力是提高服务可用性，保障您的业务稳定性的关键方式之一。

当前如果您使用裸金属形态的EBC（Elastic Baremetal Compute，弹性裸金属服务器实例）或EHC实例（Elastic High-Performance-Compute Cluster，即弹性高性能计算集群实例），在实例出现CRITICAL等级的告警事件，或是因部分意外操作导致宕机且无法快速恢复时，百度智能云提供了疏散实例的能力。通过该能力，您可以在一台无故障的宿主机上恢复您的实例，实例的关键信息皆与原实例保持一致，包括：

- 实例ID、名称、主机名等实例基本信息
- VPC、子网IP等，以及主网卡的辅助IP、弹性网卡IP
- RDMA IP（如有）等信息
- 云盘（数据盘）、弹性网卡等的挂载状态

前置条件

如果需要正常使用疏散功能，您需要保证EBC/EHC实例的 `/etc/fstab` 配置文件中，所有数据盘都已经加入nofail参数。您可以

按照如下实例，使用vim命令调整数据盘的nofail属性：

```
/dev/nvme0n1 /data1 ext4 defaults,barrier=0,nofail 0 0
```

参数	说明
/dev/nvme0n1	本地盘设备名，在实例中可以通过df -h或lsblk等命令查看。 格式根据本地盘类型可能有多种，与Linux下各设备的展示逻辑相同，如/dev/sda，或vdb，或nvme0n1，不需要体现分区。
/data1	本地盘挂载点，可通过mount命令的grep语句，或直接通过lsblk的MOUNTPOINT查询。
ext4	ext4文件系统类型，可通过blkid /dev/nvme0n1命令查询，默认逻辑下为ext4。
barrier=0	挂载选项，在文件系统中禁用barrier。
nofail	本地盘出现在文件系统中但实际缺失时，实例的启动流程不会中断。

使用限制

- 由于疏散功能对于底层资源逻辑有一定要求，而且一般需要技术客户经理提供相应支持，当前该功能仅针对部分用户开放。后续本能力将作为维修平台的附加功能面向更多用户放开。
- 疏散实例结果可能会出现失败的情况，失败之后百度智能云将回滚您的实例，不会出现实例异常删除、数据丢失等情况。
- 当前仅限EBC/EHC产品可以选择执行针对故障实例的疏散动作。
- EBC/EHC实例在疏散后，本地盘的数据会完全丢失。**请您在使用该功能之前做好相关数据的备份工作。**

使用救援模式

操作场景

当BCC实例出现系统内grub 引导文件丢失、系统关键文件缺失、lib 动态库文件损坏/缺失等问题时，可能会导致操作系统无法进入单用户模式并完成修复，您可以使用救援模式来进行系统修复。本文介绍如何使用救援模式。

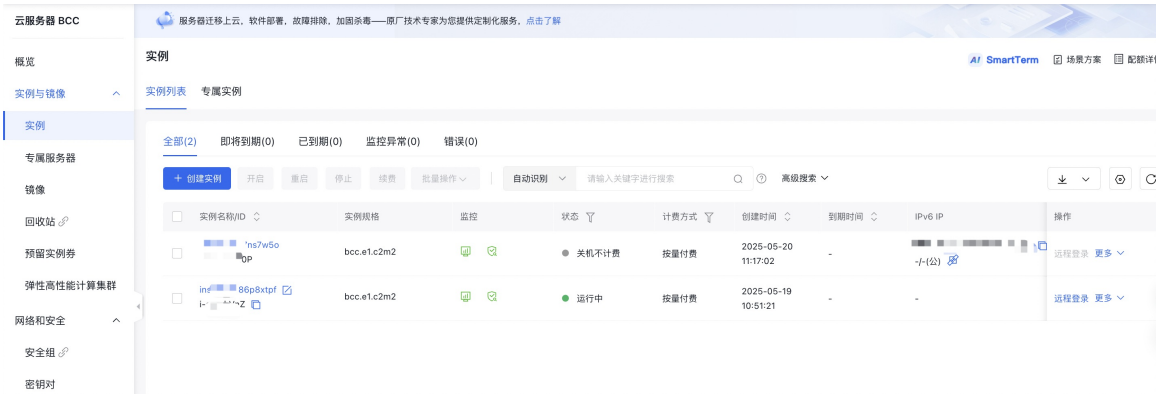
使用救援模式一般用于系统无法正常启动，需要修复系统盘场景。不建议对业务正常运行的实例操作，会影响业务运行。

前提条件

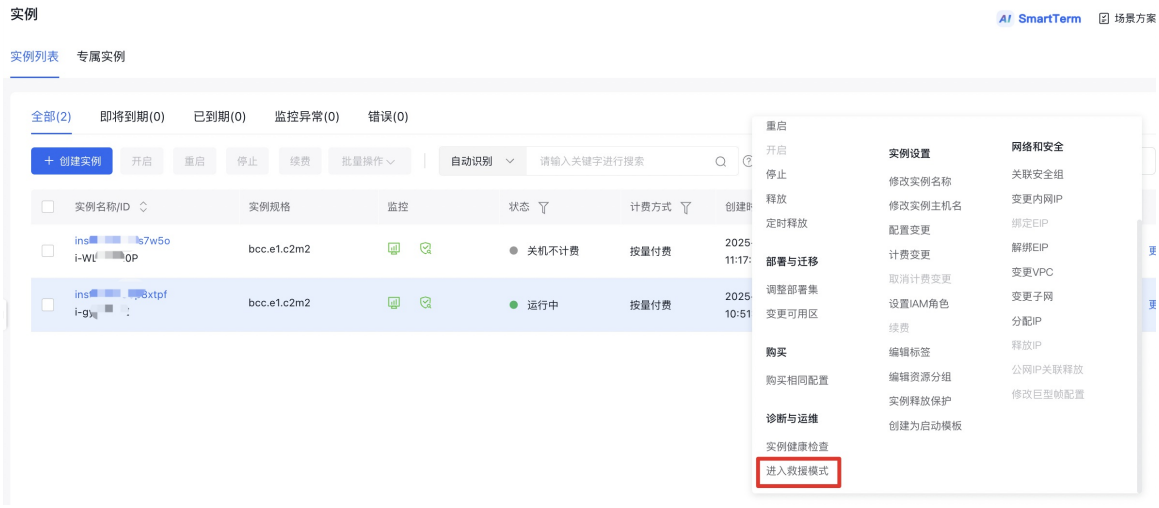
- 仅Linux系统的实例支持使用救援模式，Windows系统的实例不支持使用救援模式。
- 裸金属EBC实例暂不支持使用救援模式。
- 当实例状态为“运行中”或“已关机”时，支持使用救援模式。若实例状态为“运行中”，则进入救援模式需要允许实例重启。
- 实例进入救援模式期间不支持开机、关机、重启、配置变更、重置密码等资源操作，若需操作，请退出救援模式后操作。

操作步骤

进入救援模式 1.登录云服务器BCC控制台进入实例列表页面。



2.选择您要修复的实例，在右侧的操作栏单击“更多>>进入救援模式”，如下图所示。



3.在弹出的进入救援模式窗口中，设置救援模式期间登录实例的密码。如下图所示。

4.单击确认。

此时可查看实例状态为重启中，正在进入救援模式。当查看实例状态如下图所示，则说明已成功进入救援模式，请参考下一步修复实例。

使用救援模式进行系统修复 1.使用root账户及步骤3中设置的密码远程登录实例。

- 若实例有公网IP，您可以通过SSH登录实例；
- 若实例无公网IP，您可以登录控制台通过SmartTerm/VNC登录实例。

  2.登录成功后，您需要挂载系统盘根分区。

> 救援模式下实例系统盘设备名为 vda，根分区为 vda1，默认未挂载。不同操作系统的根分区会有所不同，根分区也会是vda2或者其他分区，可以依据实例所使用的操作系统进行判断，并进行挂载。
>
> 挂载成功后，您即可操作根分区中的数据，进行修改或修复。

以Ubuntu 22.04操作系统的实例为例：

救援模式下实例系统盘设备名为 vda，根分区为 vda2，默认未挂载。

- 挂载实例系统盘根分区

```
 mkdir -p /mnt/sysroot
mount /dev/vda2 /mnt/sysroot
```

挂载成功后，您即可操作根分区中的数据。

- 您可以通过chroot到实例系统盘的根目录下运行指令，具体操作命令如下：

```
 mount -o bind /dev /mnt/sysroot/dev
mount -o bind /dev/pts /mnt/sysroot/dev/pts
mount -o bind /proc /mnt/sysroot/proc
mount -o bind /run /mnt/sysroot/run
mount -o bind /sys /mnt/sysroot/sys
chroot /mnt/sysroot /bin/bash
```

退出救援模式 1.实例修复完成后，返回实例列表，选择实例，在右侧操作栏单击“更多>>退出救援模式”，如下图所示。

2.实例退出救援模式后，实例状态变为运行中。 **通过API使用救援模式** 您可以通过进入救援模式接口对指定实例进入救援模式；

您可以通过退出救援模式接口对指定实例退出救援模式。

弹性高性能计算集群

弹性高性能计算集群概述

弹性高性能计算集群概述

弹性高性能计算集群（Elastic Hyper-Computing Cluster，简称EHC）是基于高性能RDMA网络，将多个云服务器进行互联，EHC集群中的云服务器实例之间可通过RDMA网络高速互联，可提供高带宽、低时延的通信能力，极大提升计算任务的加速比的计算集群，不同EHC集群中的云服务器之间的RDMA网络相互隔离，可满足企业多业务部门之间的安全需求。

适用范围

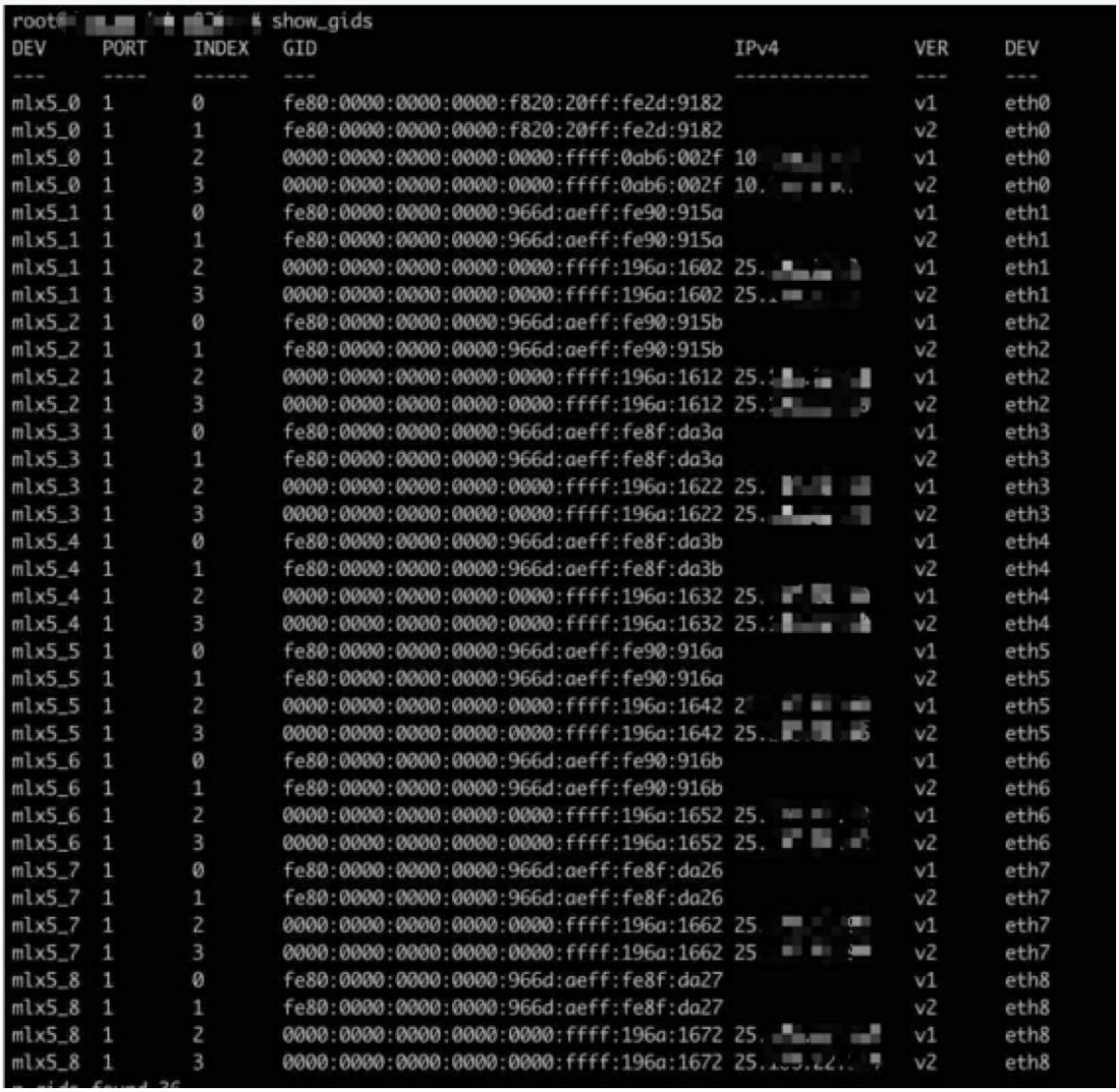
- EHC集群适用支持高性能RDMA网络的实例，可参考EHC实例规格。
- 同一EHC集群可支持不同实例规格加入。

应用场景

- 大模型训练
- 高性能计算

产品特性

- 高速网络通信：在同一EHC集群中支持实例以高带宽网络进行通信，通过RDMA协议可极大降低通信时延与开销。
- 安全可控：不同EHC集群中的实例的高性能RDMA网络互相隔离，满足企业多业务部门、测试与生产环境的合规与安全要求。
- 高扩展性：EHC集群最大可支持扩展到千台实例规模，满足业务通过scale out方式提高应用性能。
- 即开即用：创建EHC实例后百度智能云会为您分配并配置好用于通信的系统配置，如IP（如图1，系统会分配25网段用于高性能网络通信）、无损网络配置等，您无需关注复杂的网络运维即可使用高性能网络，登陆实例OS也可查看高性能网络设备和其身份，如图二所示。



```
root# show_gids
DEV     PORT    INDEX   GID                                     IPv4            VER     DEV
---     ----    -----   ---                                     ------------    ---     ---
mlx5_0  1       0       fe80:0000:0000:0000:f820:20ff:fe2d:9182                 v1      eth0
mlx5_0  1       1       fe80:0000:0000:0000:f820:20ff:fe2d:9182                 v2      eth0
mlx5_0  1       2       0000:0000:0000:0000:0000:ffff:0ab6:002f 10              v1      eth0
mlx5_0  1       3       0000:0000:0000:0000:0000:ffff:0ab6:002f 10.             v2      eth0
mlx5_1  1       0       fe80:0000:0000:0000:966d:aeff:fe90:915a                 v1      eth1
mlx5_1  1       1       fe80:0000:0000:0000:966d:aeff:fe90:915a                 v2      eth1
mlx5_1  1       2       0000:0000:0000:0000:0000:ffff:196a:1602 25.             v1      eth1
mlx5_1  1       3       0000:0000:0000:0000:0000:ffff:196a:1602 25.             v2      eth1
mlx5_2  1       0       fe80:0000:0000:0000:966d:aeff:fe90:915b                 v1      eth2
mlx5_2  1       1       fe80:0000:0000:0000:966d:aeff:fe90:915b                 v2      eth2
mlx5_2  1       2       0000:0000:0000:0000:0000:ffff:196a:1612 25.             v1      eth2
mlx5_2  1       3       0000:0000:0000:0000:0000:ffff:196a:1612 25.             v2      eth2
mlx5_3  1       0       fe80:0000:0000:0000:966d:aeff:fe8f:da3a                 v1      eth3
mlx5_3  1       1       fe80:0000:0000:0000:966d:aeff:fe8f:da3a                 v2      eth3
mlx5_3  1       2       0000:0000:0000:0000:0000:ffff:196a:1622 25.             v1      eth3
mlx5_3  1       3       0000:0000:0000:0000:0000:ffff:196a:1622 25.             v2      eth3
mlx5_4  1       0       fe80:0000:0000:0000:966d:aeff:fe8f:da3b                 v1      eth4
mlx5_4  1       1       fe80:0000:0000:0000:966d:aeff:fe8f:da3b                 v2      eth4
mlx5_4  1       2       0000:0000:0000:0000:0000:ffff:196a:1632 25.             v1      eth4
mlx5_4  1       3       0000:0000:0000:0000:0000:ffff:196a:1632 25.             v2      eth4
mlx5_5  1       0       fe80:0000:0000:0000:966d:aeff:fe90:916a                 v1      eth5
mlx5_5  1       1       fe80:0000:0000:0000:966d:aeff:fe90:916a                 v2      eth5
mlx5_5  1       2       0000:0000:0000:0000:0000:ffff:196a:1642 2               v1      eth5
mlx5_5  1       3       0000:0000:0000:0000:0000:ffff:196a:1642 25.             v2      eth5
mlx5_6  1       0       fe80:0000:0000:0000:966d:aeff:fe90:916b                 v1      eth6
mlx5_6  1       1       fe80:0000:0000:0000:966d:aeff:fe90:916b                 v2      eth6
mlx5_6  1       2       0000:0000:0000:0000:0000:ffff:196a:1652 25.             v1      eth6
mlx5_6  1       3       0000:0000:0000:0000:0000:ffff:196a:1652 25.             v2      eth6
mlx5_7  1       0       fe80:0000:0000:0000:966d:aeff:fe8f:da26                 v1      eth7
mlx5_7  1       1       fe80:0000:0000:0000:966d:aeff:fe8f:da26                 v2      eth7
mlx5_7  1       2       0000:0000:0000:0000:0000:ffff:196a:1662 25              v1      eth7
mlx5_7  1       3       0000:0000:0000:0000:0000:ffff:196a:1662 25              v2      eth7
mlx5_8  1       0       fe80:0000:0000:0000:966d:aeff:fe8f:da27                 v1      eth8
mlx5_8  1       1       fe80:0000:0000:0000:966d:aeff:fe8f:da27                 v2      eth8
mlx5_8  1       2       0000:0000:0000:0000:0000:ffff:196a:1672 25.             v1      eth8
mlx5_8  1       3       0000:0000:0000:0000:0000:ffff:196a:1672 25.             v2      eth8
```

创建弹性高性能计算集群

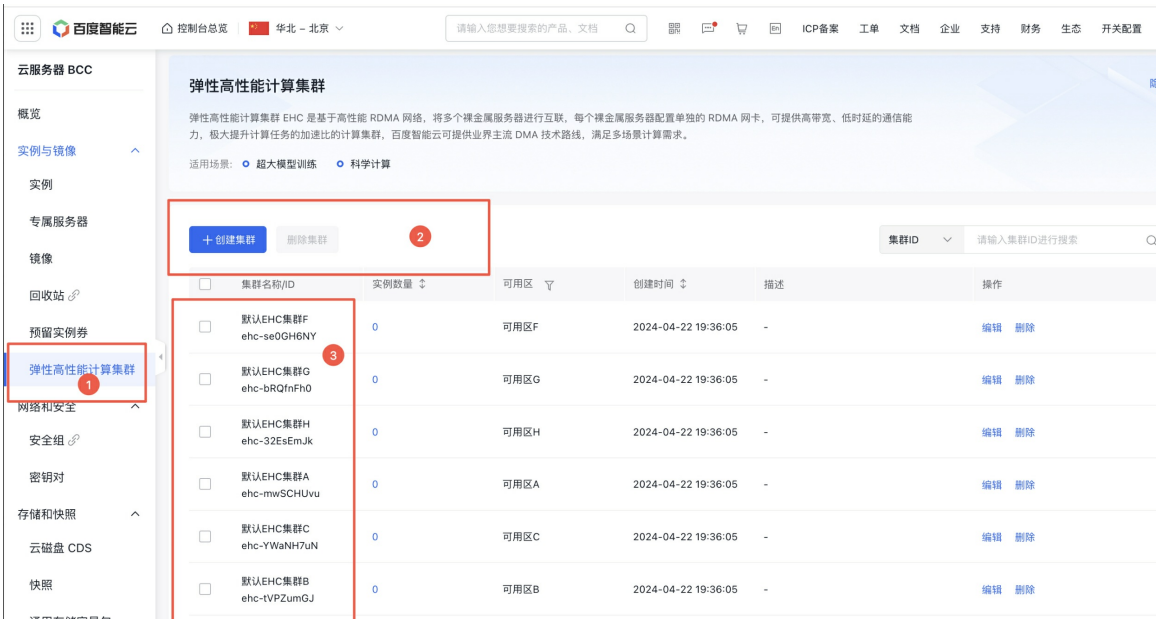
概述

本文介绍如何在控制台完成弹性高性能计算集群（EHC集群）的管理。

创建EHC集群并创建BCC实例

按以下步骤在控制台上创建EHC集群：

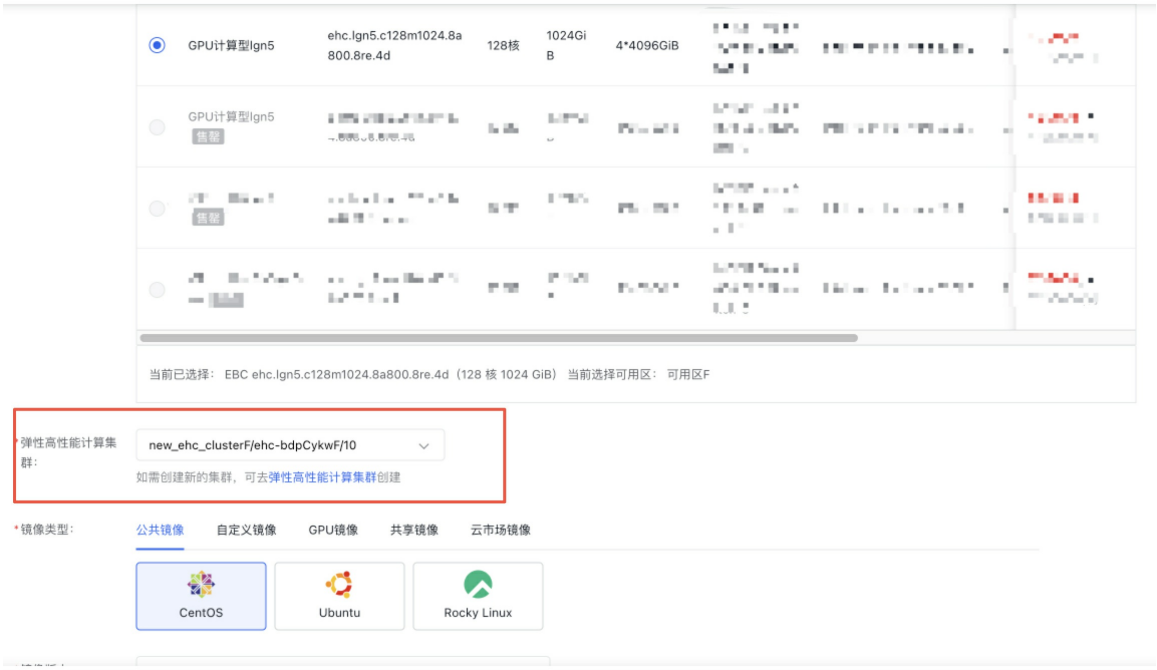
1. 登录 云管理控制台，选择“产品服务 > 计算 > 云服务器 BCC”。
2. 选择云服务器所在的区域，
3. 如图数字1，侧边栏选择弹性高性能计算集群模块，在列表中可看到系统为您默认生成的EHC集群，如您不需要多个EHC集群来隔离网络，使用默认EHC集群即可。您也可以通过数字2点击创建实例来创建更多集群。



4. 在弹出的窗口中选择可用区，填写集群名称及描述，点击确认。

5. 控制台侧边栏选择实例并点击创建实例

6. 创建BCC中的EHC套餐，在弹性高性能计算集群选择新创建的ehc集群并完成此次创建



删除EHC集群

在EHC集群的列表页中选择操作中的删除即可删除EHC集群，删除集群需要该EHC集群中没有实例，您可点击列表中的实例数量跳转到BCC实例列表查看该集群中有哪些实例。

创建EHC实例

对比其他BCC实例规格，创建EHC实例需要填写EHC集群参数，您可参考创建实例查看相关操作。

创建EHC实例的预留实例券

对比其他BCC实例规格，创建EHC实例的预留实例券需要填写EHC集群参数，您可参考创建预留实例券查看相关操作。

资源调研

概述

若您从其他云厂商或者IDC迁移至百度云服务器BCC时，资源调研能根据调研的源端资源地域、规格、磁盘、公网等属性，进行云服务器BCC的选型推荐，同时也支持用户对推荐配置进行自定义调整。

使用限制

- 资源调研支持根据计算规格、磁盘、公网推荐匹配的云服务器BCC、云磁盘CDS、弹性公网EIP产品规格。
- 资源调研仅支持导入文件方式。
- 资源清单文件导入支持csv、xls、xlsx格式，且文件大小不得超过5MB。
- 一次导入任务仅支持上传一个文件。

操作前准备

资源调研支持对阿里云、腾讯云的实例清单进行选型匹配，同时也提供导入模板。

- 若您源端资源来自阿里云或者腾讯云，可以登录源云厂商控制台导出实例列表，需要选中全部字段操作导出，且不可删除表头。

> 若您有数据安全考量，规格、计费方式、盘、网络等资源相关字段信息需要保留，其他字段的信息可以清空。
>
> 详细字段可见控制台资源调研导入页面说明。

- 若您源端资源来自其他云厂商或者自建IDC，您可以下载导入模板填充数据，并将其上传。

导入

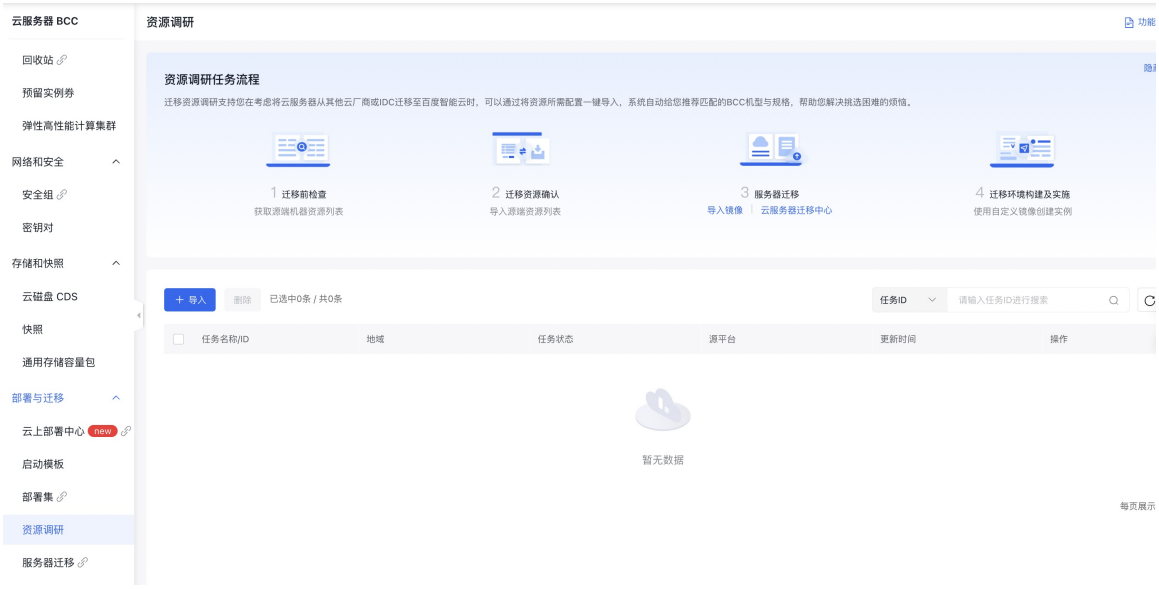
功能说明

您可以通过资源文件导入一键获取推荐机型和价格。

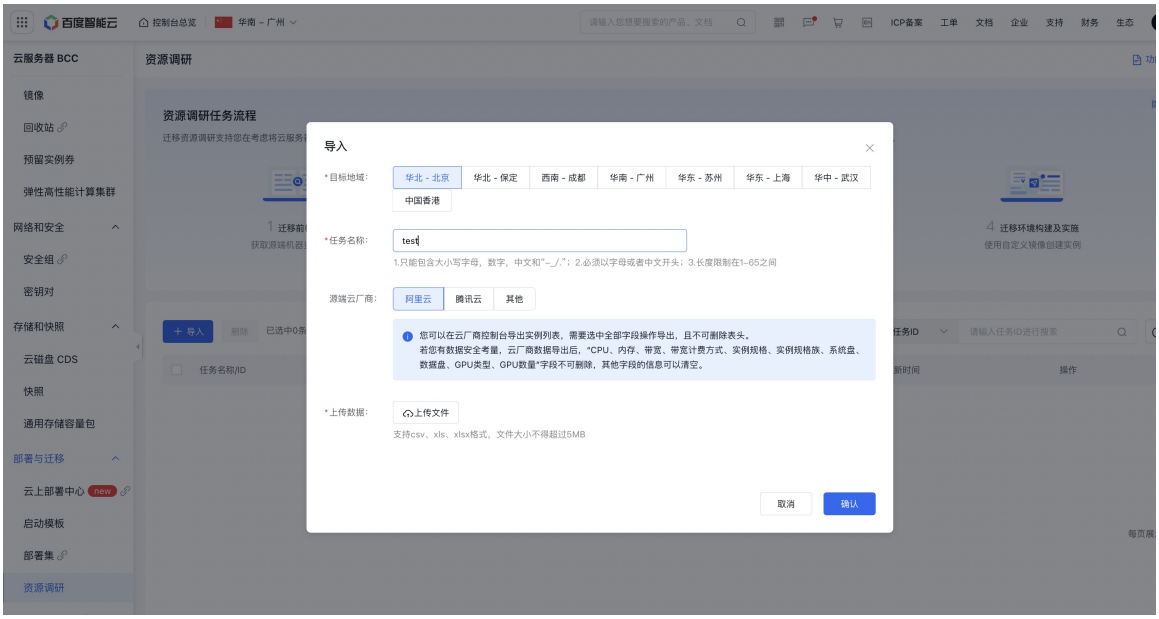
操作步骤

1.登录云服务器BCC控制台进入实例列表页。

2.在左侧导航选择"**部署与迁移-资源调研**"，进入资源调研列表页面。



3.点击**导入**，选择目标地域和源端云厂商，并上传文件，完成后点击**确认**。



4.列表页出现一条调研任务，任务状态为**创建中**。

5.待任务状态变为**已完成**后，点击任务名称进行详情，查看资源推荐。

下载

任务完成后您可以点击下载，将该任务里所有的源资源信息和匹配的机型、价格信息以excel方式导入，方便进行成本比对和核算。

修改推荐配置

若默认推荐配置不太满足您的需求，您可以点击“**修改推荐配置**”来选择您的目标配置，确认后资源列表会根据您的选择来自动刷新推荐配置和价格。

创建实例

点击“**创建实例**”则默认会带着推荐配置跳转至云服务器BCC创建页。

删除

若您不需要某条资源数据，可以操作删除。删除后数据无法找回，若您需要找回，可以重新导入文件。

视频专区

产品简介

百度云服务器BCC简介

- 视频简介：百度云服务器BCC简介，帮助您快速了解云服务器BCC。

操作指南

基础功能

BCC远程连接实例

- 介绍如何快速连接BCC实例，主要包括Linux环境下连接Linux实例、Windows环境下连接Linux实例和在Windows环境下连接Windows实例。

BCC创建自定义镜像

- 介绍如何在控制台为云服务器BCC创建自定义镜像。

BCC扩容系统盘

- 介绍如何对云服务器BCC系统盘进行扩容。

BCC创建实例快照

- 介绍如何在控制台为云服务器BCC创建实例快照。

使用BCC标签功能

- 介绍标签管理功能的使用方法，通过给每个云资源添加标签，从而快速分类和识别管理这些资源。

使用密钥登录Linux云服务器

- 介绍如何通过SSH密钥登录Linux云服务器。

网络相关

BCC配置IPv6

- 介绍BCC实例如何配置IPv6。

BCC CentOS实例添加辅助IP

- 介绍BCC CentOS实例如何添加辅助IP。

BCC CentOS实例配置EIP直通

- 介绍BCC中CentOS实例配置EIP直通的方法。

BCC实例通过NAT网关访问公网

- 介绍BCC实例如何通过NAT网关访问公网的方法。

BCC配置跨账号对等连接

- 介绍BCC实例如何配置跨账号对等连接功能。

密码功能

BCC重置密码

- 介绍如何在控制台重置云服务器BCC密码。

Linux通过单用户模式重置密码

- 介绍如何通过单用户模式重置Linux实例root密码。

环境搭建

BCC Linux系统安装宝塔

- 介绍BCC Linux系统如何安装宝塔面板。

BCC Windows Server 2012安装宝塔面板

- 介绍如何安装BCC Windows Server 2012宝塔面板。

BCC CentOS 7 yum安装MySQL 8.0

- 介绍CentOS 7 yum安装MySQL 8.0的过程。

BCC Windows服务器实例共享文件夹

- 介绍BCC中Windows服务器实例如何共享文件夹

BCC CentOS安装Nginx服务并部署SSL证书

- 介绍CentOS安装Nginx服务并部署SSL证书的方法

BCC Linux服务器安装Java环境和部署Tomcat服务

- 介绍Linux服务器安装Java环境和部署Tomcat服务的方法

BCC CentOS7.5安装FTP服务

- 介绍CentOS安装FTP服务的方法

运维参考

挂载CDS磁盘和格式化分区

- 介绍如何快速挂载CDS磁盘，主要包括Linux环境和Windows环境下如何挂载CDS磁盘。

Windows 2008关闭IE增强

- 介绍Windows Server 2008服务器如何关闭IE增强。

Windows 2012关闭IE增强

- 介绍Windows Server 2012服务器如何关闭IE增强。

Linux修改SSH默认端口号

- 介绍如何修改Linux中SSH的默认端口号。

Windows服务器修改默认远程端口

- 介绍如何修改Windows服务器默认远程端口。

故障处理

BCC Windows实例重置密码不成功

- 介绍BCC Windows实例重置密码不成功的原因以及解决办法。

BCC Windows实例磁盘脱机

- 介绍BCC Windows实例磁盘脱机问题的解决方法。

典型实践

标签管理实践

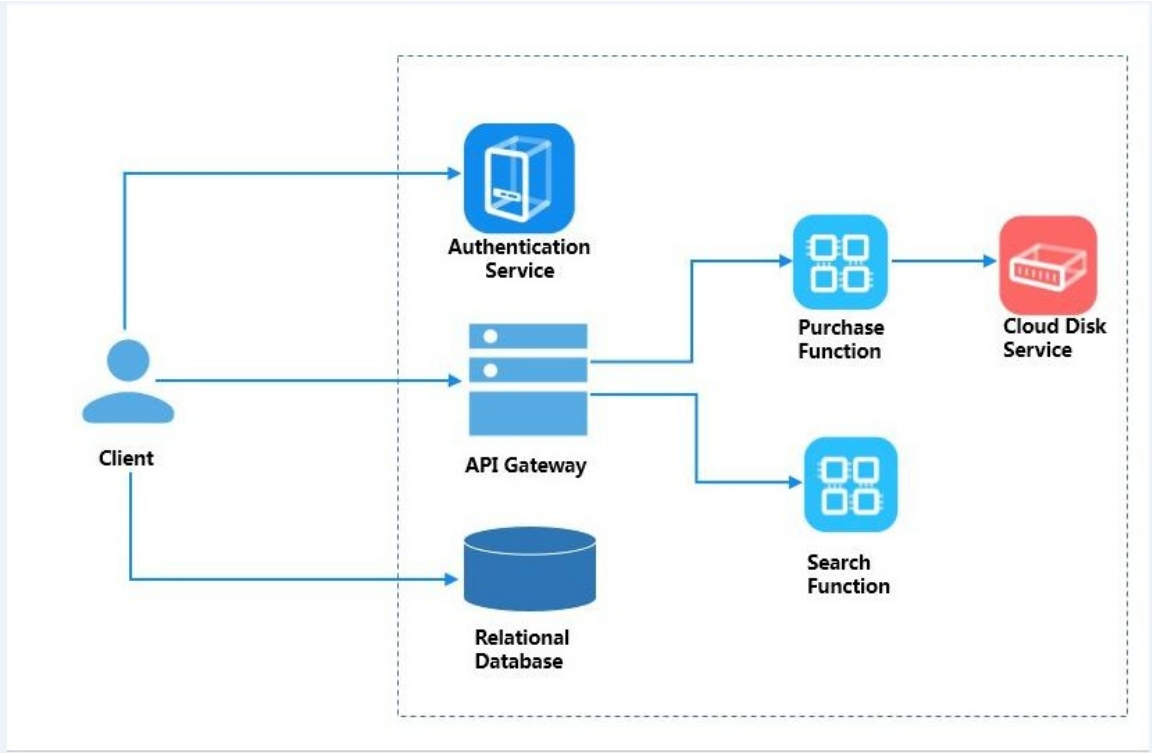
概览

通过标签管理可支持按照各种标准（如用途、所有者或项目）跨服务、跨区域对资源添加标签，从而快速分类和管理资源。

需求场景

为了方便拥有大量BCC实例的客户进行实例管理，期望可以按照各种标准（如用途、所有者或项目）对资源进行分类添加标签，便于客户快速分类和识别资源。

方案概述



上图包含（认证服务、API网关、收费功能、服务功能、产品数据服务器这六个实例以及一个挂载在收费功能上的磁盘）的网站系统，以此进行创建标签命名的示范。

配置步骤

1. API Gateway 标签名命名为— —Gateway 001（键名为：Gateway，值为001）步骤如下：

2. 由于Purchase Funciton和Search Function功能属于同一类型，从项目管理的角度上把两个标签名的定义键名相同值不同，便于管理。

Purchase Function标签名命名为— —Function 002（键名为：Function，值为002）；

3. Cloud Disk Service是Purchase Function挂载的磁盘，直接点击关联资源CDS、EIP并统一加标签按钮自动命名为— —Function 002；

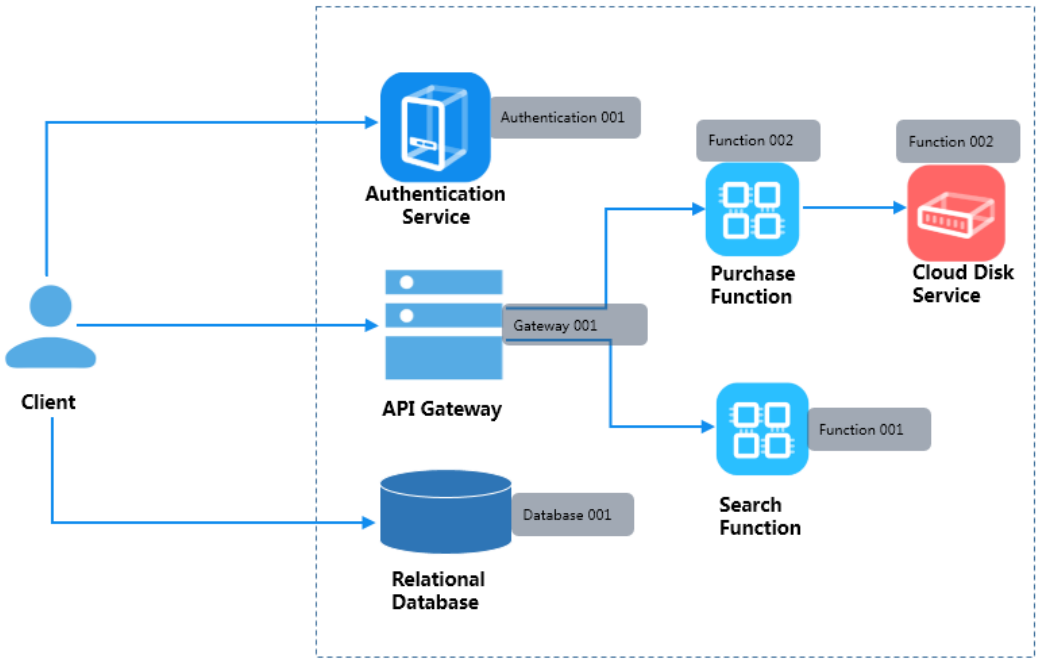
4. 依次为其他资源添加标签：

Search Function标签名命名为— —Function 001（键名为：Function，值为001）；

Authentication Service 标签名命名为— —Authentication 001 （键名为：Authentication，值为001）；

Product Database标签名命名为— —Database 001（键名为：Database，值为001）；

5. 系统标记标签名后如下图所示，深色文本框内为实例标签名：



环境搭建

AccessVPN搭建典型实践

概览

通过VPN拨号的方式，实现外网终端与云平台内部资源的安全连接，方便用户在任何地点远程安全接入智能云。

需求场景

需求场景1：面向企业 IT 人员，提升云上资源的管理维护

可以简化云上资源运维，消除云上和云下资源的运维差异。

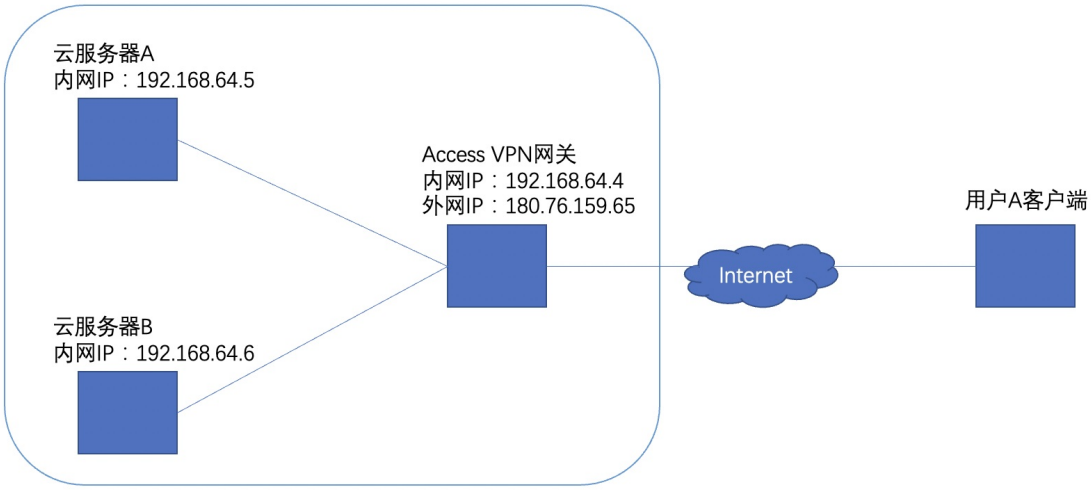
需求场景2：面向企业普通员工，提供混合云下的移动办公解决方案

加速企业IT基础设施的数字化转型，实现真正的移动办公，让公司员工无论何时何地都可以如同在公司内网一样接入企业IT系统。

方案概述

如下图所示，用户可以使用云服务器BCC产品搭建Access VPN服务器作为客户端远程连接智能云上资源的VPN网关。云平台为用户提供了集成开源Open VPN Access Server的服务集成镜像，用户在创建BCC时可以使用该镜像快速搭建Access VPN服务。

服务搭建完成后，用户可在安装了VPN客户端的PC上接入VPN，实现在远程内网访问到云服务器资源（如下图中云服务器A和云服务器B）的需求场景。



配置步骤

环境准备

- 客户端：可以连接到Internet的PC或者笔记本电脑
- 服务端：一台绑定了公网IP的BCC，作为Access VPN 网关
- 服务端配置建议：500个用户并发连接规模推荐采用bcc.g3.c4m16型号（Intel Xeon(Skylake) Gold 6148，4核CPU，16GB内存）来承载，如果仅是几名运维人员登录，采用较低配置服务器（如bcc.g3.c1m4）即可

> 说明：EIP带宽可根据用户实际使用情况灵活调整。

服务端（Access VPN网关）配置示例

1. 云平台为用户提供了集成了开源Open VPN Access Server的服务镜像帮助用户快速搭建VPN网关。用户在智能云Console中购买BCC，使用镜像选择"服务集成镜像 > Access VPN CentOS6.5 (64位) 版"。

> 说明：用户也可以选择合适的公共镜像自行下载VPN Server安装包搭建VPN网关。

2. 在购买页面同时选择“购买弹性公网IP”，并根据您的需要选择适合的计费方式和带宽峰值。

3. 完成服务器创建后，在实例列表界面复制第一步创建的Access VPN服务器的公网IP，如下图所示为180.76.159.65。

4. SSH登录Access VPN服务器，修改/root/client.ovpn文件中第四行的为服务器的公网IP。

以本例中的公网IP 187.76.159.65为例，可以使用命令`sed -ri "s/remote\s+\s+1194/remote 187.76.159.65 1194/g" /root/client.ovpn`。命令仅供参考，用其他方法修改也是可以的。

下图为修改后的client.ovpn文件：

5. 将client.ovpn文件从Access VPN服务器里取出，并复制到客户端所在机器。
6. 如果需要下发路由和DNS配置，可以登录Access VPN服务器，在/etc/openvpn/server.conf 增加push配置项，然后service openvpn restart 重启VPN服务。

客户端（Open VPN Client）配置示例

1. Windows/Linux/Mac系统，登录Open VPN的官网，下载并安装对应操作系统的客户端。
2. 将从服务端复制好的配置文件client.ovpn加载到配置中。

3. 启动Open VPN Client，选择加载好的配置文件进行连接，登录成功。

测试验证

在完成全部配置的A用户的PC或者笔记本电脑上，通过Ping测试连接云服务器BCC的内网IP（云服务器A的内网IP地址是192.168.64.5，云服务器B的内网IP地址是192.168.64.6）。若能Ping通，则说明VPN搭建成功。

至此，用户A通过Access VPN方式，成功连接到云平台内部的BCC资源。

相关产品

云服务器BCC、弹性公网IP

Linux系统搭建FTP服务

Centos搭建FTP服务

此部分介绍百度智能云“CentOS7.5安装FTP服务”的方法（含操作指导视频），您可以参考以下视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《CentOS7.5安装FTP服务》。

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

本教程以 百度智能云公共镜像CentOS6.5/CentOS7.5 64位操作系统为例，说明如何在 Linux 实例上安装并配置 vsftpd。

- 步骤一： 安装 vsftpd
- 步骤二： 配置 vsftpd
  - 方式一：配置匿名用户上传文件权限
  - 方式二：配置本地用户登录
- 步骤三： 设置安全组
- 步骤四： 客户端测试

步骤一： 安装 vsftpd

1.远程连接并登录到 Linux 实例。

2.运行命令yum install -y vsftpd安装 vsftpd。

3.运行以下命令进入/etc/vsftpd目录，并查看该目录下的文件。

```
 cd /etc/vsftpd
ls
```

说明:

- /etc/vsftpd/vsftpd.conf 是 vsftpd 的核心配置文件。
- /etc/vsftpd/user_list 是白名单文件，此文件里的用户允许访问 FTP 服务器。
- /etc/vsftpd/ftpusers 是黑名单文件，此文件里的用户不允许访问 FTP 服务器。

4.运行以下命令设置FTP服务开机自启动。

Centos6.5: `chkconfig vsftpd on`

Centos7.5: `systemctl enable vsftpd.service`

5.运行以下命令启动 FTP 服务。

Centos6.5:`/etc/init.d/vsftpd start`

Centos7.5:`systemctl start vsftpd.service`

6运行以下命令查看 FTP 服务监听的端口。

netstat -antup | grep ftp

步骤二： 配置 vsftpd

vsftpd 安装后默认开启了匿名访问 FTP 服务器的功能。使用匿名访问，您无需输入用户名密码即可登录 FTP 服务器，但没有权限修改或上传文件。

本教程介绍了以下两种配置 vsftpd 的方法，并提供了相关的参数说明，您可以根据具体需要进行参考。

- 配置匿名用户上传文件权限
- 配置本地用户登录

方式一：配置匿名用户上传文件权限

匿名访问FTP服务器是一种不安全的访问模式，任何人无需密码验证就可以登录到FTP服务器，这种模式一般只用来保存不重要的公开文件，不推荐在生产环境中使用。如果您需要配置匿名用户上传文件的权限，可以参考以下步骤配置：

1.修改/etc/vsftpd/vsftpd.conf。

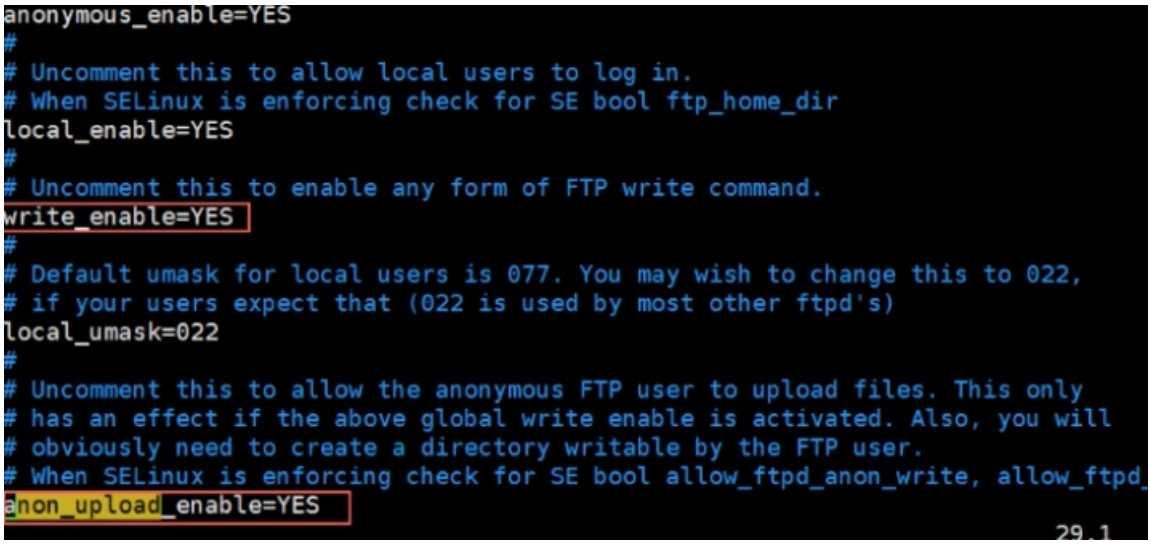
a.运行vim /etc/vsftpd/vsftpd.conf。

b.按i 键进入编辑模式。

c.将写权限修改为write_enable=YES。

d.将匿名上传权限修改为anon_upload_enable=YES。

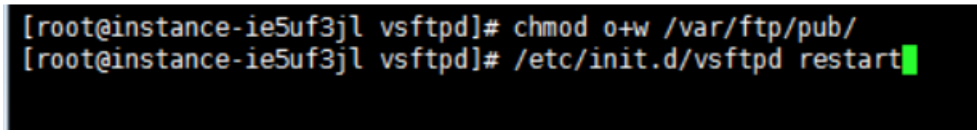
e.按Esc键退出编辑模式，然后输入:wq保存并退出文件。



2.运行以下命令更改/var/ftp/pub目录的权限，为 FTP 用户添加写权限，并重新加载配置文件。

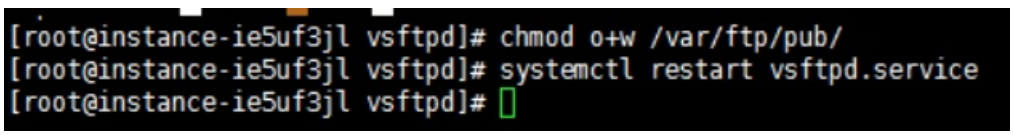
Centos6.5:

```
 chmod o+w /var/ftp/pub/
/etc/init.d/vsftpd restart
```

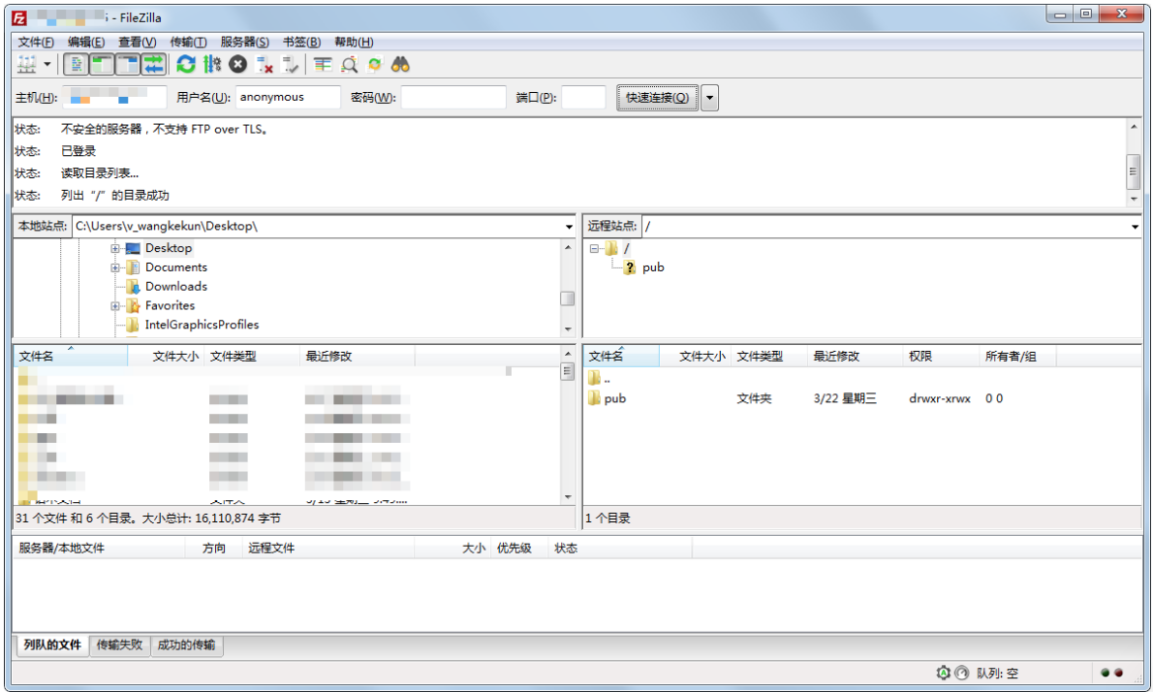


CentOS7.5:

```
 chmod o+w /var/ftp/pub/
systemctl restart vsftpd.service
```



此时使用filezilla测试可以在不需要账户密码（匿名下）的情况下正常连接该FTP服务器



方式二：配置本地用户登录

本地用户登录是指用户使用 Linux 操作系统的账号和密码登录 FTP 服务器。

vsftpd 安装后默认只支持匿名访问 FTP 服务器，如果您试图使用 Linux 操作系统中的账号登录服务器，将会被 vsftpd 服务拒绝。您可以参考以下步骤配置 vsftpd 服务，以使用Linux账号和密码访问FTP服务器。

1.运行以下命令为FTP服务创建一个Linux用户。本示例中，该用户名为ftpbaidu。

useradd ftpbaidu

```
[root@instance-ie5uf3jl vsftpd]# useradd ftpbaidu
[root@instance-ie5uf3jl vsftpd]#
```

2.运行以下命令修改ftptest 用户的密码。

passwd ftpbaidu

3.创建一个供FTP服务使用的文件目录。

mkdir /var/ftp/baidu

4.更改/var/ftp/baidu目录的拥有者为ftpbaidu。

chown -R ftpbaidu:ftpbaidu /var/ftp/baidu

5.输入命令vim /etc/vsftpd/vsftpd.conf打开vsftpd.conf配置文件并按键i进入编辑模式。

6.修改vsftpd.conf配置文件。

FTP服务器可以配置为主动模式或被动模式。

主动模式下，客户端向服务端发送数据端口的信息，由服务端主动连接客户端发送的数据端口。配置FTP为主动模式的参数如下：

```
 #禁止匿名登录FTP服务器
anonymous_enable=NO
#允许本地用户登录FTP服务器
local_enable=YES
#设置本地用户登录后所在的目录
local_root=/var/ftp/baidu
#全部用户被限制在主目录
chroot_local_user=YES
#启用例外用户名单
chroot_list_enable=YES
#指定例外用户列表，这些用户不被锁定在主目录
chroot_list_file=/etc/vsftpd/chroot_list

#配置其他参数
allow_writeable_chroot=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

（1）按Esc键退出编辑模式，然后按键:wq保存并退出文件。

（2）运行touch /etc/vsftpd/chroot_list命令创建chroot_list文件，并写入不受只可以访问其主目录限制的例外用户名单。

注意：如果没有例外用户也必须要有 chroot_list文件，内容可为空。

（3）按键Esc退出编辑模式，然后按键:wq保存并退出文件。

（4）运行以下命令重启vsftpd服务。/etc/init.d/vsftpd restart

到此完成主动模式所有配置。

被动模式下，服务端开启并发送数据端口的信息给客户端，由客户端连接服务端开启的数据端口，服务端被动接受连接。在被动模式下，您需要配置服务端可以开启的数据端口范围。配置FTP为被动模式的参数如下：

```
 #禁止匿名登录FTP服务器
anonymous_enable=NO
#允许本地用户登录FTP服务器
local_enable=YES
#设置本地用户登录后所在目录
local_root=/var/ftp/baidu
#全部用户被限制在主目录
chroot_local_user=YES
#启用例外用户名单
chroot_list_enable=YES
#指定例外用户列表，这些用户不被锁定在主目录
chroot_list_file=/etc/vsftpd/chroot_list
#开启被动模式
pasv_enable=YES
#FTP服务器公网IP
pasv_address=<FTP服务器公网IP>
#设置被动模式下，建立数据传输可使用port范围的最小值
pasv_min_port=50000
#设置被动模式下，建立数据传输可使用port范围的最大值
pasv_max_port=50010
#配置其他参数
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
xferlog_std_format=YES
tcp_wrappers=YES
allow_writeable_chroot=YES
listen=YES
listen_ipv6=NO
pam_service_name=vsftpd
userlist_enable=YES
```

注意：建议您把端口范围设在比较高的一段范围内，比如50000-50010，有助于提高访问FTP服务器的安全性。

（1）按Esc键退出编辑模式，然后按键:wq保存并退出文件。

（2）运行touch /etc/vsftpd/chroot_list命令创建chroot_list文件，并写入不受只可以访问其主目录限制的例外用户名单。

注意：如果没有例外用户也必须要有 chroot_list文件，内容可为空。

（3）按键Esc退出编辑模式，然后按键:wq保存并退出文件。

（4）运行以下命令重启vsftpd服务。/etc/init.d/vsftpd restart

到此完成被动模式所有配置。

vsftpd.conf 的配置文件参数说明

运行命令cat /etc/vsftpd/vsftpd.conf查看配置文件内容。

用户登录控制：

参数	说明
anonymous_enable=YES	接受匿名用户
no_anon_password=YES	匿名用户login时不询问口令
anon_root=(none)	匿名用户主目录
local_enable=YES	接受本地用户
local_root=(none)	本地用户主目录

用户权限控制：

参数	说明
write_enable=YES	可以上传(全局控制)
local_umask=022	本地用户上传文件的umask
file_open_mode=0666	上传文件的权限配合umask使用
anon_upload_enable=NO	匿名用户可以上传
anon_mkdir_write_enable=NO	匿名用户可以建目录
anon_other_write_enable=NO	匿名用户修改删除
chown_username=lightwiter	匿名上传文件所属用户名

步骤三： 设置安全组

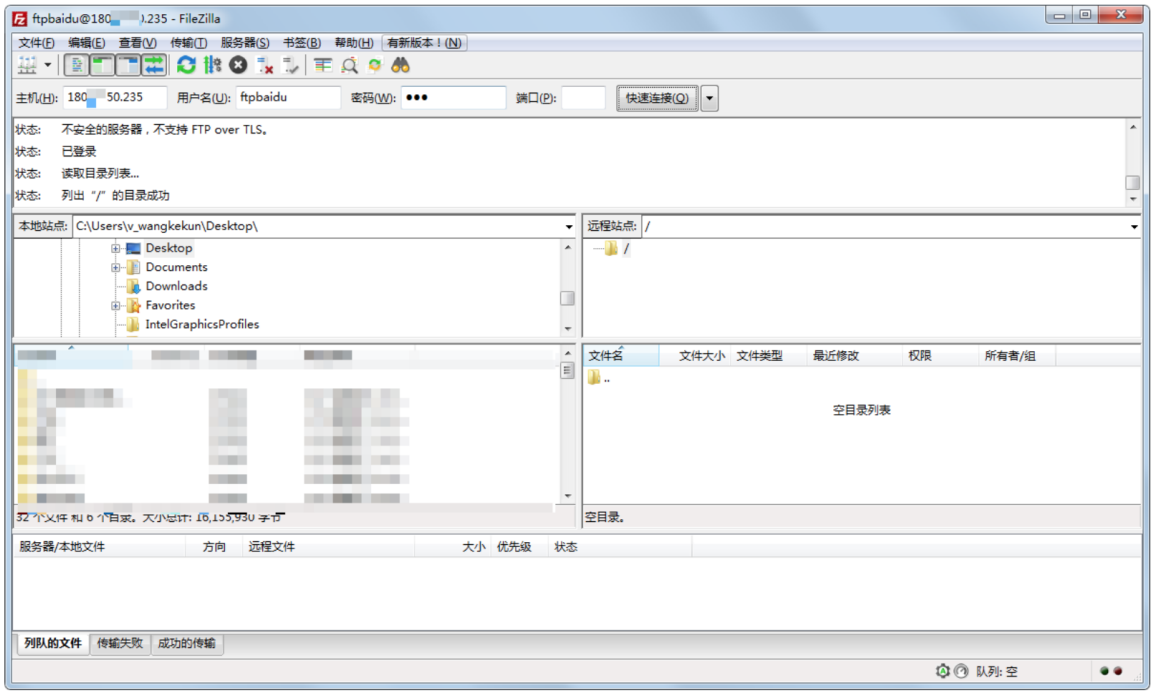
搭建好 FTP 站点后，您需要在实例安全组的入方向添加放行下列 FTP 端口的规则。

FTP 为主动模式时：端口21。
FTP 为被动模式时：端口21，以及配置文件/etc/vsftpd/vsftpd.conf中参数pasv_min_port和pasv_max_port之间的所有端口。

添加安全组规则的具体步骤，请参见添加安全组规则。

步骤四： 客户端测试

您可以通过 FTP 客户端或浏览器访问 FTP 服务器进行测试。本教程以filezilla工具为例：



ubuntu搭建FTP服务

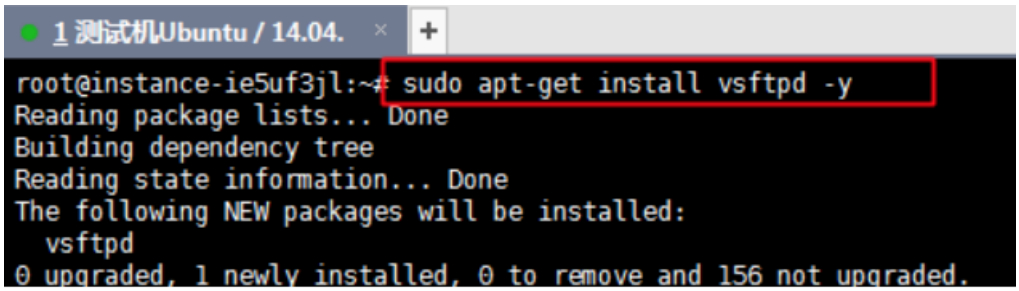
本教程以 百度智能云公共镜像ubuntu14.01 64位操作系统为例，说明如何在 Linux 实例上安装并配置 vsftpd。

- 步骤一： 安装 vsftpd
- 步骤二： 配置 vsftpd
  - 方式一：配置匿名用户上传文件权限
  - 方式二：配置本地用户登录
- 步骤三： 设置安全组
- 步骤四： 客户端测试

步骤一： 安装 vsftpd

1.远程连接并登录到 Linux 实例。

2.运行命令sudo apt-get install vsftpd -y 安装 vsftpd。



3.安装完成后 VSFTPD 会自动启动且默认开机自启，通过 netstat 命令可以看到系统已经监听

步骤二： 配置 vsftpd

vsftp分为匿名模式（不需要输入账号密码即可登录）和本地用户模式

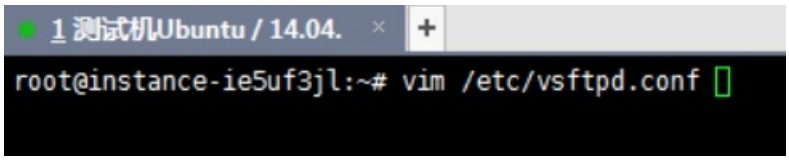
本教程介绍了以下两种配置 vsftpd 的方法，并提供了相关的参数说明，您可以根据具体需要进行参考。

- 匿名模式
- 本地用户模式

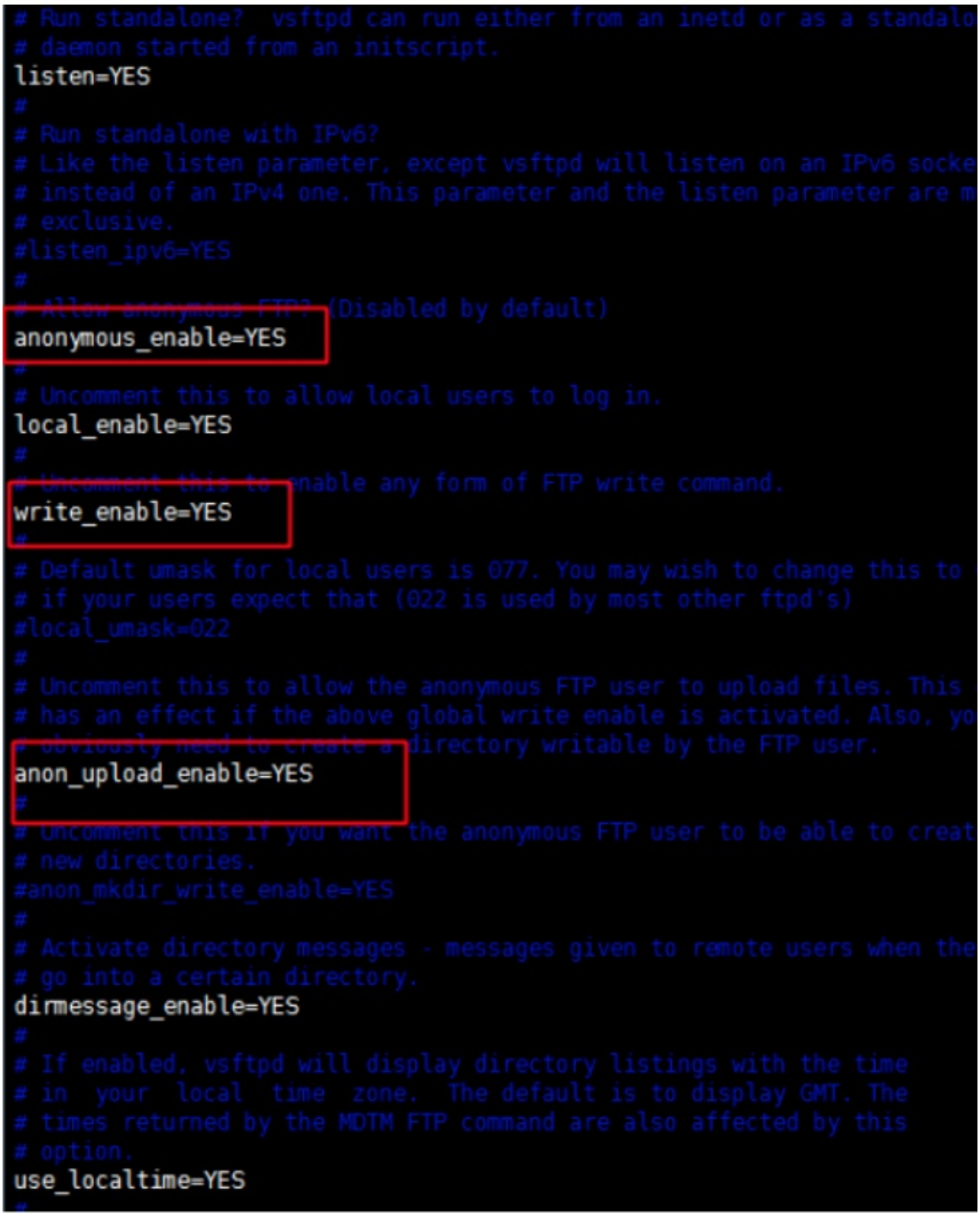
方式一：配置匿名用户上传文件权限

匿名访问FTP服务器是一种不安全的访问模式，任何人无需密码验证就可以登录到FTP服务器，这种模式一般只用来保存不重要的公开文件，不推荐在生产环境中使用。如果您需要配置匿名用户上传和下载文件的权限，可以参考以下步骤配置：

1.执行命令： vim /etc/vsftpd.conf



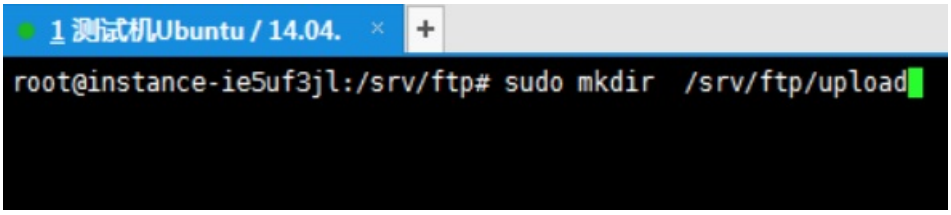
a.按i 键进入编辑模式。 b.将写权限修改为write_enable=YES。 c.将匿名上传权限修改为anon_upload_enable=YES。 d.将是否允许匿名登录修改为anonymous_enable=YES e.按Esc键退出编辑模式，然后输入:wq保存并退出文件。



2.创建FTP子目录。

执行命令：sudo mkdir /srv/ftp/upload

ubuntu搭建的FTP默认根目录是/srv/ftp/，但是匿名用户默认不可以在根目录上传文件，因此只能建立子目录实现上传.



3.对该目录设置755权限和分组。

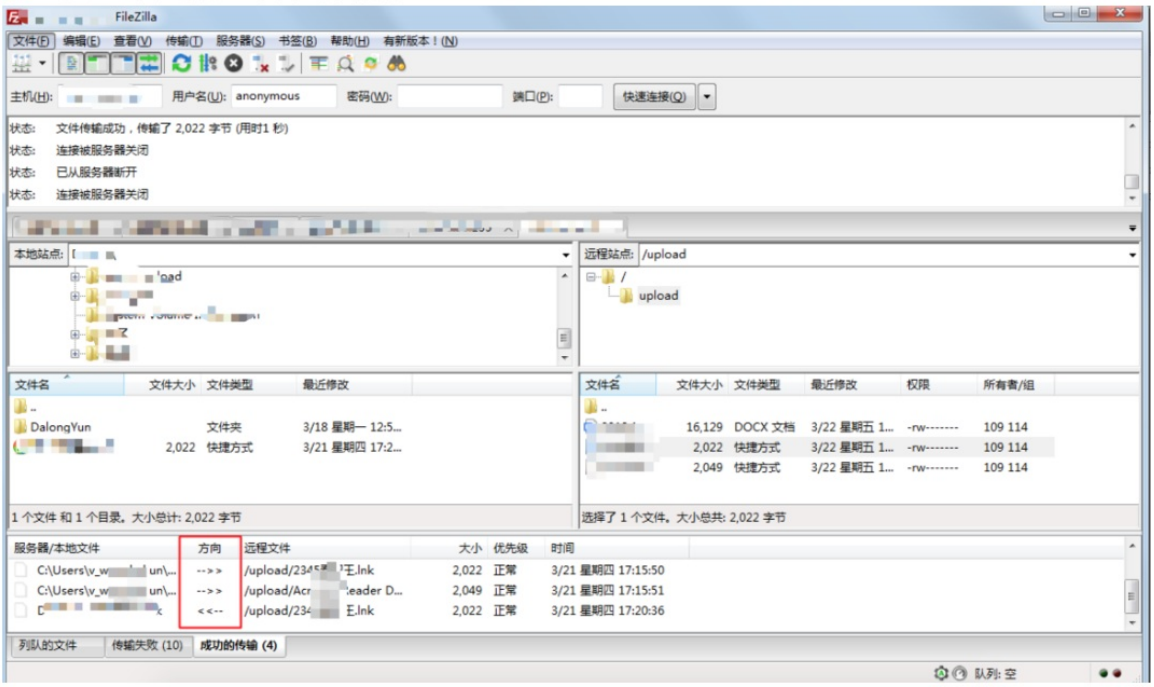
执行命令：sudo chown root:ftp /srv/ftp/upload

之后执行：sudo chmod 755 /srv/ftp/upload

4.重启服务使其生效。

执行命令： sudo service vsftpd restart

5.测试可以正常连接和上传下载文件。



方式二：配置本地用户登录

本地用户登录是指用户使用 Linux 操作系统的账号和密码登录 FTP 服务器。

您可以参考以下步骤配置 vsftpd 服务，以使用Linux账号和密码访问FTP服务器。

1.运行以下命令为FTP服务创建一个Linux用户。本示例中，该用户名为ftpbaidu。

sudo useradd ftpbaidu

```
[root@instance-ie5uf3jl vsftpd]# useradd ftpbaidu
[root@instance-ie5uf3jl vsftpd]#
```

2.运行以下命令修改ftptest 用户的密码。

sudo passwd ftpbaidu

3.运行以下命令创建一个供FTP服务使用的文件目录。

sudo mkdir /srv/ftp/baidu

4.运行以下命令更改/srv/ftp/baidu目录的拥有者为ftpbaidu。

sudo chown -R ftpbaidu:ftpbaidu /srv/ftp/baidu

5.输入命令vim /etc/vsftpd.conf打开vsftpd.conf配置文件并按键i进入编辑模式。

6.修改vsftpd.conf配置文件。

FTP服务器可以配置为主动模式或被动模式。

配置主动模式：主动模式下，客户端向服务端发送数据端口的信息，由服务端主动连接客户端发送的数据端口。配置FTP为主动模式的参数如下：

```
 #禁止匿名登录FTP服务器
anonymous_enable=NO
#允许本地用户登录FTP服务器
local_enable=YES
#设置本地用户登录后所在的目录
local_root=/var/ftp/baidu
#全部用户被限制在主目录
chroot_local_user=YES
#启用例外用户名单
chroot_list_enable=YES
#指定例外用户列表，这些用户不被锁定在主目录
chroot_list_file=/etc/chroot_list#主目录可写权限allow_writeable_chroot=YES
#配置其他参数
allow_writeable_chroot=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

（1）按Esc键退出编辑模式，然后按键:wq保存并退出文件。 （2）运行touch /etc/chroot_list命令创建chroot_list文件，并写入不受只可以访问其主目录限制的例外用户名单。注意：如果没有例外用户也必须要有 chroot_list文件，内容可为空

（3）按键Esc退出编辑模式，然后按键:wq保存并退出文件。

（4）运行以下命令重启vsftpd服务。sudu service vsftpd restart

以上为主动模式配置方法

配置被动模式：被动模式下，服务端开启并发送数据端口的信息给客户端，由客户端连接服务端开启的数据端口，服务端被动接受连接。在被动模式下，您需要配置服务端可以开启的数据端口范围。配置FTP为被动模式的参数如下：

```
 #禁止匿名登录FTP服务器
anonymous_enable=NO
#允许本地用户登录FTP服务器
local_enable=YES
#设置本地用户登录后所在目录
local_root=/var/ftp/baidu
#全部用户被限制在主目录
chroot_local_user=YES
#启用例外用户名单
chroot_list_enable=YES
#指定例外用户列表，这些用户不被锁定在主目录
chroot_list_file=/etc/chroot_list
#开启被动模式
pasv_enable=YES
#FTP服务器公网IP
pasv_address=<FTP服务器公网IP>
#设置被动模式下，建立数据传输可使用port范围的最小值
pasv_min_port=50000
#设置被动模式下，建立数据传输可使用port范围的最大值pasv_max_port=50010#主目录可写权限
allow_writeable_chroot=YES
pasv_max_port=50010
#配置其他参数
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
xferlog_std_format=YES
tcp_wrappers=YES
allow_writeable_chroot=YES
listen=YES
listen_ipv6=NO
pam_service_name=vsftpd
userlist_enable=YES
```

注意：建议您把端口范围设在比较高的一段范围内，比如50000-50010，有助于提高访问FTP服务器的安全性。

（1）按Esc键退出编辑模式，然后按键:wq保存并退出文件。

（2）运行touch /etc/chroot_list命令创建chroot_list文件，并写入不受只可以访问其主目录限制的例外用户名单。注意：如果没有例外用户也必须要有 chroot_list文件，内容可为空

（3）按键Esc退出编辑模式，然后按键:wq保存并退出文件。

（4）运行以下命令重启vsftpd服务。sudo service vsftpd restart

以上为被动模式配置方法

vsftpd.conf 的配置文件参数说明

运行命令cat /etc/vsftpd.conf查看配置文件内容。

用户登录控制：

参数	说明
anonymous_enable=YES	接受匿名用户
no_anon_password=YES	匿名用户login时不询问口令
anon_root=(none)	匿名用户主目录
local_enable=YES	接受本地用户
local_root=(none)	本地用户主目录

用户权限控制：

参数	说明
write_enable=YES	可以上传(全局控制)
local_umask=022	本地用户上传文件的umask
file_open_mode=0666	上传文件的权限配合umask使用
anon_upload_enable=NO	匿名用户可以上传
anon_mkdir_write_enable=NO	匿名用户可以建目录
anon_other_write_enable=NO	匿名用户修改删除
chown_username=lightwiter	匿名上传文件所属用户名

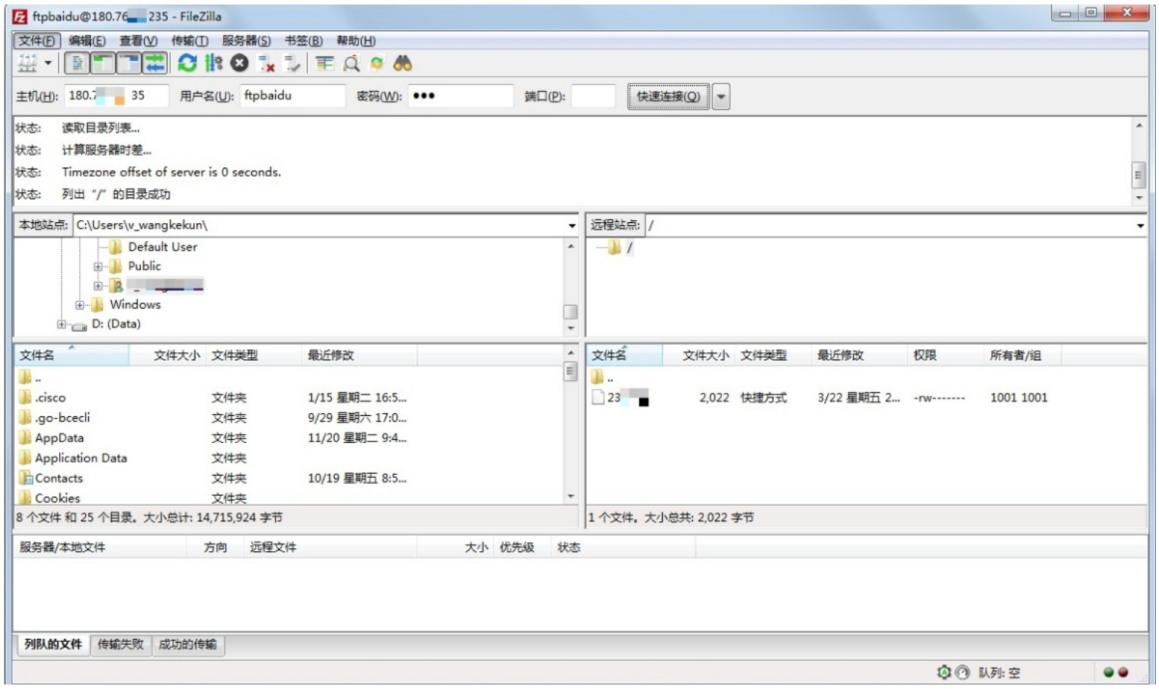
步骤三： 设置安全组

搭建好 FTP 站点后，您需要在实例安全组的入方向添加放行下列 FTP 端口的规则。

- FTP 为主动模式时：端口21。
- FTP 为被动模式时：端口21，以及配置文件/etc/vsftpd.conf中参数pasv_min_port和pasv_max_port之间的所有端口。

步骤四： 客户端测试

您可以通过 FTP 客户端或浏览器访问 FTP 服务器进行测试。本教程以filezilla工具为例：



windows搭建IIS和FTP教程

windows2012搭建IIS和FTP教程

使用Windows Server 2012 R2操作系统的Windows 实例搭建 FTP 站点具体操作步骤如下：

- 步骤一： 添加IIS以及FTP服务角色。
- 步骤二： 创建FTP用户名及密码。
- 步骤三： 设置共享文件的权限。
- 步骤四： 添加及设置FTP站点。
- 步骤五： 设置安全组及防火墙。
- 步骤六： 客户端测试。

添加IIS以及FTP服务角色

在创建 FTP 站点前，首先需要安装 IIS 及 FTP 服务。

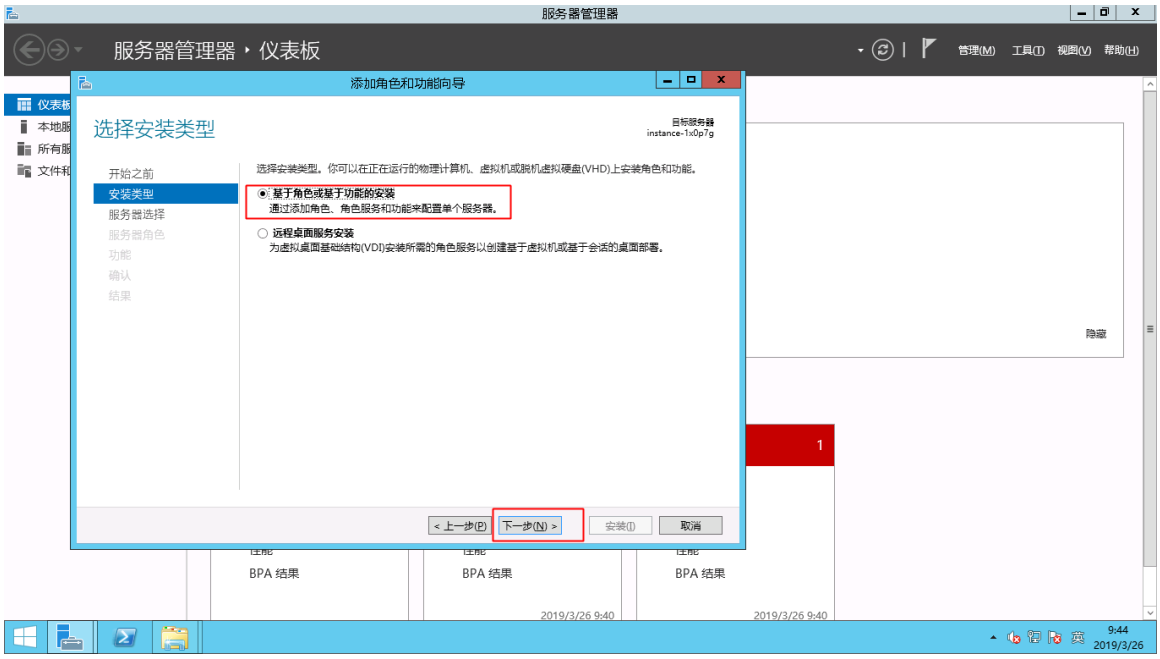
1.远程连接 并登录到 Windows 实例。

2.选择 **开始** > **服务管理器** > **仪表盘**。

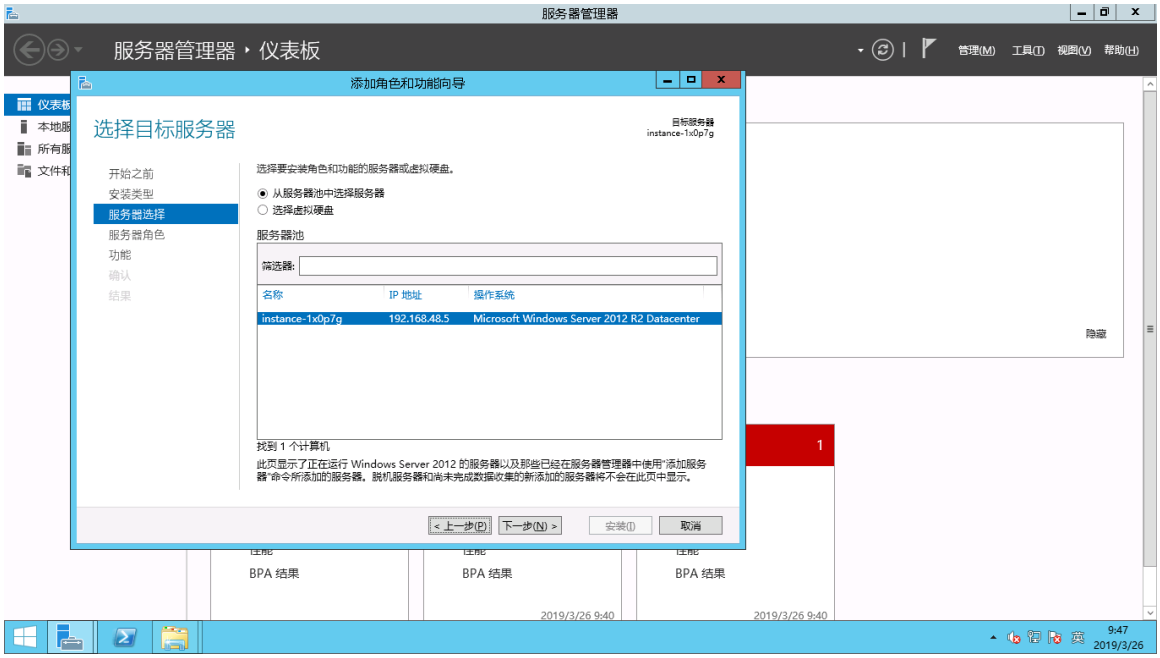
3.单击 **添加角色和功能**。

4.在弹出的对话框中，选择 **下一步**。

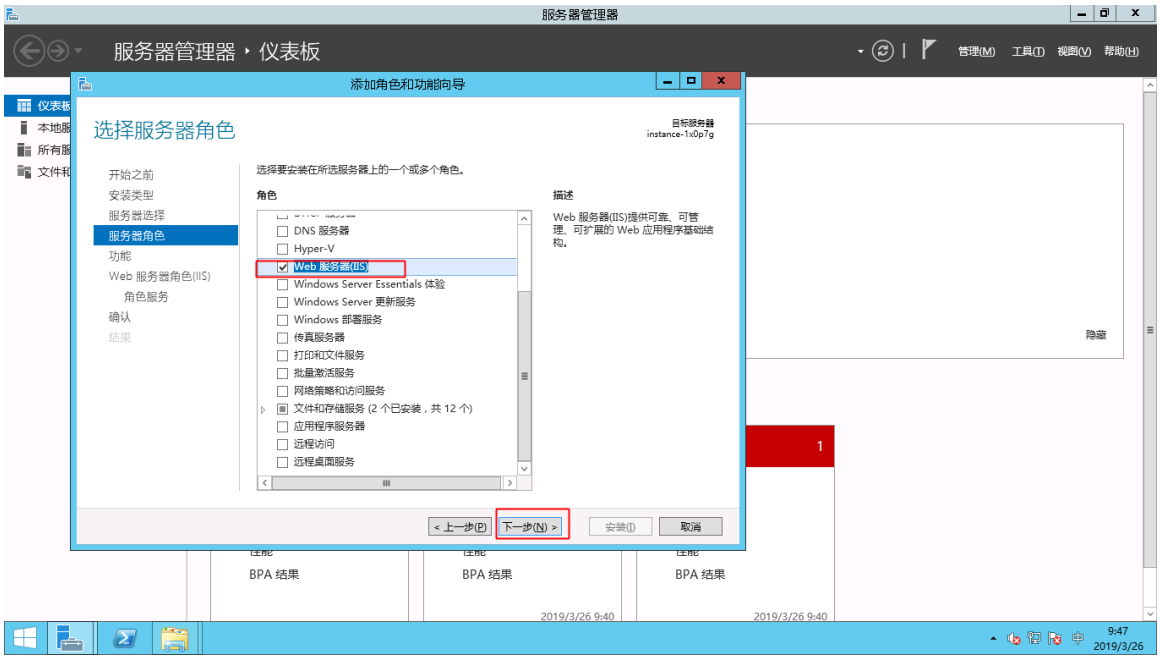
5.选择“基于角色或基于功能的安装”点击**下一步**。



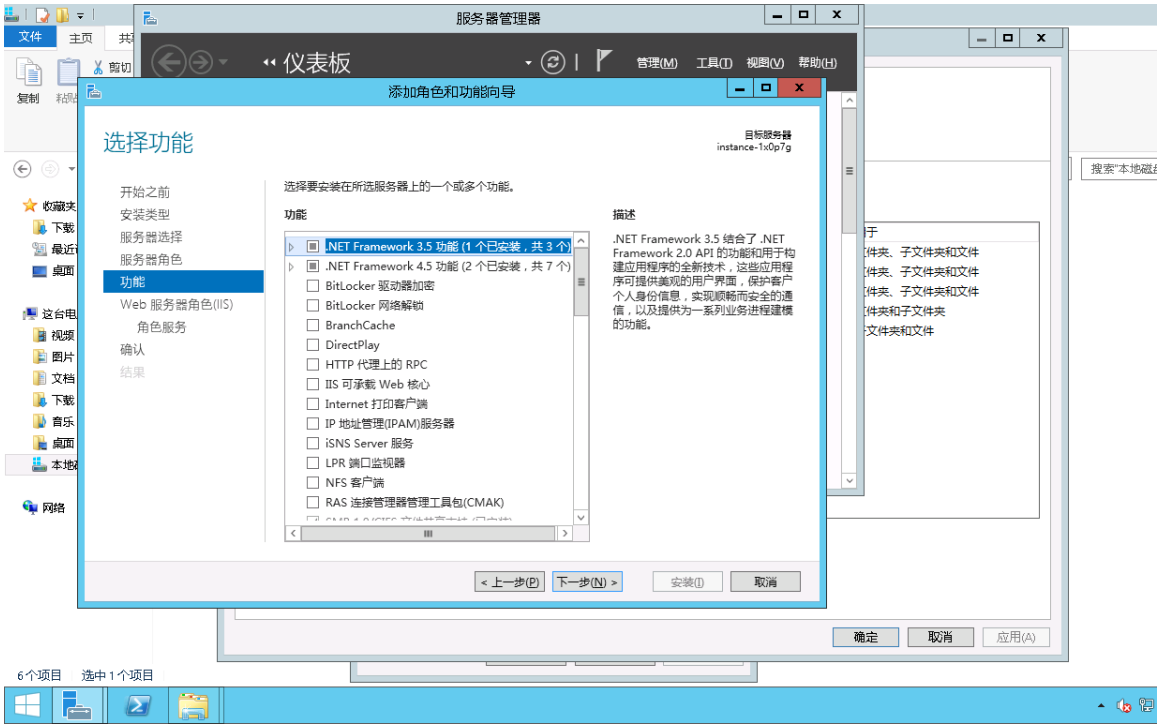
6.服务器选择界面无需操作，直接**下一步**。



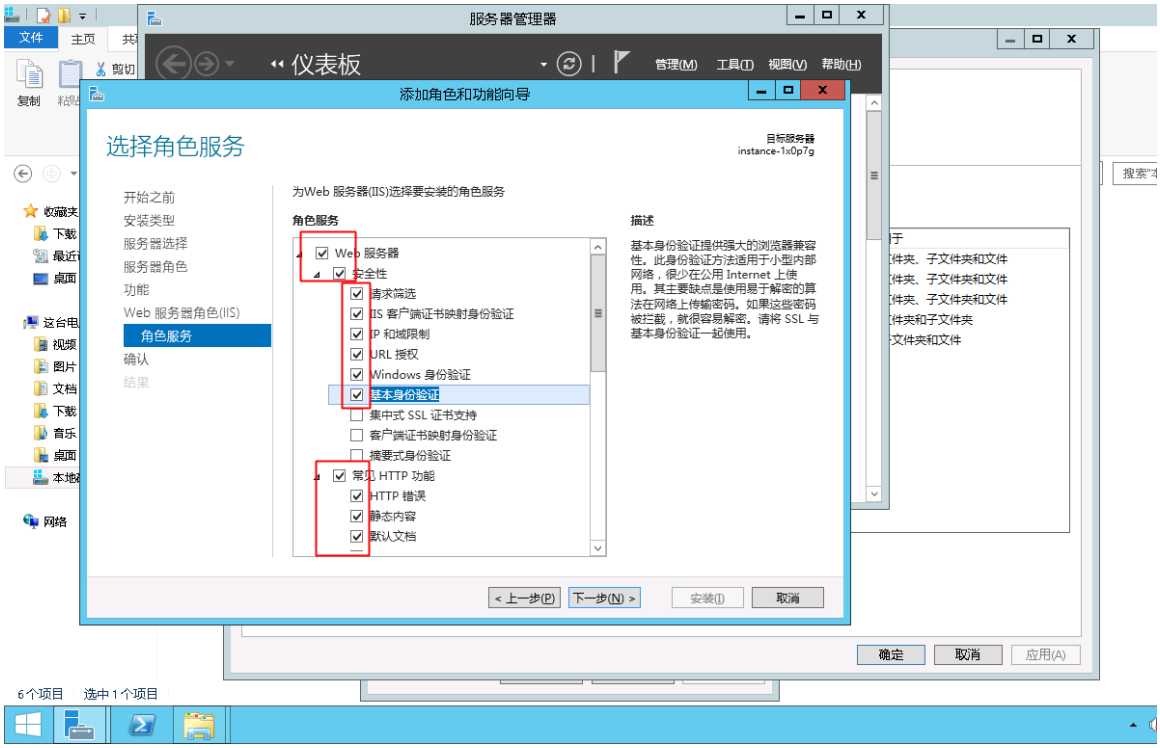
7.进入 添加角色和功能向导，选择 “**Web服务器(IIS)**”。



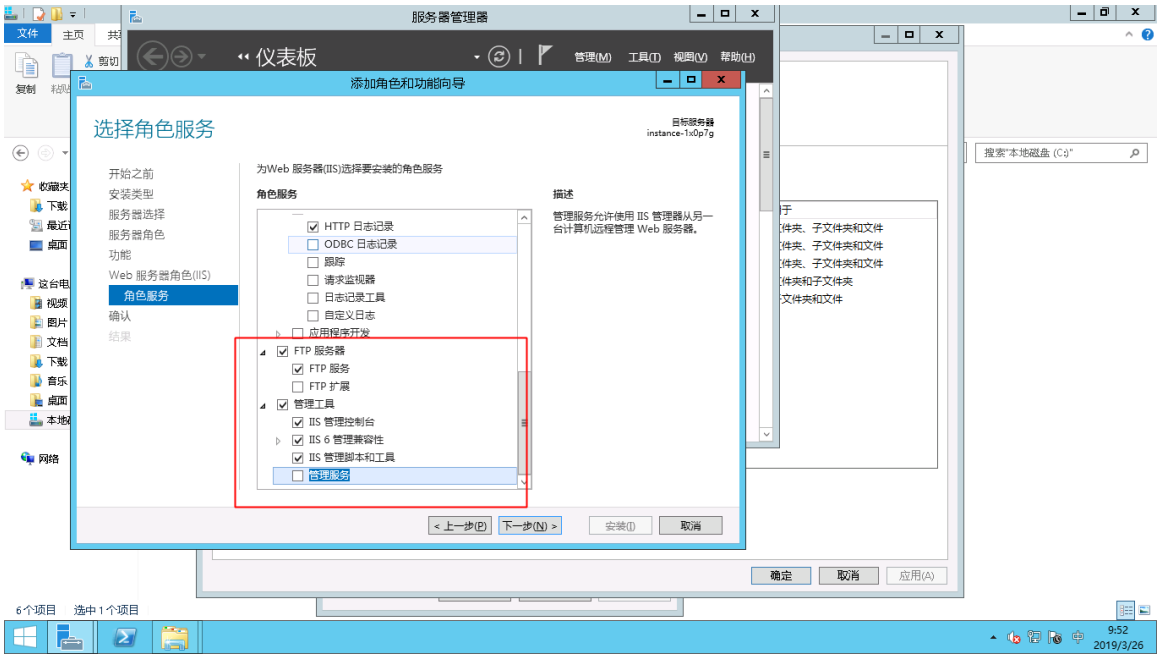
8.功能界面单击**下一步**。



9.安装**角色服务**，选择对应的安全性选项，如图。



10.选择“**FTP服务器**”，以及开设主机必须要用的“**管理工具**”。



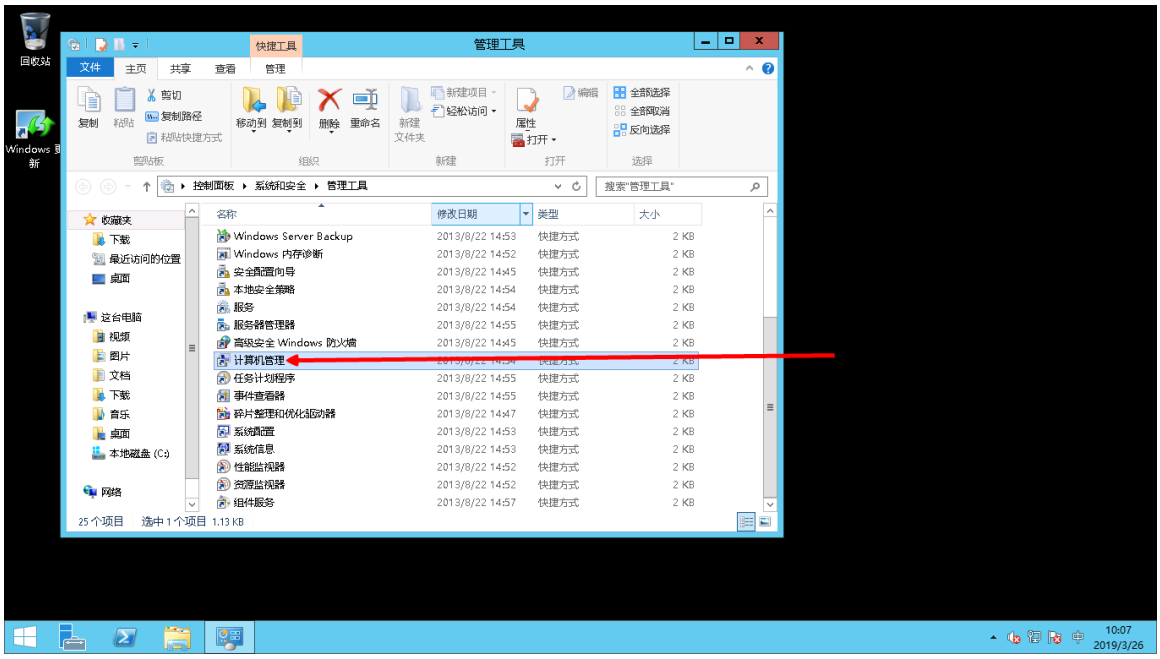
11.点击安装即可，时间较长，请耐心等待。

上述步骤完成后，IIS与FTP即搭建完毕。

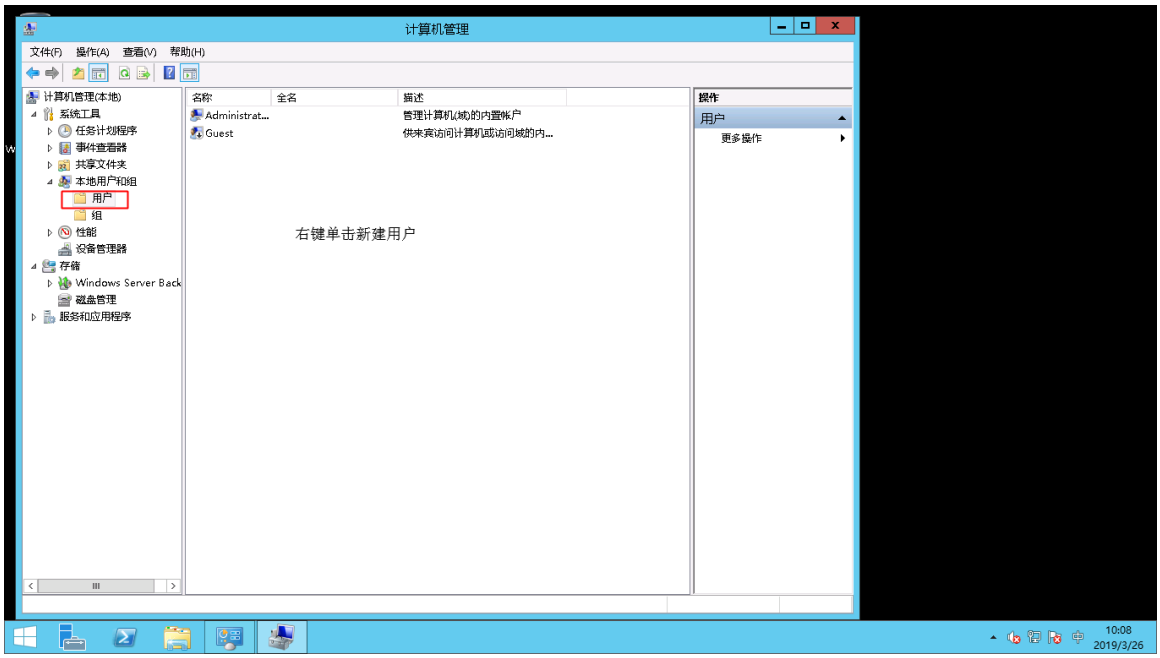
创建FTP用户名及密码

创建 Windows 用户名和密码，用于 FTP 使用。（默认可以匿名登录，此步骤可以不操作，此步骤为设置FTP需要用账号密码登录）

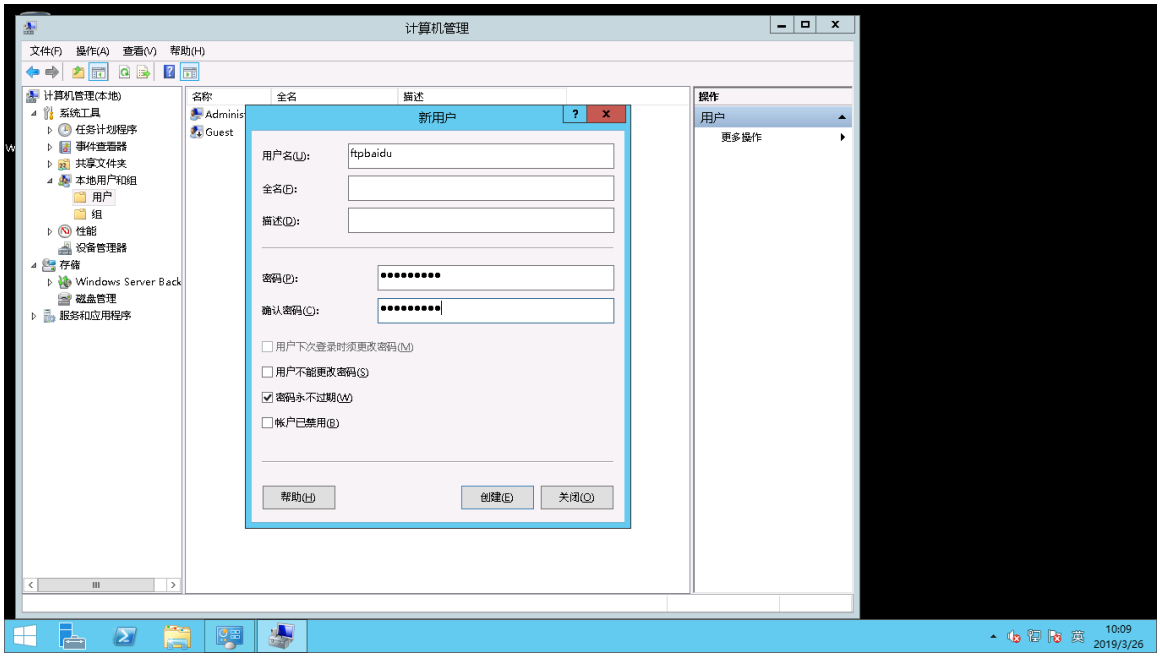
1.选择 **开始** > **管理工具** > **计算机管理**。

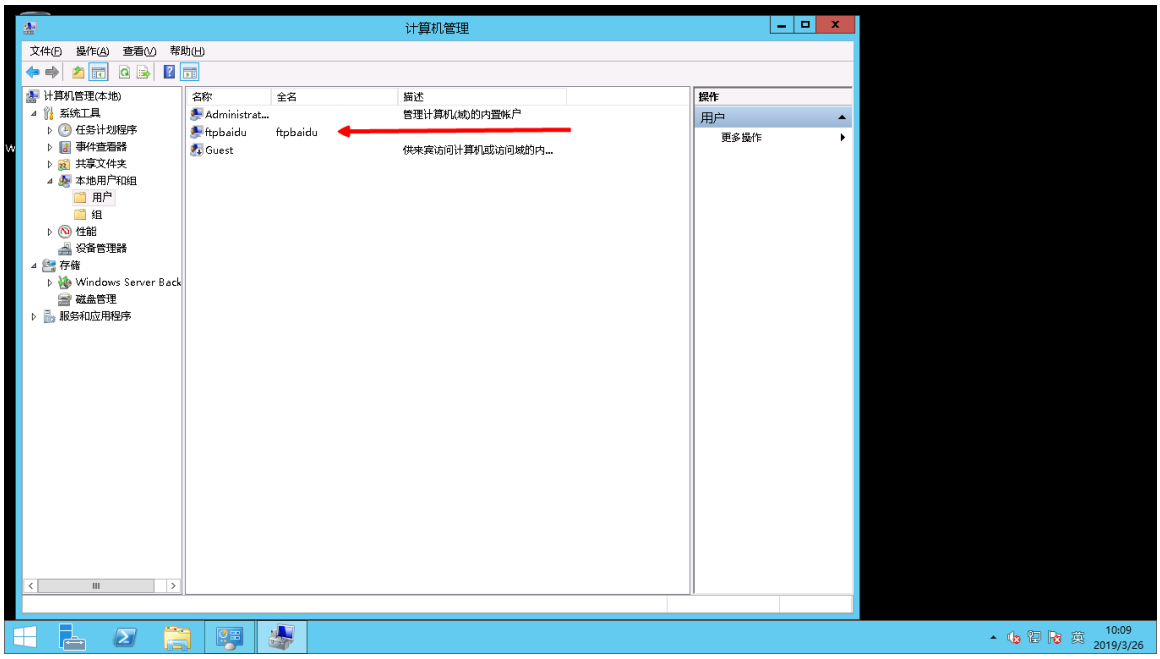


2.左侧菜单栏点击 **本地用户和组** > **用户**，并在右侧空白处单击右键，再选择 **添加用户**。本文例子中用户名使用 ftpbaidu。



注意： 密码必须包括大写字母、小写字母和数字。否则会显示无法通过密码策略。

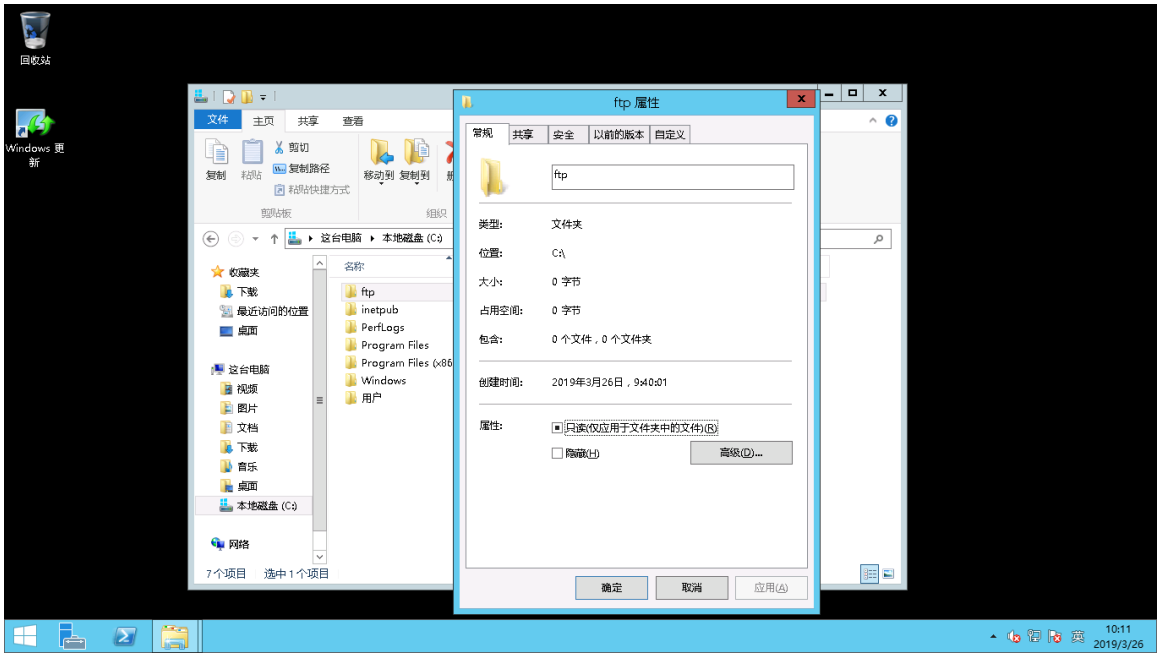




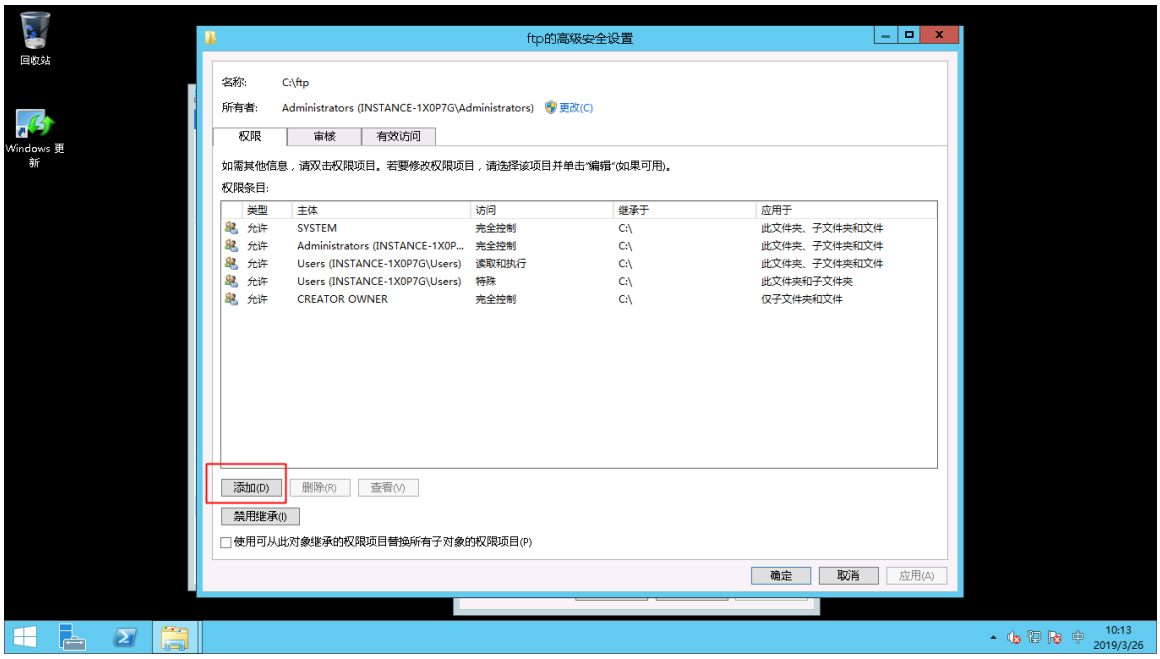
设置共享文件的权限

您需要为在 FTP 站点共享给用户的文件夹设置访问以及修改等权限。

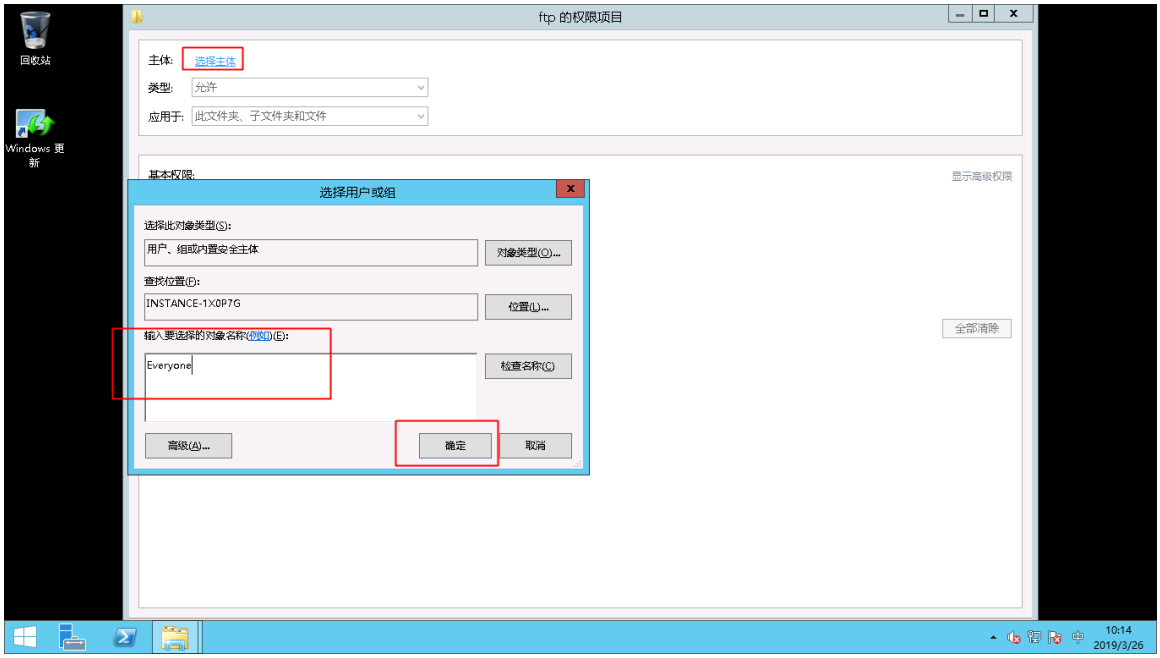
1.在服务器磁盘上创建一个供 FTP 使用的文件夹，右键单击文件夹，选择 **属性**。



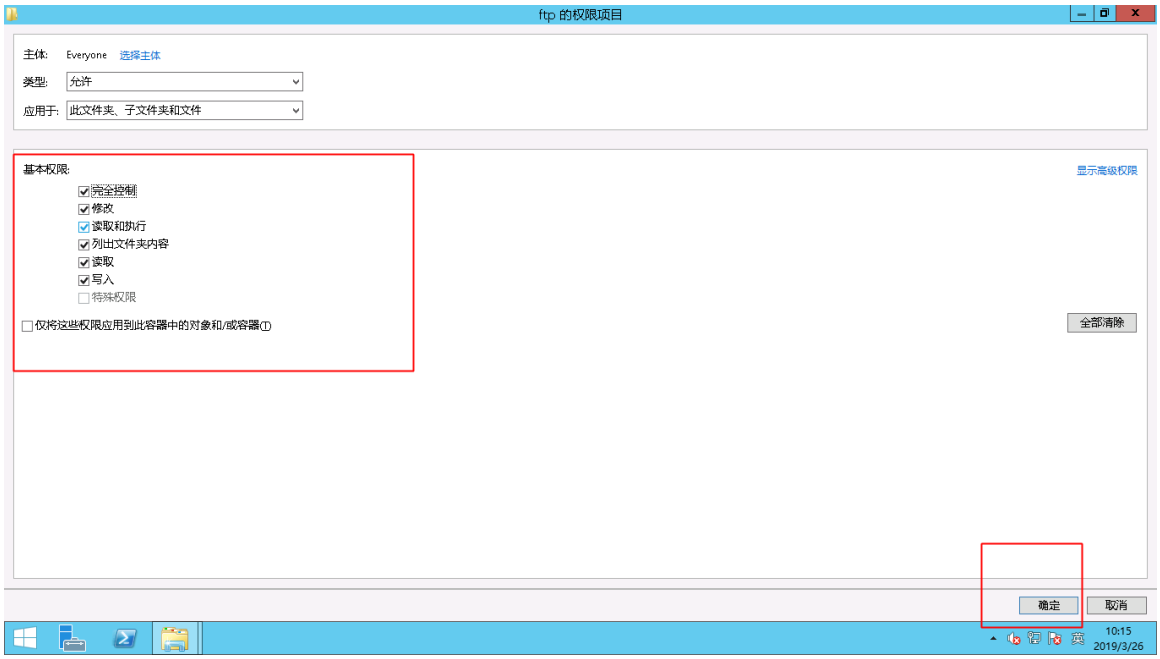
2.单击 **安全**，选择 **高级** - **更改权限** - **添加**。



3.点击选择主体，之后在框中输入**Everyone**点击**确定**。



4.按照您的需求选择适合的权限。如点击**完全控制**，之后点击**确定**，点击后所有的选项都选择**确定**。

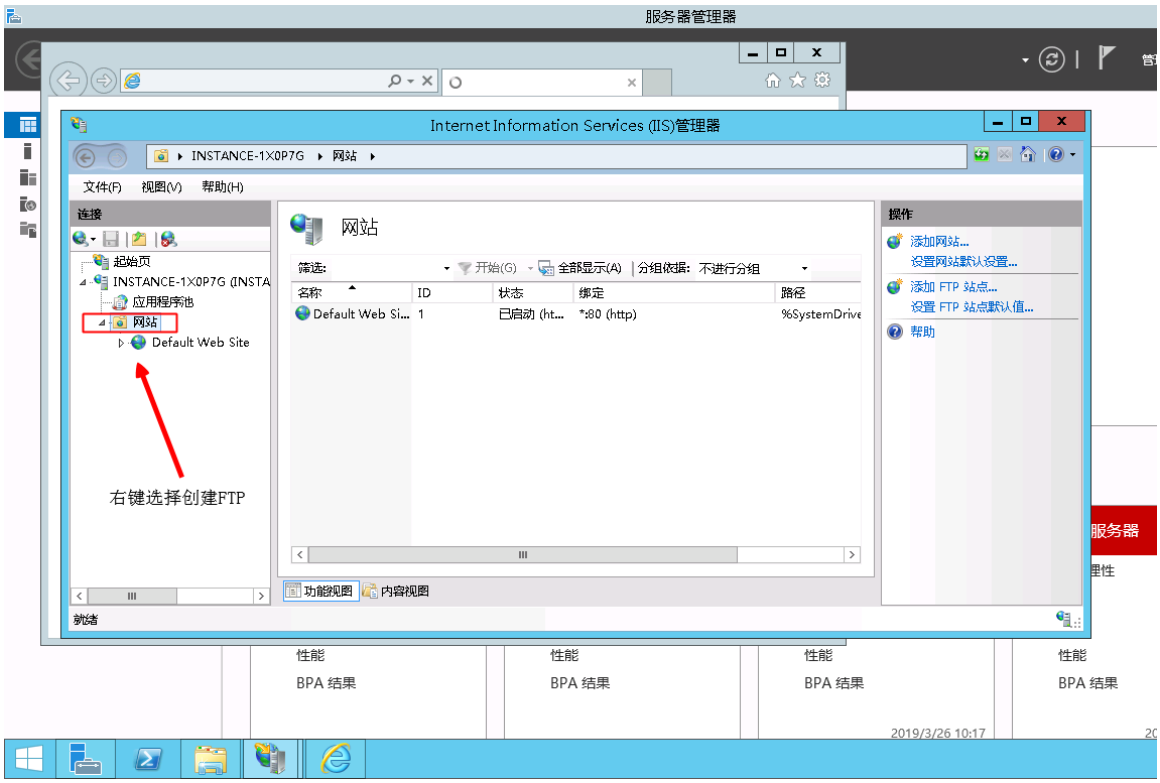


添加及设置 FTP 站点

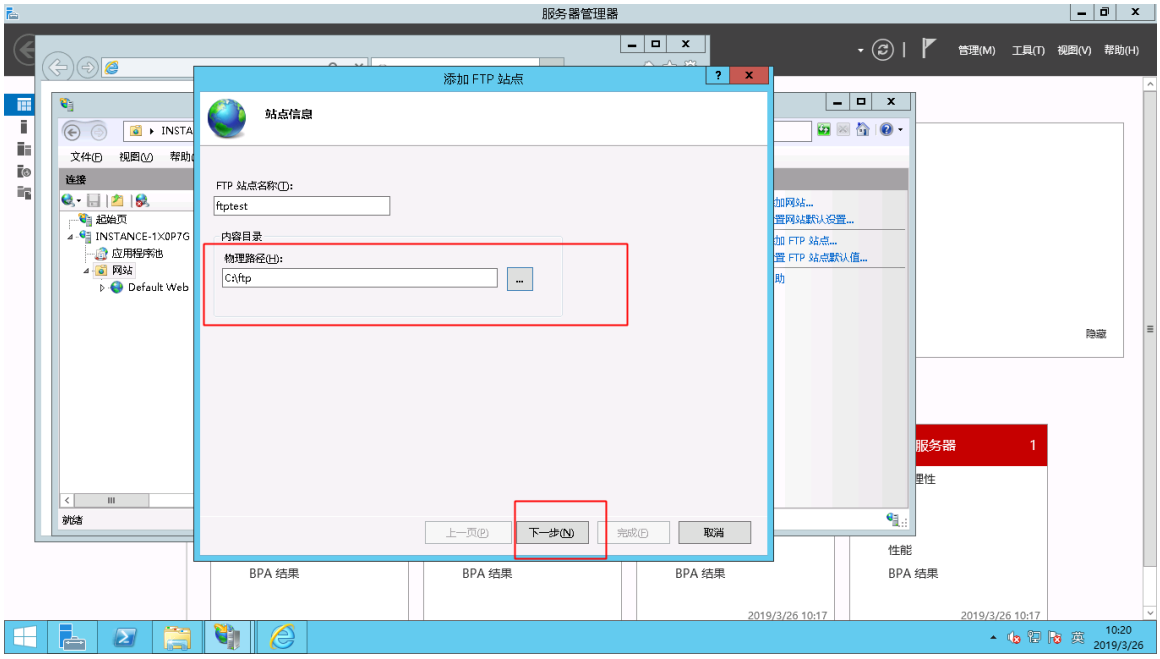
安装 FTP，设置好共享文件夹权限后，您需要创建 FTP 站点。

1.选择 **开始** > **服务器管理器** > **工具** > **Internet information Service（IIS）管理器**。

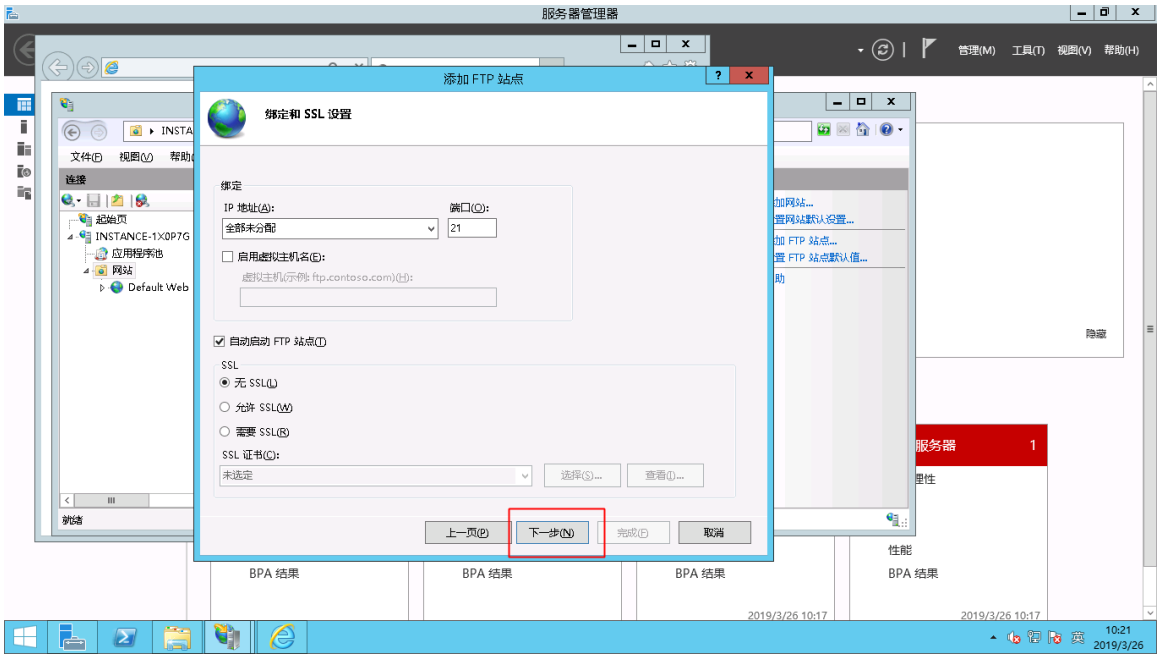
2.右键单击 **网站**，选择 **添加 FTP 站点**。



3.在弹出的窗口，填写 FTP 站点名称与共享文件夹的物理路径（之前创建的FTP的文件夹的路径），然后单击 **下一步**。

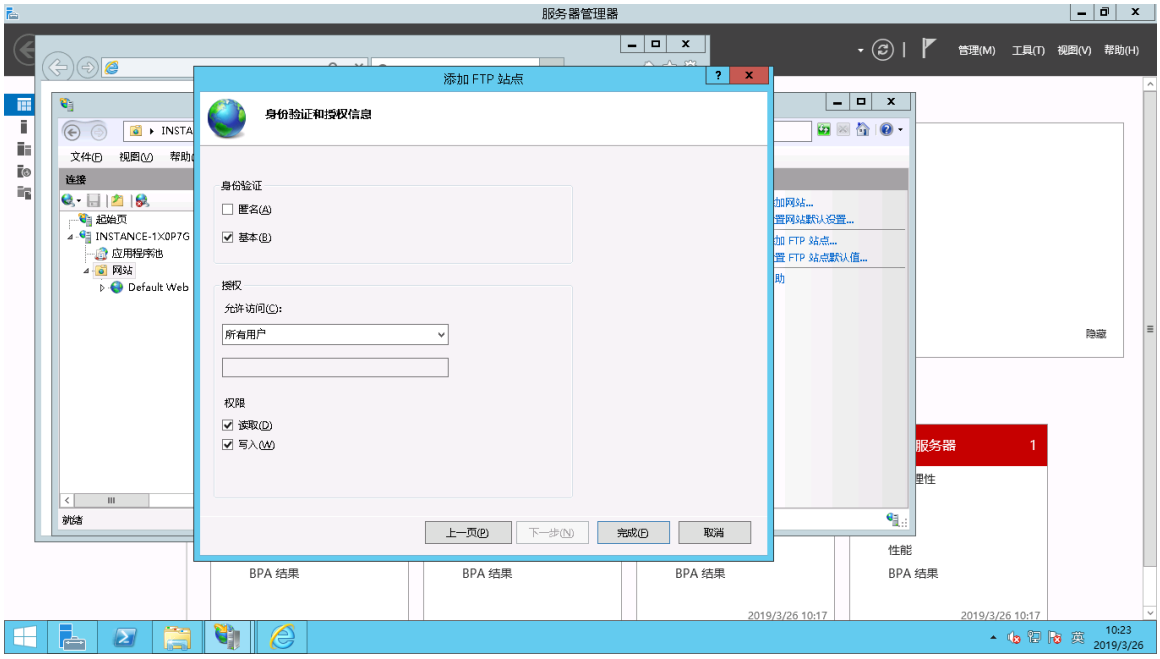


4.IP 地址默认选择 **全部未分配**。端口号可自行设置，FTP 默认端口号为 **21**。 5.选择 **SSL 设置**。您可以根据您的需求选择对应的设置选项。

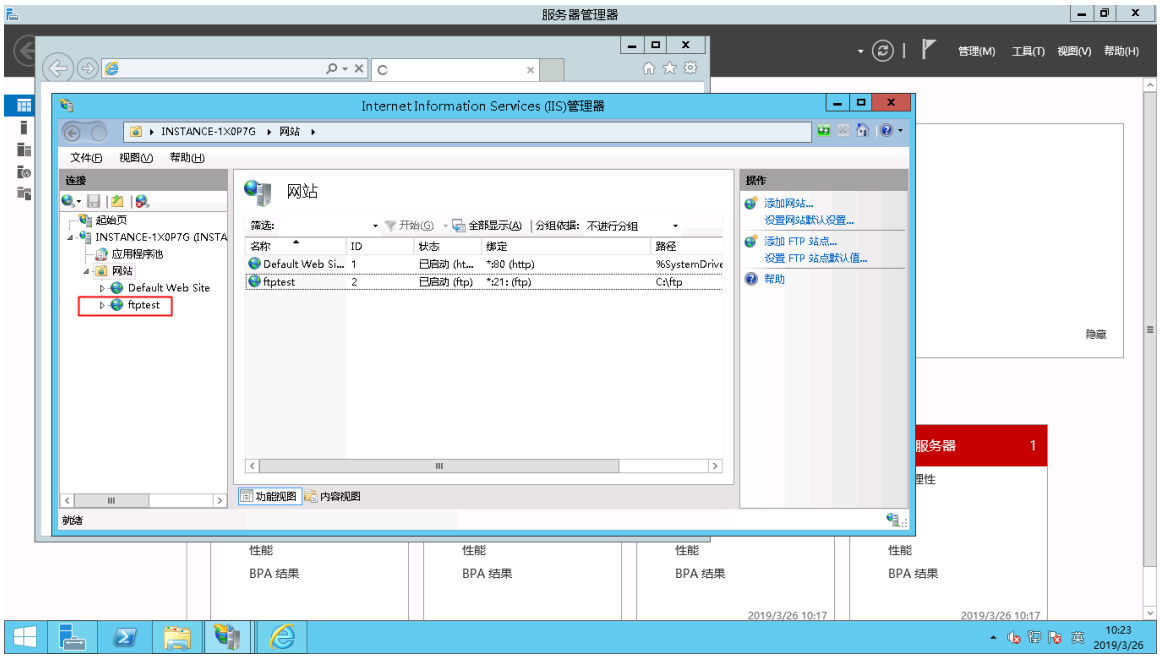


- 允许：允许 FTP 服务器支持与客户端的非 SSL 和 SSL 连接。
- 需要：需要对 FTP 服务器和客户端之间的通信进行 SSL 加密。
- 无： 不需要 SSL 加密选择 无。

  6.选择要使用的一种或多种身份验证方法。
- 匿名：允许任何仅提供用户名 anonymous 或 ftp 的用户访问内容。
- 基本：需要用户提供有效用户名和密码才能访问内容。由于基本身份验证通过网络传输未加密的密码，因此请仅在清楚客户端和 FTP 服务器之间的连接是安全的情况下（例如，使用安全套接字层 (SSL) 时）使用此身份验证方法。



上述步骤完成后，您即可以看到搭建的 FTP 站点。



设置安全组及防火墙

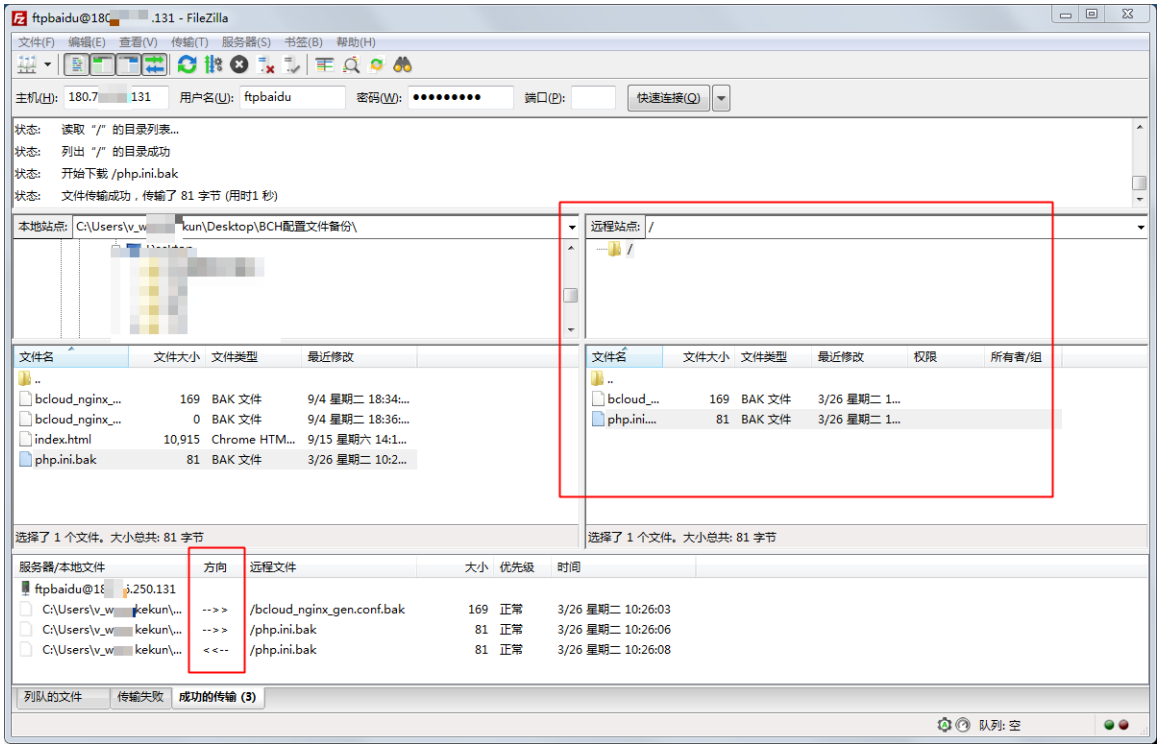
搭建好 FTP 站点后，您需要在实例安全组的入站方向添加一条放行 FTP 端口的规则，具体步骤参见 添加安全组规则

服务器防火墙需要设置放行 TCP 21 端口用于 FTP 服务。（可自行百度防火墙放行端口教程，或可以暂时关闭防火墙）

其他防火墙设置参见 微软官方文档。

客户端测试

这边使用FTP工具filezilla进行测试可正常连接和上传下载文件。



windows2008搭建IIS和FTP教程

本文介绍了如何使用 Windows 实例搭建 FTP 站点（以 Windows Server 2008 R2 为例）。Windows 实例搭建 FTP 站点具体操作步骤如下：

- 步骤一： 添加IIS以及FTP服务角色
- 步骤二： 创建FTP用户名及密码
- 步骤三： 设置共享文件的权限
- 步骤四： 添加及设置FTP站点

- 步骤五： 设置安全组及防火墙
- 步骤六： 客户端测试

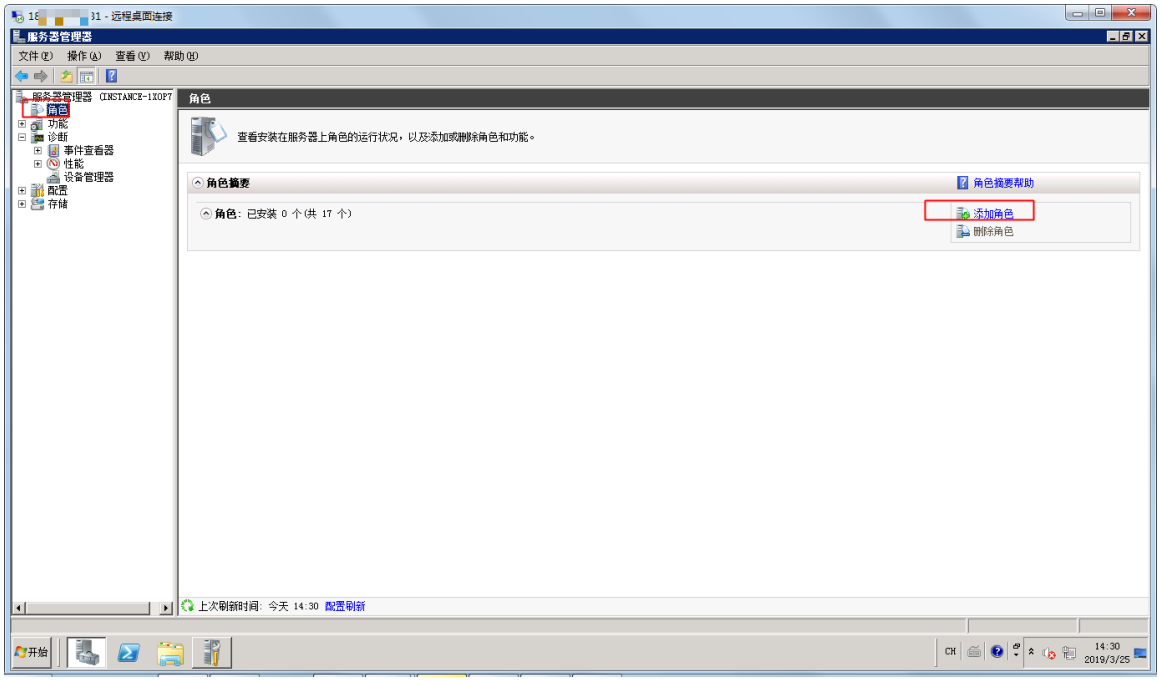
添加IIS以及FTP服务角色

在创建 FTP 站点前，首先需要安装 IIS 及 FTP 服务。

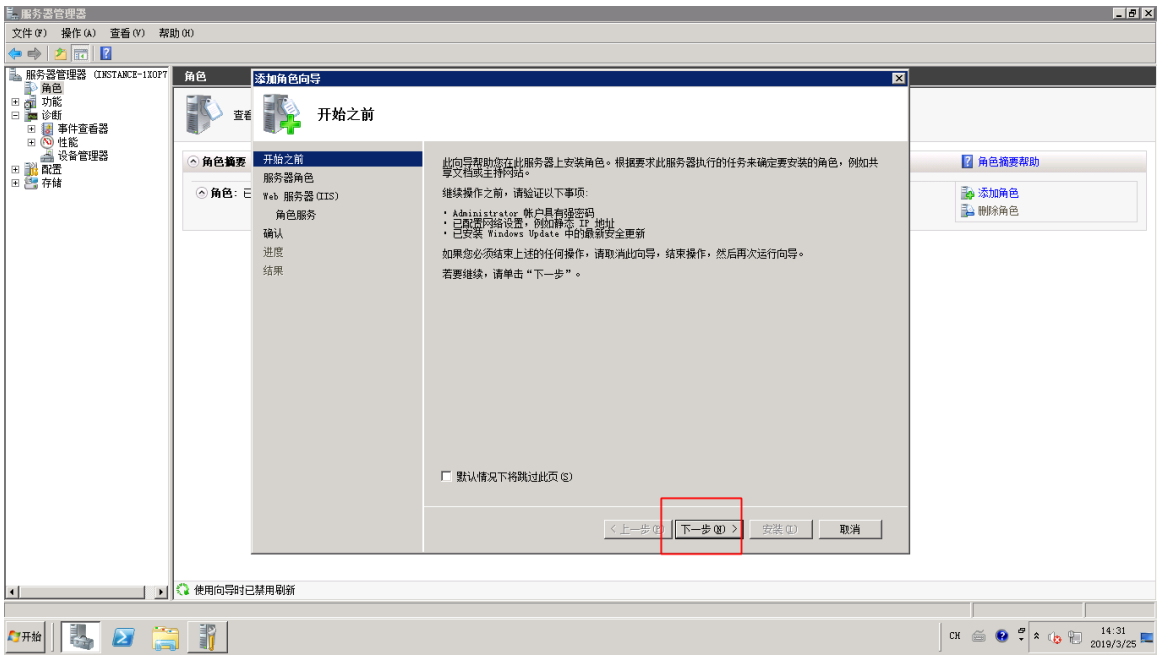
1.远程连接 并登录到 Windows 实例。

2.选择 **开始** > **所有程序** > **管理工具** > **服务管理器**。

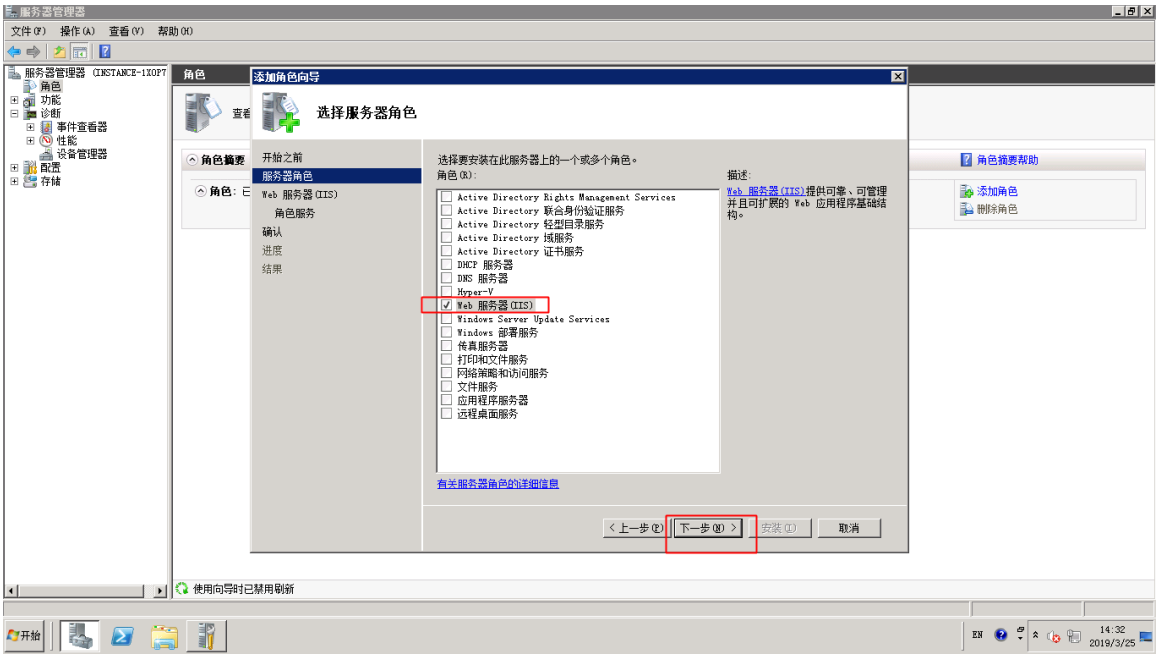
3.单击 **角色**，然后单击 **添加角色**。



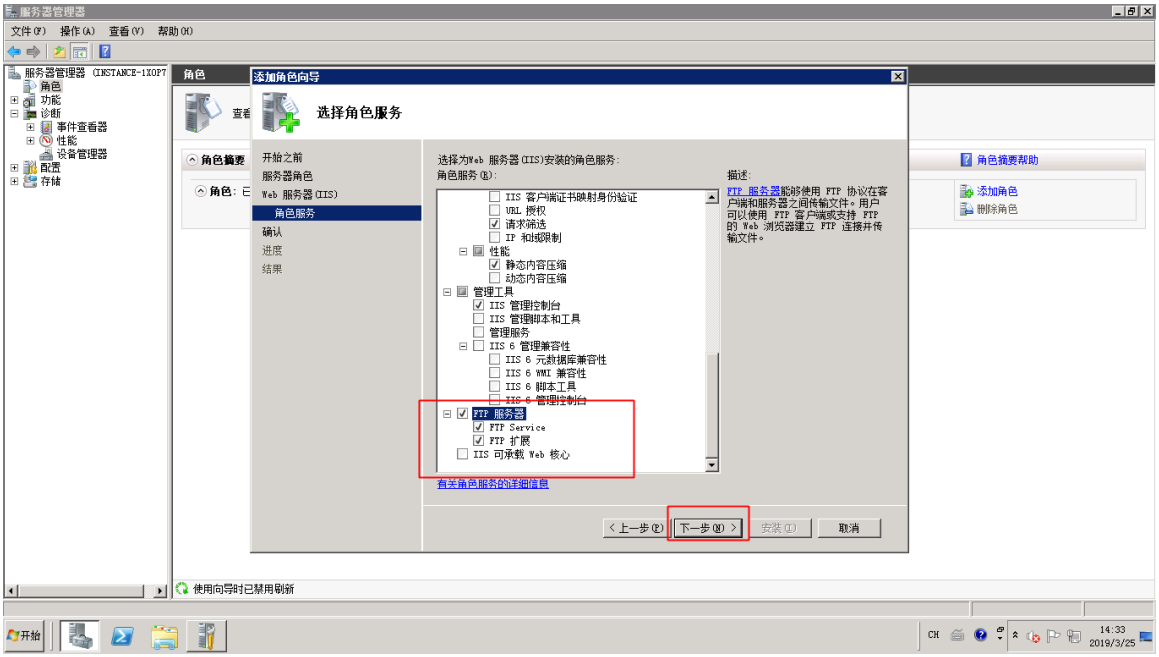
4.在弹出的对话框中，选择 **下一步**。

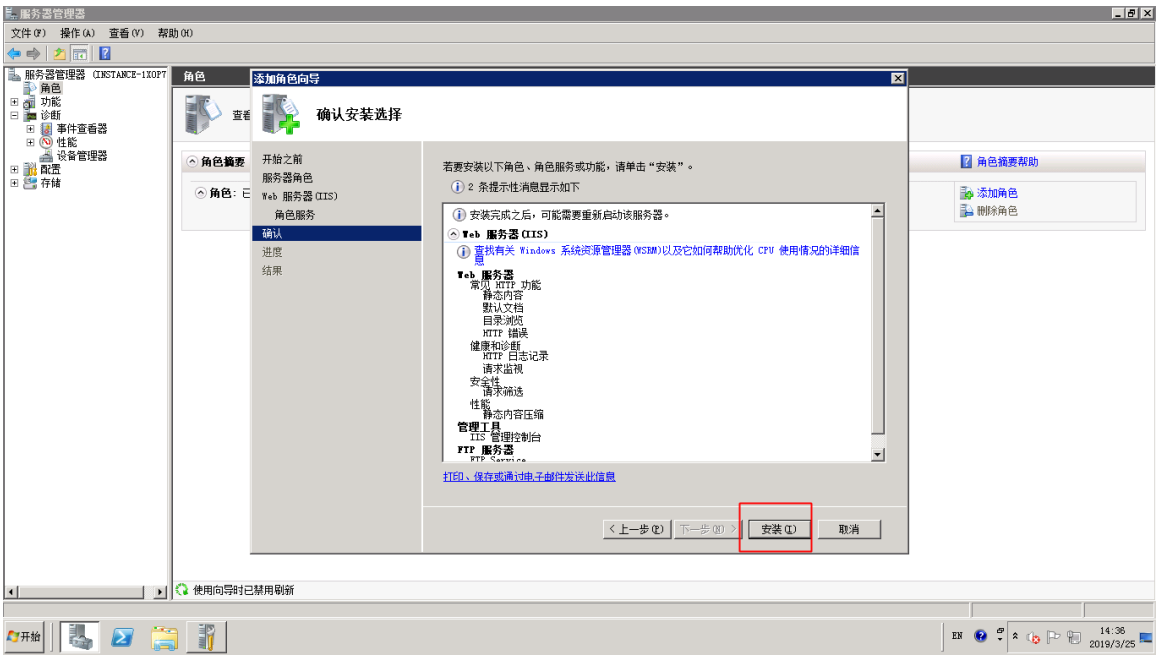


5.选择 **Web 服务器（IIS）**，然后单击 **下一步**。



6.选择 **IIS管理控制台** 以及**FTP 服务器**，选择 **下一步**，单击 **安装**。





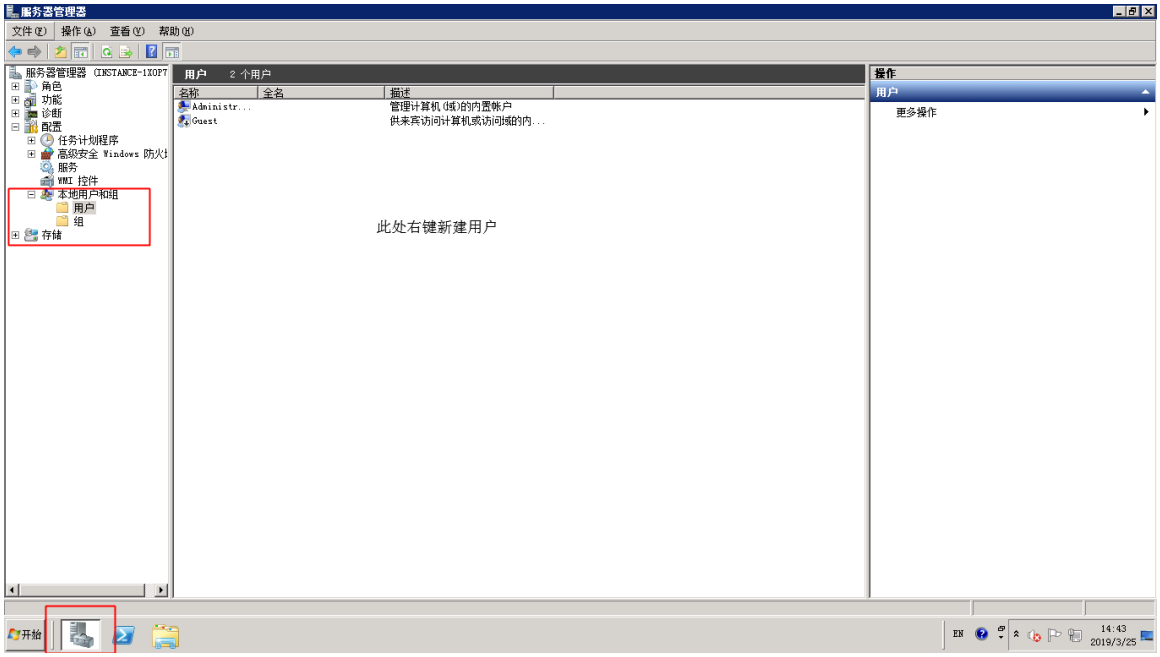
上述步骤完成后，IIS与FTP即搭建完毕。

创建FTP用户名及密码

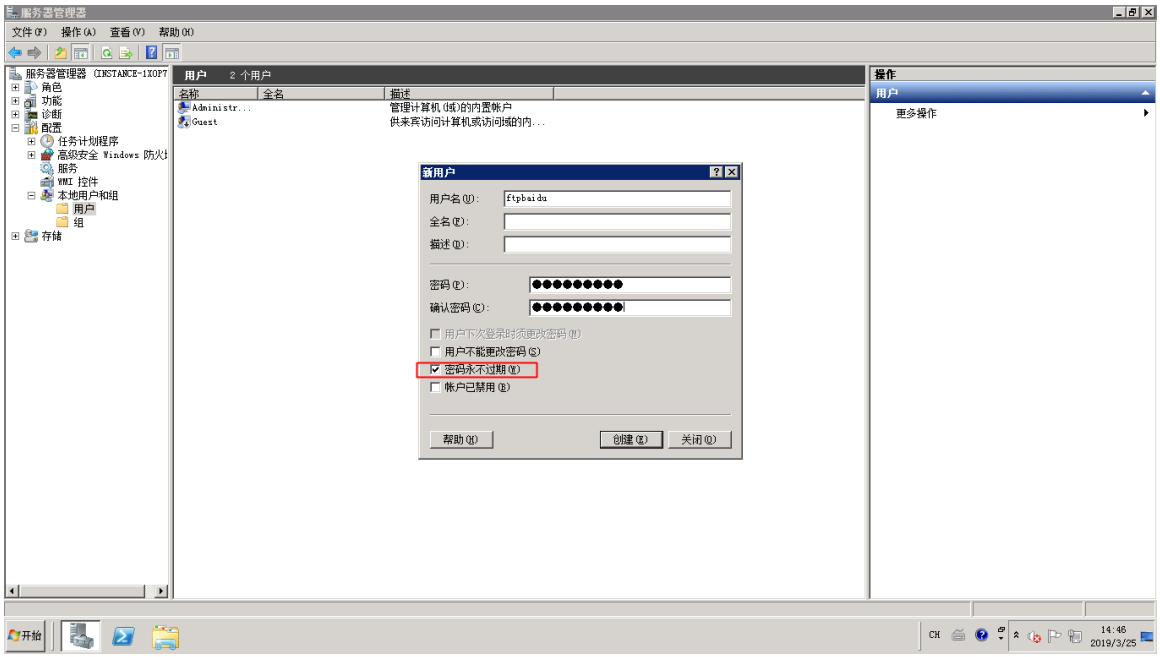
创建 Windows 用户名和密码，用于 FTP 使用。（默认可以匿名登录，此步骤可以不操作，此步骤为设置FTP需要用账号密码登录）

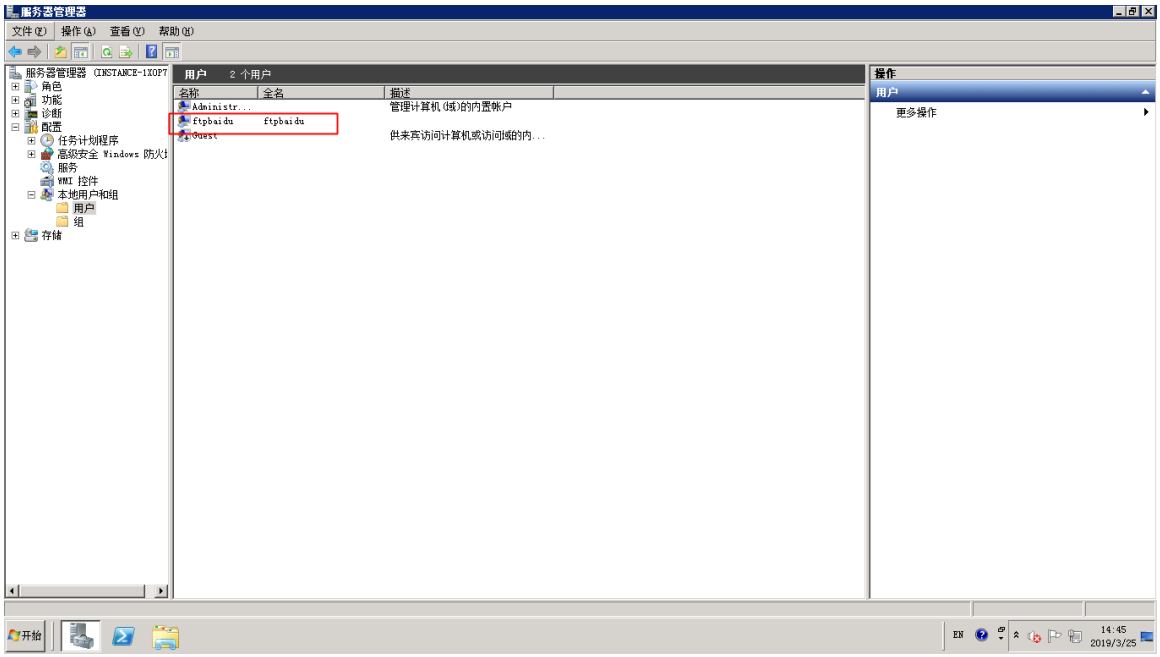
1.选择 **开始** > **管理工具** > **服务器管理器**。

2.单击 **配置** > **本地用户和组** > **用户**，并在右侧空白处单击右键，再选择 **添加用户**，本文例子中 用户名 使用 ftpbaidu。



注意： 密码必须包括大写字母、小写字母和数字。否则会显示无法通过密码策略。

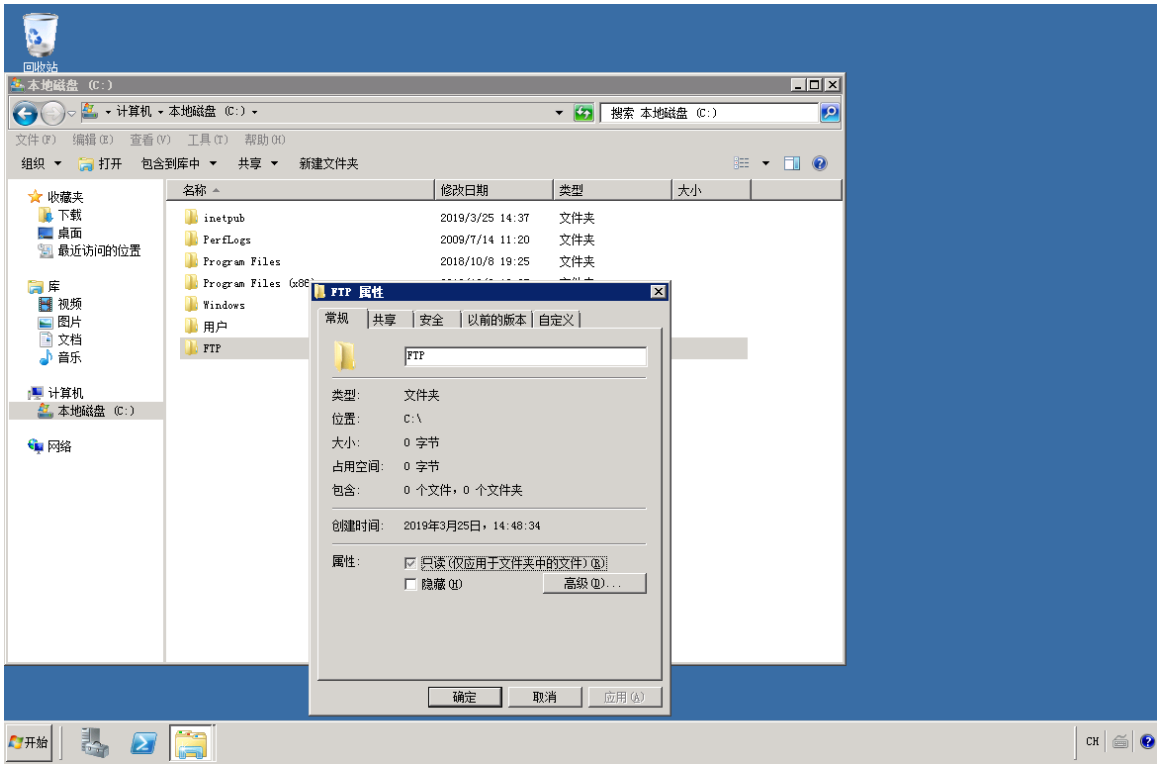




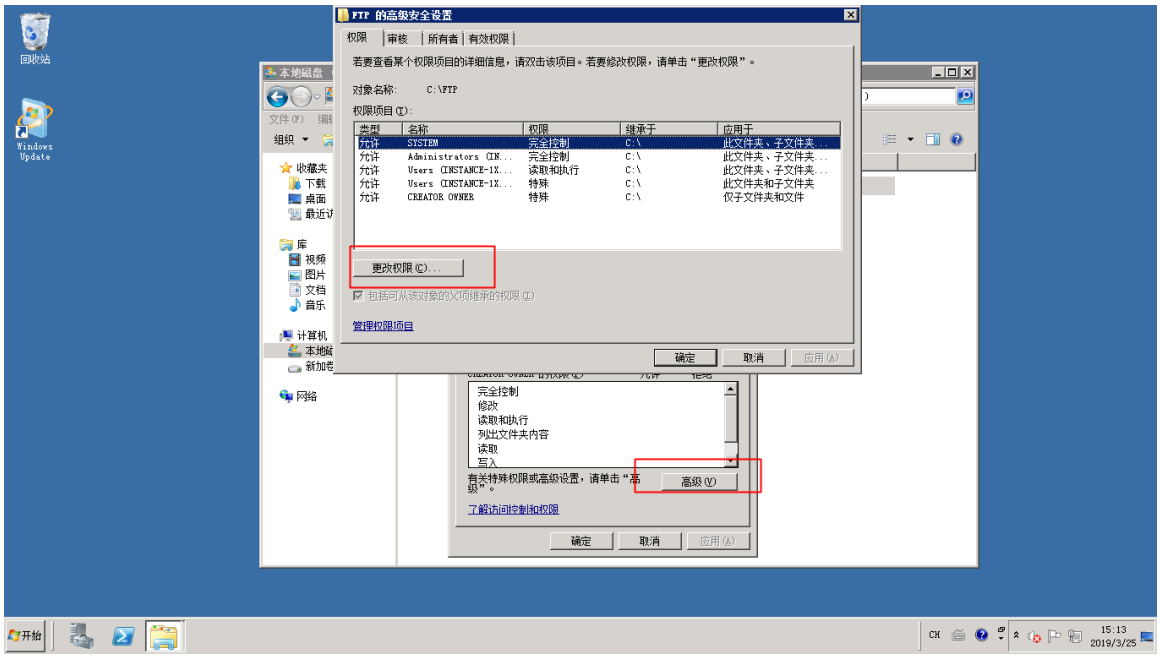
设置共享文件的权限

您需要为在 FTP 站点共享给用户的文件夹设置访问以及修改等权限。

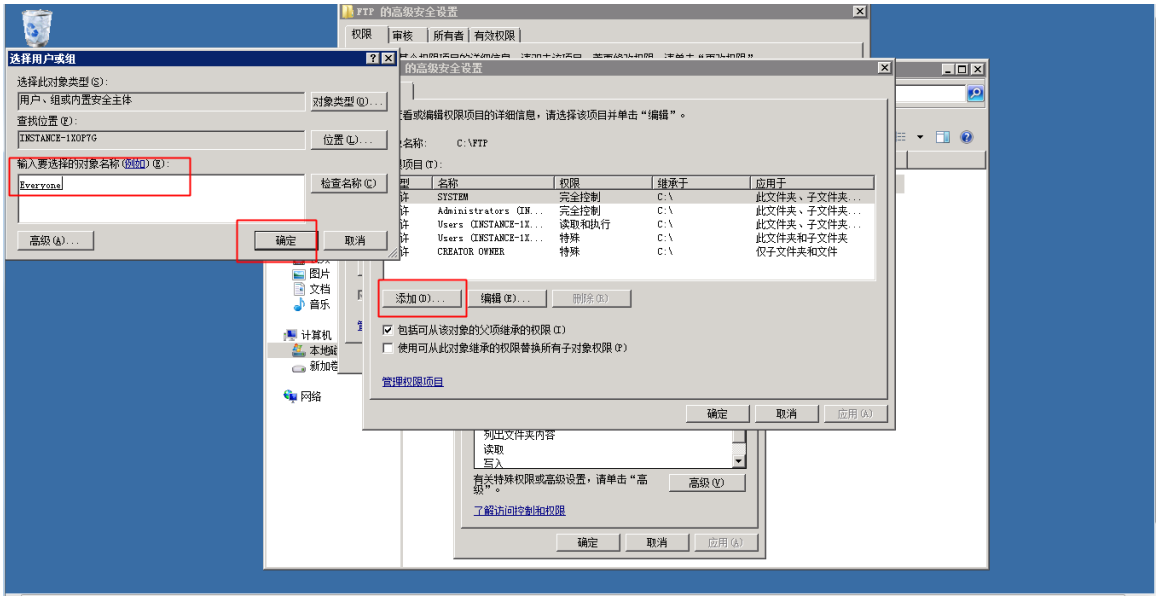
1.在服务器磁盘上创建一个供 FTP 使用的文件夹，右键单击文件夹，选择 **属性**。



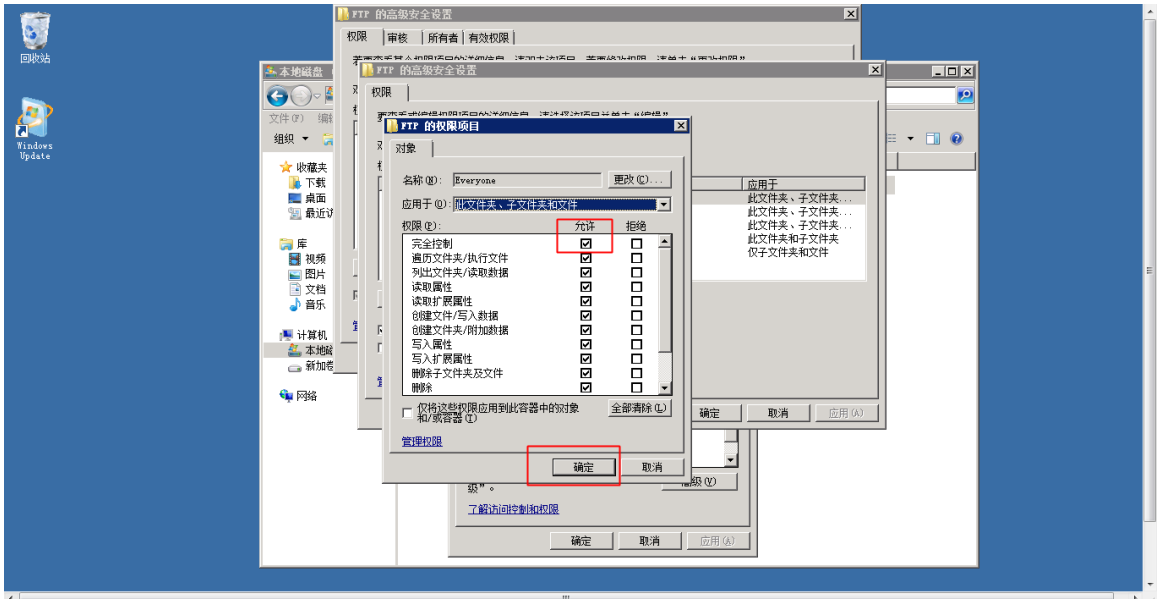
2.单击**安全**，选择**高级-更改权限**



3.选择**添加**，输入Everyone，点击**确定**。



4.按照您的需求配置权限。如点击**完全控制**，之后点击**确定**，点击后所有的选项都选择**“确定”**。

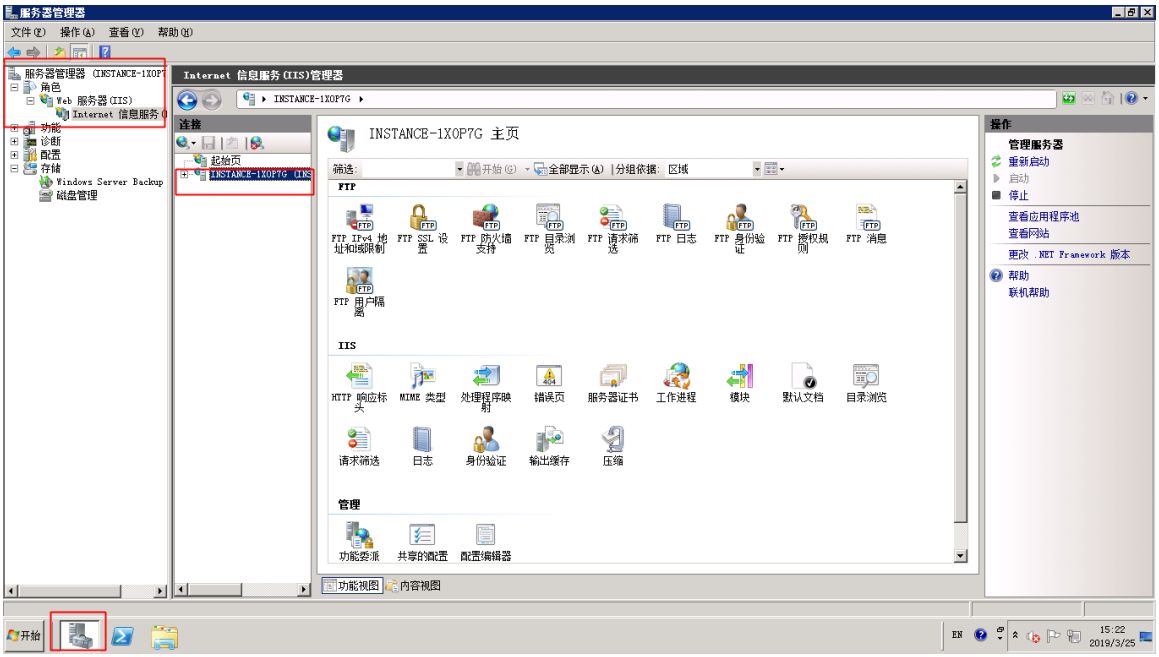


添加及设置FTP站点

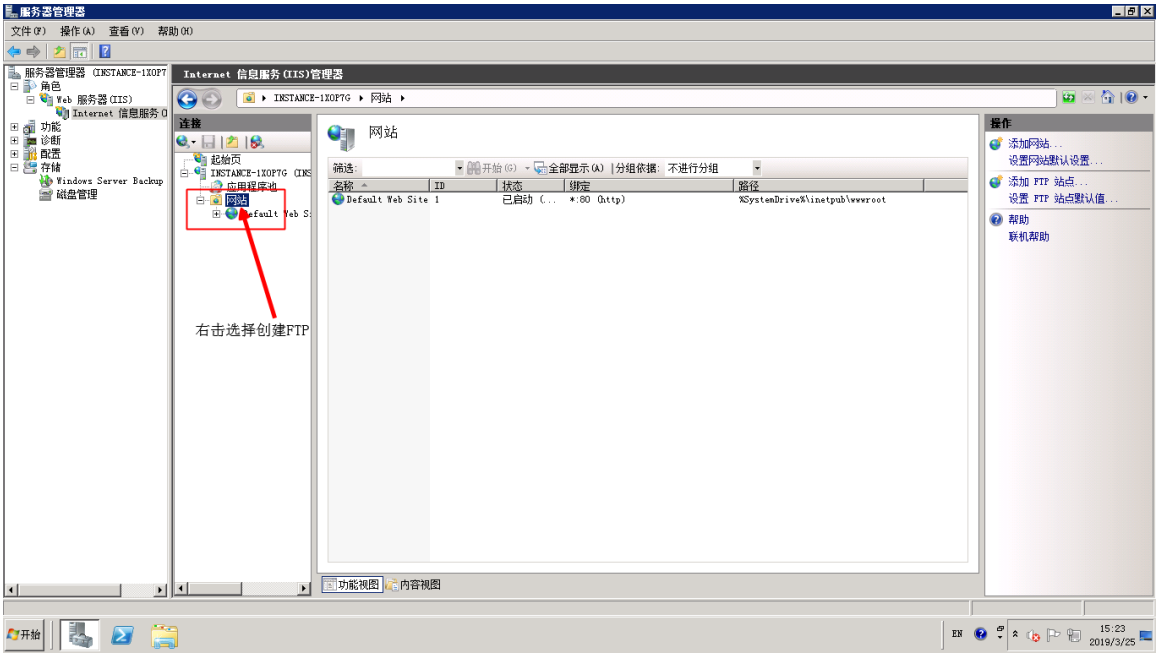
安装 FTP，设置好共享文件夹权限后，您需要创建 FTP 站点。

1. 选择 **开始** > **所有程序** > **管理工具** > **Internet 信息服务（IIS）管理器**

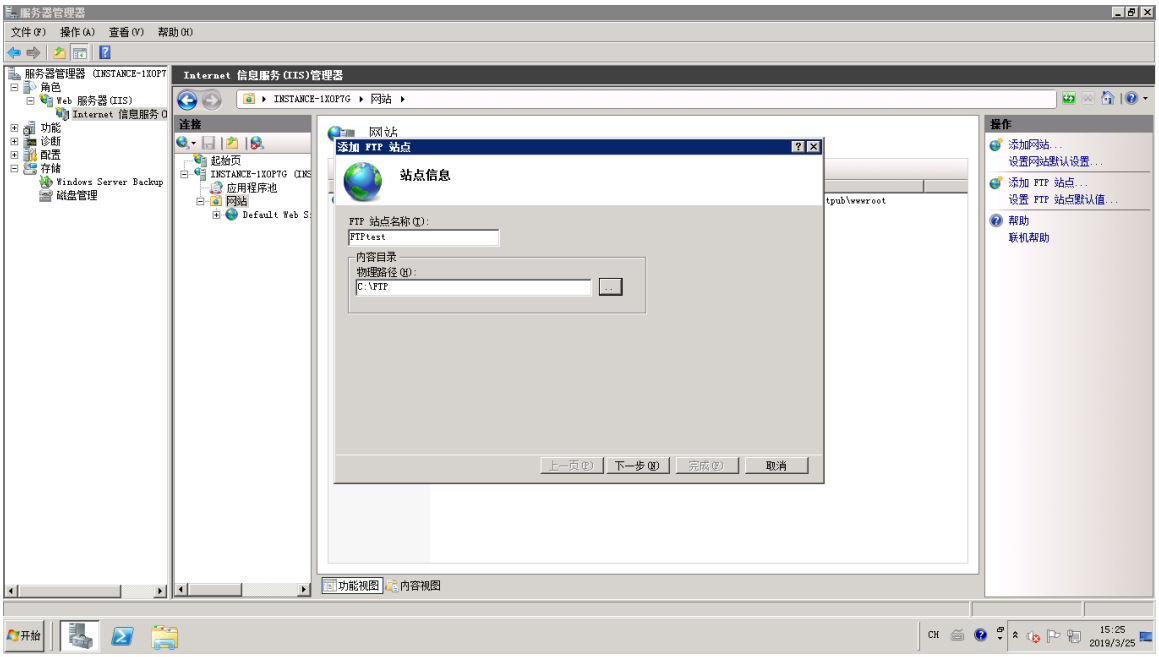
1. 选择 **开始** > **所有程序** > **管理工具** > **Internet 信息服务（IIS）管理器**。



2. 右键单击 **网站**，选择 **添加 FTP 站点**。

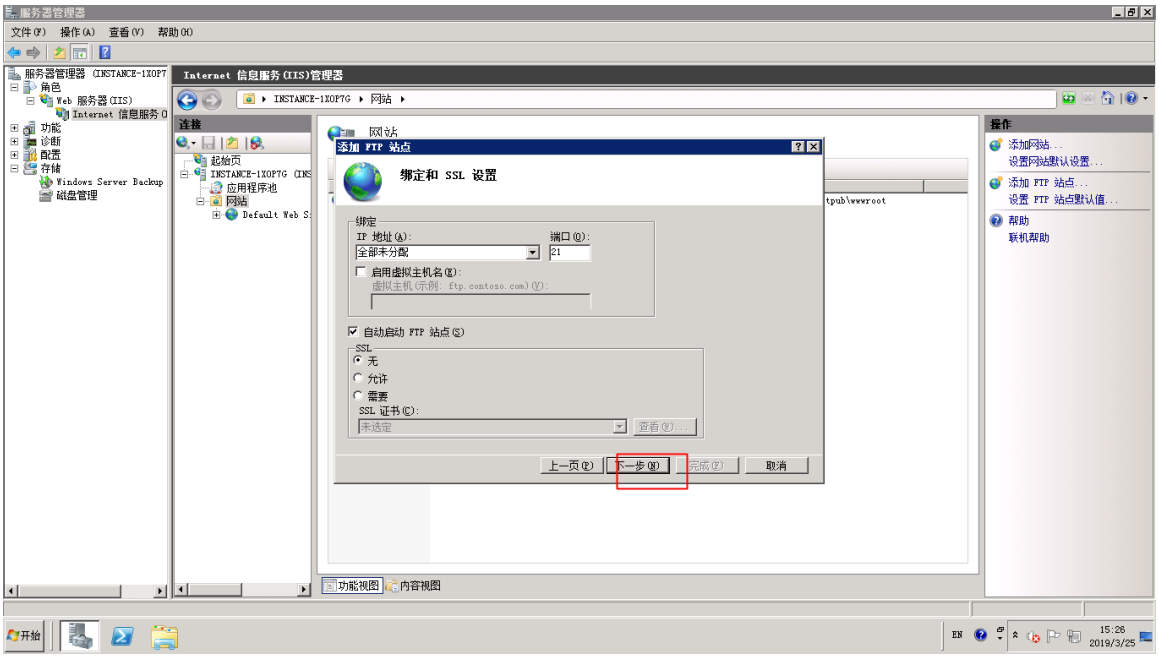


3. 在弹出的窗口，填写 FTP 站点名称与共享文件夹的物理路径（之前创建的FTP的文件夹的路径），然后单击 **下一步**。



4. IP 地址默认选择 **全部未分配**。端口号可自行设置，FTP 默认端口号为 **21**。

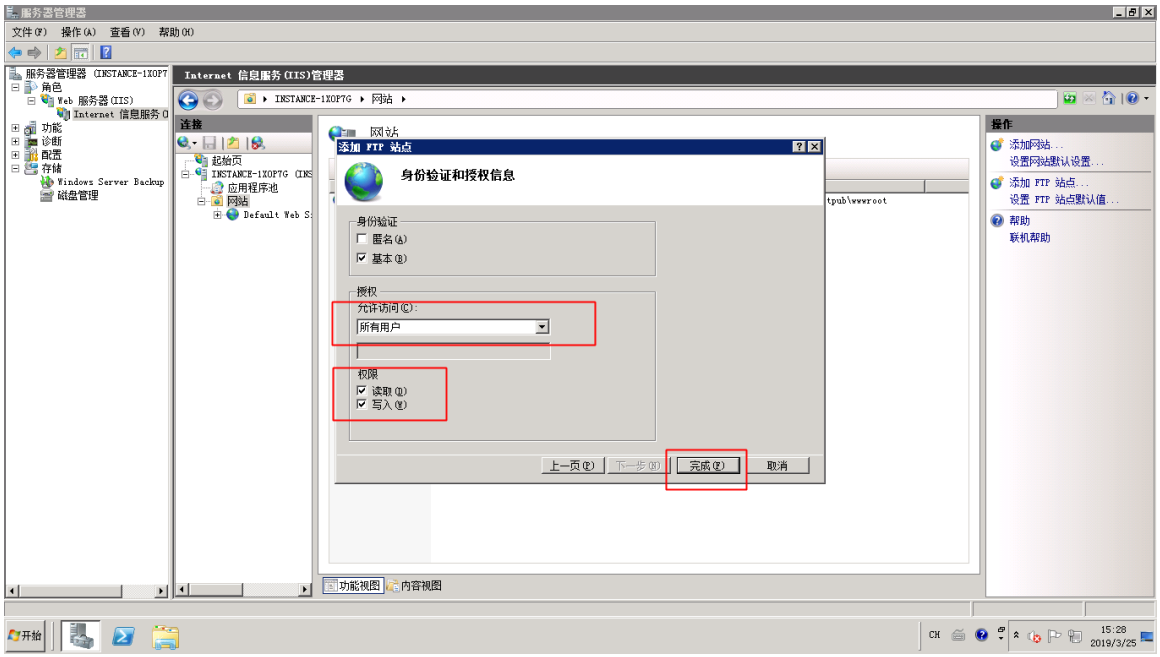
5. 选择 **SSL 设置**，根据您的实际需求进行实例选择，如**不需要**。



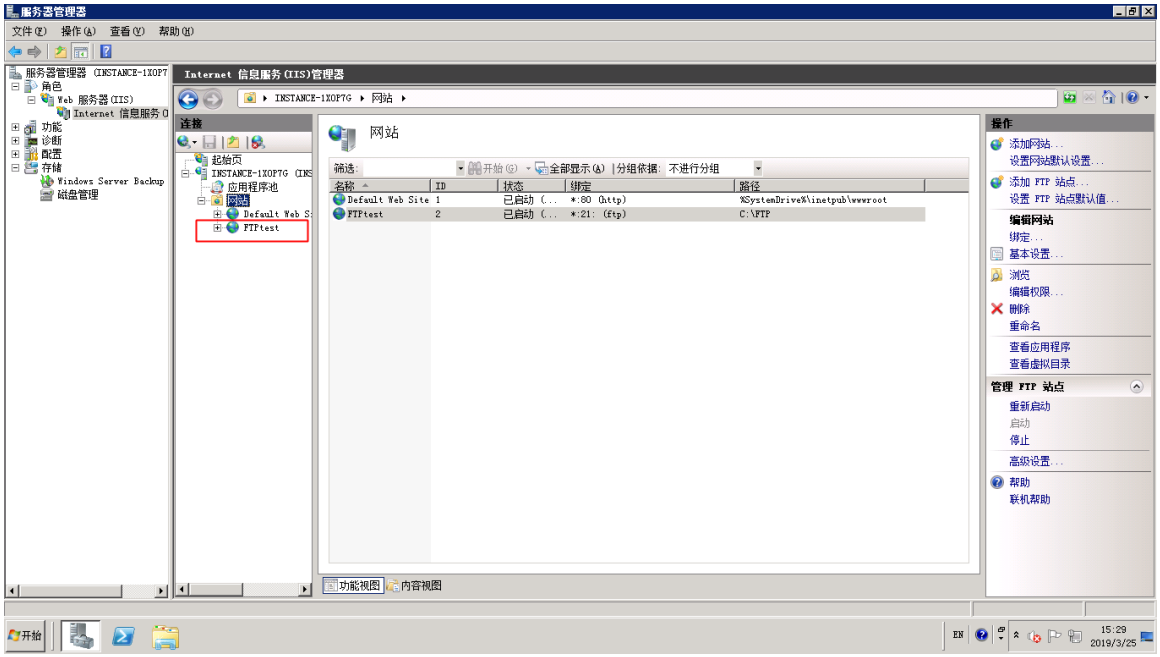
- 允许：允许 FTP 服务器支持与客户端的非 SSL 和 SSL 连接。
- 需要：需要对 FTP 服务器和客户端之间的通信进行 SSL 加密。
- 无：不需要 SSL 加密选择 无。

  5.选择要使用的一种或多种身份验证方法。

- 匿名：允许任何仅提供用户名 anonymous 或 ftp 的用户访问内容。
- 基本：需要用户提供有效用户名和密码才能访问内容。由于基本身份验证通过网络传输未加密的密码，因此请仅在清楚客户端和 FTP 服务器之间的连接是安全的情况下（例如，使用安全套接字层 (SSL) 时）使用此身份验证方法。



6.完成后可以看到搭建的 FTP 站点。



设置安全组及防火墙

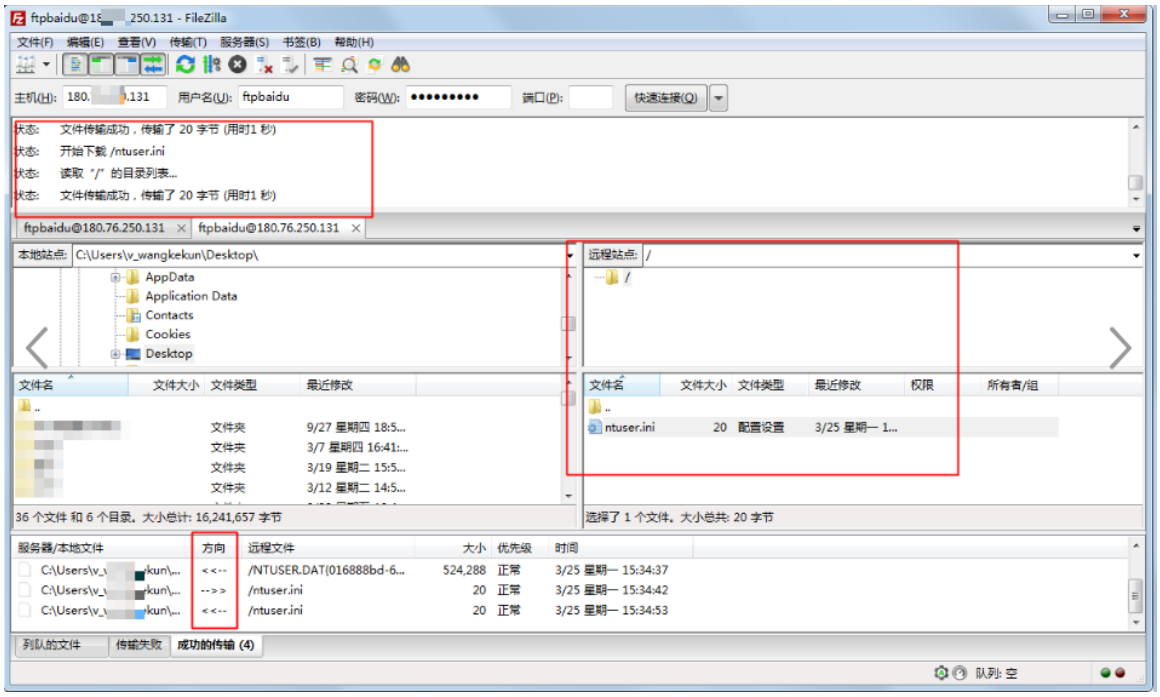
搭建好 FTP 站点后，您需要在实例安全组的入方向添加一条放行 FTP 端口的规则，具体步骤参见添加安全组规则

服务器防火墙需要设置放行 **TCP 21 端口**用于 FTP 服务。（可自行百度防火墙放行端口教程，或可以暂时关闭防火墙）

其他防火墙设置参见微软官方文档。

客户端测试

使用FTP工具filezilla进行测试可正常连接和上传下载文件。



Windows系统部署SQL Server 2008 R2 Express

步骤一 安装前的准备工作

1.下载SQL Server安装程序。

2.下载Microsoft SQL Server2008 R2 RTM - Express with Management Tools （一般是选择X64位的版本）。

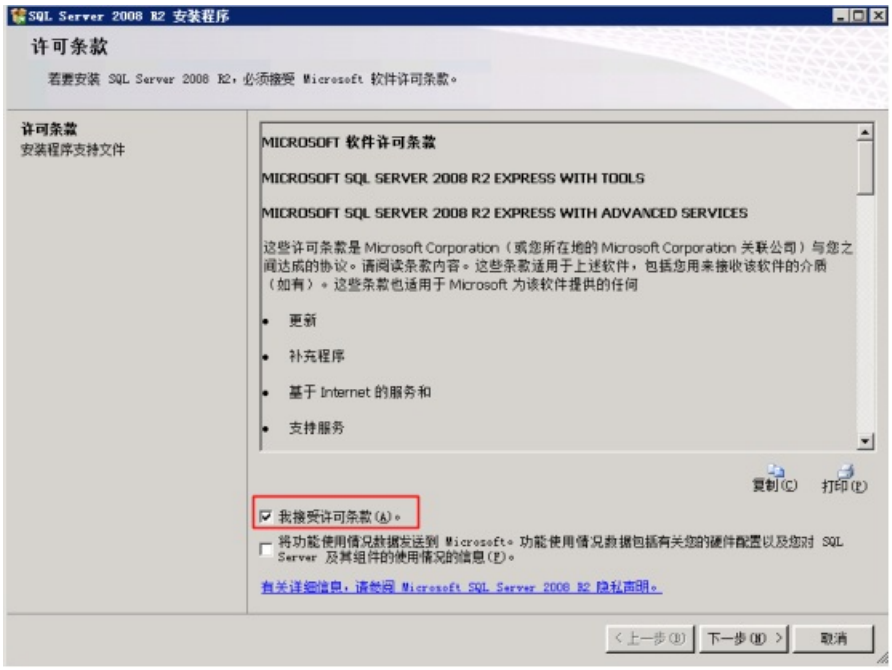
SQL sever下载地址

步骤二 安装SQL Server 1.双击运行下载的SQL Server安装程序SQLEXPRWT_x64_CHS.exe。

2.打开SQLServer安装中心，在右边选择“全新安装或向现有安装添加功能”。

3.SQL Server开始安装准备。

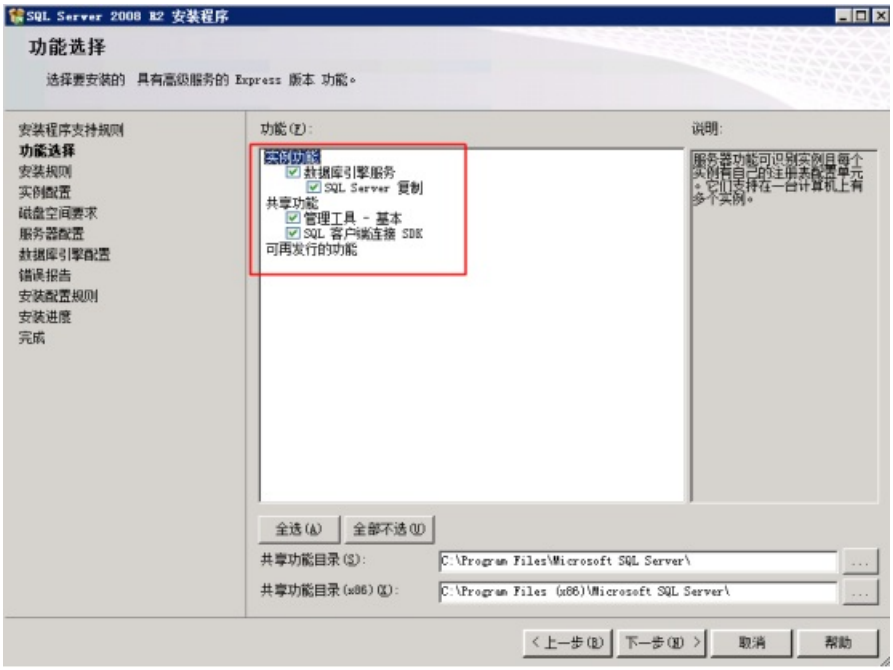
4.,在“许可条款”中选择“我接受许可条款”，然后“下一步”。



5.安装程序支持文件。在下面的界面中点击“安装”来安装程序的支持文件。

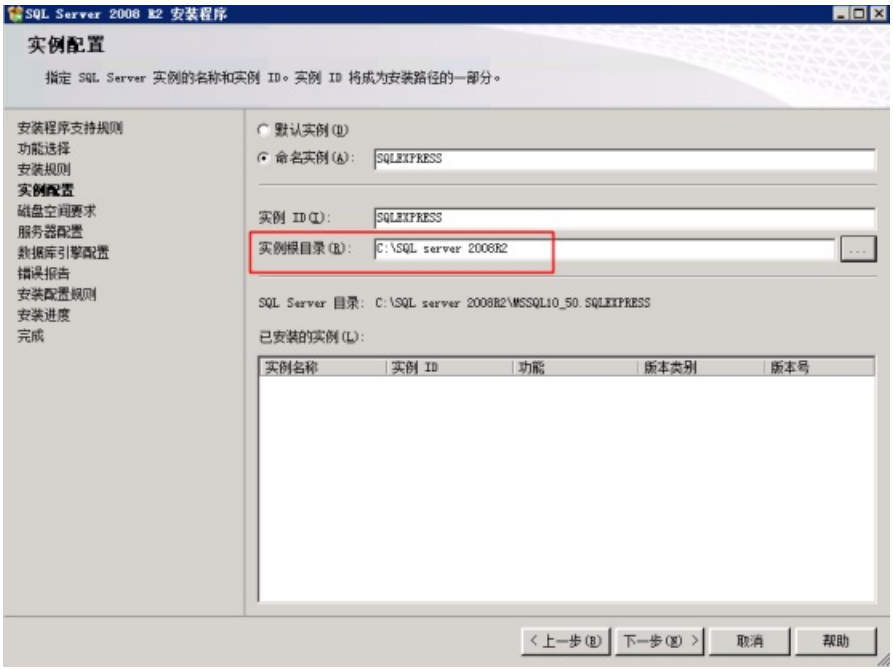
6.安装程序支持规则进行检查，对于没有通过的项进行修改，以符合支持规则。全部通过后点击“下一步”。

7.进行功能选择。根据需要选择功能，然后点击“下一步”。



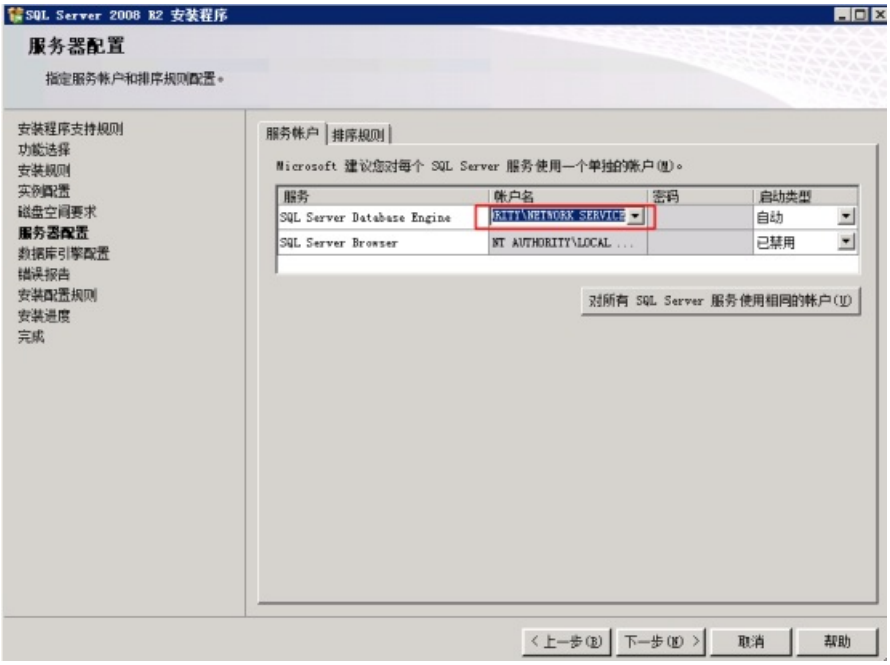
8.安装程序开始进行安装规则检查。通过后点击“下一步”。

9.在“实例配置”中指定实例的名称和实例安装的位置。

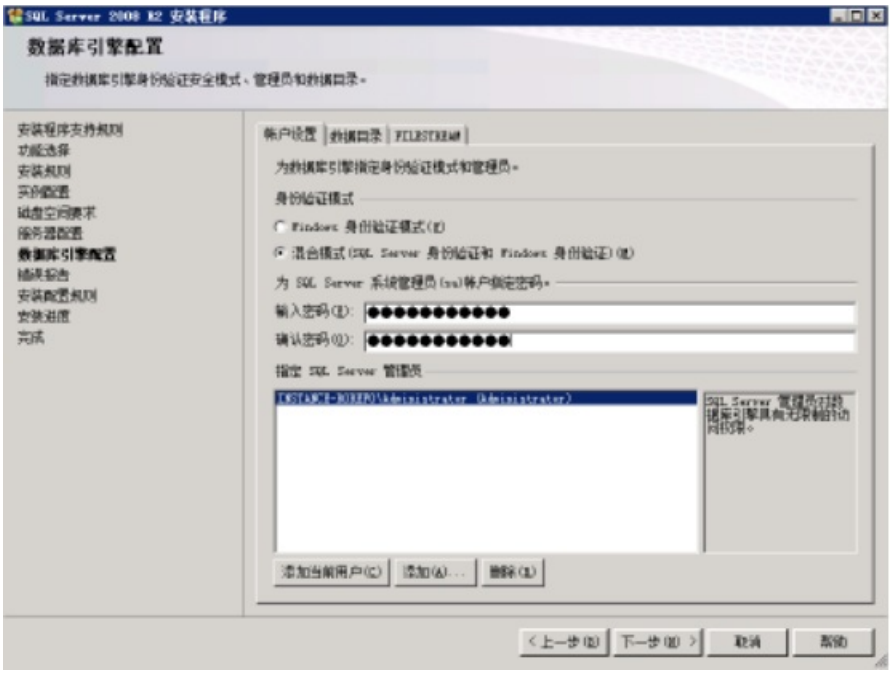


10.安装程序开始检查磁盘空间是否满足要求。满足要求后，点击“下一步”。

11.服务器配置，点击“下一步”。



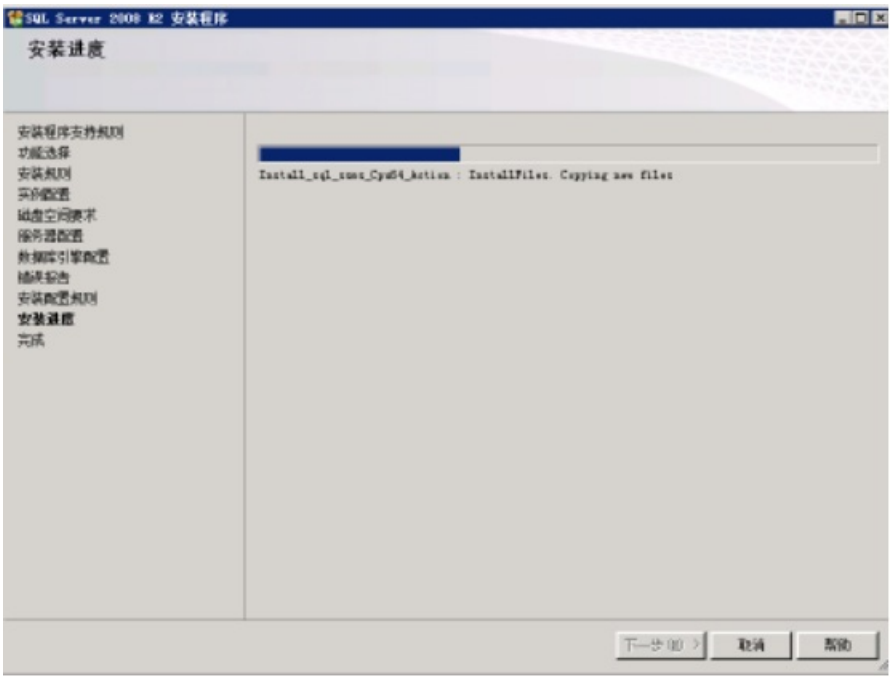
12.在“数据库引擎配置”中，选择“混合模式”，并为“SQLServer系统管理员账户（sa）”指定密码，比如“DFG22344”。



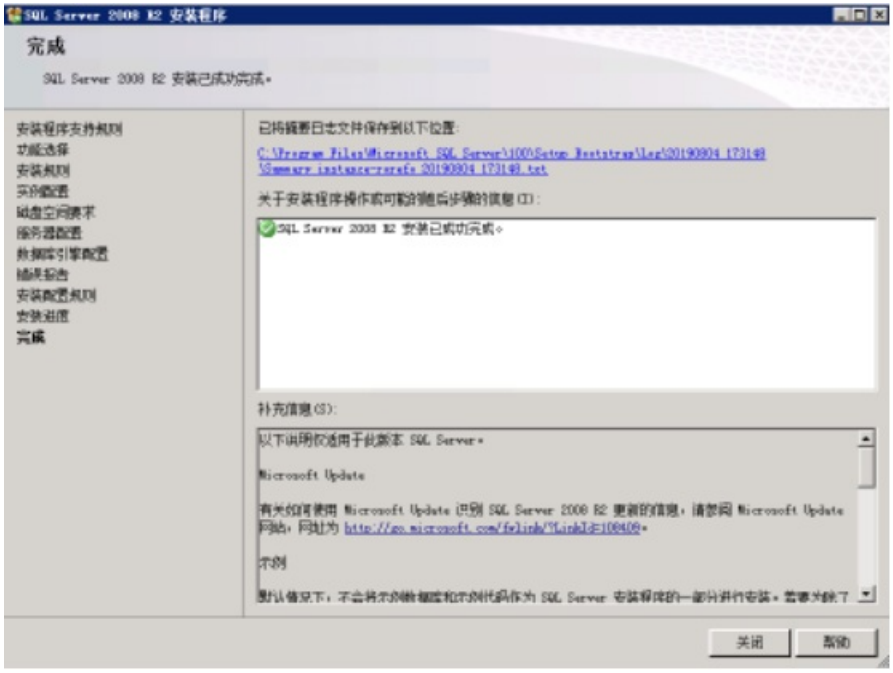
13.在“错误报告”界面，点击“下一步”。

14.安装程序运行安装配置规则。完成后点击“下一步”。

15.开始程序的安装。

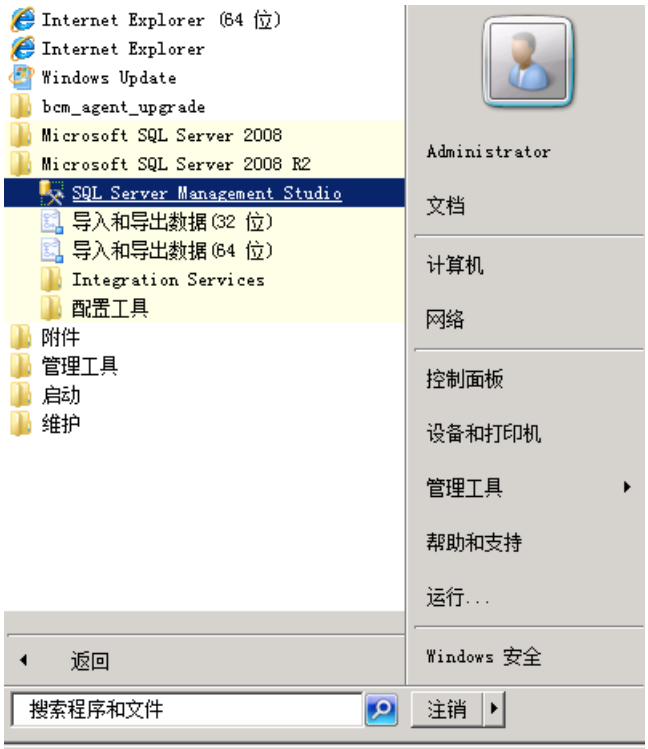


16.安装完成



步骤三 登录SQL Server 数据库

1.选择开始菜单 – 所有程序 – MicrosoftSQL Server 2008 R2 – SQL Server Management Studio。



2.打开SQL ServerManagement Studio。

3.在打开的“连接到服务器”对话框中，服务器类型选择“数据库引擎”。

到此既完成了SQL Server的搭建。

CentOS 7 yum安装MySQL 8.0

本视频介绍操作系统为CentOS 7的BCC实例yum安装MySQL 8.0的过程。

Linux系统搭建宝塔

本文介绍百度智能云“Linux系统BCC如何安装宝塔”的方法（含操作指导视频），您可以参考以下视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《Linux系统BCC如何安装宝塔》

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

本教程以百度智能云公共镜像CentOS7.6 64位操作系统为例，说明如何在 Linux 实例上安装宝塔应用。

步骤一 登录服务器

可以通过ssh远程连接服务器，如果无法远程连接，使用VNC登录。

步骤二 yum安装宝塔系统

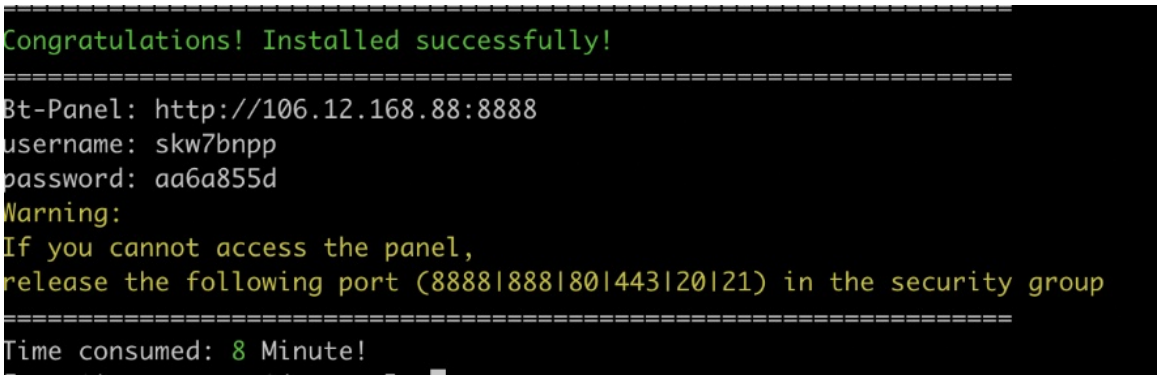
运行命令 yum install -y wget && wget -O install.sh http://download.bt.cn/install/install_6.0.sh && sh install.sh 安装宝塔系统。

步骤三 查看登录信息

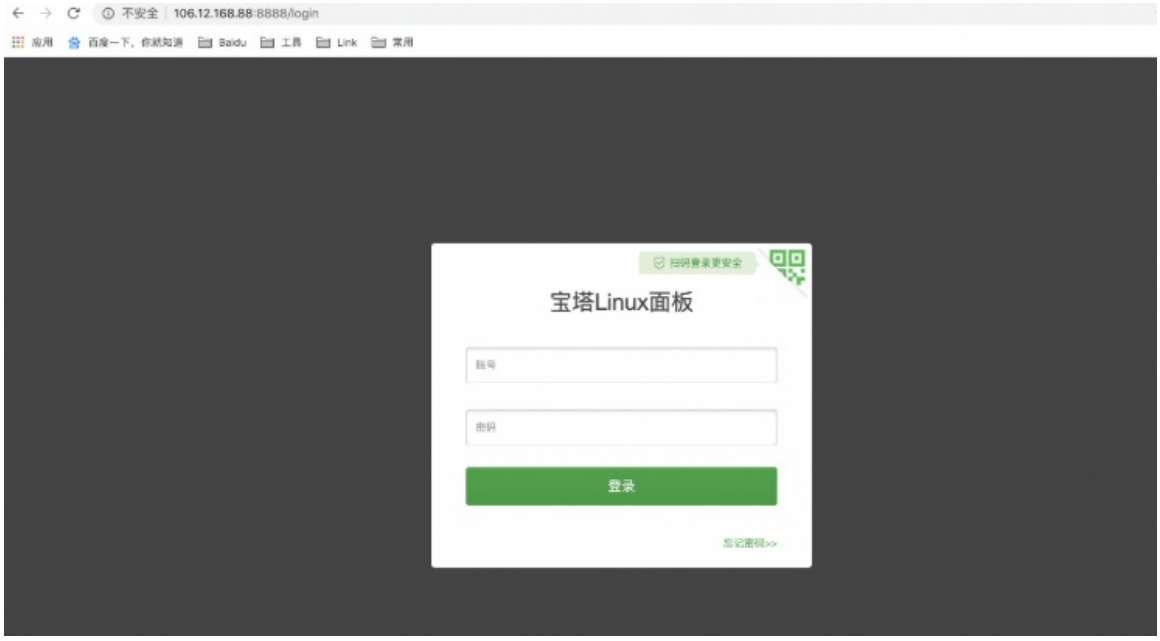
Bt-Panel：宝塔面板登录地址

Username：宝塔面板登录账号

Password：宝塔面板登录密码



步骤四 访问验证



输入用户名密码信息，既可完成登录。

Windows系统搭建宝塔

本视频介绍在BCC中Windows Server 2012如何安装宝塔面板。

Linux幻兽帕鲁服务器一键搭建

概览

《幻兽帕鲁》官方服务器太过火爆，时不时就遇到卡顿现象。不如自己搭建一个专属服务器，与小伙伴们一起畅享丝滑。

方案概述

支持 Ubuntu18 和 Centos7 以上系统，一键下载安装《幻兽帕鲁》游戏。

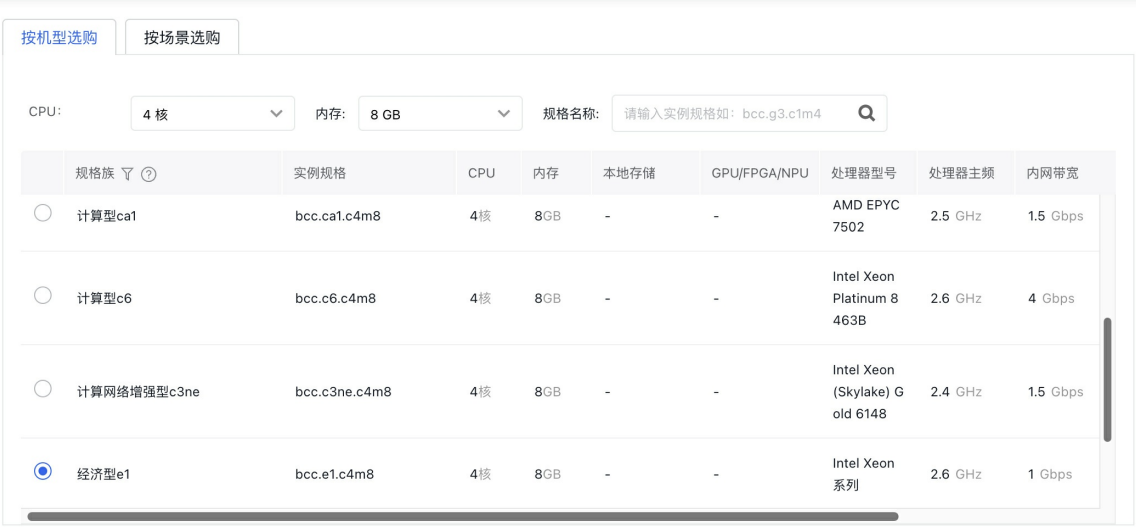
环境准备

- 客户端：安装好 Steam 的电脑
- 服务端：一台绑定了公网 IP 的 BCC，磁盘最低 20G，装完还剩 12G 左右空间
- 服务端配置建议：幻兽帕鲁官方推荐最低配置为 4 核 8G，可以使用 bcc.e1.c4m8 套餐机器

> 注：EIP带宽可根据实际体验灵活调整。多人游戏建议使用更高规格实例防止游戏服务 OOM 影响体验

Ubuntu 服务端配置示例

1. 登录控制台，按需购买云服务器，选择 Ubuntu 公共镜像。

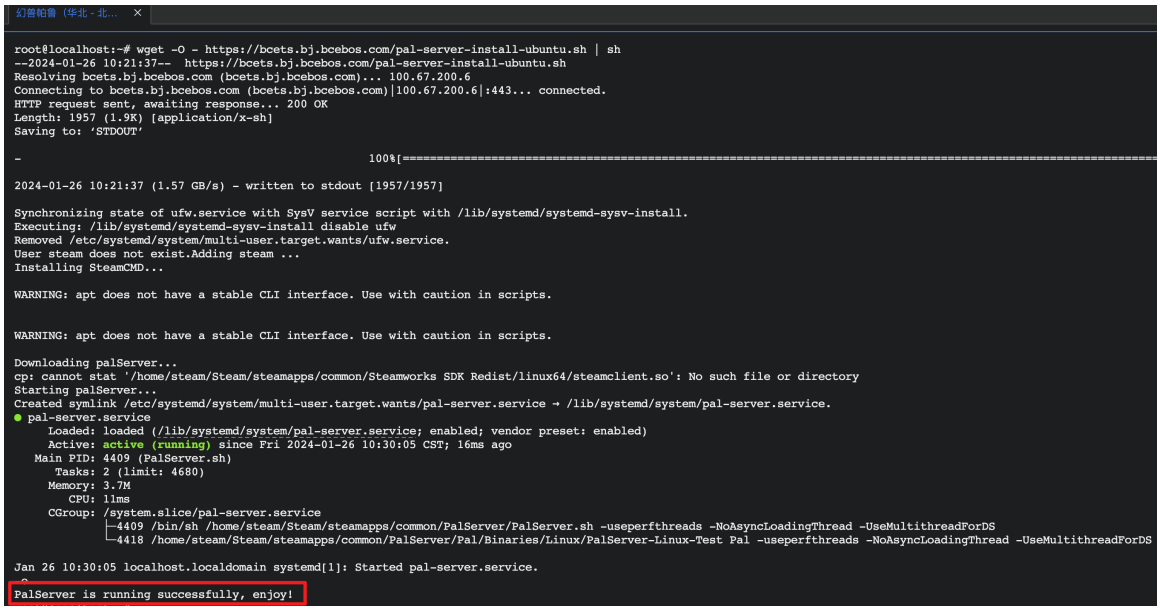


2. 在购买页面同时选择“购买弹性公网IP”，并根据您的需要选择适合的计费方式和带宽峰值。

3. 通过 SmartTerm 登录服务器，这里可以记录下服务器的弹性公网 IP 地址

4. 执行安装脚本等待安装结束

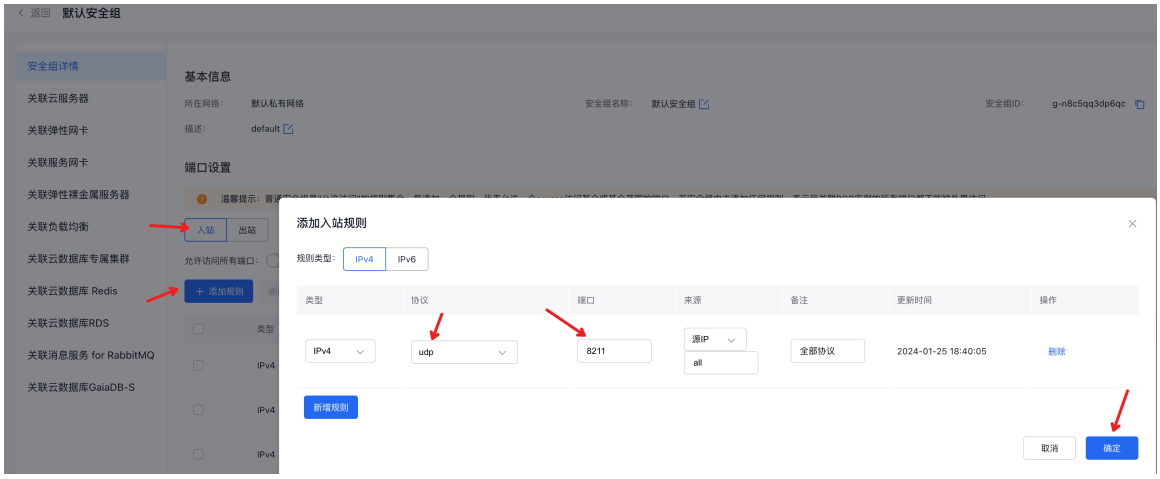
```
wget -O - https://bcets.bj.bcebos.com/pal-server-install-ubuntu.sh | sh
```



详细安装过程可以 tail -f /tmp/pal_server.log 命令查看

```
root@instance-0bvk66tk:~# tail -f /tmp/pal_server.log
 Update state (0x81) verifying update, progress: 61.86 (2925415139 / 4729437191)
 Update state (0x81) verifying update, progress: 69.54 (3289039118 / 4729437191)
 Update state (0x81) verifying update, progress: 77.33 (3657320460 / 4729437191)
 Update state (0x81) verifying update, progress: 84.84 (4012622802 / 4729437191)
 Update state (0x81) verifying update, progress: 92.34 (4367111879 / 4729437191)
 Update state (0x101) committing, progress: 0.00 (0 / 4729437191)
 Update state (0x101) committing, progress: 10.52 (497770195 / 4729437191)
 Update state (0x101) committing, progress: 99.53 (4707243775 / 4729437191)
Success! App '2394010' fully installed.
```

6. 操作安全组入向规则放行 UDP 协议 8211 端口的流量



Centos 服务端配置示例

1. 登录控制台，按需购买云服务器，选择 Centos 公共镜像。

2. 在购买页面同时选择“购买弹性公网IP”，并根据您的需要选择适合的计费方式和带宽峰值。

3. 通过 SmartTerm 登录服务器，这里可以记录下服务器的弹性公网 IP 地址

4. 执行安装脚本等待安装结束

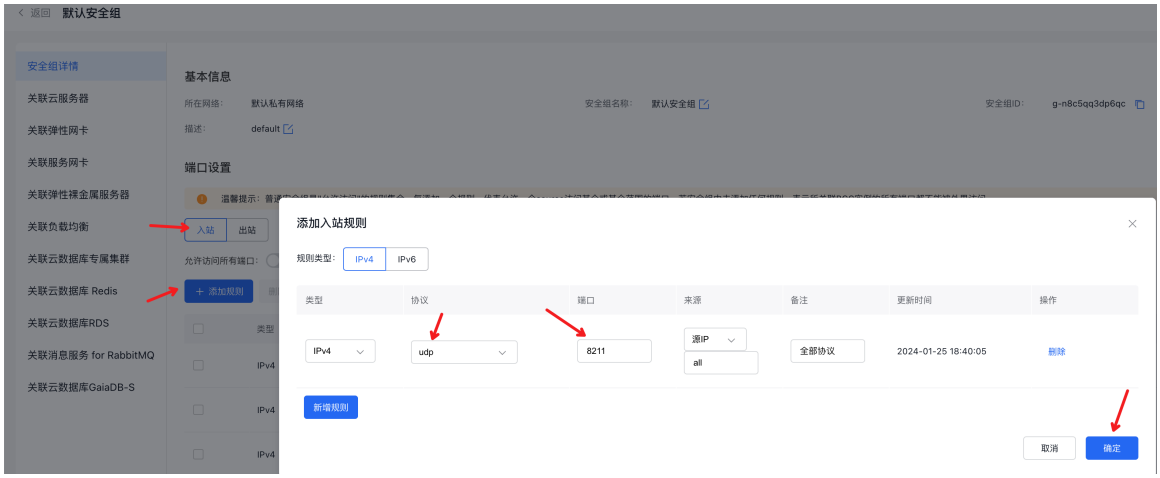
```
wget -O - https://bcets.bj.bcebos.com/pal-server-install-centos.sh | sh
```

```
[root@keep_yyy-cce ~]# wget -O - https://bcets.bj.bcebos.com/pal-server-install-centos.sh | sh
--2024-01-26 10:42:31--  https://bcets.bj.bcebos.com/pal-server-install-centos.sh
Resolving bcets.bj.bcebos.com (bcets.bj.bcebos.com)... 100.67.200.6
Connecting to bcets.bj.bcebos.com (bcets.bj.bcebos.com)|100.67.200.6|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1894 (1.8K) [text/x-sh]
Saving to: 'STDOUT'

100%[==========================================================================================================================================>] 1,894       --.-K/s   in 0s

2024-01-26 10:42:31 (761 MB/s) - written to stdout [1894/1894]

User steam does not exist. Adding steam ...
Warning: RPMDB altered outside of yum.
Error:  Multilib version problems found. This often means that the root
       cause is something else and multilib version checking is just
       pointing out that there is a problem. Eg.:

         1. You have an upgrade for libstdc++ which is missing some
            dependency that another package requires. Yum is trying to
            solve this by installing an older version of libstdc++ of the
            different architecture. If you exclude the bad architecture
            yum will tell you what the root cause is (which package
            requires what). You can try redoing the upgrade with
            --exclude libstdc++.otherarch ... this should give you an error
            message showing the root cause of the problem.

         2. You have multiple architectures of libstdc++ installed, but
            yum can only see an upgrade for one of those architectures.
            If you don't want/need both architectures anymore then you
            can remove the one with the missing update and everything
            will work.

         3. You have duplicate versions of libstdc++ installed already.
            You can use "yum check" to get yum show these errors.

       ...you can also use --setopt=protected_multilib=false to remove
       this checking, however this is almost never the correct thing to
       do as something else is very likely to go wrong (often causing
       much more problems).

       Protected multilib versions: libstdc++-4.8.5-44.el7.i686 != libstdc++-4.8.5-36.el7.x86_64
Installing SteamCMD...
Downloading palServer...
Starting palServer...
Created symlink from /etc/systemd/system/multi-user.target.wants/pal-server.service to /usr/lib/systemd/system/pal-server.service.
● pal-server.service
   Loaded: loaded (/usr/lib/systemd/system/pal-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2024-01-26 10:46:23 CST; 14ms ago
 Main PID: 2595 (PalServer.sh)
   CGroup: /system.slice/pal-server.service
           ├─2595 /bin/sh /home/steam/Steam/steamapps/common/PalServer/PalServer.sh -useperfthreads -NoAsyncLoadingThread -UseMultithreadForDS
           └─2604 /home/steam/Steam/steamapps/common/PalServer/Pal/Binaries/Linux/PalServer-Linux-Test Pal -useperfthreads -NoAsyncLoadingThread -UseMultithreadForDS

Jan 26 10:46:23 keep_yyy-cce systemd[1]: Started pal-server.service.

PalServer is running successfully, enjoy!
```

详细安装过程可以 tail -f /tmp/pal_server.log 命令查看

```
[root@keep_yyy-cce ~]# tail -f /tmp/pal_server.log
 Update state (0x61) downloading, progress: 89.71 (2016836536 / 2248053389)
 Update state (0x61) downloading, progress: 91.12 (2048361761 / 2248053389)
 Update state (0x61) downloading, progress: 92.71 (2084246544 / 2248053389)
 Update state (0x61) downloading, progress: 94.11 (2115703252 / 2248053389)
 Update state (0x61) downloading, progress: 95.64 (2149957716 / 2248053389)
 Update state (0x61) downloading, progress: 97.13 (2183446104 / 2248053389)
 Update state (0x81) verifying update, progress: 2.12 (47650608 / 2248053389)
 Update state (0x81) verifying update, progress: 45.69 (1027120148 / 2248053389)
 Update state (0x81) verifying update, progress: 88.98 (2000351807 / 2248053389)
Success! App '2394010' fully installed.
```

6. 操作安全组入向规则放行 UDP 协议 8211 端口的流量



服务端版本更新 如您进入游戏时提示：「您正尝试加入的比赛正在运行不兼容的游戏版本，请尝试升级游戏版本」，您可以通过如下操作步骤更新幻兽帕鲁服务器。

1. 通过 SmartTerm 或者远程桌面的方式 登录服务器，以root账户登录。
2. 根据操作系统选择对应的命令，执行更新 Palworld 服务器程序，此命令要下载游戏服务器文件，请耐心等待其执行完成。

- Ubuntu

```
 sudo systemctl stop pal-server
sudo -u steam $(which steamcmd) +login anonymous +app_update 2394010 validate +quit
sudo systemctl restart pal-server
sudo systemctl status pal-server
```

- CentOS

```
 sudo systemctl stop pal-server
sudo -u steam /home/steam/steamcmd.sh +login anonymous +app_update 2394010 validate +quit
sudo systemctl restart pal-server
sudo systemctl status pal-server
```

Steam 客户端配置示例

1. 进入游戏，选择 "加入多人游戏（专用服务器）"

2. 输入上述云服务器绑定的弹性公网 IP 和端口，点击联机

3. 进入到创建人物界面，则说明连接服务器成功

Windows幻兽帕鲁服务器一键搭建

概览

《幻兽帕鲁》官方服务器太过火爆，时不时就遇到卡顿现象。不如自己搭建一个专属服务器，与小伙伴们一起畅享丝滑。

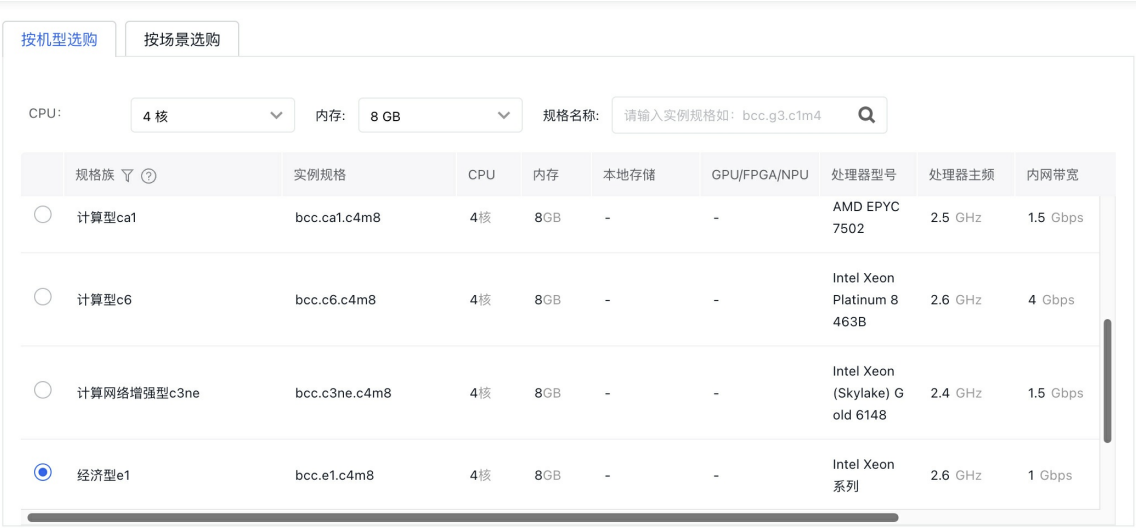
环境准备

- 客户端：安装好 Steam 的电脑
- 服务端：一台绑定了公网 IP 的 BCC，磁盘最低 40G，装完还剩 20G 左右空间
- 服务端配置建议：幻兽帕鲁官方推荐最低配置为 4 核 8G，可以使用 bcc.e1.c4m8 套餐机器

> 注：EIP带宽可根据实际体验灵活调整。多人游戏建议使用更高规格实例防止游戏服务 OOM 影响体验

服务端配置示例

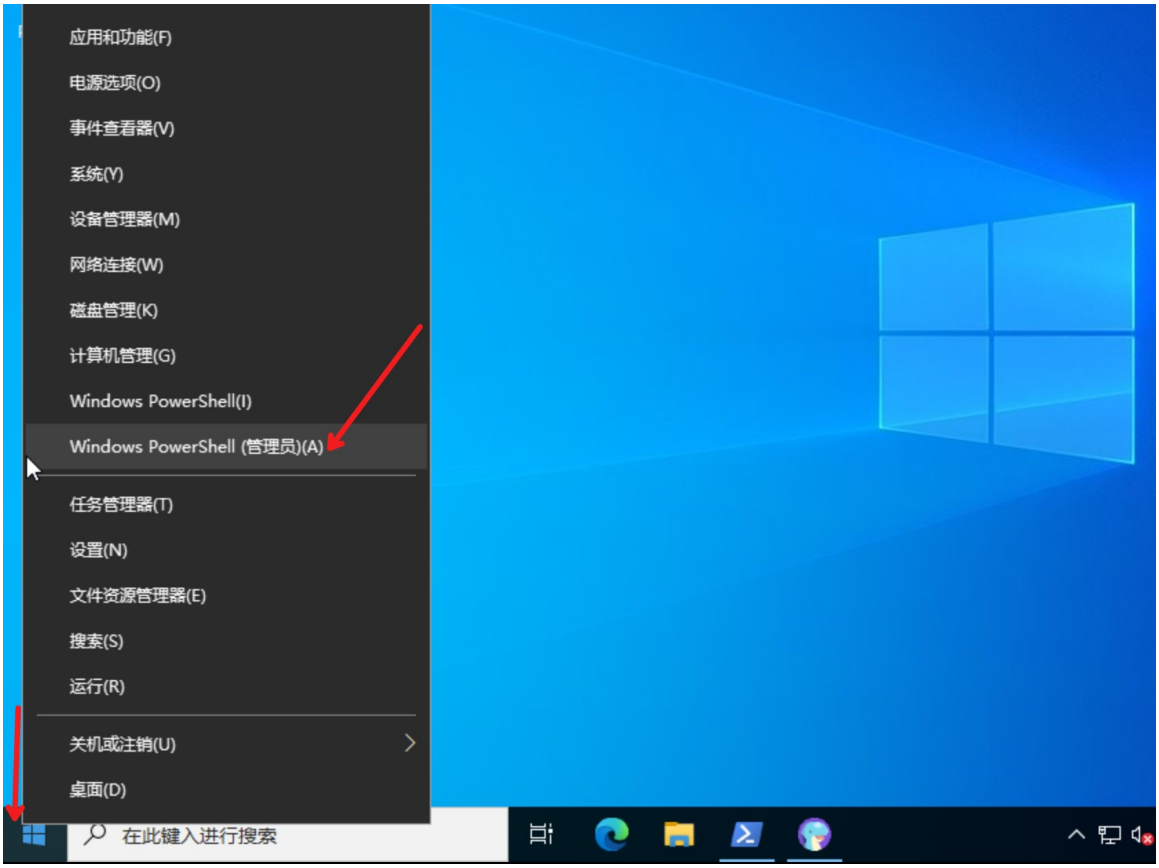
1. 登录控制台，按需购买云服务器，选择 Windows 公共镜像。



2. 在购买页面同时选择“购买弹性公网IP”，并根据您的需要选择适合的计费方式和带宽峰值。

3. 通过 VNC 或者远程桌面的方式 登录服务器，这里可以记录下服务器的弹性公网 IP 地址

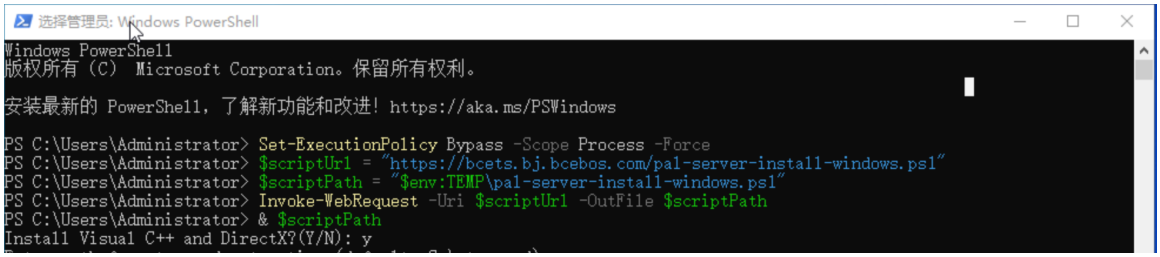
4. 鼠标放到左下角 Windows 徽标上右击，选择 Windows Powershell（管理员）打开



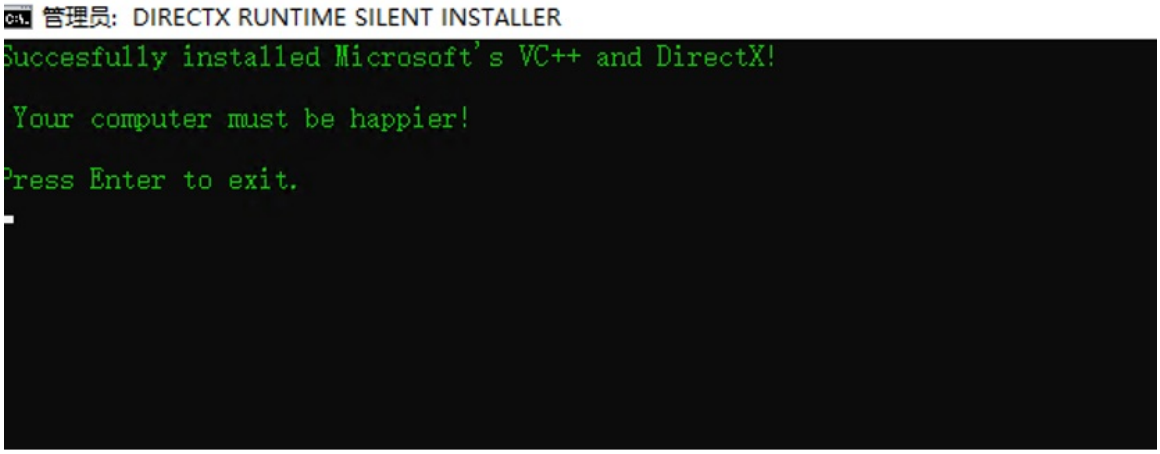
5. 执行安装脚本等待安装结束

```
 Set-ExecutionPolicy Bypass -Scope Process -Force
$scriptUrl = "https://bcets.bj.bcebos.com/pal-server-install-windows.ps1"
$scriptPath = "$env:TEMP\pal-server-install-windows.ps1"
Invoke-WebRequest -Uri $scriptUrl -OutFile $scriptPath
& $scriptPath
```

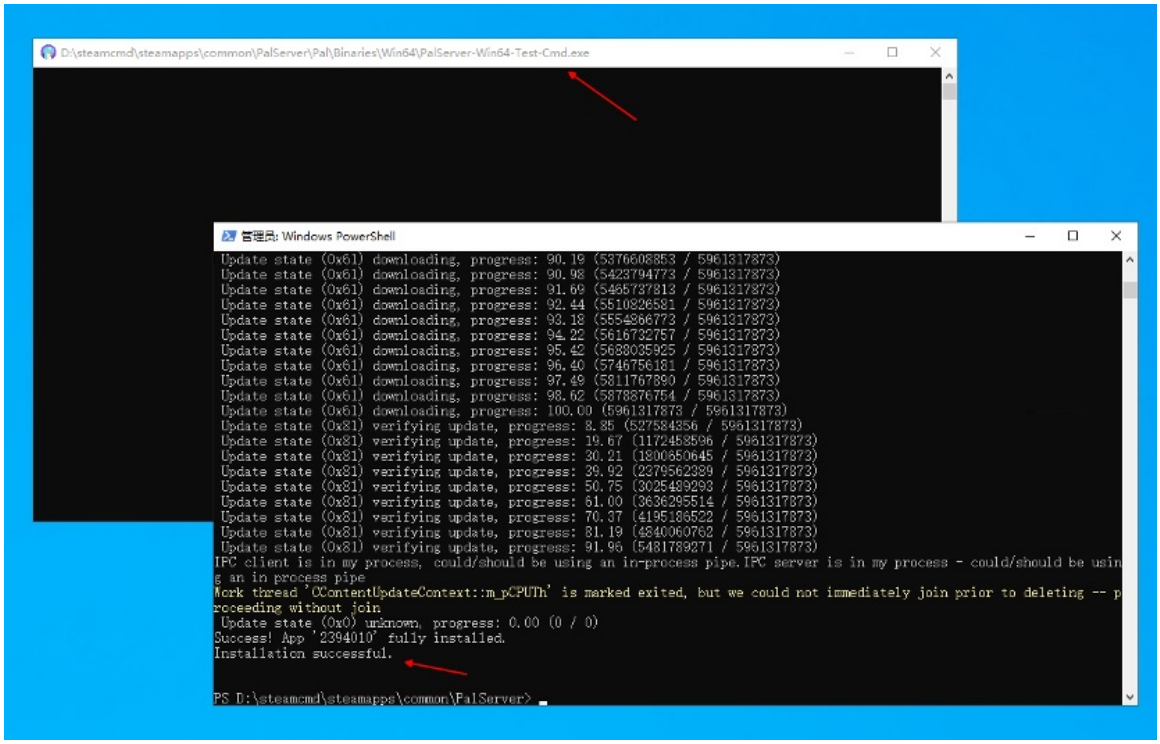
此期间会询问是否安装运行游戏必需的VC++及DirectX组件，如果没有安装过输入Y进行安装；否则输入N来跳过此步骤



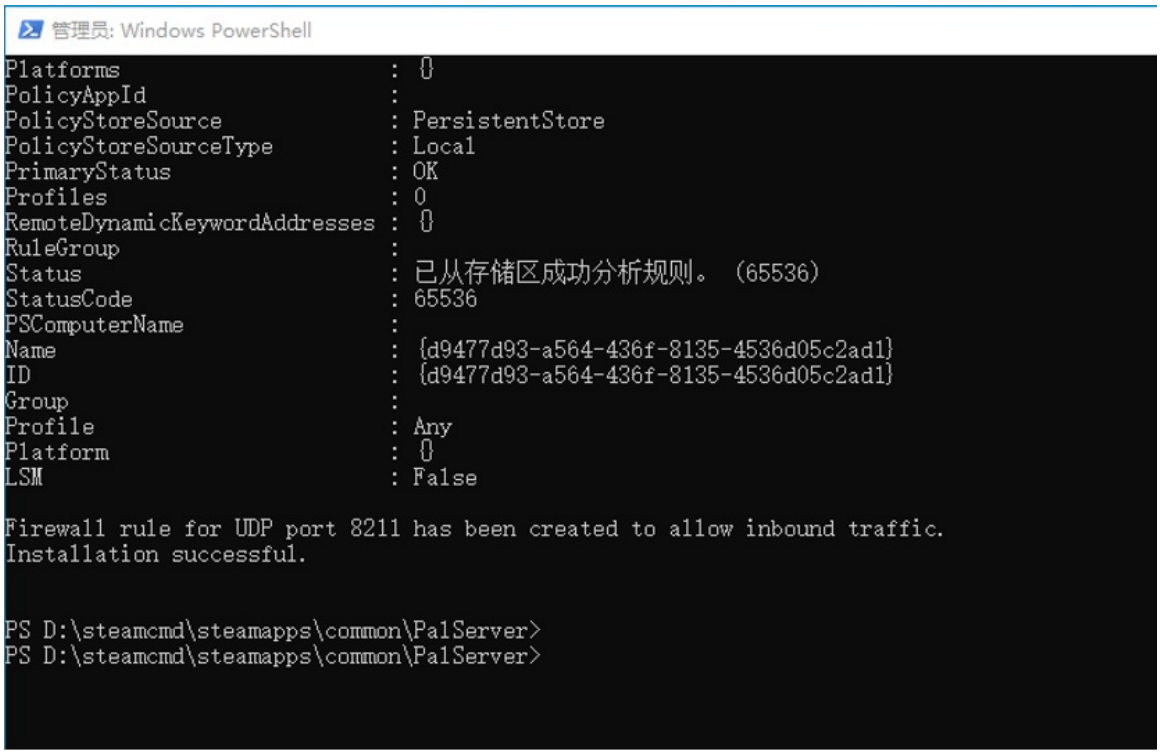
看到以下提示说明VC++及DirectX组件安装成功，关闭该弹窗即可，脚本会继续安装游戏服务



此时会提示用户输入安装路径，默认是C:\steamcmd，如不需修改直接回车。安装完成后会自动打开新的窗口来运行游戏服务

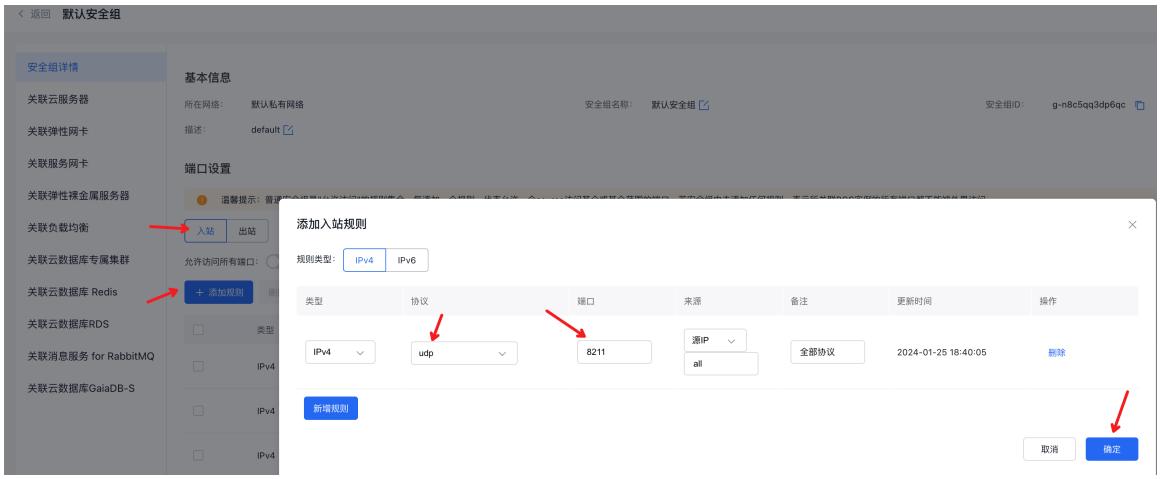


且会自动添加服务器内防火墙规则，放行 UDP 协议 8211 端口的入站流量



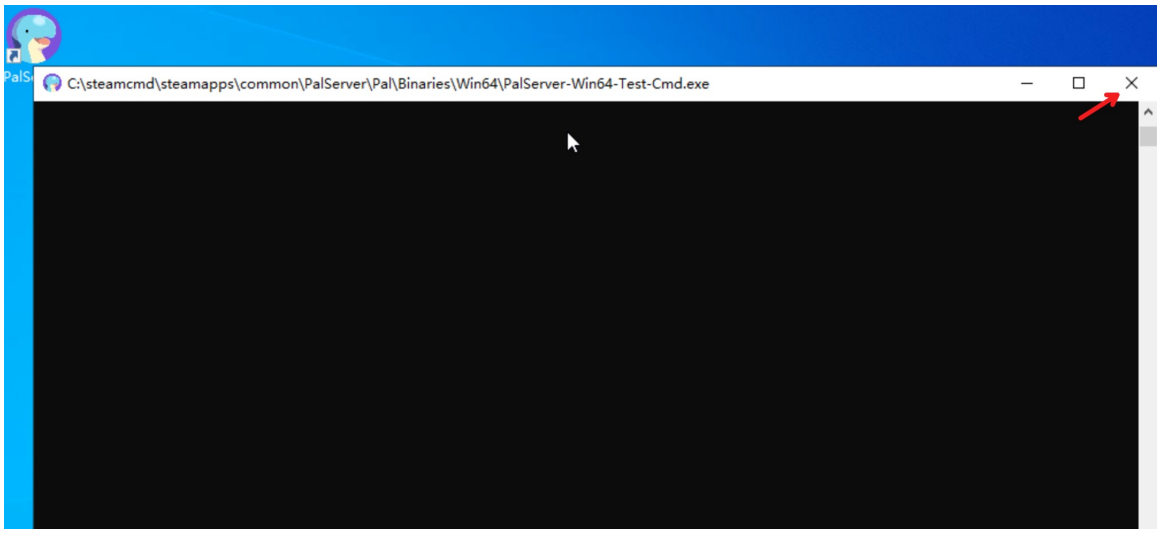
安装完成后会在桌面自动生成快捷方式，可以快速从这里启动游戏服务端

6. 操作安全组入向规则放行 UDP 协议 8211 端口的流量

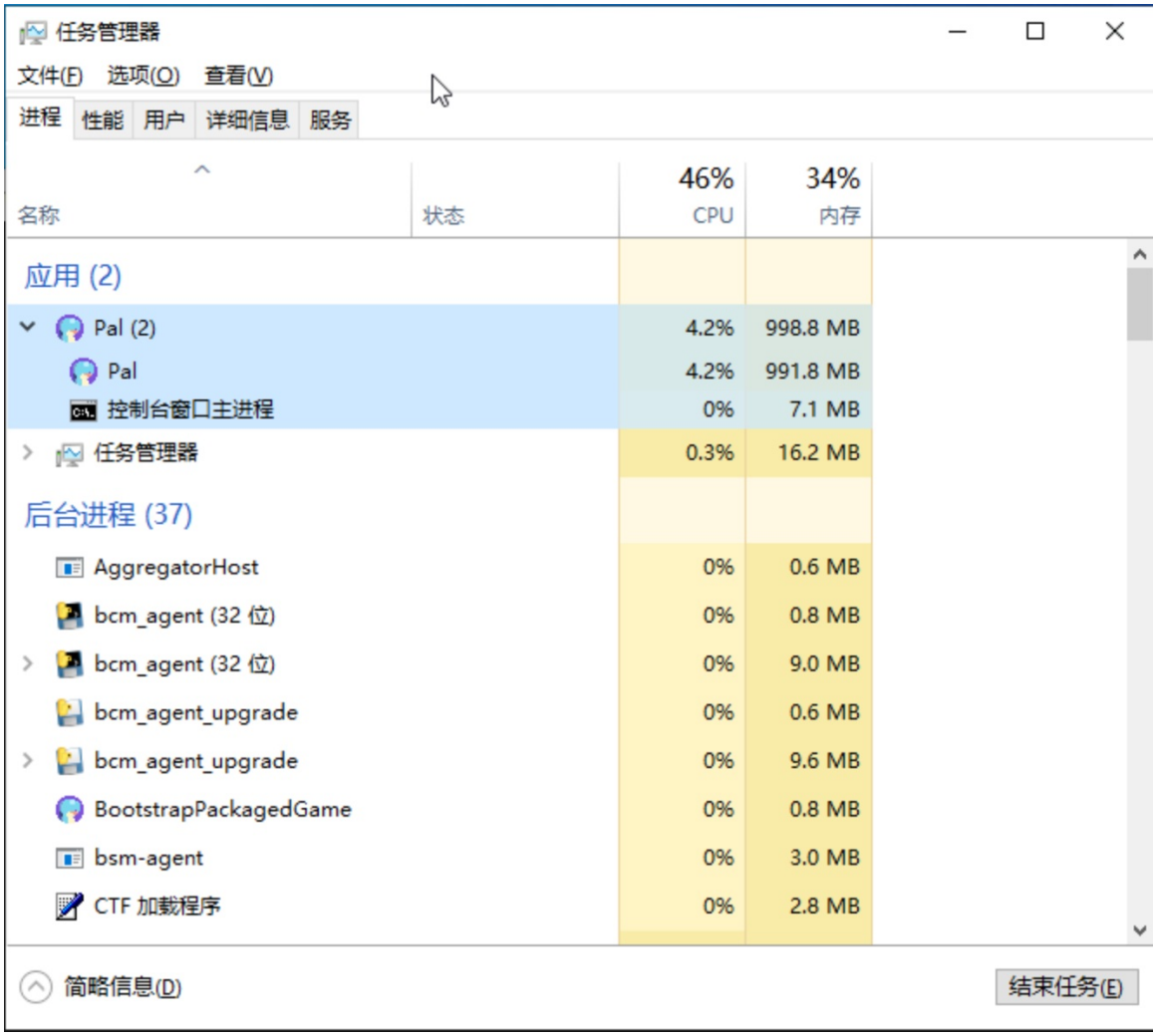


服务端版本更新 如您进入游戏时提示：「您正尝试加入的比赛正在运行不兼容的游戏版本，请尝试升级游戏版本」，您可以通过如下操作步骤更新幻兽帕鲁服务器。

1. 通过 VNC 或者远程桌面的方式 登录服务器，关闭服务端窗口。



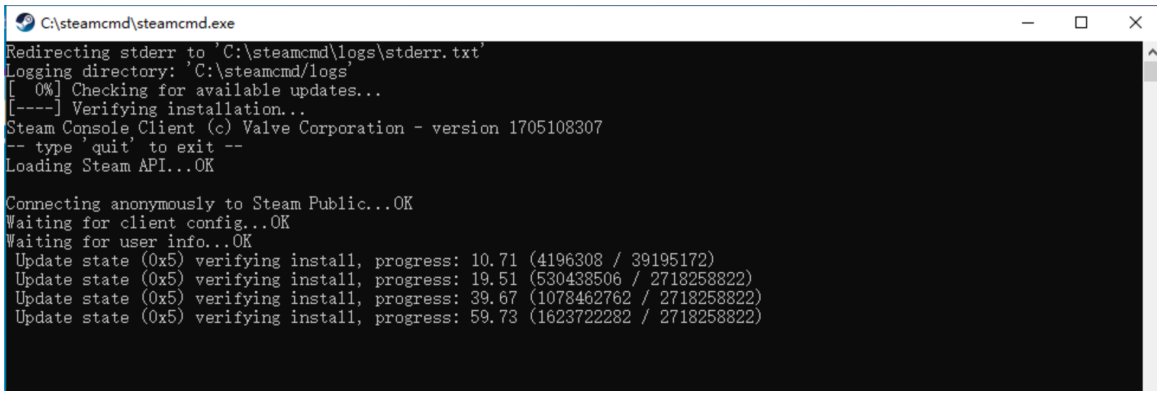
或者打开 任务管理器， 找到Pal应用，右键结束任务。



2. 打开 Windows Powershell（管理员），执行如下命令更新 Palworld 服务器程序，此命令要下载游戏服务器文件，请耐心等待其执行完成。

   注意“C:\steamcmd”是服务端默认的安装路径，请根据实际安装路径进行替换。

   ```
   Start-Process "C:\steamcmd\steamcmd.exe" -ArgumentList "+login anonymous +app_update 2394010 validate +quit" -Wait
   ```



3. 服务器更新完成后，桌面上双击游戏快捷方式图标，重新启动游戏服务端。

Steam 客户端配置示例

1. 进入游戏，选择 "加入多人游戏（专用服务器）"

2. 输入上述云服务器绑定的弹性公网 IP 和端口，点击联机

3. 进入到创建人物界面，则说明连接服务器成功

建站教程

Centos-7.2部署LNMP环境

本文介绍了如何使用centos 7.2系统搭建LNMP环境。centos7.2搭建LNMP具体步骤如下：

1. 配置防火墙

CentOS 7.0以上的系统默认使用的是firewall作为防火墙。

关闭**firewall**：

```
 systemctl stop firewalld.service       #停止firewall
systemctl disable firewalld.service     #禁止firewall开机启动
```

2. 安装以及配置nginx

（1）可直接使用服务器内自带的yum源进行安装；

```
yum install -y nginx
```

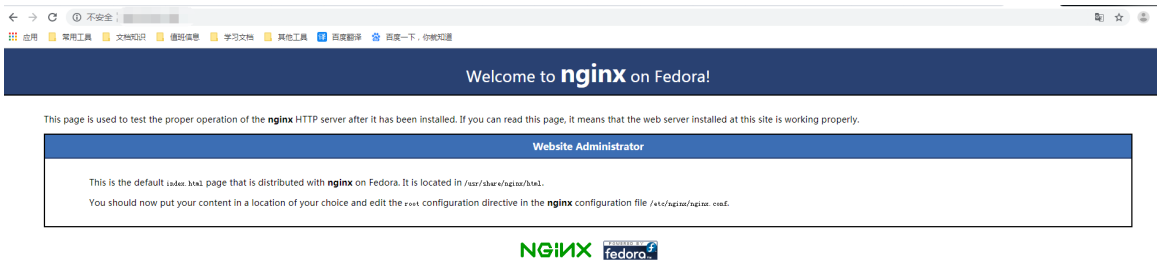
（2）修改nginx默认配置文件；

```
 cd    /etc/nginx/                       #nginx的默认配置目录
mv    nginx.conf    nginx.conf.swf          #将通过yum安装自动生成的配置文件换名，不再使用。
mv     nginx.conf.default  nginx.conf        #将default文件修改为加载配置文件
```

（3）启动nginx，并访问。

```
 systemctl start nginx.service                #启动nginx
systemctl stop nginx.service                 #停止
systemctl restart nginx.service              #重启
systemctl enable nginx.service
```

此时使用公网IP访问，即可打开nginx的默认欢迎页面。



3. 安装php环境以及依赖扩展，并开启php

（1）安装环境；

```
 yum install -y    php-fpm    php php-mysql php-gd libjpeg* php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring php-bcmath php-mhash
```

（2）启动php-fpm；

```
 systemctl start php-fpm.service            #启动php-fpm
systemctl enable php-fpm.service            #设置开机启动
```

（3）修改配置文件，使nginx兼容php；

`vim    /etc/nginx/nginx.conf`

在location内添加上index.php，将php识别之前的#注释去掉，并且改一下fastcgi_param。

```
 location / {
root html;
index index.php index.html index.htm;
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
root html;
}

location ~ \.php$ {
root html;
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME    $document_root$fastcgi_script_name;
include fastcgi_params;
}
```

```
server {
    listen       80;
    server_name  localhost;


    location / {
        root   html;
        index  index.php index.html index.htm;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }

    location ~ \.php$ {
        root           html;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include        fastcgi_params;
    }
}
```

（4）重启nginx和php-fpm；

```
systemctl restart php-fpm.service
systemctl restart nginx.service
```

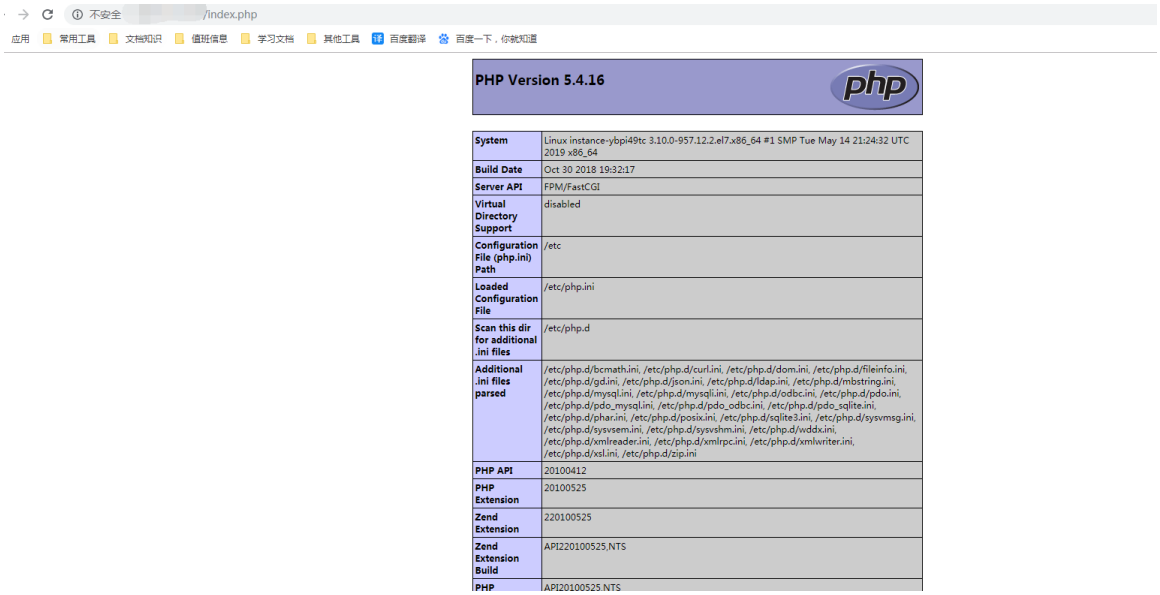
（5）访问phpinfo验证；

进入nginx默认根目录 `cd /usr/share/nginx/html/`

`vim index.php`

```
<?php
echo phpinfo();
?>
```

此时访问公网IP/index.php，可看到php的探针页面：



PHP Version 5.4.16

System	Linux instance-ybpi49tc 3.10.0-957.12.2.el7.x86_64 #1 SMP Tue May 14 21:24:32 UTC 2019 x86_64
Build Date	Oct 30 2018 19:32:17
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/bcmath.ini, /etc/php.d/curl.ini, /etc/php.d/dom.ini, /etc/php.d/fileinfo.ini, /etc/php.d/gd.ini, /etc/php.d/json.ini, /etc/php.d/ldap.ini, /etc/php.d/mbstring.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/odbc.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_odbc.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/posix.ini, /etc/php.d/sqlite3.ini, /etc/php.d/sysvmsg.ini, /etc/php.d/sysvsem.ini, /etc/php.d/sysvshm.ini, /etc/php.d/wddx.ini, /etc/php.d/xmlreader.ini, /etc/php.d/xmlrpc.ini, /etc/php.d/xmlwriter.ini, /etc/php.d/xsl.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP	API20100525,NTS

4. 安装mysql，并连接php进行验证

（1）centos7以上的系统，默认数据库为MariaDB，需要下载mysql源进行安装。

```
wget http://dev.mysql.com/get/mysql-community-release-el7-5.noarch.rpm
rpm -ivh mysql-community-release-el7-5.noarch.rpm
yum install -y mysql-community-server
```

成功安装之后重启mysql服务:

`systemctl start mysqld`

（2）安装后的mysql数据库是没有密码的，可进入数据库后授权，设置密码。

```
 mysql -u root
mysql> use mysql;
mysql> update user set password=PASSWORD("这里输入root用户密码") where user='root';
mysql> flush privileges;
mysql> exit
```

（3）编写php测试mysql是否可连接的代码，并访问。

`vim /usr/share/nginx/html/test.php`

```
 <?php
$link=mysql_connect("localhost","root","刚才所设置的数据库密码");
if(!$link) echo "FAILD!error";
else echo "OK!You succeeded.";
?>
```

保存后，此时访问IP/test.php，可看到

至此，基于centos7.2系统所安装的LNMP环境已经全部搭建完毕。

Centos-7.2部署LAMP环境

本文介绍了如何搭建LAMP环境，所涉及到的演示内容皆基于centos7.2版本的BCC实例，具体步骤如下。

1.配置防火墙

CentOS 7.0以上的系统默认使用的是firewall作为防火墙。

关闭firewall：

```
 systemctl stop firewalld.service    #停止firewall
systemctl disable firewalld.service      #禁止firewall开机启动
```

2.安装以及配置apache

（1）可直接使用服务器内自带的yum源进行安装；

`yum install -y httpd`

（2）启动httpd；

`systemctl start httpd`

（3）查看以及访问； 通过netstat -anplt可查看到80端口已经开启，使用公网IP也可打开apache的默认首页。

```
[root@instance-ybpi49tc html]# netstat -anplt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN      3520/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      2937/master
tcp        0      0 10.0.0.22:53910         100.67.65.19:80         ESTABLISHED 3418/./bcm-agent
tcp        0     52 10.0.0.22:2222          61.135.169.80:61156     ESTABLISHED 3528/sshd: root@pts
tcp        0      0 127.0.0.1:48680         127.0.0.1:781           ESTABLISHED 3430/./bcm-si
tcp        0      0 10.0.0.22:45568         100.67.65.16:1814       ESTABLISHED 3100/hosteye
tcp        0      0 10.0.0.22:57020         100.67.65.16:1813       ESTABLISHED 3100/hosteye
tcp6       0      0 :::3306                 :::*                    LISTEN      3965/mysqld
tcp6       0      0 :::781                  :::*                    LISTEN      3418/./bcm-agent
tcp6       0      0 :::2222                 :::*                    LISTEN      3520/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      4133/httpd
tcp6       0      0 ::1:25                  :::*                    LISTEN      2937/master
tcp6       0      0 127.0.0.1:781           127.0.0.1:48680         ESTABLISHED 3418/./bcm-agent
tcp6       0      0 127.0.0.1:781           127.0.0.1:48676         TIME_WAIT   -
tcp6       0      0 127.0.0.1:781           127.0.0.1:48678         TIME_WAIT   -
```

3.安装php以及依赖扩展，并开启php

（1）安装php环境；

`yum install -y php`

（2）安装扩展项；

`yum install -y php-gd php-mysql libjpeg* php-imap php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring php-mcrypt php-bcmath php-mhash libmcrypt`

（3）修改apache配置文件；

- 通过httpd -V命令可查看到apache的默认配置文件为：`/etc/httpd/conf/httpd.conf`

```
Server compiled with....
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D DYNAMIC_MODULE_LIMIT=256
 -D HTTPD_ROOT="/etc/httpd"
 -D SUEXEC_BIN="/usr/sbin/suexec"
 -D DEFAULT_PIDLOG="/run/httpd/httpd.pid"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_ERRORLOG="logs/error_log"
 -D AP_TYPES_CONFIG_FILE="conf/mime.types"
 -D SERVER_CONFIG_FILE="conf/httpd.conf"
[root@instance-ybpi49tc html]#
```

`vim /etc/httpd/conf/httpd.conf`

- 在DirectoryIndex后添加上`index.php`
- 在配置文件最后，添加上：

```
LoadModule php5_module modules/libphp5.so

AddType application/x-httpd-php .php
```

```
</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.php index.html
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
```

（4）重启apache，并访问测试。

`systemctl restart httpd`

进入apache的默认跟目录/var/www/html/，编写phpinfo文件。

`vim /var/www/html/index.php`

```
<?php
    phpinfo();
?>
```

此时通过ip/index.php即可看到

4.安装mysql，并连接php进行验证

（1）centos7以上的系统，默认数据库为MariaDB，需要下载mysql源进行安装。

```
wget http://dev.mysql.com/get/mysql-community-release-el7-5.noarch.rpm
rpm -ivh mysql-community-release-el7-5.noarch.rpm
yum install -y mysql-community-server
```

成功安装之后重启mysql服务:

`systemctl start mysqld`

（2） 安装后的mysql数据库是没有密码的，可进入数据库后授权，设置密码。

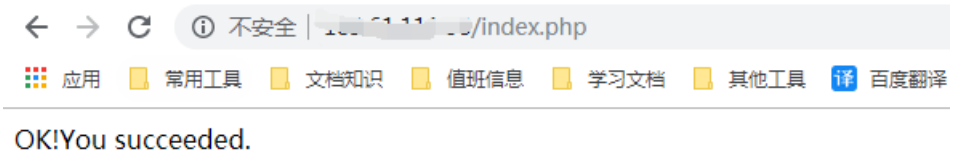
```
mysql -u root
mysql> use mysql;
mysql> update user set password=PASSWORD("这里输入root用户密码") where user='root';
mysql> flush privileges;
mysql> exit
```

（3）编写php测试mysql是否可连接的代码，并访问。

`vim /var/www/html/index.php`

```
 <?php
$link=mysql_connect("localhost","root","刚才所设置的数据库密码");
if(!$link) echo "FAILD!error";
else echo "OK!You succeeded.";
?>
```

此时使用IP/index.php，即可看到



至此，基于centos7.2系统所安装的LAMP环境已经全部搭建完毕。

云服务器部署SSL证书-Nginx

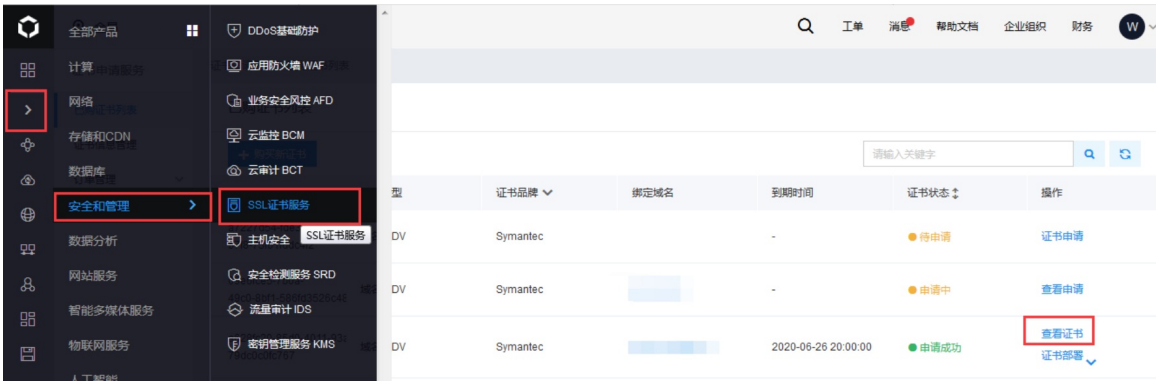
本文介绍百度智能云“在Nginx服务中部署SSL证书”的方法（含操作指导视频），您可以参考以下视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《CentOS安装Nginx服务并部署SSL证书》

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

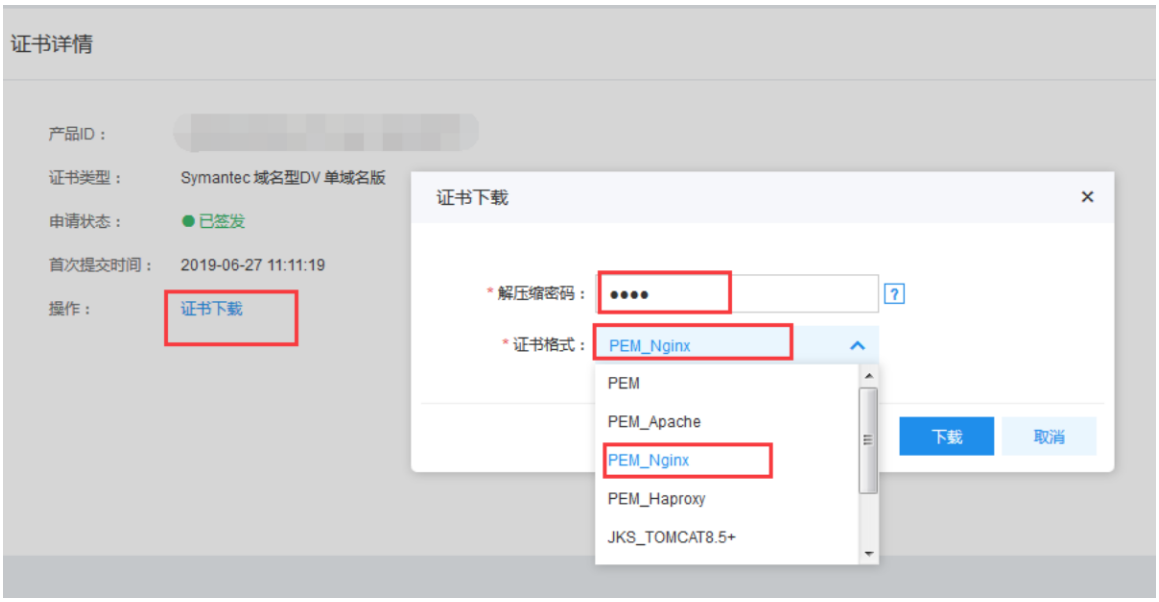
基于Centos-7.2部署LNMP环境 教程所搭建的Nginx环境，为web服务配置SSL证书（需要提前准备好域名）。以下步骤为配置部署证书的全部过程。

下载证书并解压

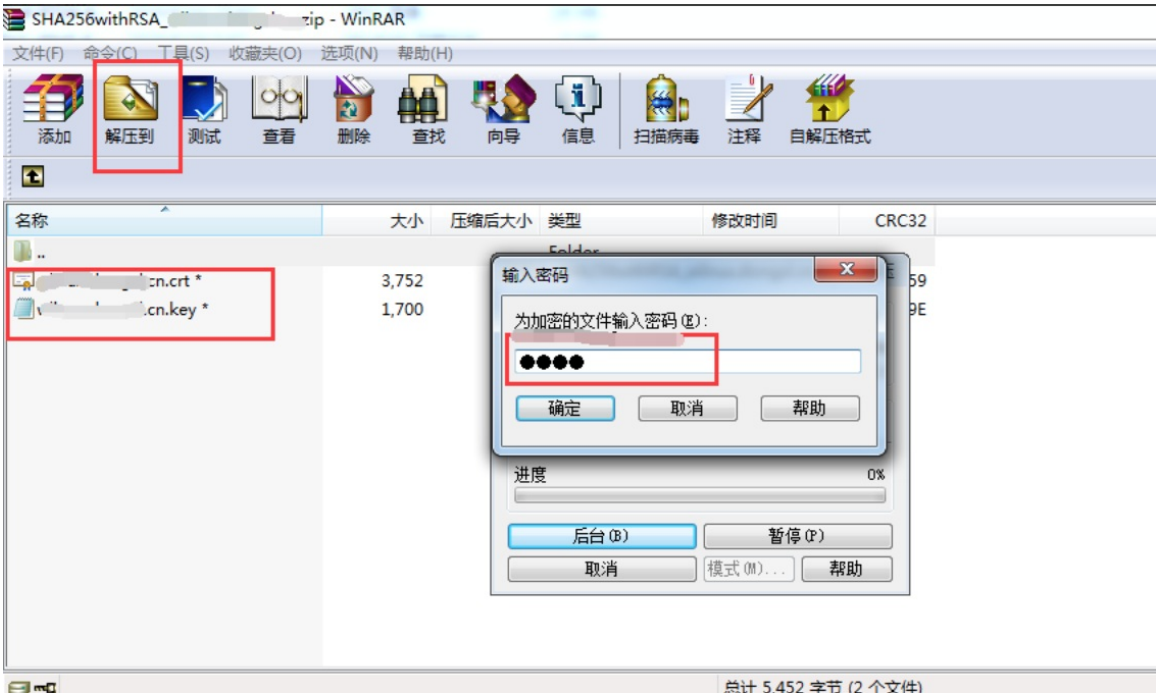
进入百度智能云控制台--安全和管理--SSL证书服务–已购证书列表中，点击之前基于该服务器所绑定域名申请的证书信息--查看证书。



点击下载证书，选择PEM_Nginx格式，并设置四位数字的解压缩密码。

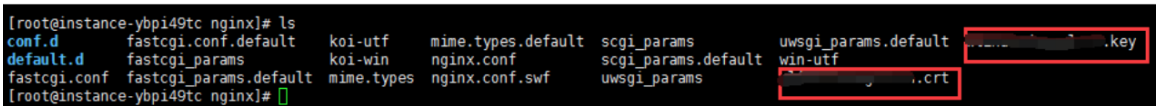


在本地下载完毕后，双击打开，可看到有crt以及key两个文件，选择解压到，设置路径后，输入刚在控制台上设置的四位数密码。



上传证书文件到服务器中

使用FTP或者其他工具将上一步中解压得到的crt以及key文件上传到服务器的nginx配置目录/etc/nginx/中。

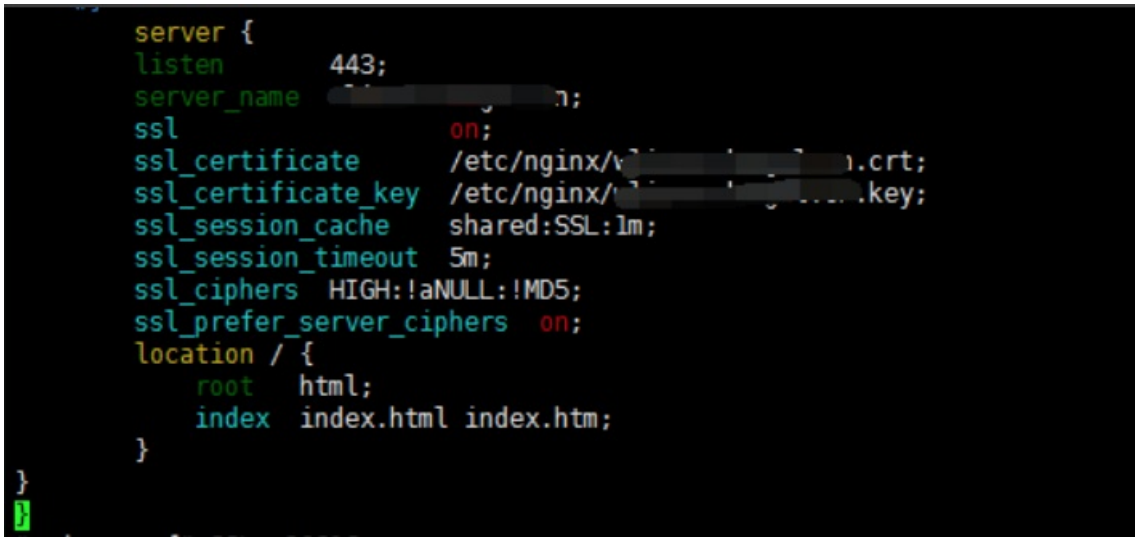


修改配置文件

```
vim  /etc/nginx/nginx.conf
```

添加上以下虚拟主机配置，或者取消HTTPS主机的注释。

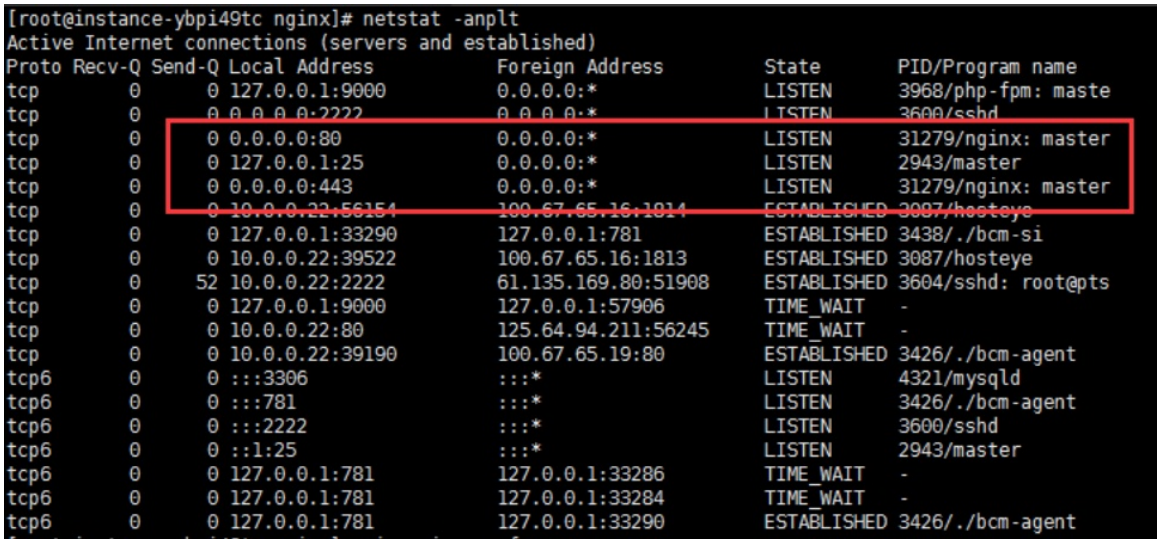
```
 server {
  listen       443;
  server_name  ********.com;   #更换上所绑定的域名，一定要是申请了证书的域名
  ssl              on;    #这一行是另外添加的，意思是打开ssl功能，一定要添加。
  ssl_certificate      /etc/nginx/********.com.crt; #这是下载下来的nginx证书的crt文件路径，绝对或者相对路径都可以
  ssl_certificate_key  /etc/nginx/*********.com.key;  #和crt的规则一样
  ssl_session_cache    shared:SSL:1m;
  ssl_session_timeout  5m;
  ssl_ciphers  HIGH:!aNULL:!MD5;
  ssl_prefer_server_ciphers  on;
  location / {
     root   html;
     index  index.html index.htm;
  }
}
```



保存配置后，重启nginx服务

```
 systemctl restart nginx
```

此时可通过netstat -anplt查看到443端口已经开放。使用https://域名也可正常打开网站。

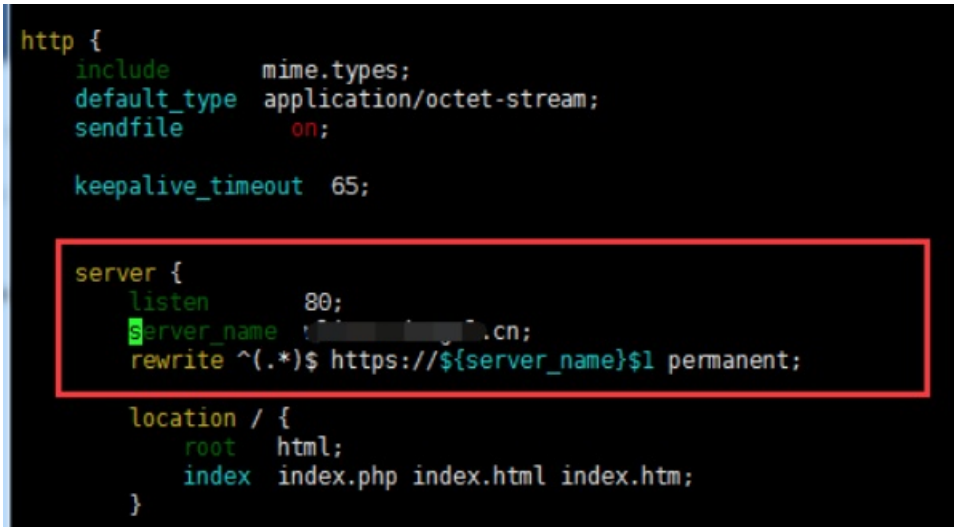


设置http强制跳转https

```
vim  /etc/nginx/nginx.conf
```

在80的server中，先将server_name后的localhost修改为此证书域名。然后在下面添加上一句：

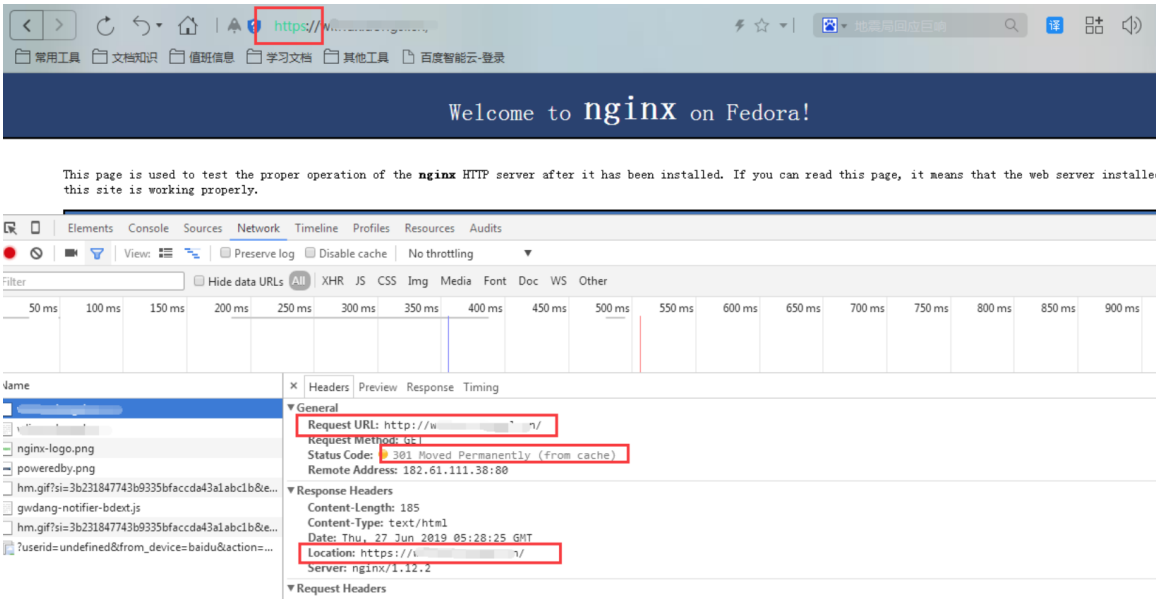
```
rewrite ^(.*)$ https://${server_name}$1 permanent;
```



保存后，重启nginx：

```
systemctl restart nginx
```

此时在直接访问域名时，会全部301跳转到https请求上。



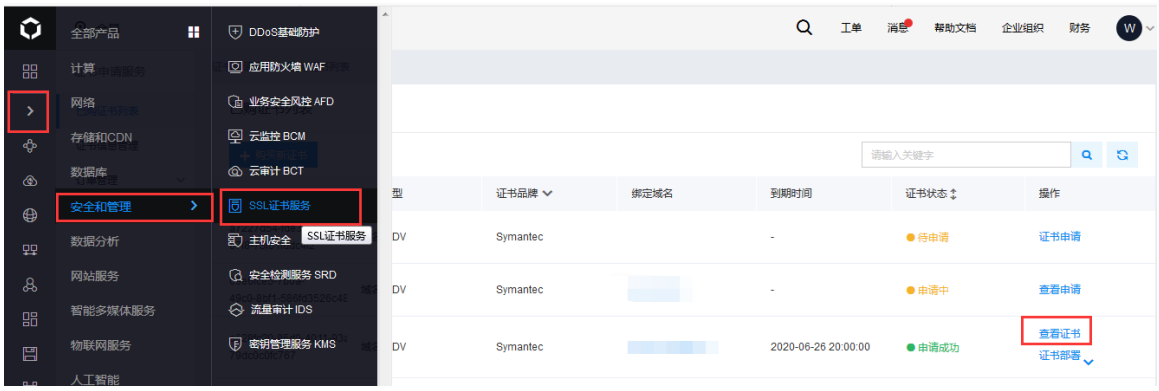
云服务器部署SSL证书-Apache

基于Centos-7.2部署LAMP环境所搭建的apache环境，为Web服务配置SSL证书（需要提前准备好域名）。以下步骤为配置部署

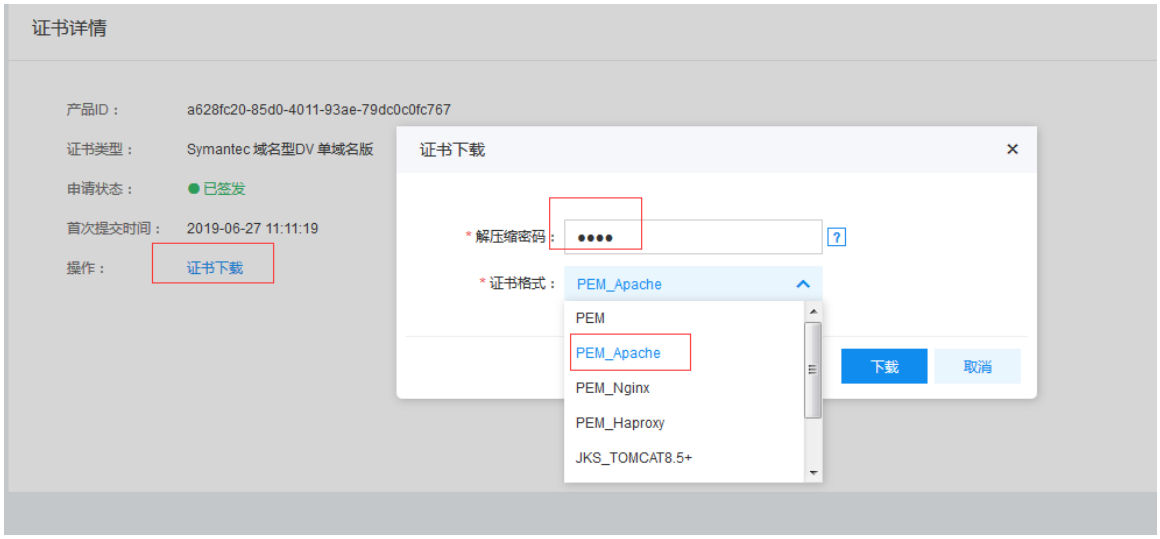
证书的全部过程。

1.下载证书并解压

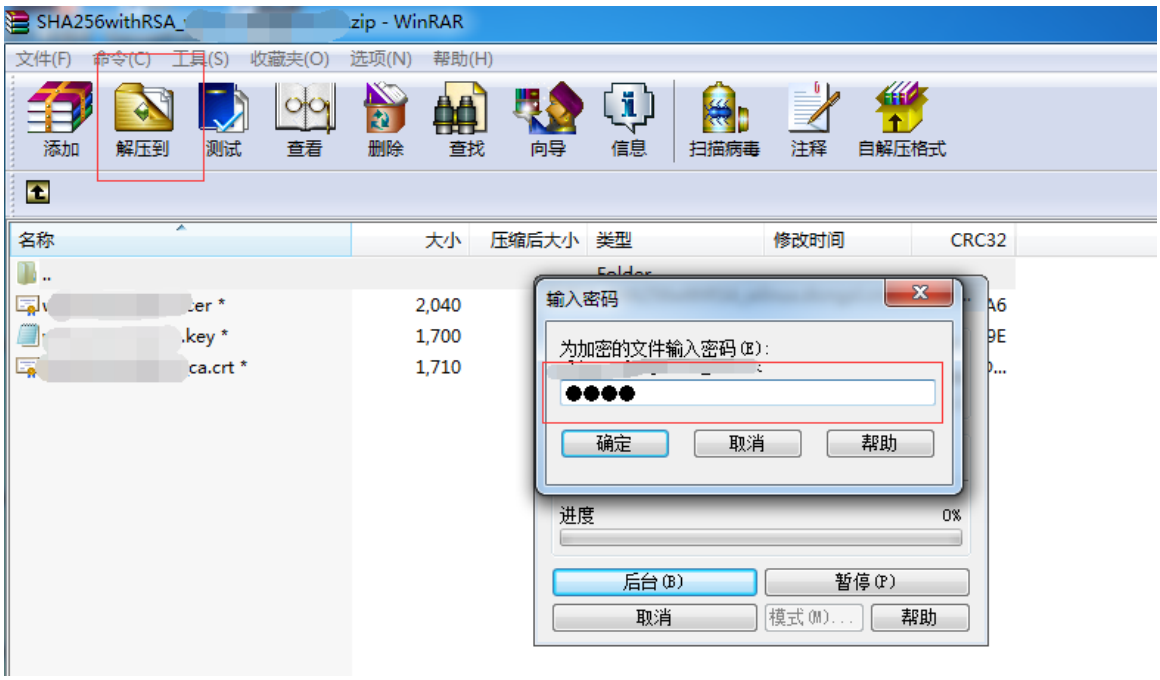
进入百度智能云控制台--安全和管理--SSL证书服务--已购证书列表中，点击之前基于该服务器所绑定域名申请的证书信息--查看证书。



点击下载证书，选择PEM_Apache格式，并设置四位数字的解压缩密码。



在本地下载完毕后，双击打开，可看到有cer、crt以及key三个文件，选择解压到，设置路径后，输入刚在控制台上设置的四位数密码。



2.上传证书文件到服务器中

使用FTP或者其他工具将上一步中解压得到的cer、crt以及key文件上传到服务器的apche配置目录/etc/httpd/conf中

```
[root@instance-ybpi49tc conf]# ls
httpd.conf  magic  [redacted]_ca.crt  [redacted].cer  [redacted].key
[root@instance-ybpi49tc conf]#
```

3.安装mod_ssl，并配置证书

因为是使用yum直接安装的apache，无法在安装的时候，直接编译添加ssl模块，需要再进行安装。

`yum install mod_ssl openssl`

安装之后，在/etc/httpd/conf.d下多了一个ssl.conf文件。

`vim /etc/httpd/conf.d/ssl.conf`

搜索ServerName 取消该行的注释，并修改域名为证书域名。

将证书三个文件的位置路径填写正确

```
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/httpd/conf.d/[redacted].cer

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/httpd/conf.d/[redacted].key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
SSLCertificateChainFile /etc/httpd/conf.d/[redacted]_ca.crt

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
```

重启httpd后，可通过https://域名打开该网站

4.http强制跳转https

因为是yum安装的，在apache的配置文件中没有ssl.so和httpd-ssl.conf模块路径，可手动添加rewrite规则实现。

`vim vim /etc/httpd/conf/httpd.conf`

在`<Directory>`

```
  ......
</Directory>
```

中添加：

```
  RewriteEngine on
RewriteCond %{SERVER_PORT} !^443$
RewriteRule ^(.*)?$ https://%{SERVER_NAME}/$1 [L,R]
```

```
# Further relax access to the default document root:
<Directory "/var/www/html">
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
    RewriteEngine on
    RewriteCond %{SERVER_PORT} !^443$
    RewriteRule ^(.*)?$ https://%{SERVER_NAME}/$1 [L,R]
</Directory>
```

再次重启apache后，访问网站时，http会强制跳转到https。

云服务器部署SSL证书-tomcat

安装JKS格式证书

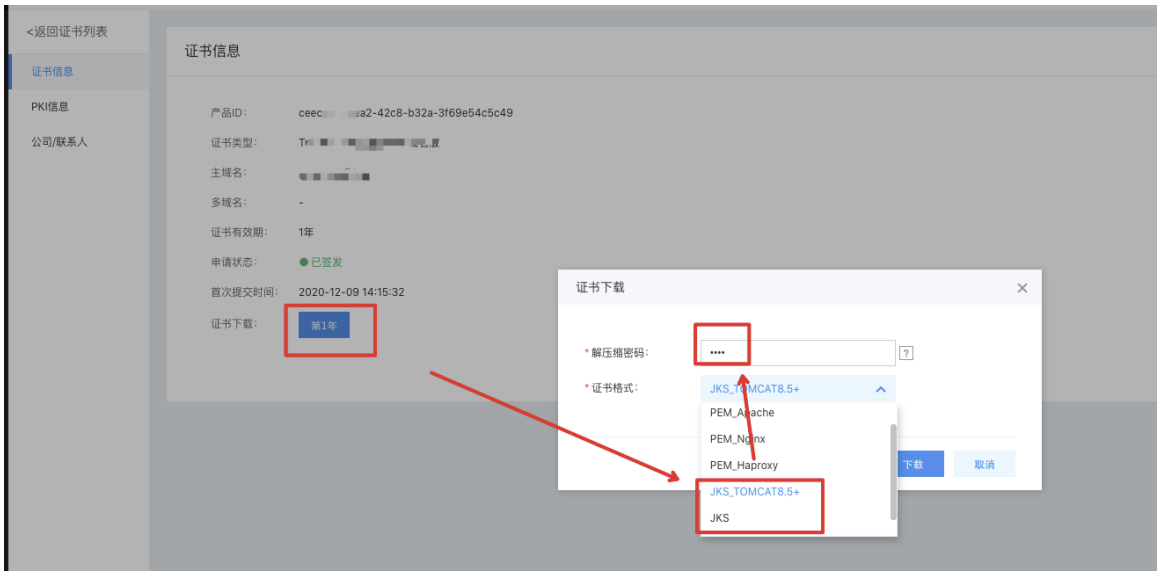
您可以将下载的证书安装到Tomcat服务器上。Tomcat支持PFX格式和JKS两种格式的证书，您可根据您Tomcat的版本选择其中一种格式的证书安装到Tomcat上。本文档介绍了JKS格式证书安装的具体步骤。

背景信息

本文档证书名称以domain.com为示例，如证书文件名称为domain.com.jks（百度智能云中的jks格式和pfx格式证书部署可以填写解压密码）。

操作指南

1. 登录百度智能云SSL证书控制台。
2. 在SSL证书页面，定位到需要下载的证书并单击证书条目右下角的查看证书



3. 打开后点击证书下载对话框。选择JKS格式并且键入证书压缩密码
4. 在Tomcat安装目录下新建cert目录，将证书文件拷贝到cert目录下
5. 打开Tomcat安装目录 > conf文件夹 > server.xml文件，在server.xml文件中找到 <Connector port=”8443”标签并进行修改。

参考以下完整配置（其中port属性请根据您的实际情况修改）：<Connector port="443" protocol="HTTP/1.1" SSLEnabled="true" scheme="https" secure="true" keystoreFile="cert/domain.jks" #此处keystoreFile代表证书文件的路径，请用您证书的文件名替换domain。 keystoreType="PKCS12" keystorePass="证书密码" #请用您证书密码文件中的密码替换“证书密码”。 clientAuth="false" SSLProtocol="TLSv1+TLSv1.1+TLSv1.2" ciphers="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA256"/>

6. 保存server.xml文件配置。
7. （可选步骤）配置web.xml文件开启HTTP强制跳转HTTPS。

在< /welcome-file-list >后添加以下内容：

```
 <login-config>
  <!-- Authorization setting for SSL -->
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>Client Cert Users-only Area</realm-name>
</login-config>
<security-constraint>
    <!-- Authorization setting for SSL -->
    <web-resource-collection>
        <web-resource-name>SSL</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
         <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

8. 重启Tomcat。

9. 验证。

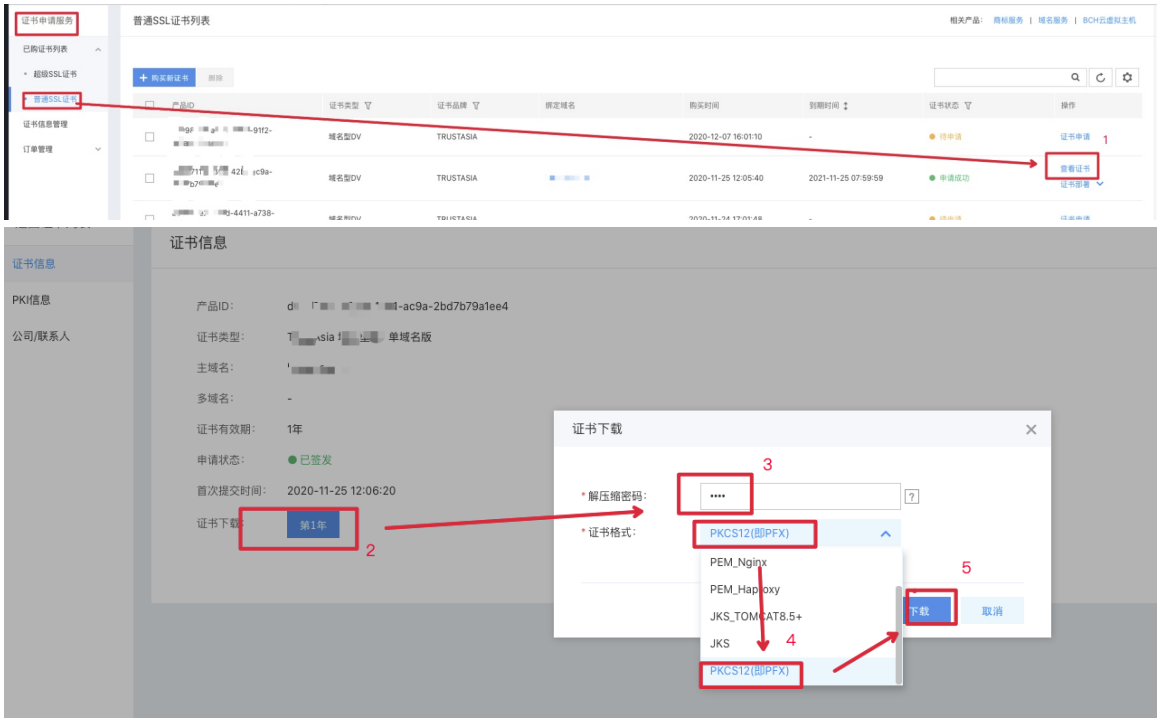
证书安装完成后，可通过登录证书绑定域名的方式验证证书是否安装成功。

在浏览器中输入地址 `https://domain:port`，将domain替换成证书绑定的域名，默认443端口可以忽略不输入。如果网页地址栏出现绿色小锁标志，表示证书安装成功。

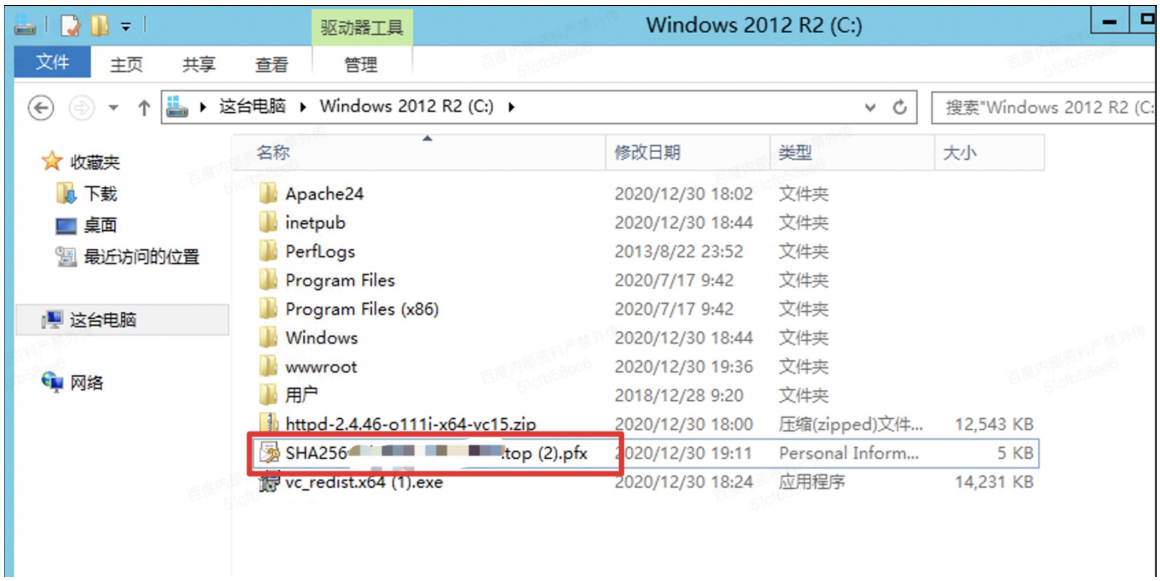
> 验证证书是否安装成功时，如果网站无法通过https正常访问，需确认您安装证书的服务器443端口是否已开启或被其他工具拦截。

云服务器部署SSL证书-IIS

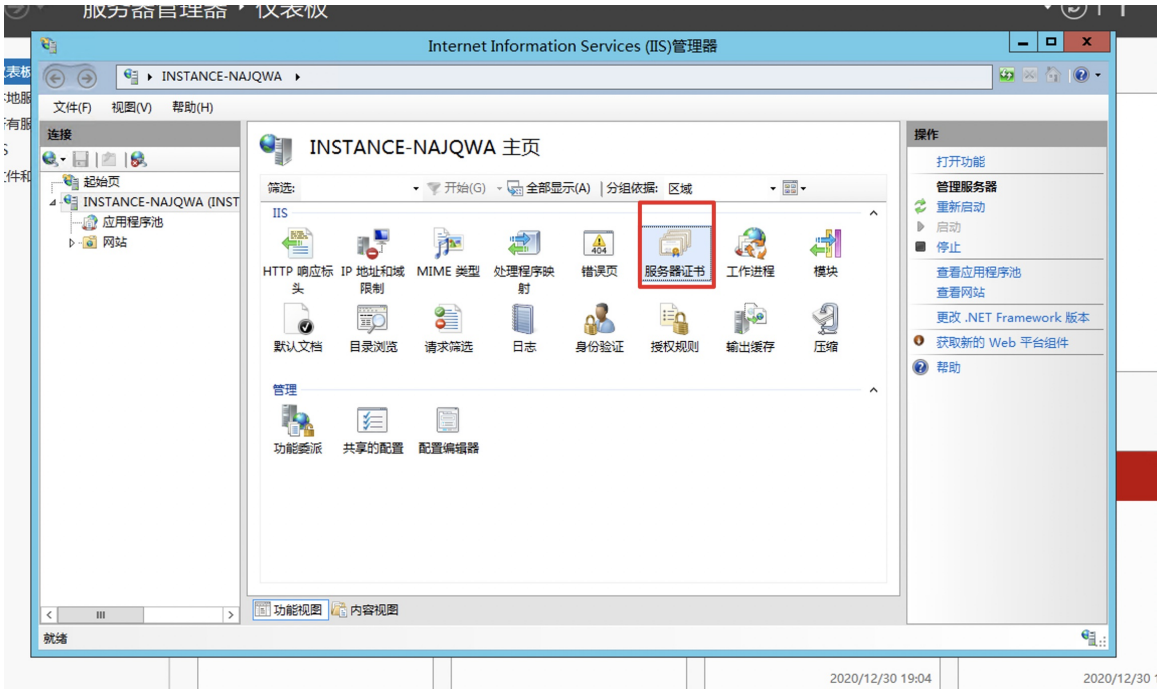
1. 您需要先将IIS要部署的证书下载到服务器，IIS需要使用.pfx格式证书，百度智能云证书下载方法如图：



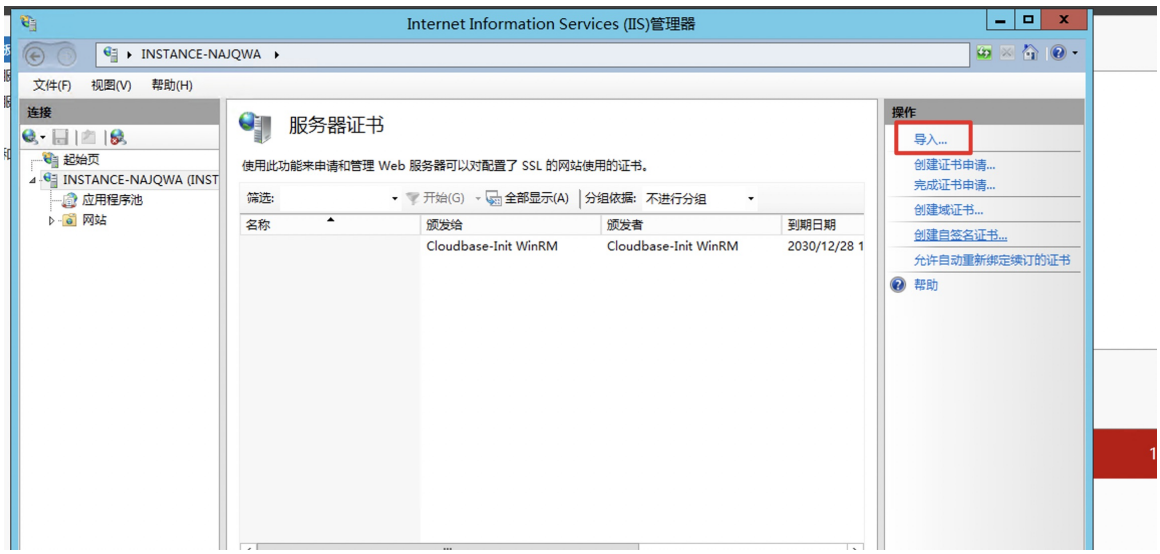
2. 将证书上传到服务器中，如图：

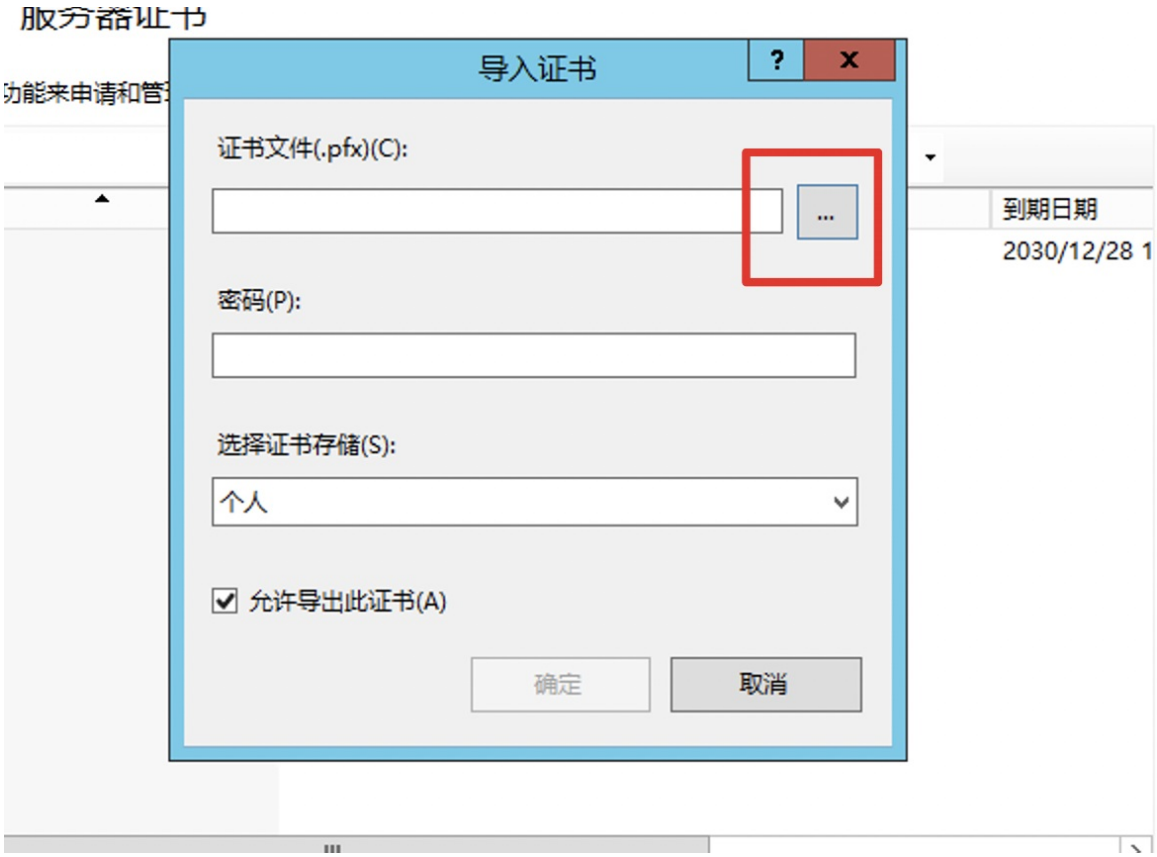


3. 打开IIS管理器--服务器证书，如图：

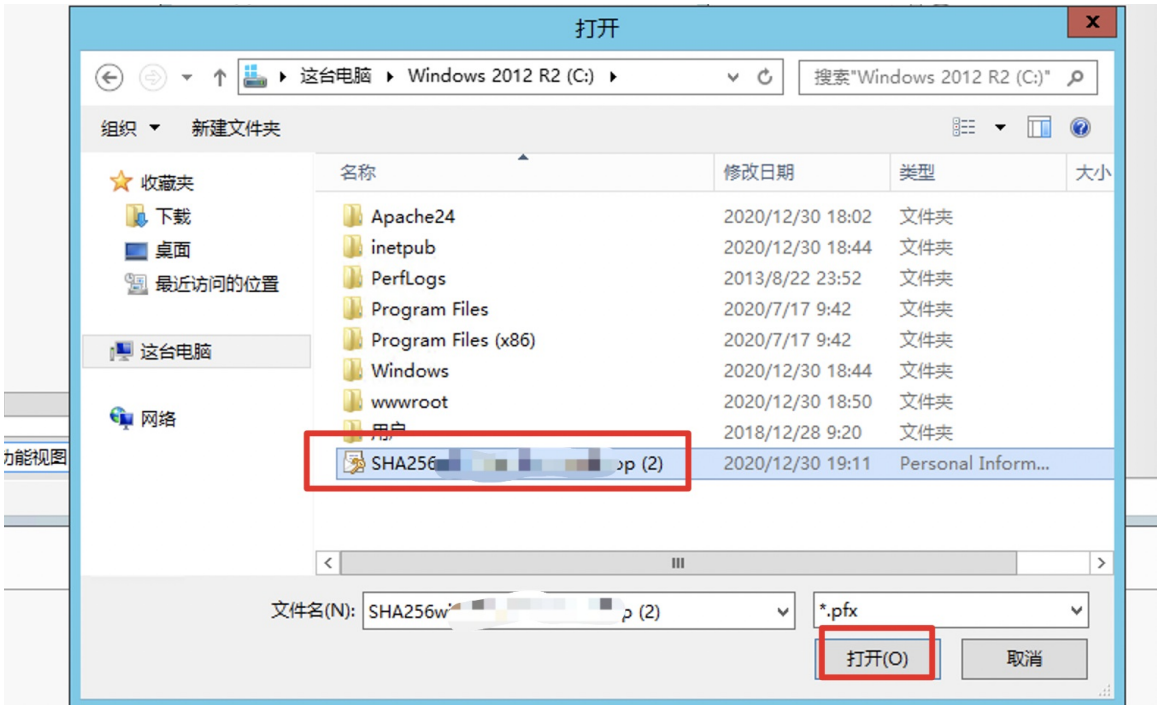


4. 导入证书：

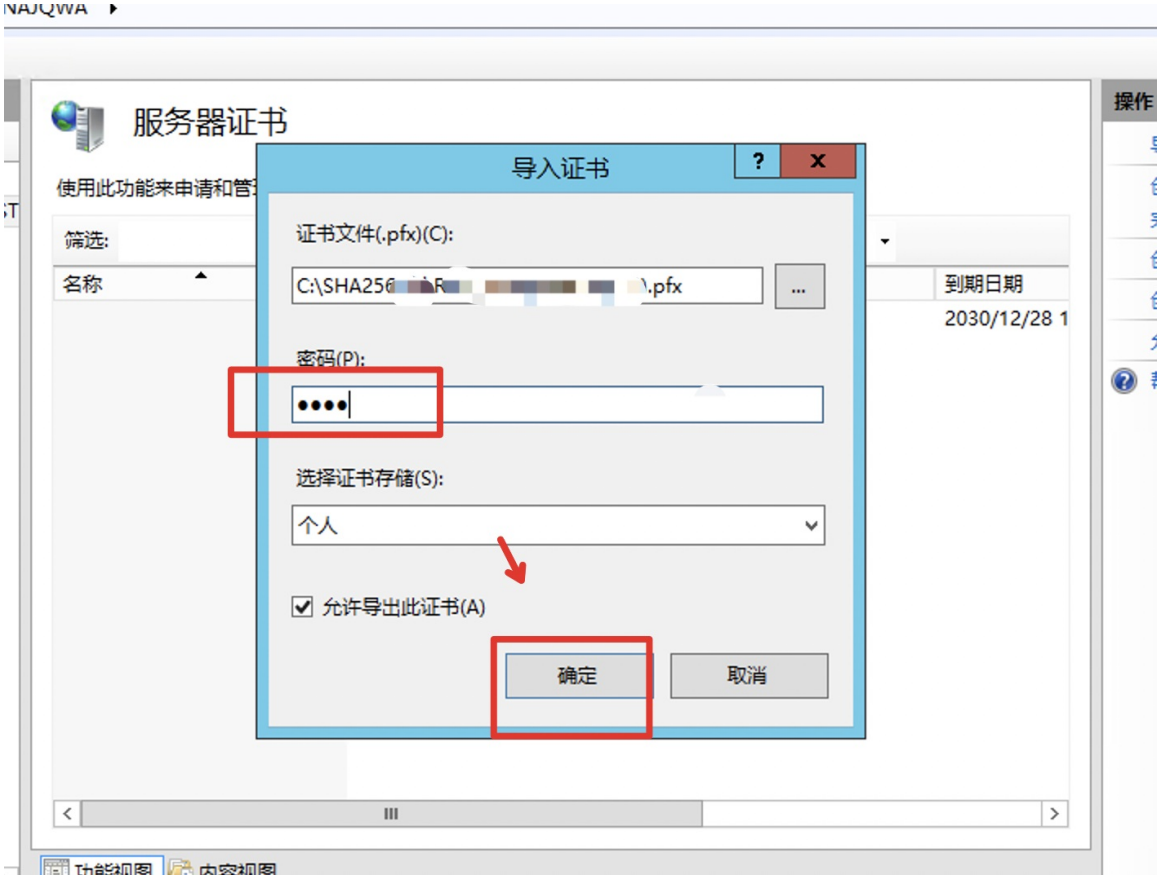


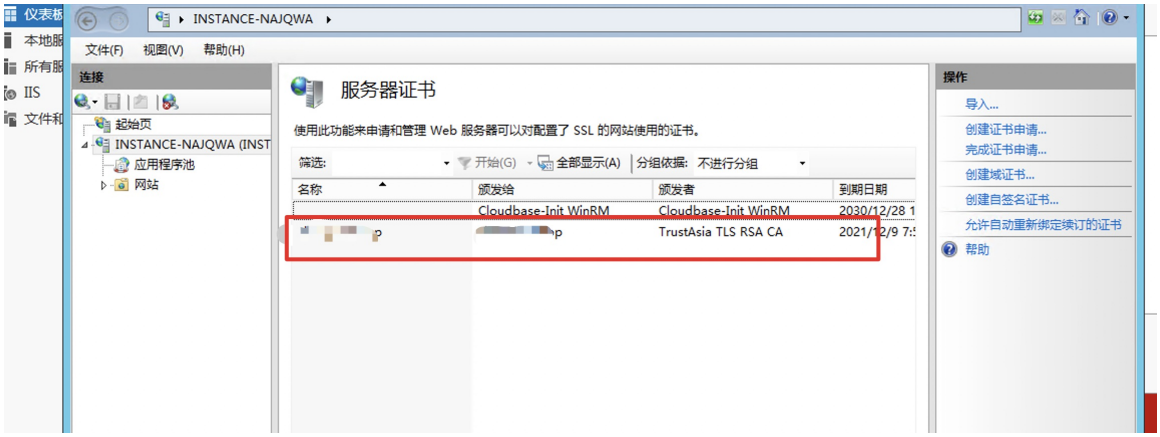


选择已经下载好的证书：

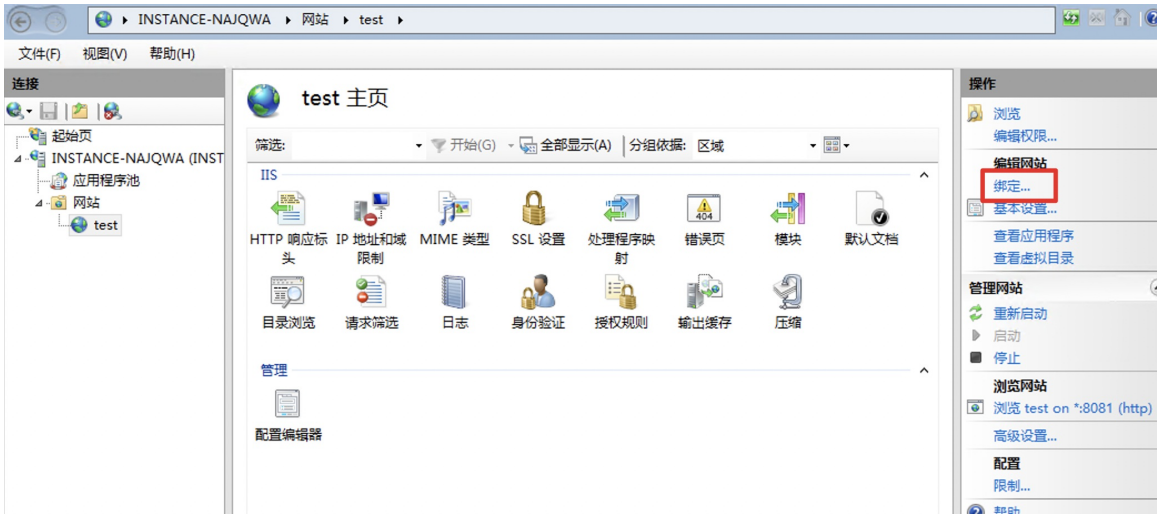


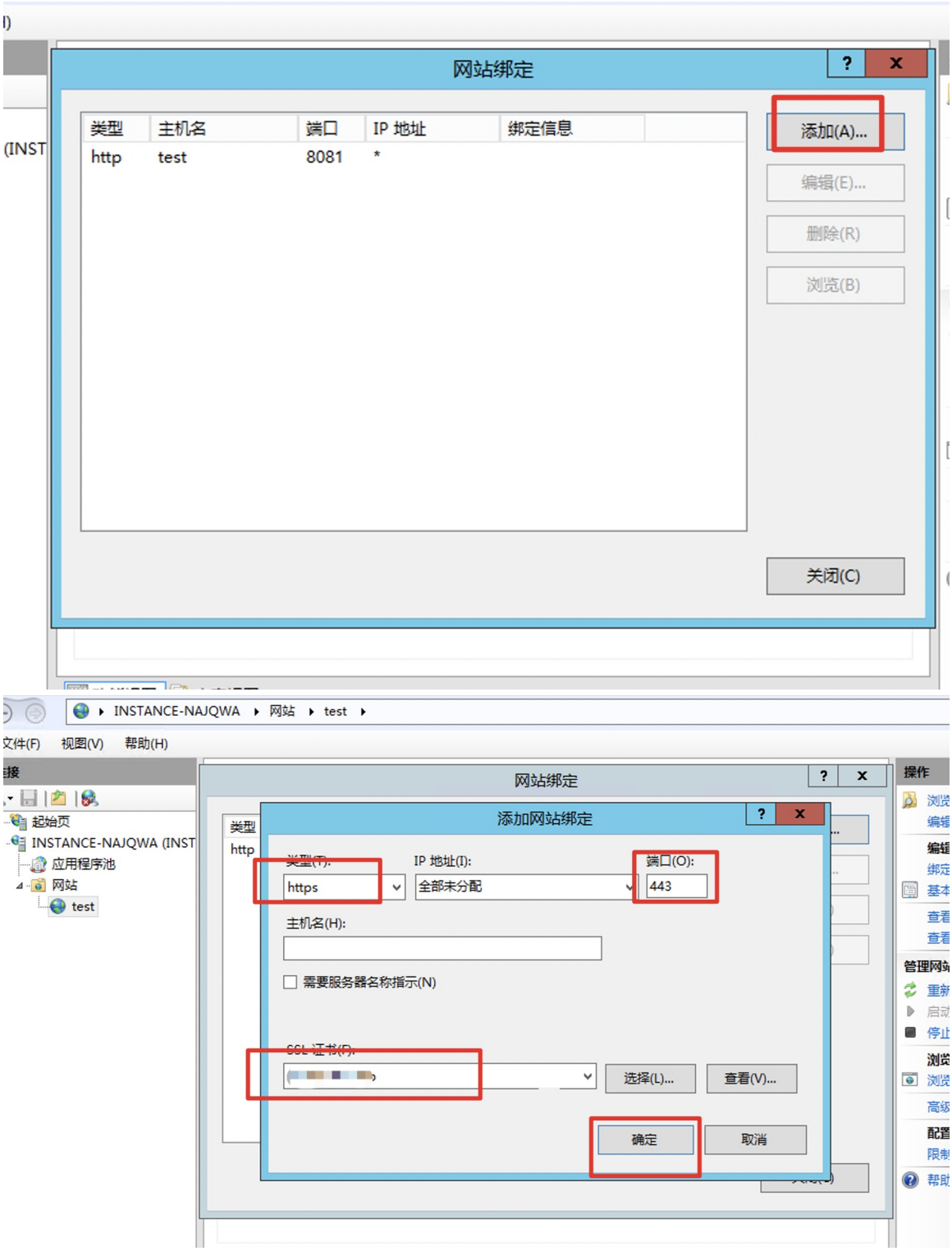
密码为下载证书时的“解压缩密码”：


服务器证书
导入证书
证书文件(.pfx)(C):
密码(P):
选择证书存储(S):
个人
允许导出此证书(A)
确定
取消
操作
筛选:
名称
到期日期
2030/12/28 1

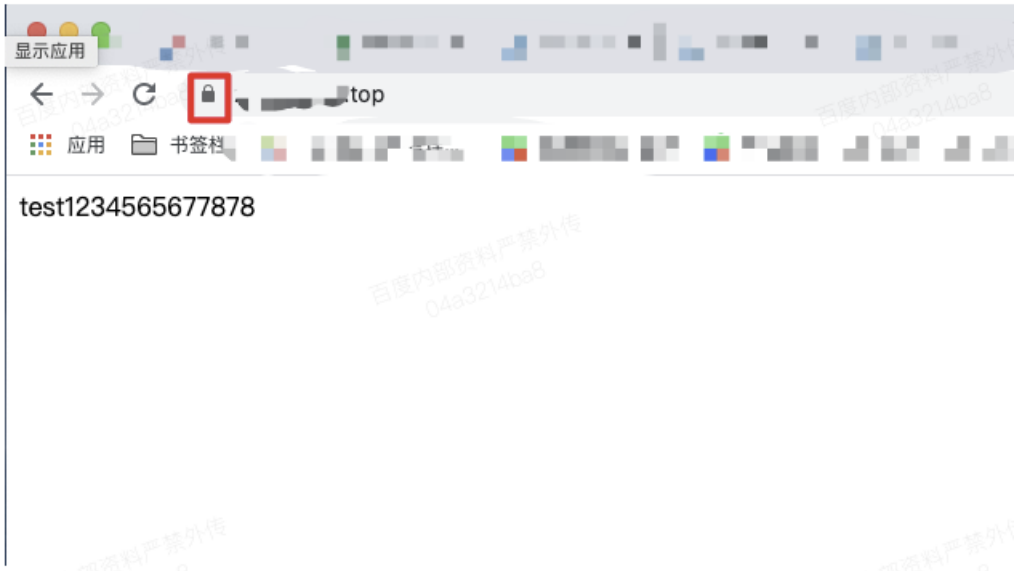

INSTANCE-NAJQWA
服务器证书
使用此功能来申请和管理 Web 服务器可以对配置了 SSL 的网站使用的证书。
Cloudbase-Init WinRM
TrustAsia TLS RSA CA
操作
导入...
创建证书申请...
完成证书申请...
创建域证书...
创建自签名证书...
允许自动重新绑定续订的证书
帮助

5. 给站点绑定证书：


INSTANCE-NAJQWA ▸ 网站 ▸ test
test 主页
IIS
管理
操作
浏览
编辑权限...
编辑网站
绑定...
基本设置...
查看应用程序
查看虚拟目录
管理网站
重新启动
启动
停止
浏览网站
浏览 test on *:8081 (http)
高级设置...
配置
限制...



6. 验证https访问，没有不安全提示，可以正常使用https访问：



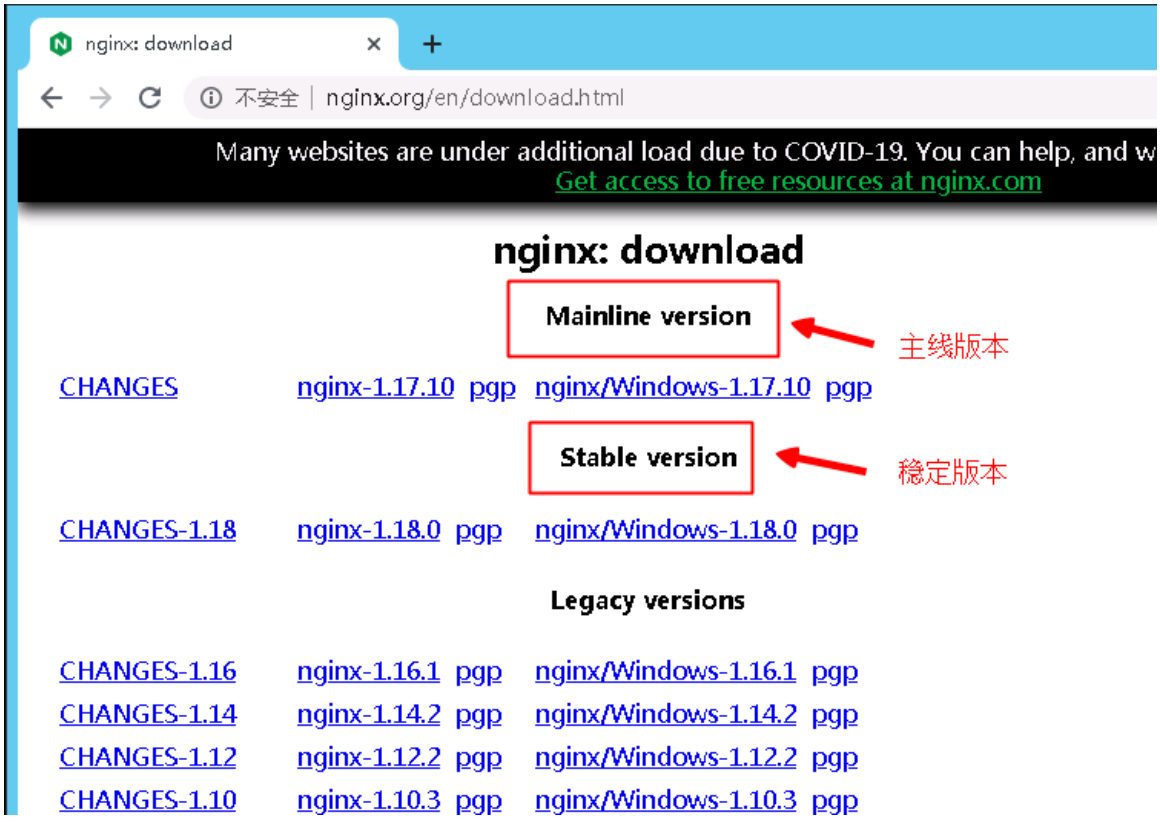
windows安装配置Nginx环境

1. 关闭IE安全增强模式

Windows服务器自带的IE浏览器默认开启了 IE增强模块，为了方便稍后的资源下载以及测试访问，现将该功能关闭，操作步骤可参考文档链接：https://cloud.baidu.com/doc/BCC/s/pjxlpzqce 关闭后，可根据需求继续使用IE浏览器，或者下载其他浏览器工具，如下面步骤中所使用的chrome浏览器。

2. 下载nginx

进入nginx官方下载地址：http://nginx.org/en/download.html 下载稳定版本，下图以nginx/Windows-1.18.0为例



下载后解压，获取到如下文件

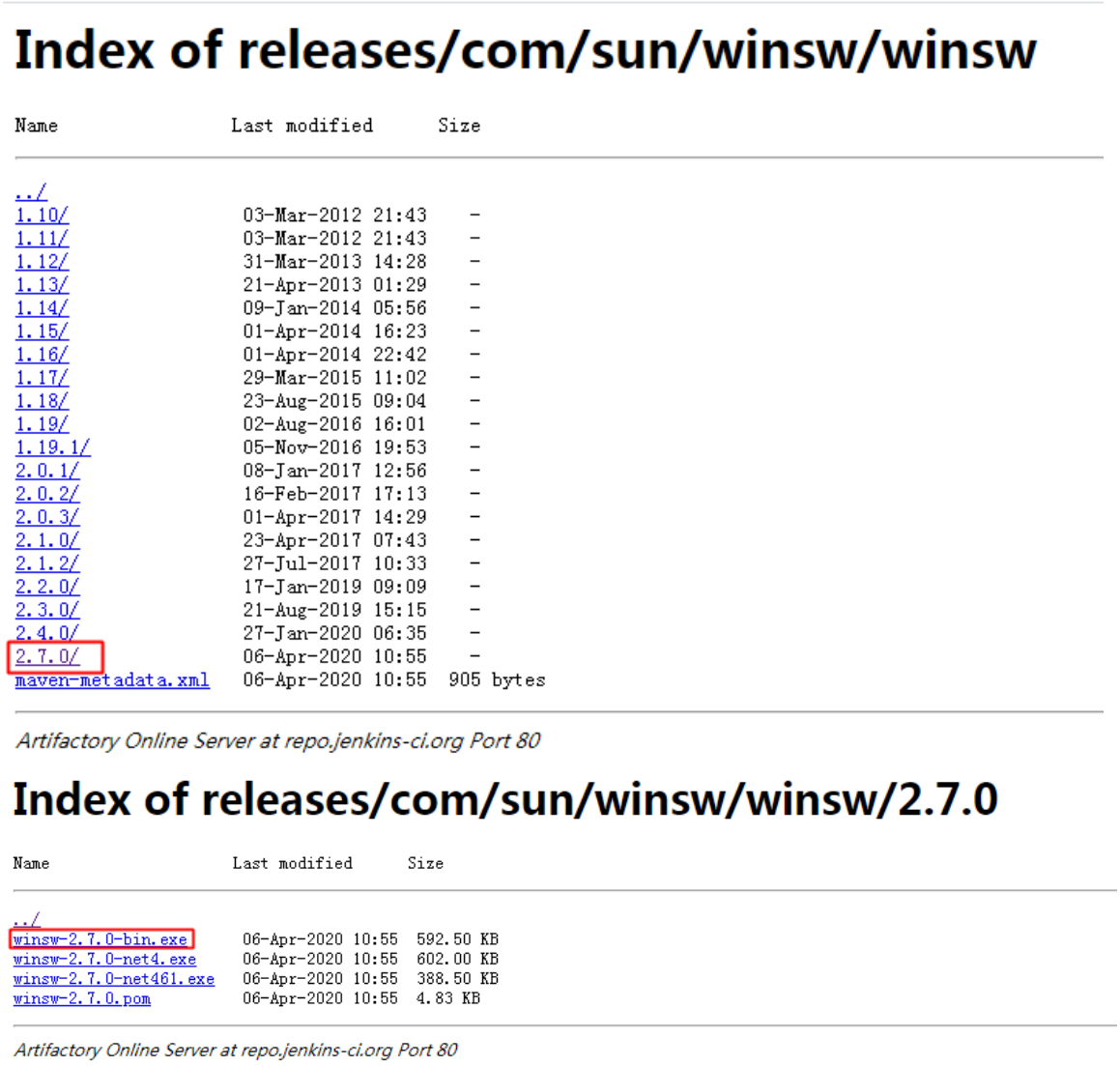
名称	修改日期	类型	大小
conf	2019/8/13 16:42	文件夹	
contrib	2019/8/13 16:42	文件夹	
docs	2019/8/13 16:42	文件夹	
html	2019/8/13 16:42	文件夹	
logs	2019/8/13 16:42	文件夹	
temp	2019/8/13 16:42	文件夹	
nginx	2020/5/13 15:46	应用程序	3,611 KB

3. 安装nginx并启动

直接双击nginx.exe，双击后一个黑色的弹窗一闪而过；或者打开cmd命令窗口，切换到nginx解压目录下，输入命令 nginx.exe 或者 start nginx ，输入回车；因为通过命令进行nginx的启动和停止比较麻烦，不方便后期维护，也可将nginx直接添加进windows的services服务列表内，进行开启和停止，可参考步骤4

4. 将nginx添加进windows服务

下载Wrapper服务：http://repo.jenkins-ci.org/releases/com/sun/winsw/winsw/ 该工具可设置nginx为windows服务，选择合适的版本，此处选择最新的2.7.0-bin版本，下载页面如下图：



Index of releases/com/sun/winsw/winsw

Name	Last modified	Size
../		
1.10/	03-Mar-2012 21:43	-
1.11/	03-Mar-2012 21:43	-
1.12/	31-Mar-2013 14:28	-
1.13/	21-Apr-2013 01:29	-
1.14/	09-Jan-2014 05:56	-
1.15/	01-Apr-2014 16:23	-
1.16/	01-Apr-2014 22:42	-
1.17/	29-Mar-2015 11:02	-
1.18/	23-Aug-2015 09:04	-
1.19/	02-Aug-2016 16:01	-
1.19.1/	05-Nov-2016 19:53	-
2.0.1/	08-Jan-2017 12:56	-
2.0.2/	16-Feb-2017 17:13	-
2.0.3/	01-Apr-2017 14:29	-
2.1.0/	23-Apr-2017 07:43	-
2.1.2/	27-Jul-2017 10:33	-
2.2.0/	17-Jan-2019 09:09	-
2.3.0/	21-Aug-2019 15:15	-
2.4.0/	27-Jan-2020 06:35	-
2.7.0/	06-Apr-2020 10:55	-
maven-metadata.xml	06-Apr-2020 10:55	905 bytes

Artifactory Online Server at repo.jenkins-ci.org Port 80

Index of releases/com/sun/winsw/winsw/2.7.0

Name	Last modified	Size
../		
winsw-2.7.0-bin.exe	06-Apr-2020 10:55	592.50 KB
winsw-2.7.0-net4.exe	06-Apr-2020 10:55	602.00 KB
winsw-2.7.0-net461.exe	06-Apr-2020 10:55	388.50 KB
winsw-2.7.0.pom	06-Apr-2020 10:55	4.83 KB

Artifactory Online Server at repo.jenkins-ci.org Port 80

将下载的winsw-2.7.0-bin.exe文件放在nginx的安装目录（此处为：C:\nginx-1.18.0\），并修改名称为：nginx-service.exe ，然后创建nginx-service.exe.config，nginx-service.xml文件，把这两个文件放在Nginx安装目录下，如下图

nginx-service.exe.config内容如下：

```
<configuration>
<startup>
<supportedRuntime version="v2.0.50727" />
<supportedRuntime version="v4.0" />
</startup>
<runtime>
<generatePublisherEvidence enabled="false"/>
</runtime>
</configuration>
```

nginx-service.xml内容如下：

```
<service>
<id>nginx</id>
<name>Nginx Service</name>
<description>High Performance Nginx Service</description>
<logpath>C:\nginx-1.18.0\logs</logpath>
<log mode="roll-by-size">
<sizeThreshold>10240</sizeThreshold>
<keepFiles>8</keepFiles>
</log>
<executable>C:\nginx-1.18.0\nginx.exe</executable>
<startarguments>-p C:\nginx-1.18.0\</startarguments>
<stopexecutable>C:\nginx-1.18.0\nginx.exe</stopexecutable>
<stoparguments>-p C:\nginx-1.18.0\nginx -s stop</stoparguments>
</service>
```

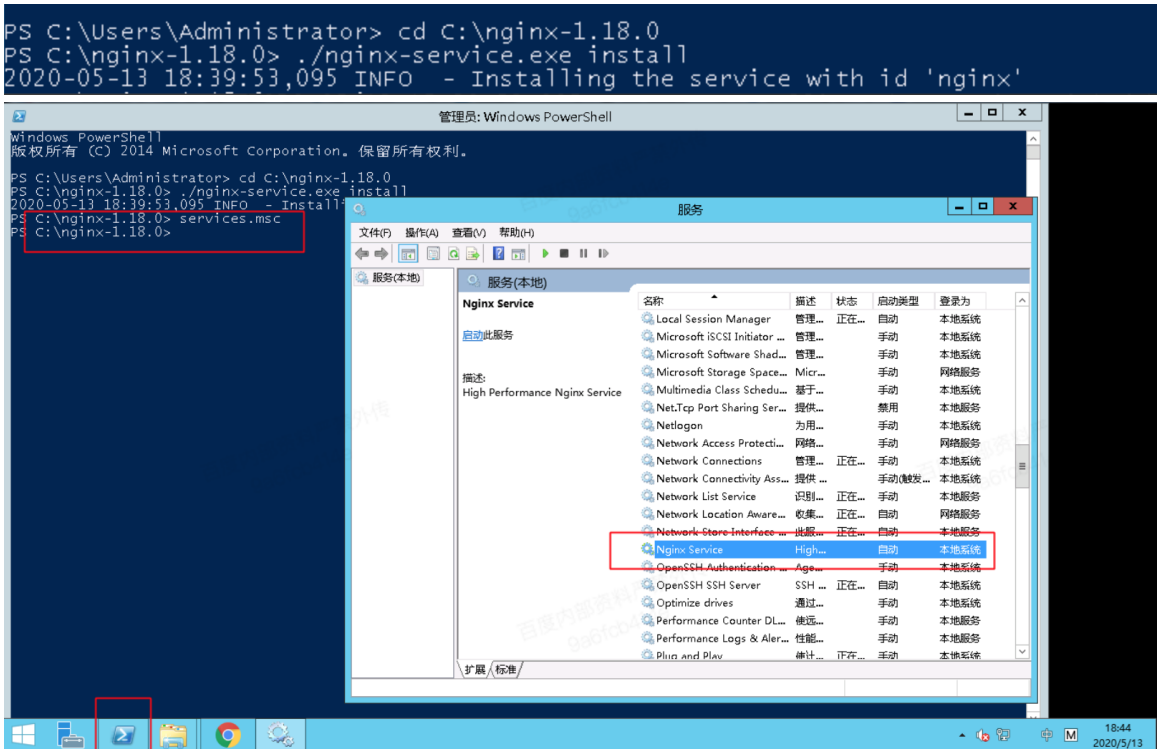
> 注意：nginx-service.xml文件内的C:\nginx-1.18.0\替换为您nginx的安装目录地址

5. 安装NET4+环境

winsw程序需要基于NET环境才可运行，windows2012默认安装了NET4.5，此步骤可忽略

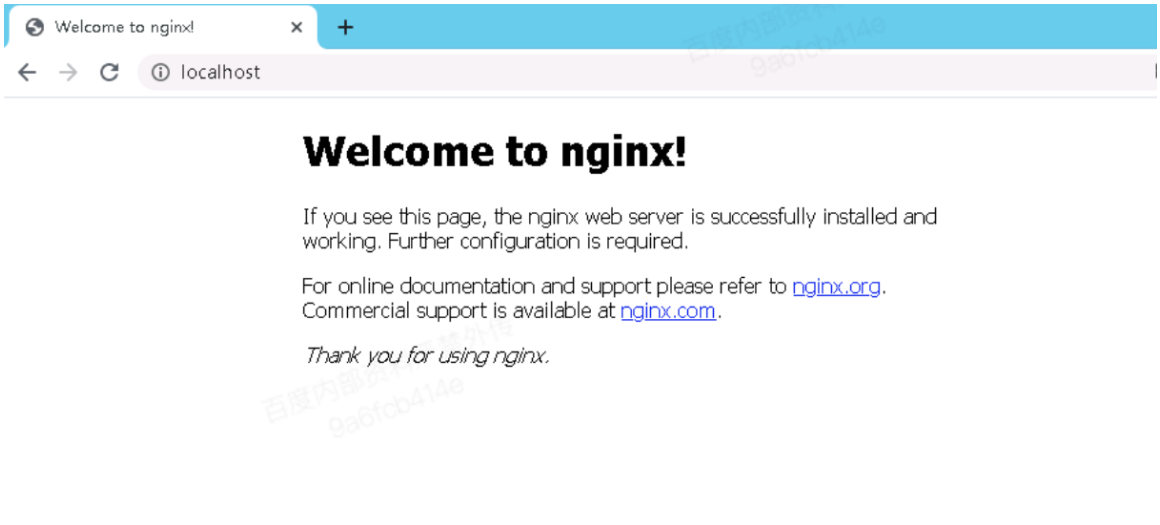
6. 执行winsw，添加nginx服务

进入cmd命令行模式，或者点击任务栏内的powershell进入命令行模式，然后进入nginx安装目录“cd C:\nginx-1.18.0”，执行“nginx-service.exe install” 安装程序，结果如下：



7. 启动nginx，并测试访问

鼠标右键点击nginx service服务，可看到有启动的选项，选择启动，nginx服务会自动开启，状态会变为正在运行，打开浏览器，直接访问http://locahost ，可查看到nginx的默认页面，说明nginx已部署完毕

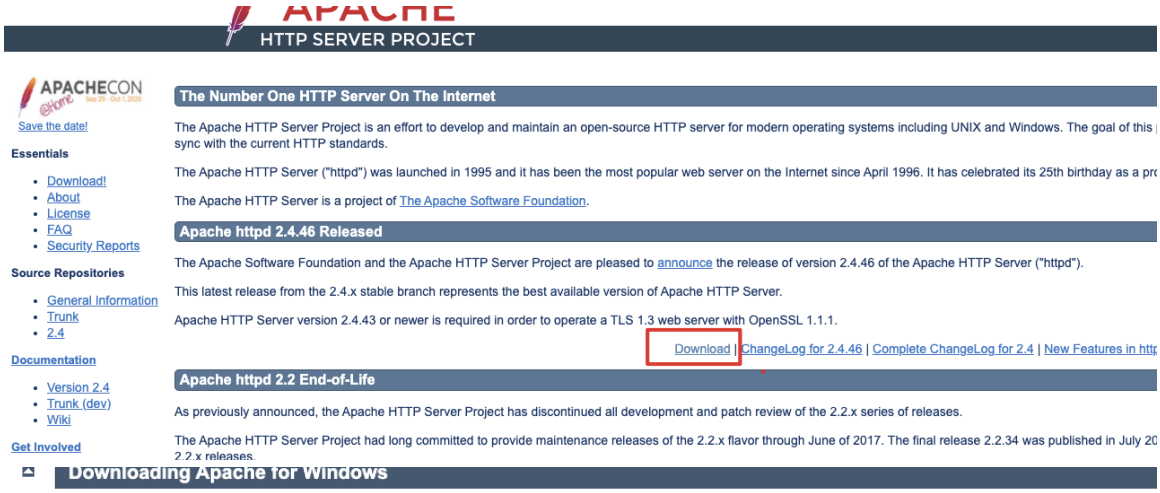


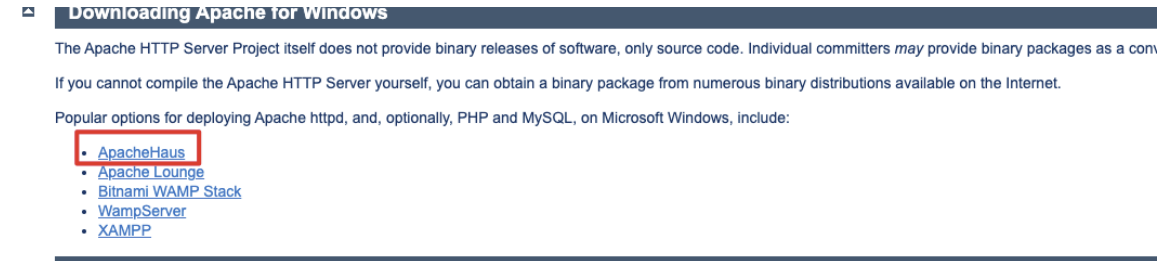
8. 开启服务器内高级防火墙的入站规则

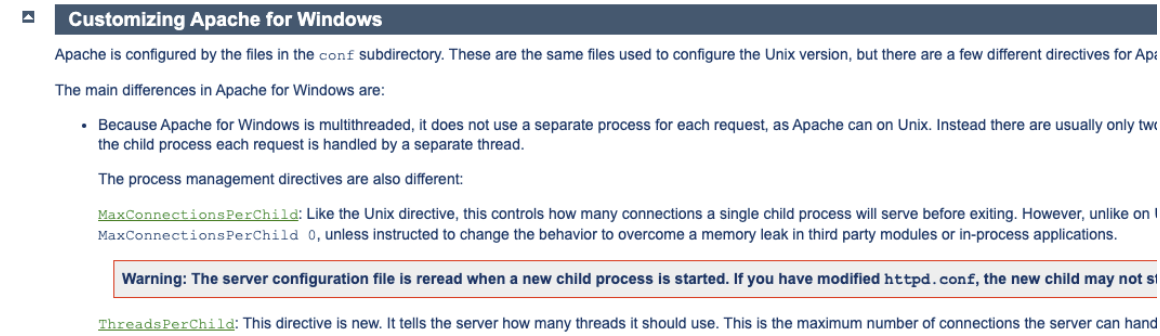
Windows服务器默认开启高级防火墙，服务搭建完毕后，还需要在防火墙的入站规则内开启对应的端口或者关闭防火墙，关闭防火墙步骤可参考文档：https://cloud.baidu.com/doc/BCC/s/bjxlq3hzn

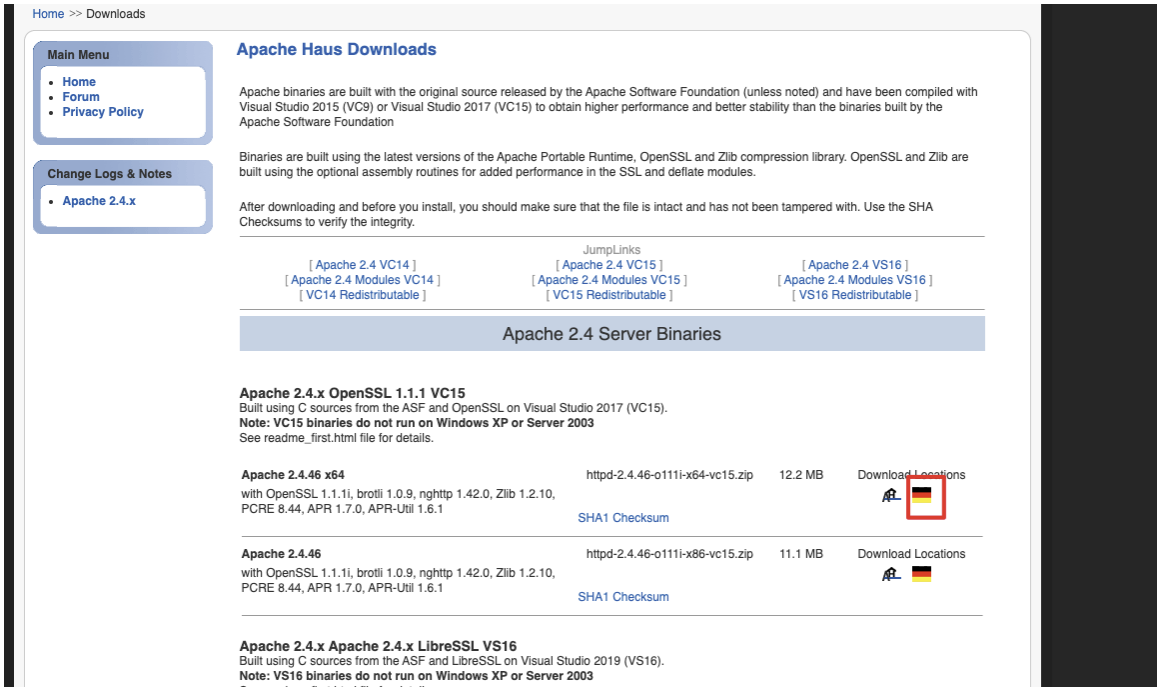
windows安装配置Apache环境

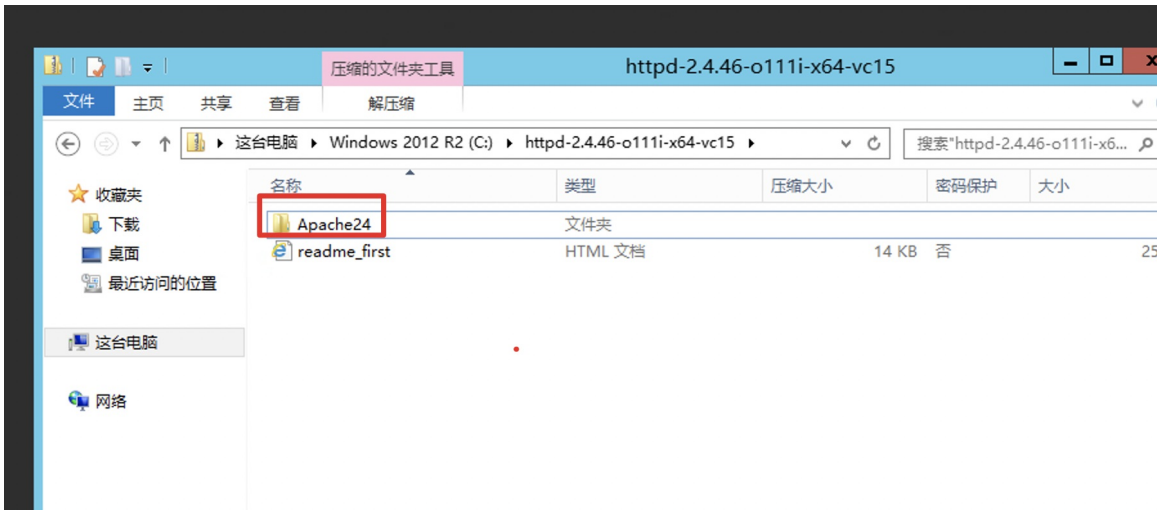
1. 在apache官网下载



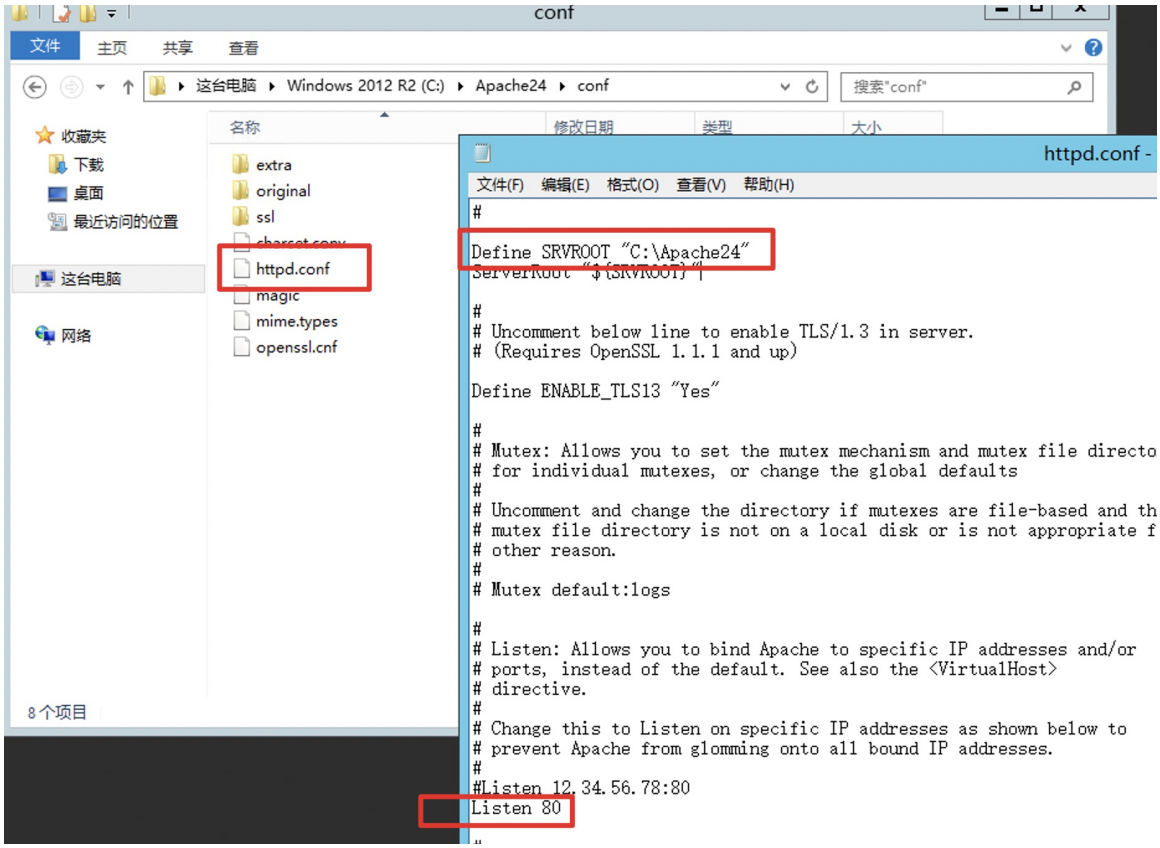




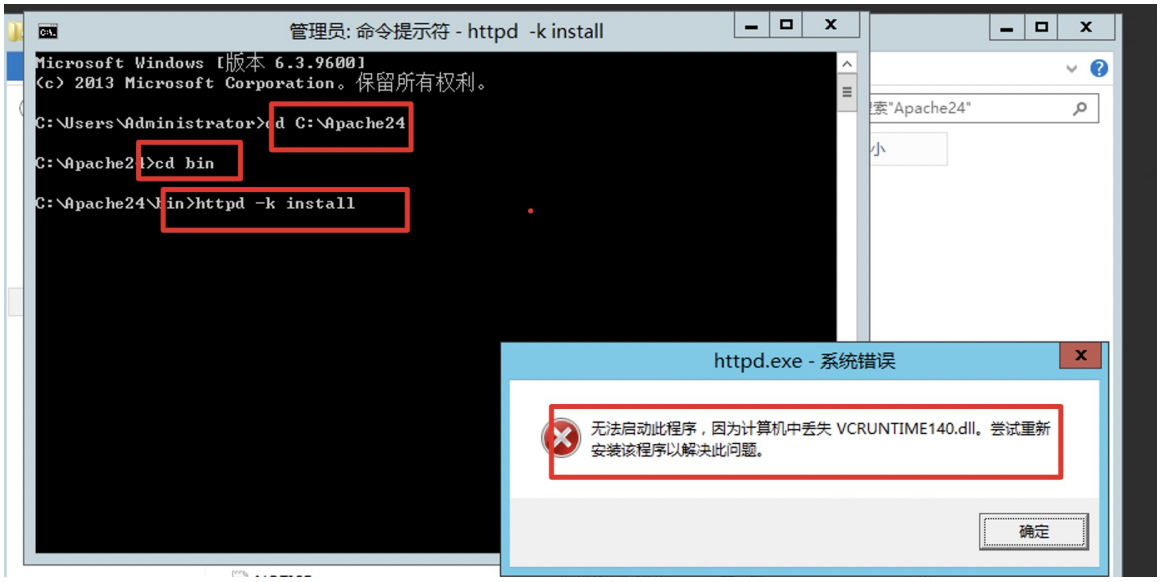




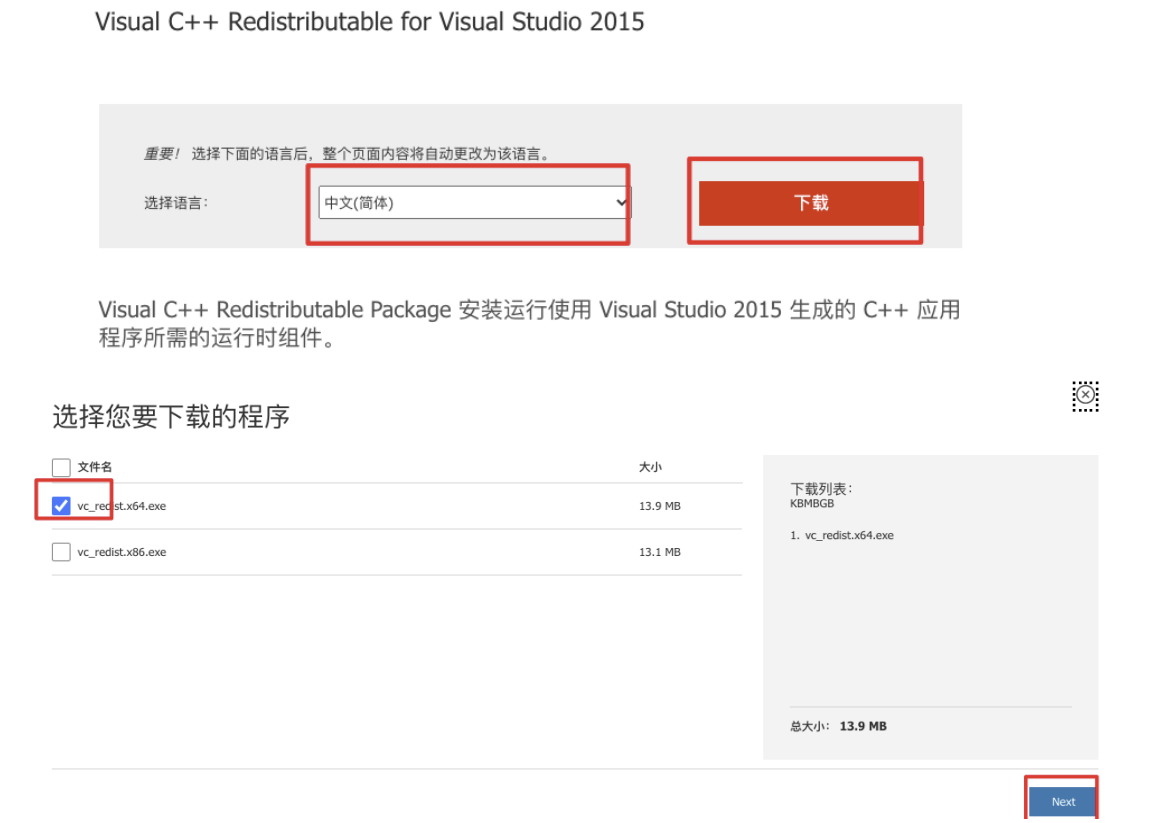
2. 打开解压后的文件，在conf中修改http.conf的服务路径和端口

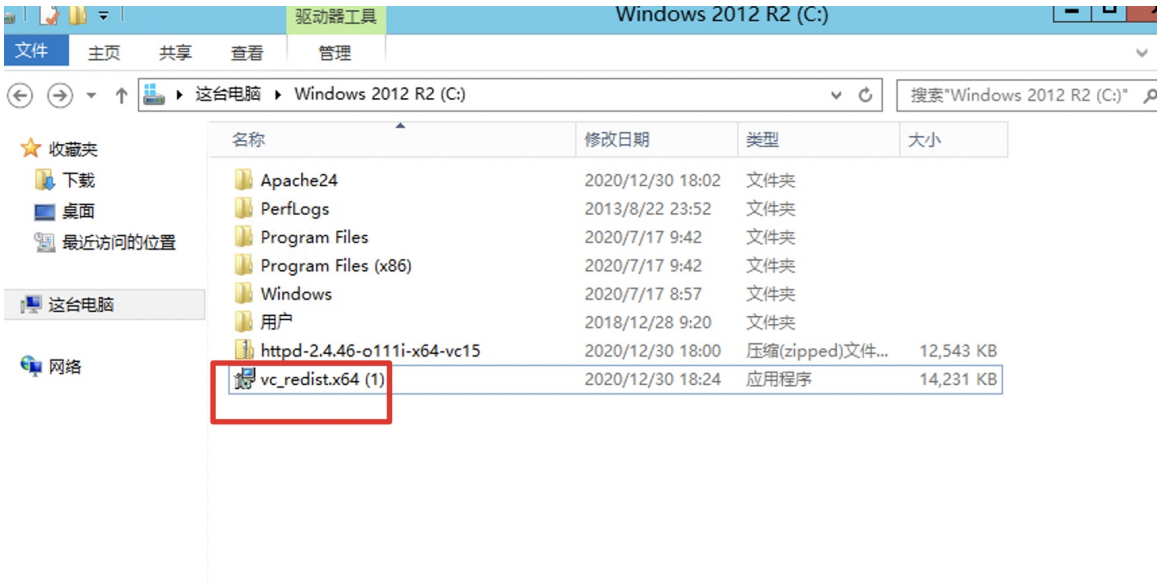


3. 安装apache



报错是由于Apache安装需要最新的Visual C++Redistributable 2015 ，下载地址： https://www.microsoft.com/zh-CN/download/details.aspx?id=48145

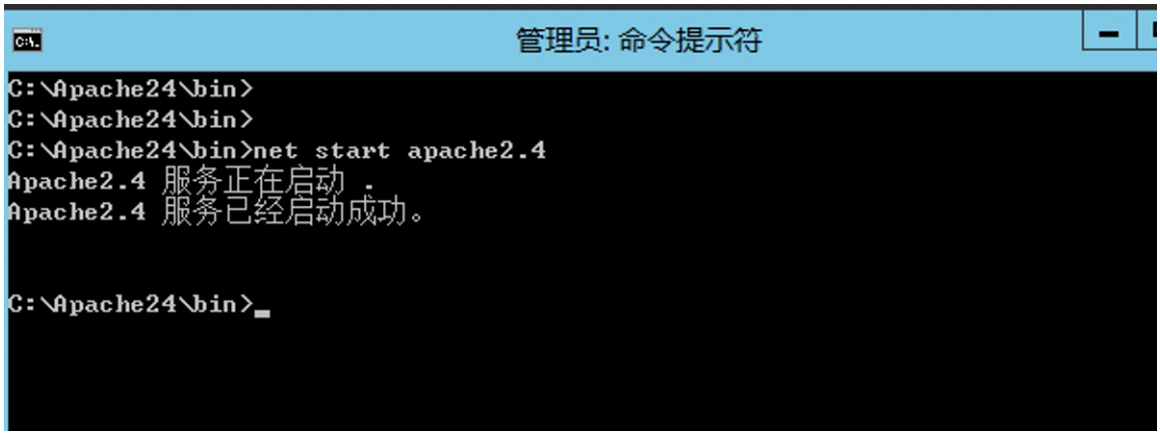




安装Visual C++Redistributable 2015 再执行 httpd -k install

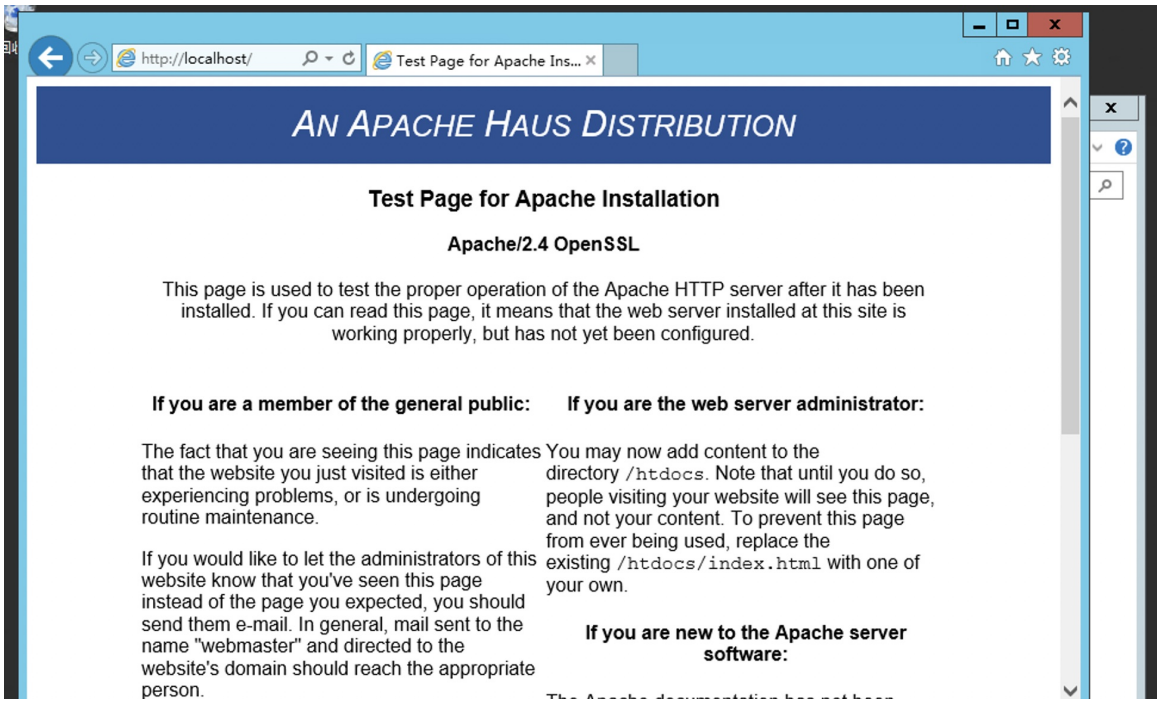
4. 启动apache

输入 net start apche2.4 启动Apache,如图：

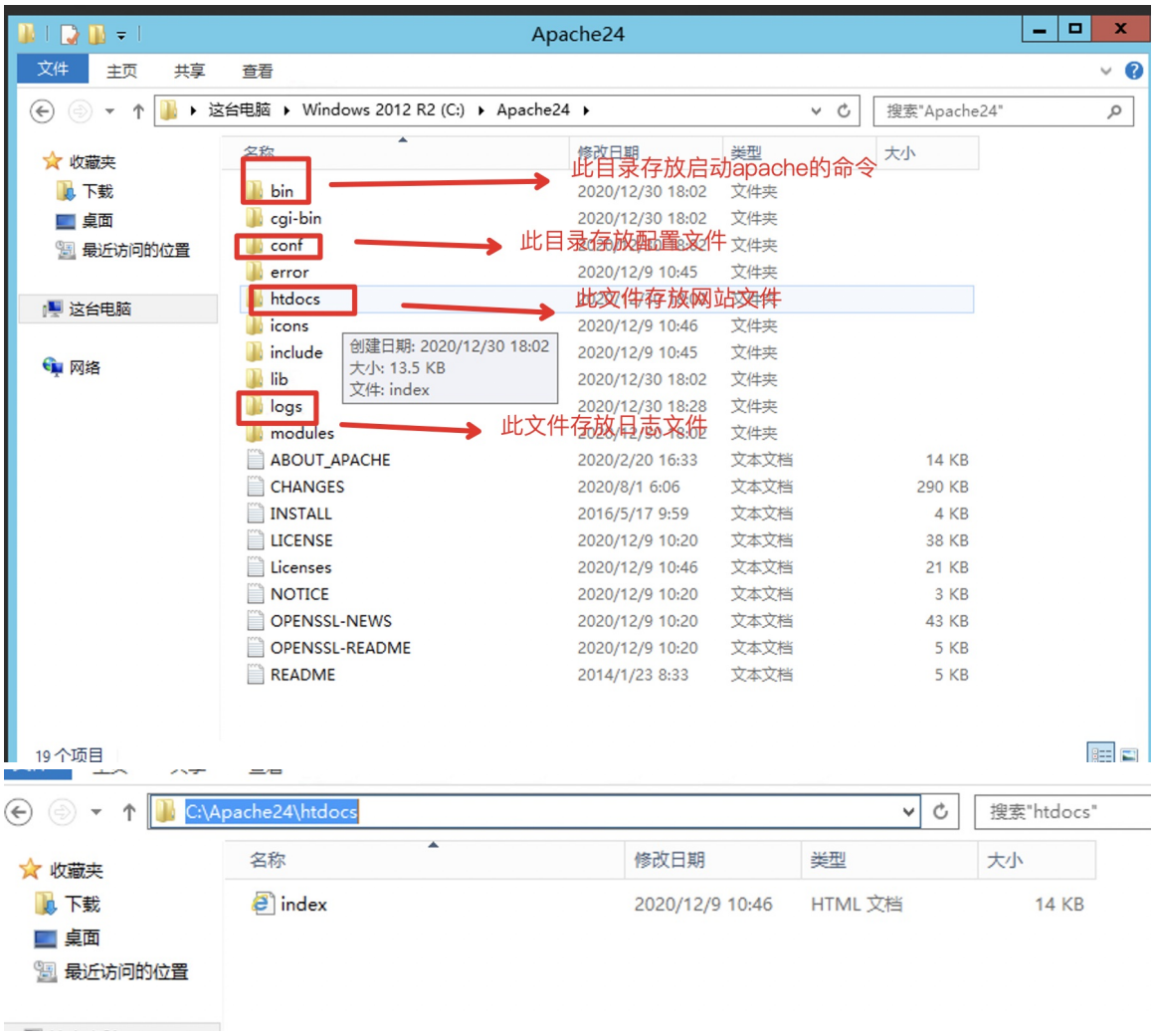


5. 验证

访问 http://localhost 显示默认页面



6. 网页文件需要上传到 C:\Apache24\htdocs 路径下，主要的文件信息如图：



7. 配置防火墙规则

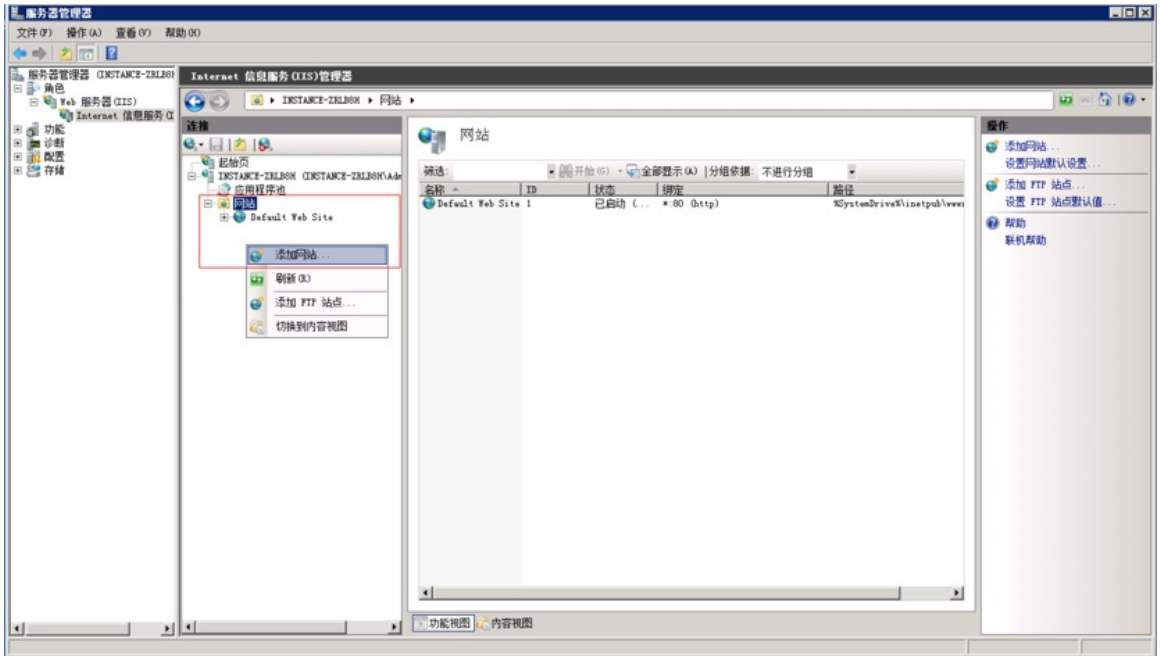
Windows服务器默认开启高级防火墙，服务搭建完毕后，还需要在防火墙的入站规则内开启对应的端口或者关闭防火墙。

关闭防火墙步骤可参考文档：https://cloud.baidu.com/doc/BCC/s/bjxlq3hzn

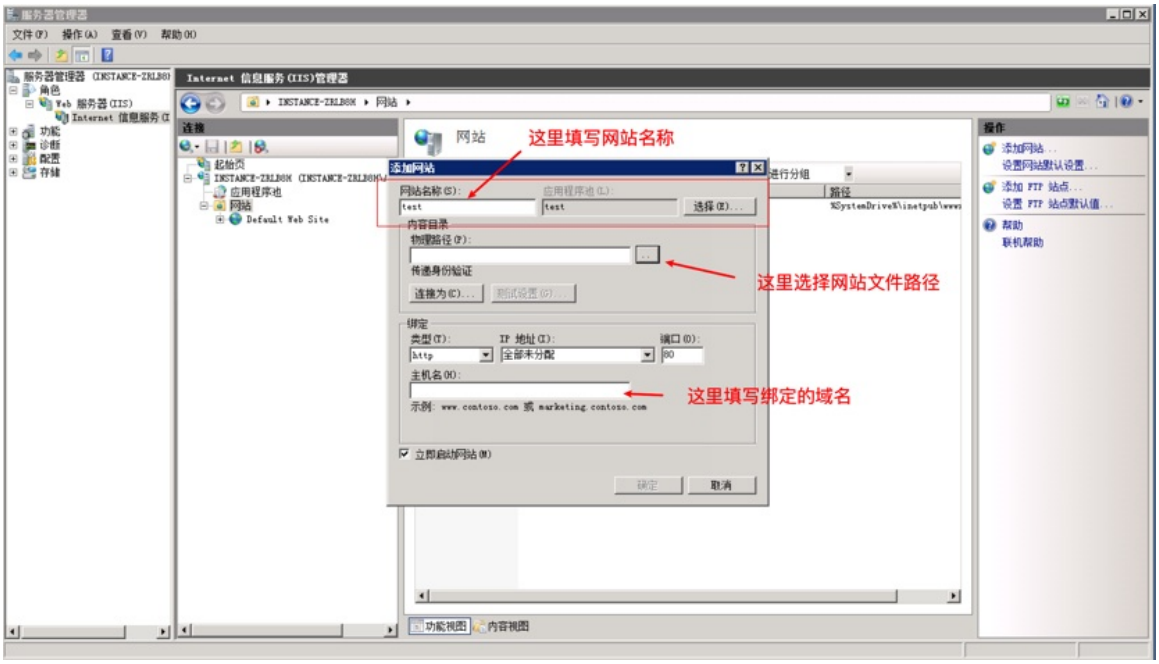
Windows2008与Windows2012下使用IIS服务搭建网站

windowsServer 2008

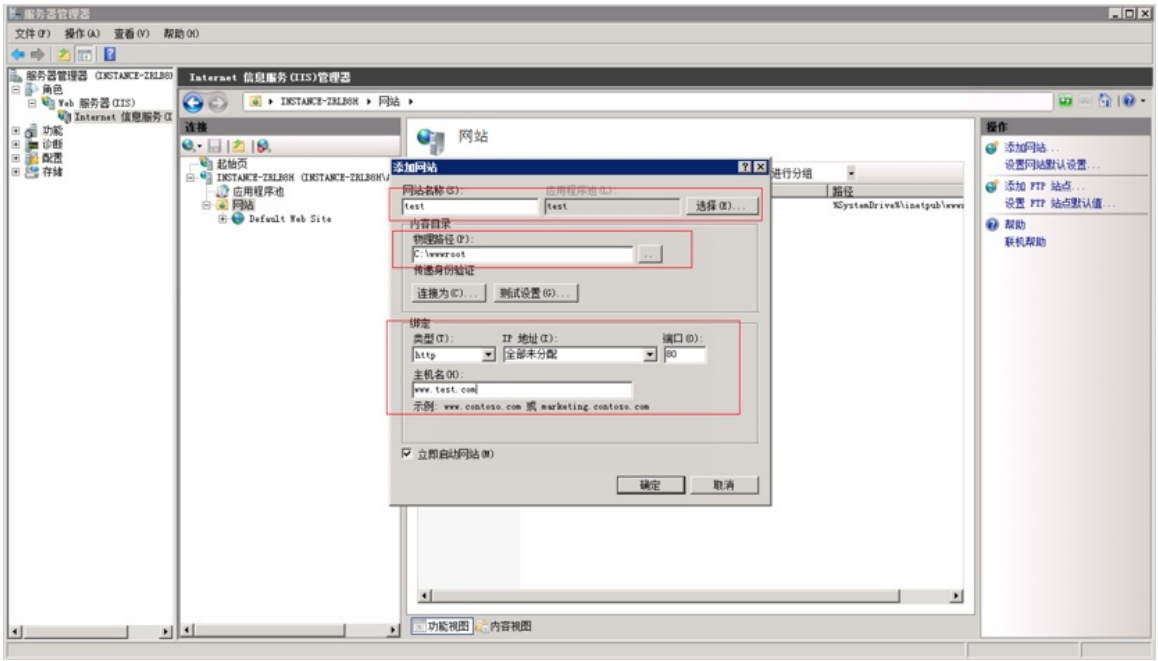
1. 鼠标左键选中网站，右键空白的地方，点击添加网站。



2. 填写网站名称，物理路径填写网站文件所在目录，一般为wwwroot（如果不存在，可以自行创建），主机名填写您的域名，完成后点击确定，示例如下：

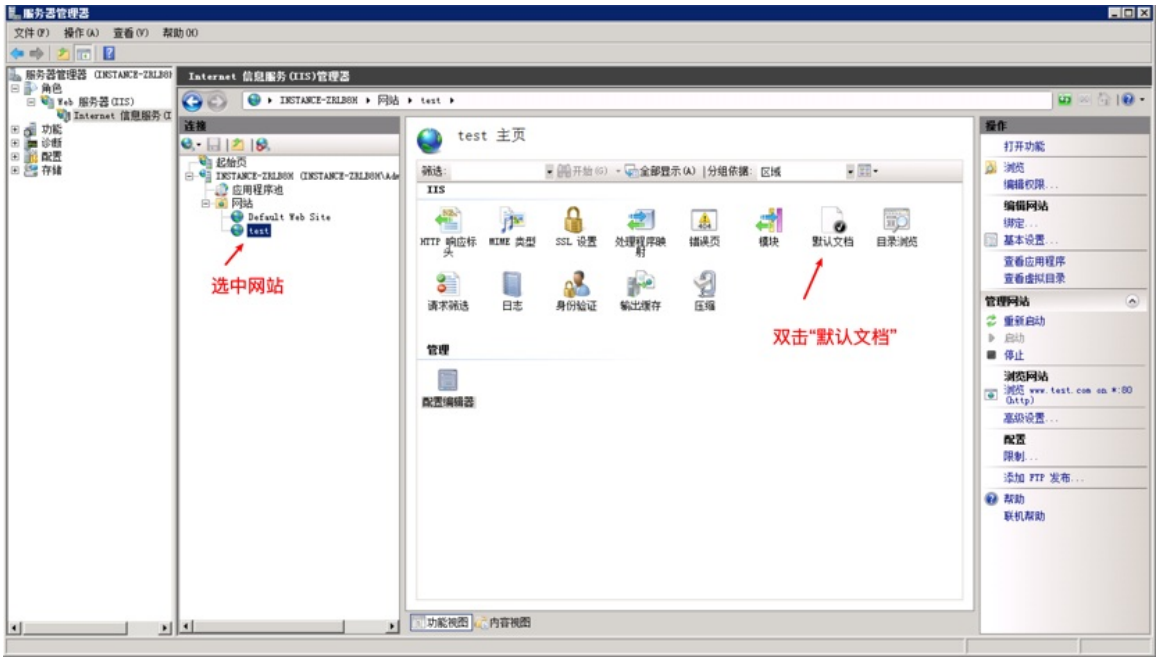


3. 绑定IP地址保持默认全部未分配即可，端口保持默认的80端口。填写自己的域名后，还需将域名解析至BCC的公网ip，这样就能通过域名访问BCC上搭建的网站。

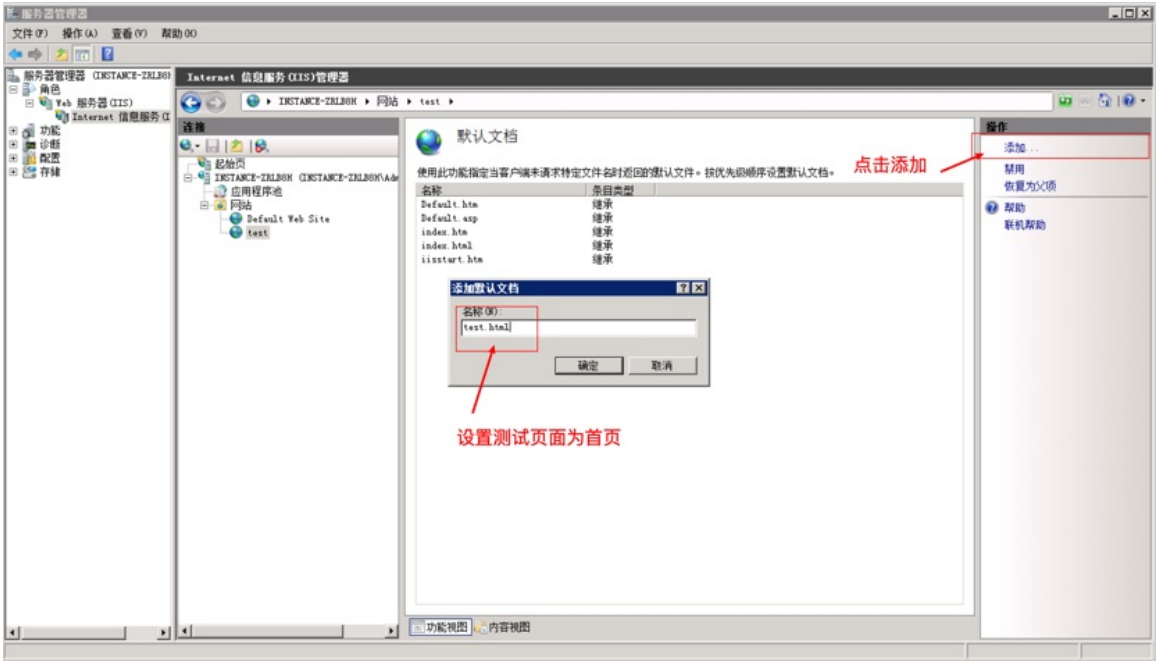


4. 将一个测试页面通过ftp上传至服务器并放在网站根目录wwwroot下。

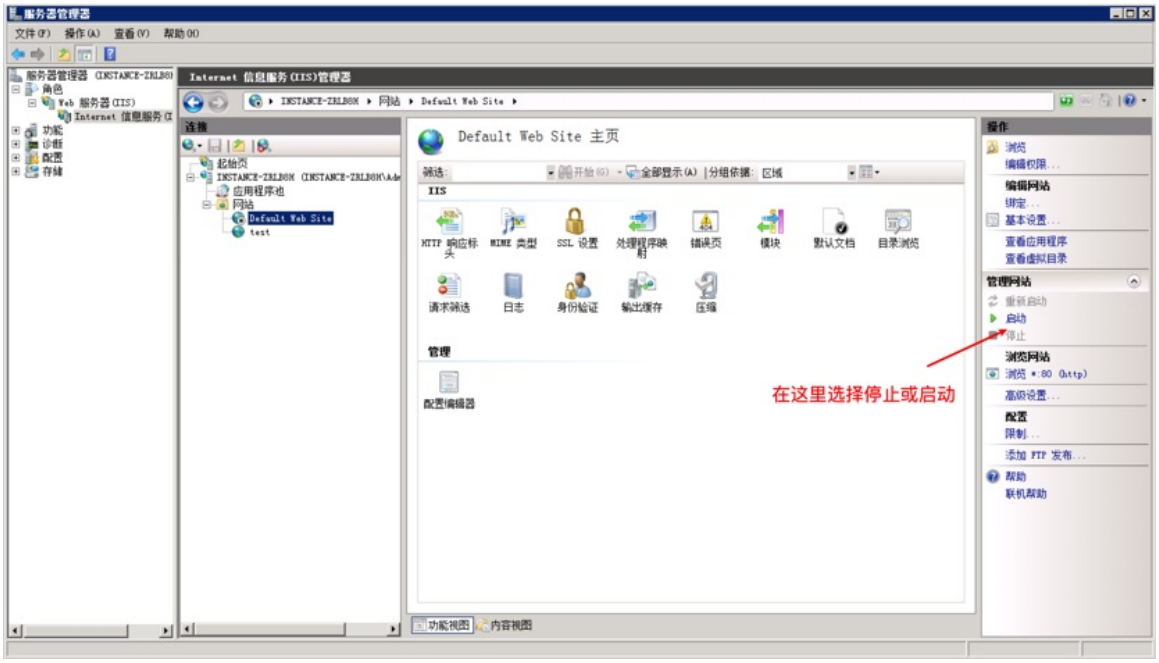
5. 配置网站的默认首页，选中站点，双点击默认文档。



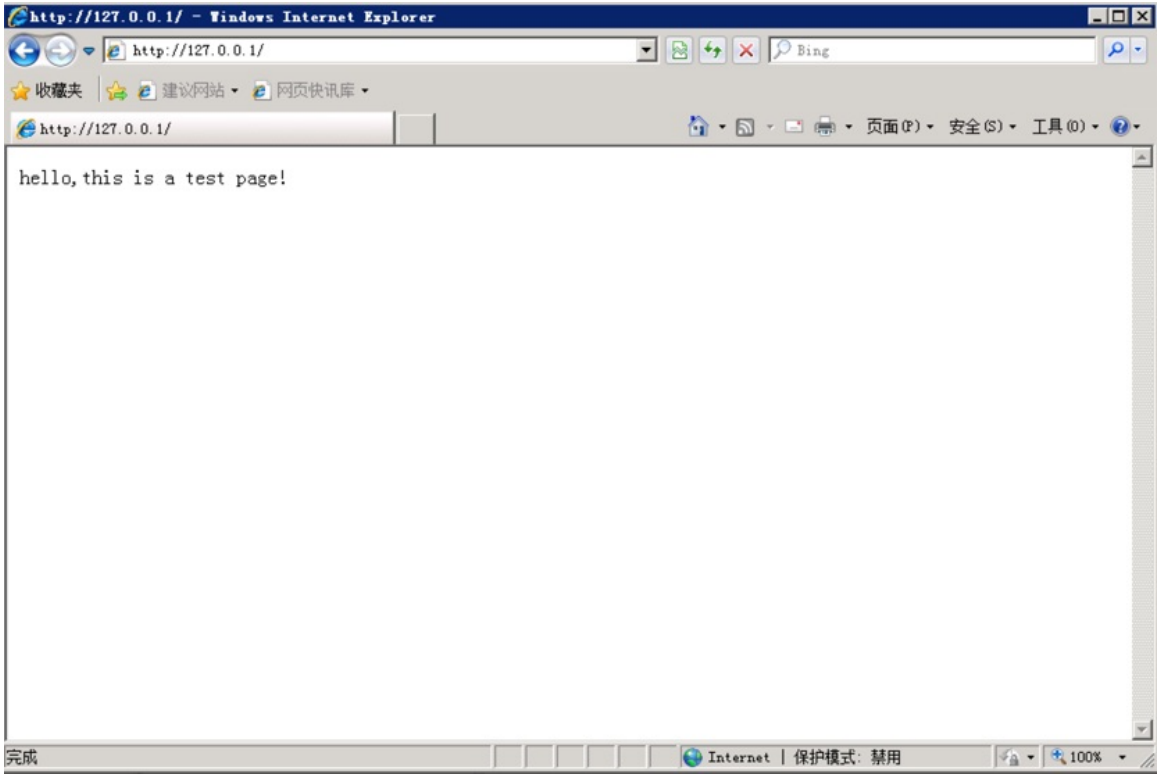
6.点击添加，将测试页面的名称填进去，点击确定。



7.将默认自带的网站停止，将刚才自建的网站启动。

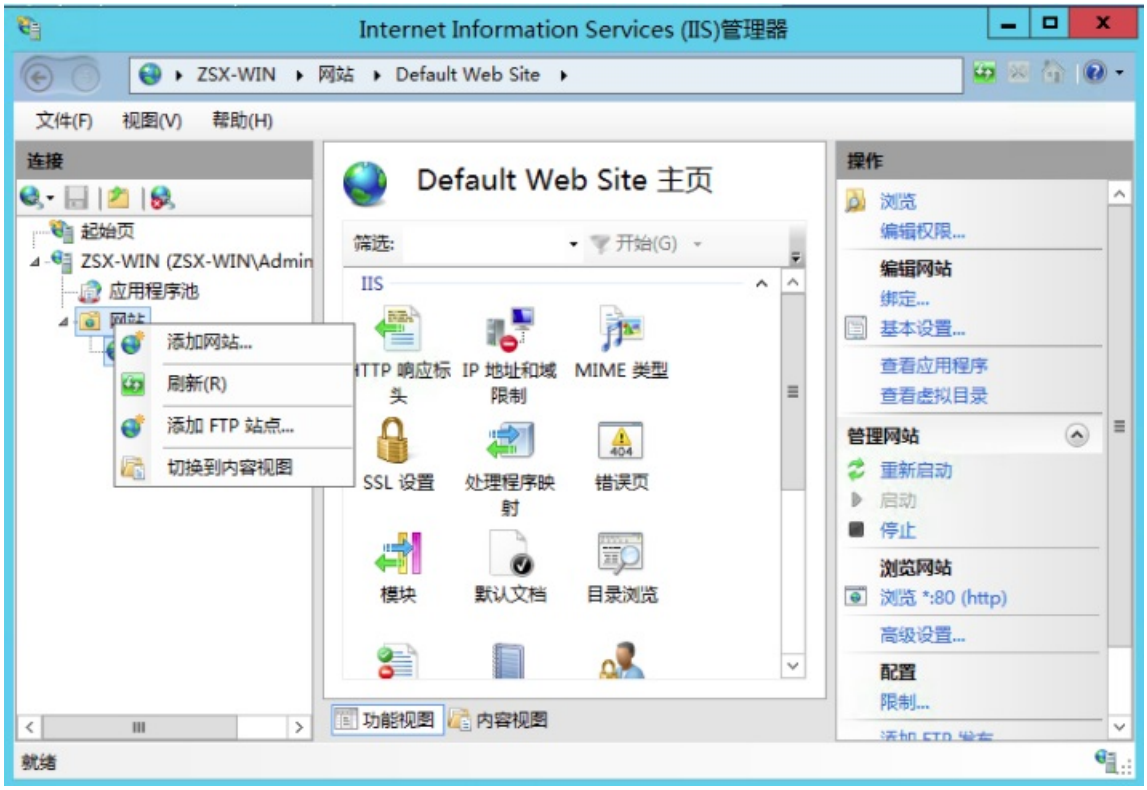


8. 打开浏览器访问域名或者BCC的eip或者本地环回地址127.0.0.1，首页正常打开。

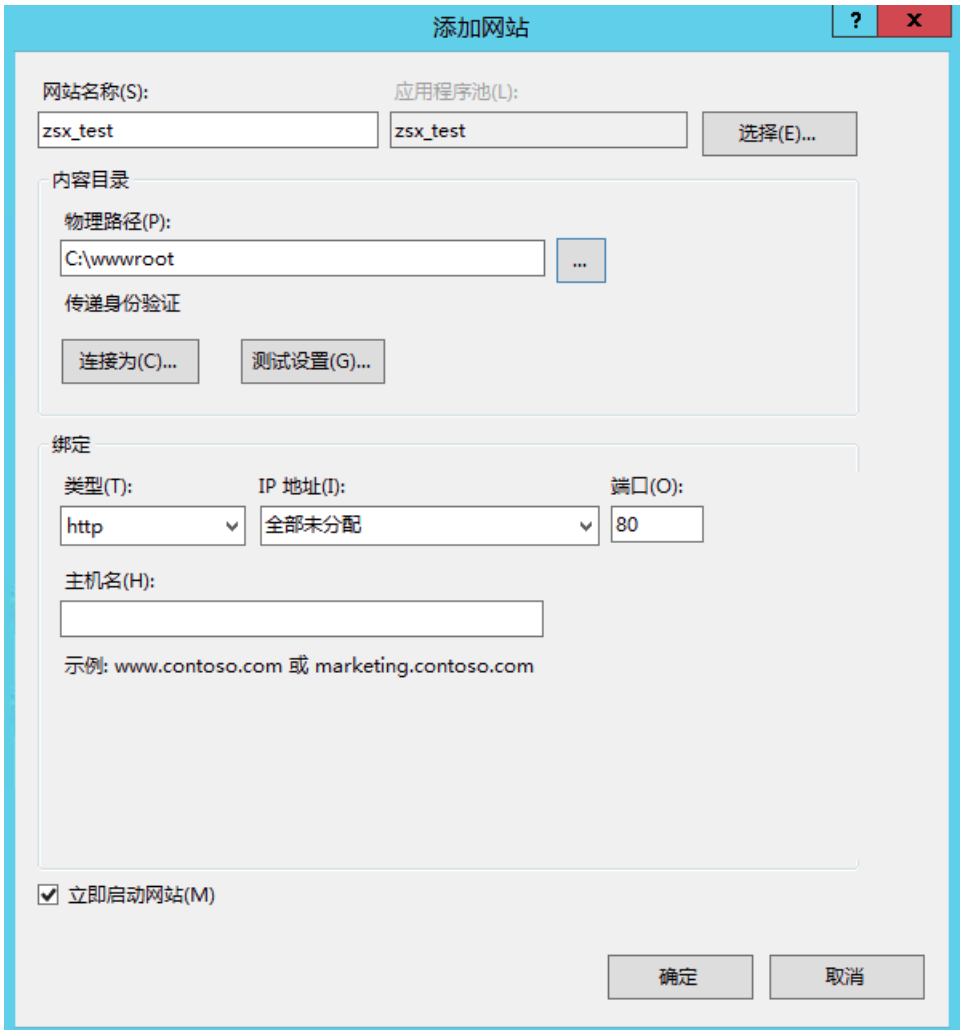


Windows Server 2012

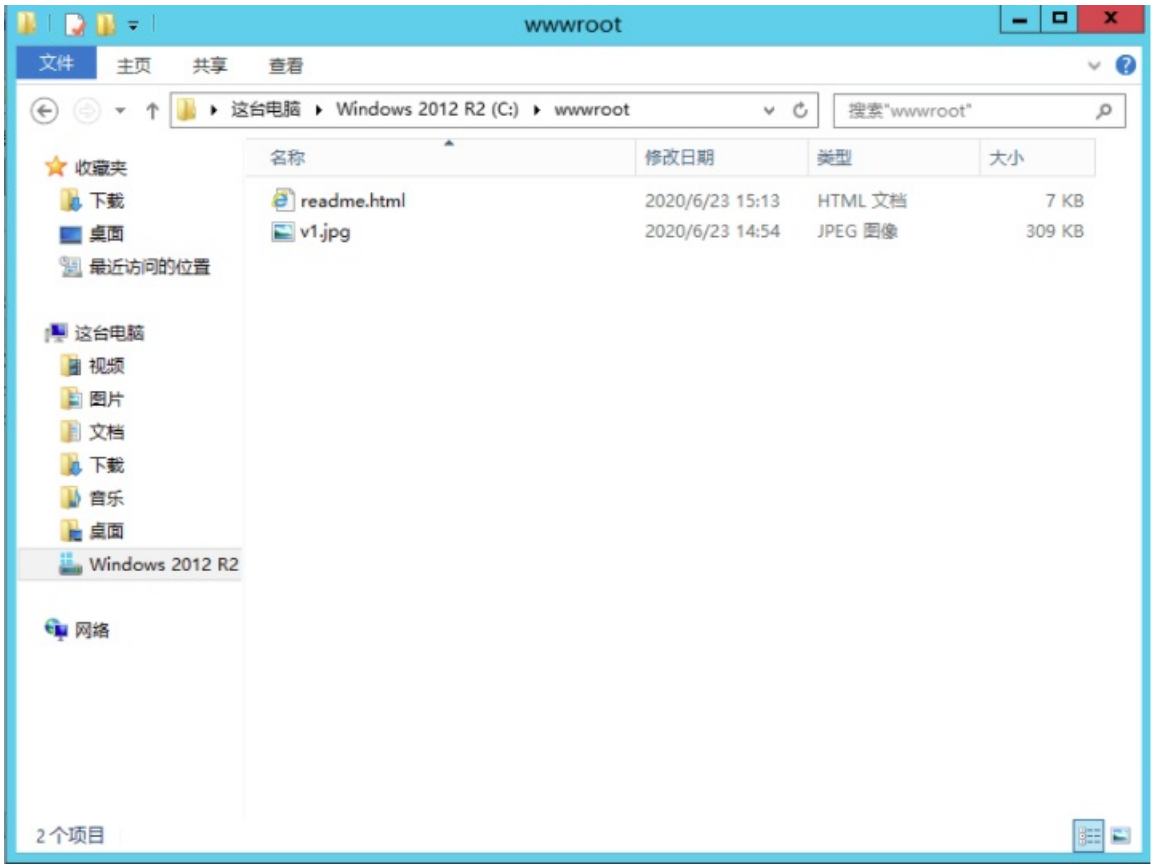
1. 左键选中网站，右键空白的地方，点击添加网站。



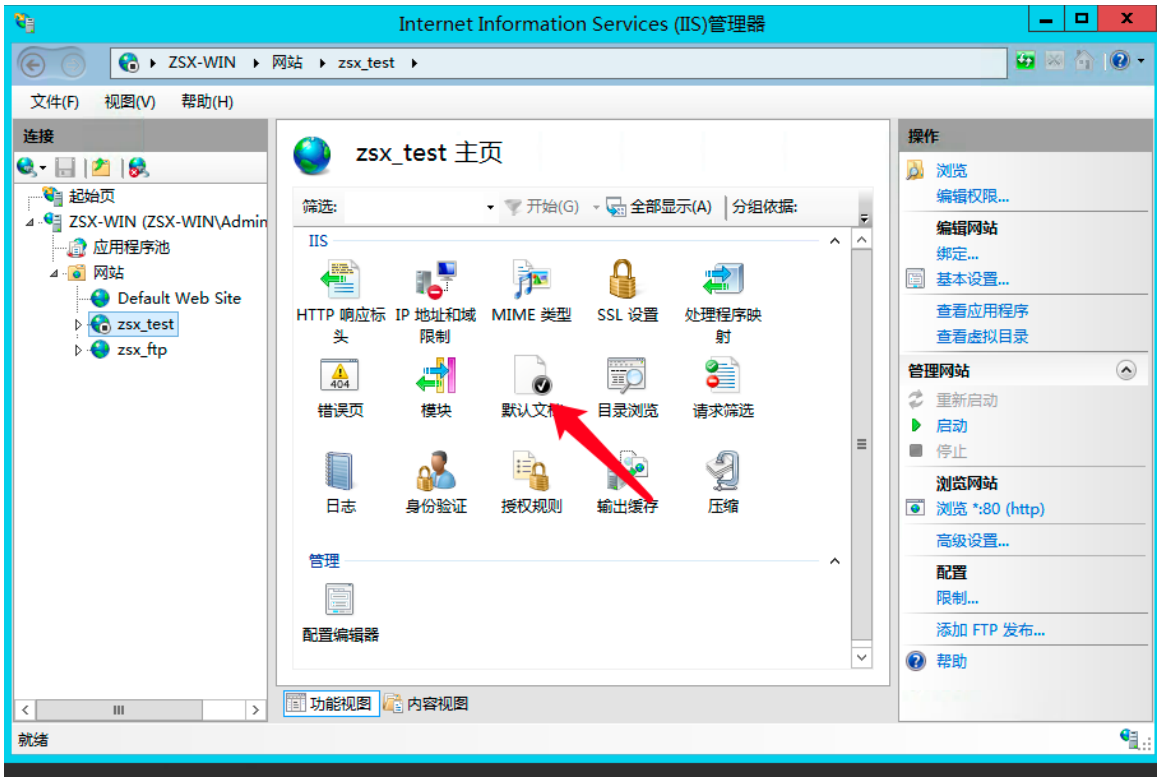
2. 填写网站名称，物理路径可以浏览选择网站所在的目录，一般为wwwroot（如果不存在，可以自行创建），绑定IP地址保持默认全部未分配即可，端口保持默认的80端口。如果有自己的域名，可以在主机地址那填写域名，然后将域名解析至BCC的公网ip，这样就能通过域名访问BCC上搭建的网站。



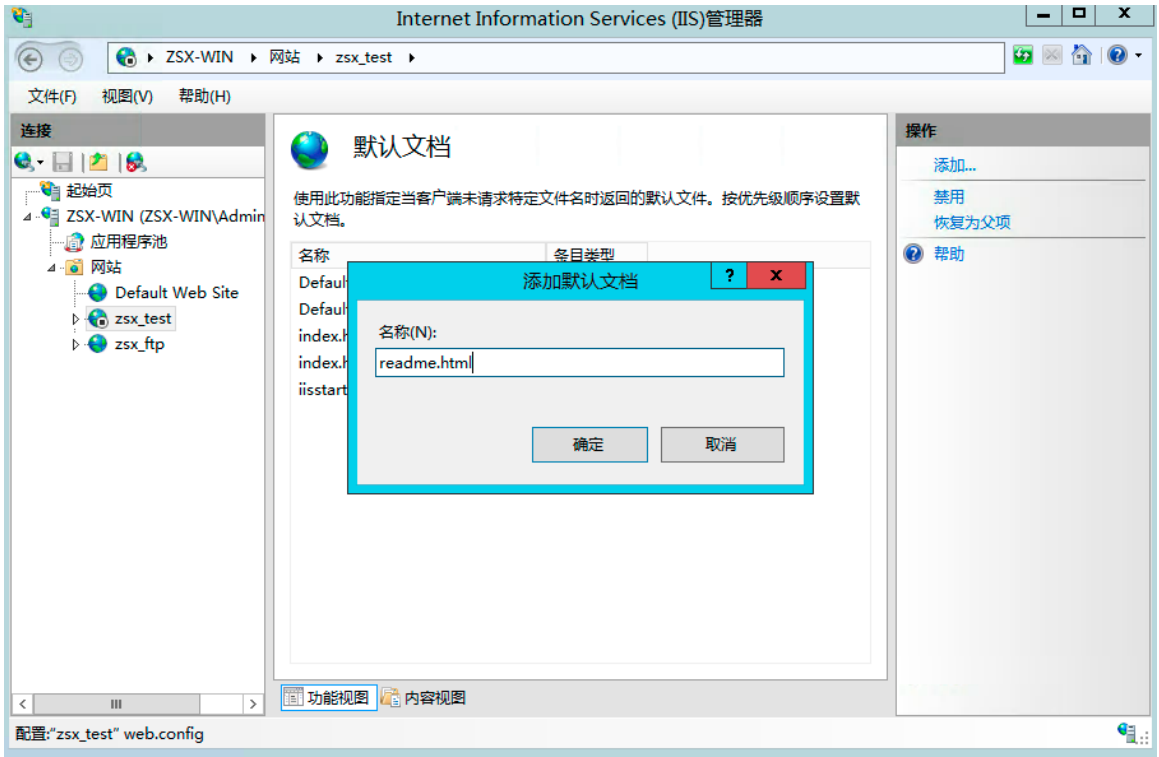
3. 将一个测试页面通过ftp上传至服务器并放在网站根目录wwwroot下。

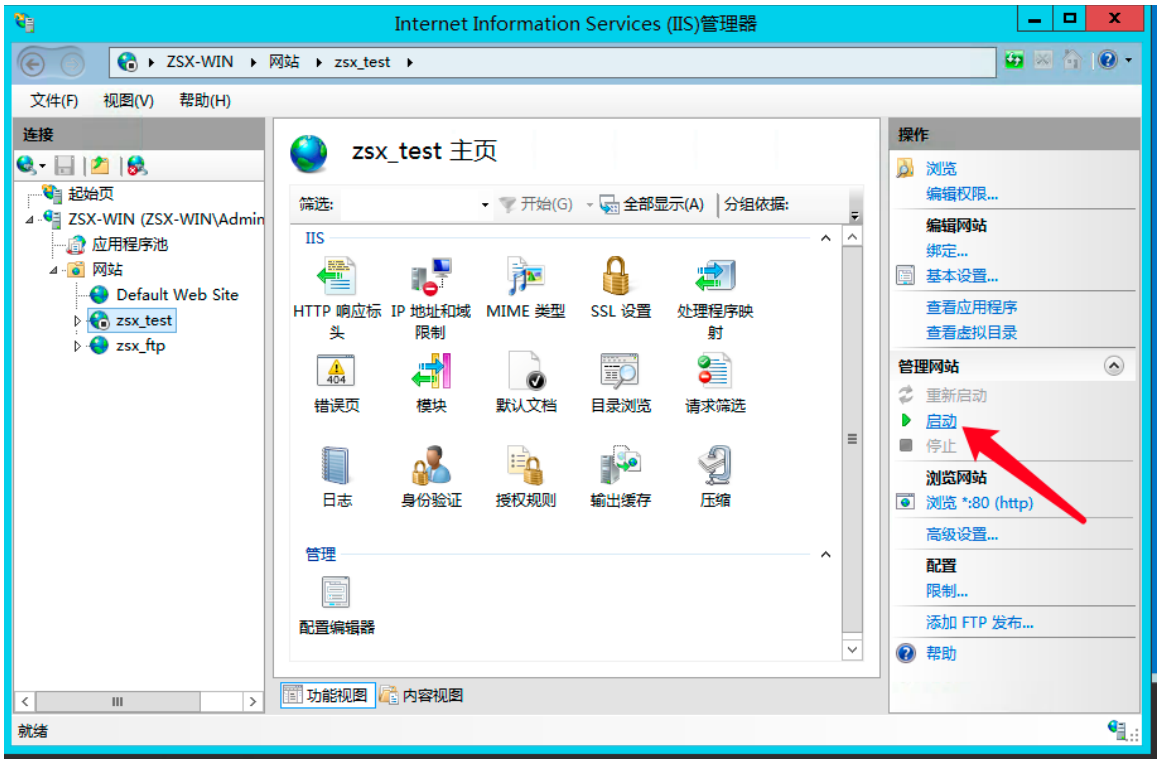


4. 配置网站的默认首页，选中站点，双点击默认文档。



5. 点击添加，将测试页面的名称填进去，点击确定，启动站点。





6. 打开浏览器访问域名或者BCC的公网eip或者localhost也即本地环回地址127.0.0.1，测试可以正常访问。

API参考

概述

欢迎使用百度智能云的核心产品——云服务器BCC (Baidu Cloud Compute)。

云服务器BCC是基于百度多年积累的虚拟化、分布式集群等技术构建的云端计算服务。BCC支持弹性伸缩，分钟级丰富灵活的计费模式，搭配镜像、快照、云安全等增值服务，为您提供超高效费比的高性能云服务器。

您可以使用本文档介绍的API对BCC服务进行灵活的操作。

如果您是初次调用百度智能云产品的API，可以观看API入门视频指南，快速掌握调用API的能力。也可以通过可视化API调试工具 - 示例代码中心，进行学习与调用测试。

BCC API 提供下列接口类型：

接口类型	描述
实例接口	实例 Instance 是一个虚拟的计算环境，BCC API 提供创建、停止、查询、重装、释放实例等接口。
磁盘接口	CDS 磁盘为 BCC 实例提供扩展存储，BCC API 提供创建、查询、挂载和卸载CDS磁盘、磁盘扩容和续费等接口。
镜像接口	镜像是 BCC 实例运行环境的模板，包括操作系统和应用软件。BCC API提供自定义镜像，查询镜像列表和详情等接口。
快照接口	快照应用于当设备发生应用故障或者文件损坏时，能够快速恢复数据，将数据恢复到某个时间点的状态。BCC API 提供创建快照、查询快照列表和详情等接口。
自动快照策略接口	自动快照策略可以帮助您按一定规则自动生成快照。BCC API 提供创建自动快照策略、绑定自动快照策略、查询自动快照策略详情等接口。
安全组接口	通过安全组规则设置 BCC 实例的入站出站访问策略，BCC API 提供创建和删除安全组等接口。
抢占实例接口	抢占实例较后付费有一定折扣，其价格随市场供需关系实时变化。BCC API提供创建抢占实例、查询抢占实例套餐、查询市场价和取消抢占实例订单等接口。
密钥对接口	密钥对是为Linux实例提供的一种安全便捷的登录认证方式。BCC API提供创建、绑定、解绑和删除等接口。
部署集接口	部署集是通过控制BCC实例在不同底层硬件分布的策略来实现服务的高可用。BCC API 提供创建、查询、调整和删除等接口。
专属集群接口	专属集群是以集群为售卖单位的云磁盘服务。BCC API 提供创建、查询、变配和续费等接口。
预留实例券接口	预留实例券是一种可以抵扣按量付费实例（不含抢占式实例）的账单，也能够预留实例资源的抵扣券。BCC API 提供创建、查询、调整和续费等接口。
维修平台相关接口	维修平台是提供云服务器实例硬件故障处理、授权维修方案的平台，BCC API提供维修任务操作、预授权操作、维修记录查询等接口。
其他接口	BCC API 还支持查询实例套餐列表等接口。
地域相关接口	查询BCC支持的地域名称及域名（endpoint）。

通用说明

API调用遵循HTTP协议，各Region采用不同的域名，具体域名为bcc.{region}.baidubce.com。 例：bcc.bj.baidubce.com。数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

实名认证

使用云服务器BCC API的用户需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。没有通过实名认证的用户请求将会得到以下错误提示码：

错误码	错误描述	HTTP状态码	中文解释
QualifyNotPass	The User has not pass qualify.	403	账号没有通过实名认证

API认证机制

所有API的安全认证一律采用Access Key与请求签名机制。 Access Key由Access Key ID和Secret Access Key组成，均为字符串。 对于每个HTTP请求，使用下面所描述的算法生成一个认证字符串。提交认证字符串放在Authorization头域里。服务端根据生成算法验证认证字符串的正确性。 认证字符串的格式为`bce-auth-v{version}/{accessKeyId}/{timestamp}/{expirationPeriodInSeconds}/{signedHeaders}/{signature}`。

- version是正整数。
- timestamp是生成签名时的UTC时间。
- expirationPeriodInSeconds表示签名有效期限。

- signedHeaders是签名算法中涉及到的头域列表。头域名之间用分号（;）分隔，如host;x-bce-date。列表按照字典序排列。（本API签名仅使用host和x-bce-date两个header）
- signature是256位签名的十六进制表示，由64个小写字母组成。

当百度智能云接收到用户的请求后，系统将使用相同的SK和同样的认证机制生成认证字符串，并与用户请求中包含的认证字符串进行比对。如果认证字符串相同，系统认为用户拥有指定的操作权限，并执行相关操作；如果认证字符串不同，系统将忽略该操作并返回错误码。

鉴权认证机制的详细内容请参见鉴权认证。

通信协议

BCC API支持HTTP和HTTPS两种调用方式。为了提升数据的安全性，建议通过HTTPS调用。

请求结构说明

数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

请求参数包括如下4种：

参数类型	说明
URI	通常用于指明操作实体,如:POST /v{version}/instance/{instanceId}
Query参数	URL中携带的请求参数
HEADER	通过HTTP头域传入,如:x-bce-date
RequestBody	通过JSON格式组织的请求数据体

响应结构说明

响应值分为两种形式：

返回内容	说明
HTTP STATUS CODE	如200,400,403,404等
ResponseBody	JSON格式组织的响应数据体

API版本号

参数	类型	参数位置	描述	是否必须
version	String	URL参数	API版本号，当前值为2。	是

幂等性

当调用创建接口时如果遇到了请求超时或服务器内部错误，用户可能会尝试重发请求，这时用户通过clientToken参数避免创建出比预期要多的资源，即保证请求的幂等性。

幂等性基于clientToken，clientToken是一个长度不超过64位的ASCII字符串，通常放在query string里，如`http://bcc.bj.baidubce.com/v1/instance?clientToken=be31b98c-5e41-4838-9830-9be700de5a20`。

如果用户使用同一个clientToken值调用创建接口，则服务端会返回相同的请求结果。因此用户在遇到错误进行重试的时候，可以通过提供相同的clientToken值，来确保只创建一个资源；如果用户提供了一个已经使用过的clientToken，但其他请求参数（包括queryString和requestBody）不同甚至url Path不同，则会返回IdempotentParameterMismatch的错误代码。

clientToken的有效期为24小时，以服务端最后一次收到该clientToken为准。也就是说，如果客户端不断发送同一个clientToken，那么该clientToken将长期有效。

密码加密传输规范

所有涉及密码的接口参数都需要加密，禁止明文传输。密码一律采用AES 128位加密算法进行加密，用SK的前16位作为密钥，加密后生成的二进制字节流需要转成十六进制，并以字符串的形式传到服务端。具体步骤如下：

- byte[] bCiphertext= AES(明文,SK)
- String strHex = HexStr(bCiphertext)

日期与时间规范

日期与时间的表示有多种方式。为统一起见，除非是约定俗成或者有相应规范的，凡需要日期时间表示的地方一律采用UTC时间，遵循ISO 8601，并做以下约束：

1. 表示日期一律采用YYYY-MM-DD方式，例如2014-06-01表示2014年6月1日
2. 表示时间一律采用hh:mm:ss方式，并在最后加一个大写字母Z表示UTC时间。例如23:00:10Z表示UTC时间23点0分10秒。
3. 凡涉及日期和时间合并表示时，在两者中间加大写字母T，例如2014-06-01T23:00:10Z表示UTC时间2014年6月1日23点0分10秒。

规范化字符串

通常一个字符串中可以包含任何Unicode字符。在编程中这种灵活性会带来不少困扰。因此引入“规范字符串”的概念。一个规范字符串只包含百分号编码字符以及URI（Uniform Resource Identifier）非保留字符（Unreserved Characters）。

RFC 3986规定URI非保留字符包括以下字符：字母（A-Z，a-z）、数字（0-9）、连字号（-）、点号（.）、下划线（_）、波浪线（~）。

将任意一个字符串转换为规范字符串的方式是:

- 将字符串转换成UTF-8编码的字节流。
- 保留所有URI非保留字符原样不变。
- 对其余字节做一次RFC 3986中规定的百分号编码（Percent-Encoding），即一个%后面跟着两个表示该字节值的十六进制字母。字母一律采用大写形式。

示例：

原字符串：`this is an example for 测试`，对应的规范字符串：`this%20is%20an%20example%20for%20%E6%B5%8B%E8%AF%95`。

排版约定

排版格式	含义
< >	变量
[ ]	可选项
{ }	必选项
|	互斥关系
等宽字体Courier New	屏幕输出

服务域名

BCC API的服务域名为：

Region	Endpoint	Protocol	Zone
北京	bcc.bj.baidubce.com	HTTP and HTTPS	cn-bj-b、cn-bj-d、cn-bj-e、cn-bj-f、cn-bj-h
保定	bcc.bd.baidubce.com	HTTP and HTTPS	cn-bd-b、cn-bd-c、cn-bd-d
广州	bcc.gz.baidubce.com	HTTP and HTTPS	cn-gz-c、cn-gz-d、cn-gz-f
苏州	bcc.su.baidubce.com	HTTP and HTTPS	cn-su-a、cn-su-d、cn-su-f、cn-su-h
香港	bcc.hkg.baidubce.com	HTTP and HTTPS	cn-hkg-a
武汉	bcc.fwh.baidubce.com	HTTP and HTTPS	cn-fwh-a
成都	bcc.cd.baidubce.com	HTTP and HTTPS	cn-cd-a
上海	bcc.fsh.baidubce.com	HTTP and HTTPS	cn-fsh-a

说明：BCC API支持HTTP和HTTPS两种调用方式。为了提升数据的安全性，建议通过HTTPS调用。

公共请求头与响应头

公共请求头

公共头部	描述
Authorization	包含Access Key与请求签名。具体请参考鉴权认证
Content-Type	application/json; charset=utf-8
x-bce-date	表示日期的字符串，符合API规范。具体请参考日期与时间规范

HTTP协议的标准头域不在这里列出。公共头域将在每个BCC API中出现，是必需的头域。POST、PUT、DELETE等请求数据需放在request body中。

公共响应头

公共头部	描述
Content-Type	application/json; charset=utf-8。
x-bce-request-id	BCC后端生成，并自动设置到响应头域中。

其中，x-bce-request-id 使用UUID version4由BCC服务生成。

错误返回

当用户访问API出现错误时，会返回相应的错误码和错误信息，便于定位问题，并做出适当的处理。请求发生错误时通过Response Body返回详细错误信息，遵循如下格式：

参数名	类型	说明
code	String	表示具体错误类型
message	String	有关该错误的详细说明
requestId	String	本次请求的requestId

示例：

```
{
  "requestId" : "ae2225f7-1c2e-427a-a1ad-5413b762957d",
  "code" : "NoSuchKey",
  "message" : "The resource you requested does not exist"
}
```

公共错误码

下表列出了百度智能云API的公共错误码。

错误码	错误消息	HTTP状态码	描述
AccessDenied	Access denied.	403Forbidden	无权限访问对应的资源。
InappropriateJSON	The JSON you provided was well-formed and valid, but not appropriate for this operation.	400 Bad Request	请求中的JSON格式正确，但语义上不符合要求。如缺少某个必需项，或值类型不匹配等。出于兼容性考虑，对于所有无法识别的项应直接忽略，不应该返回这个错误。
InternalError	We encountered an internal error Please try again.	500 Internal Server Error	所有未定义的其他错误。在有明确对应的其他类型的错误时（包括通用的和服务自定义的）不应该使用。
InvalidAccessKeyI	The Access Key ID you provided doesnot exist in our records.	403Forbidden	Access Key ID不存在。

d			
InvalidHTTPAuthHeader	The Access Key ID you provided does notexist in our records.	400 BadRequest	Authorization头域格式错误。
InvalidHTTPRequest	There was an error in the body of your HTTP request.	400 Bad Request	HTTP body格式错误。例如不符合指定的Encoding等。
InvalidURI	Could not parse the specified URI.	400 Bad Request	URI形式不正确。例如一些服务定义的关键词不匹配等。对于ID不匹配的问题，应定义更加具体的错误码，如NoSuchKey。
MalformedJSON	The JSON you provided was not well-formed.	400 BadRequest	JSON格式不合法。
InvalidVersion	The API version specified was invalid.	404 NotFound	URI的版本号不合法。
OptInRequired	A subscription for the service is required.	403Forbidden	没有开通对应的服务。
PreconditionFailed	The specified If-Match header doesn’tmatch the ETag header.	412PreconditionFailed	详见Etag。
RequestExpired	Request has expired. Timestamp date is <Data>.	400 BadRequest	请求超时。要改成x-bce-date。若请求中只有Date，需将Date转成datetime。
IdempotentParameterMismatch	The request uses the same client token asa previous, but non-identical request.	403Forbidden	clientToken对应的API参数不一样。
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check yourSecret Access Key and signing method. Consultthe service documentation for details.	400 Bad Request	Authorization头域中附带的签名和服务端验证不一致。

BCC错误返回码

下表列出了BCC API特有错误码。

错误码	错误消息	HTTP状态码	描述
QualifyNotPass	The User has not pass qualify.	403	账号没有通过实名认证
BadRequest	Bad request parameters or illegal request.	400	请求有误
AuthError	No authentication or authentication error.	401	权限错误
AccessDenied	No permission to accessor or permission error.	403	禁止操作
NoSuchObject	The specified object is not found or resource do not exist.	404	对象未找到
IllegalOperation	Illegal operation type.	405	操作类型错误

OperationDenied	Operation not available.	409	不可操作（重复名字、重建正在进行中等）
LimitExceeded	Limit exceeded for current request.	413	超出限制
ServiceInternalError	Internal service occurs error.	500	服务内部错误
ServiceUnavailable	Service is unavailable.	503	服务不可用
Volume.DiskIdIsEmpty	Volume id must be provided.	400	磁盘id不允许为空
Volume.DiskNameInvalid	Volume name is invalid.	400	名称格式不正确
Volume.DiskDescriptionInvalid	Volume description is invalid.	400	描述格式不正确
Volume.DiskSizeInvalid	Volume size is invalid.	400	大小不合法
Volume.VolumeTypeInvalid	Volume type is invalid.	400	磁盘类型不正确
Volume.DiskQuotaExceedLimit	Number of volumes exceeds limit.	413	cds磁盘超额
Volume.DiskSizeQuotaExceededLimit	Volume size exceeds total capacity quota.	413	cds磁盘大小超额
Instance.ReleaseTimeInvalid	Fail to update the instance scheduled release time settings.	400	定时释放参数不正确
Instance.AttachedVolumeExceededSnapshotLimit	Scheduled snapshot quota of your new mount CDS disk is exceeded, please change the policy or manually deleted non-scheduled snapshot.	413	挂载cds磁盘数目超出限制
Instance.SnapshotPolicyRetainsDaysInvalid	Days format of retention of scheduled snapshot is invalid.	400	自动快照保留天数不合法
Instance.SnapshotPolicyNameInvalid	The policy name of scheduled snapshot is invalid.	400	自动快照策略名称不合法
Instance.AdminPassDecryptionException	Invalid encryption code found for the admin password.	400	无效的加密密码
PaymentFailed	payment failed, please check remaining balance.	403	支付失败
Asp.NameInvalid	asp name is invalid.	400	名称非法
Asp.RetentionDaysInvalid	RetentionDays should in [1,10000]	400	保留天数非法
Asp.TimePointsInvalid	TimePoints should in [0,23]	400	执行时间点参数异常
Asp.RepeatWeekdaysInvalid	RepeatWeekdays should in [1,7]	400	执行日期参数异常
Asp.VolumeTypeInvalid	Volume type should be Cds or System	400	volume类型非法
Asp.VolumeStatusInvalid	Volume must be in-use	400	volume状态必须为in-use
BadRequest	missing parameter.	400	缺少参数
ServiceInternalError	Internal service occurs error.	500	内部错误
BadRequest	Bad request parameters or illegal request.	400	请求参数错误

NoSuchObject	The specified object is not found or resource do not exist.	404	指定的asp不存在
BadRequest	Bad request parameters or illegal request.	400	请求参数无效

实例相关接口

创建实例（通过指定实例套餐规格）

前提条件

创建实例需要实名认证，没有通过实名认证的可以前往百度智能云官网控制台中的安全认证下的个人认证或企业认证中进行认证。

BCC实例

- 计费注意事项：
  - 创建计费方式为后付费的实例需要账户现金余额+通用代金券大于100；预付费方式的实例则需要账户现金余额大于等于实例费用。
  - 指定公网IP带宽创建，计费方式为按照带宽计费。
- 实例注意事项：
  - 通过spec字段指定需要创建的虚机类型，通过实例套餐规格列表接口来查看可使用实例套餐及套餐规格。
  - 同一实例不可以同时购买GPU卡和FPGA卡，购买带有GPU卡或FPGA卡的实例时必须通过cardCount指定要购买的虚机所携带的GPU卡或FPGA卡数量。
- 存储注意事项：
  - 创建实例时可以指定系统盘大小，指定系统盘大小范围为[40, 500]GB，可指定系统盘磁盘类型可参见StorageType。
  - 创建实例时添加的系统盘在创建之后无法更改，且不支持单独创建或添加系统盘。
  - 系统盘不支持挂载、卸载、删除。
  - 创建CDS系统盘和数据盘时，支持按照1GB步长选择磁盘大小。
  - 缺省情形下，一个实例最多只能挂载5个云磁盘。
  - BCC实例的系统盘最大不能超过500G。
  - 每个实例最多只能购买一块临时数据盘，临时数据盘默认只有hp1类型。
  - 创建存储优化型实例时必须购买临时数据盘，通过ephemeralDisks指定临时盘数据盘大小，默认nvme类型数据盘，无需指定。
- 网络注意事项：
  - BCC实例支持用户自定义安全组和子网，两者需要同时指定在同一个VPC下。如果未指定安全组和子网，在创建专属实例时会自动加入系统默认的安全组和子网内。
- 其他注意事项：
  - 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
  - 创建接口为异步创建，可通过查询指定实例详情接口查询实例状态。

请求

请求结构

```
 POST /v{version}/instanceBySpec?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

```
{
    "spec": spec,
    "rootDiskSizeInGb":rootDiskSizeInGb,
    "rootDiskStorageType":"rootDiskStorageType",
    "adminPass": "adminPass",
    "ephemeralDisks": [
        {
            "storageType": storageType,
            "sizeInGB": size
        },
        {
            "storageType": storageType,
            "sizeInGB": size
        }
    ],
    "createCdsList": [
        {
            "storageType": storageType,
            "cdsSizeInGB": cdsSizeInGB,
            "snapshotId": snapshotId
        }
    ],
    "name": "name",
    "hostname": "hostname",
    "autoSeqSuffix": autoSeqSuffix,
    "isOpenHostnameDomain": isOpenHostnameDomain,
    "imageId": "imageId",
    "networkCapacityInMbps": networkCapacityInMbps,
    "eipName": "eipName",
    "autoRenewTimeUnit": "autoRenewTimeUnit",
    "autoRenewTime": autoRenewTime,
    "internetChargeType": internetChargeType,
    "dedicatedHostId": dedicatedHostId,
    "purchaseCount": purchaseCount,
    "billing":{
        "paymentTiming": paymentTiming,
        "reservation": {
            "reservationLength": reservationLength
        }
    },
    "zoneName": zoneName,
    "subnetId": subnetId,
    "securityGroupId": securityGroupId,
    "relationTag": relationTag,
    "isOpenIpv6": isOpenIpv6,
    "tags": [
      {
        "tagKey": tagKey,
        "tagValue": tagValue
      }
    ],
    "userData": "",
    "keypairId": keypairId,
    "aspId": "aspId",
    "specId": "specId",
    "resGroupId": "resGroupId",
    "ehcClusterId": "ehcClusterId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
imageId	String	是	RequestBody参数	待创建虚拟机实例的镜像ID，可通过调用查询镜像列表接口选择获取所需镜像ID。
spec	String	否	RequestBody参数	待创建虚拟机实例的套餐规格（使用该参数可替代填写instanceType，cpuCount，memoryCapacityInGB，gpuCard，fpgaCard，cardCount，ephemeralDisks参数），通过使用实例套餐规格列表接口来查看可使用实例套餐及套餐规格。
rootDiskSizeInGb	int	否	RequestBody参数	待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 100]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
rootDiskStorageType	String	否	RequestBody参数	待创建虚拟机实例系统盘介质，默认使用高性能云磁盘（hp1），可指定系统盘磁盘类型可参见StorageType。
ephemeralDisks	List<EphemeralDisk>	否	RequestBody参数	BCC实例最多只能使用一块本地盘，使用默认磁盘类型，需要指定磁盘大小。创建FPGA实例或GPU实例该参数无效，默认使用指定大小的本地磁盘。
createCdsList	List<CreateCdsModel>	否	RequestBody参数	待创建的CDS磁盘列表，具体数据格式参见下述CreateCdsModel
networkCapacityInMbps	int	否	RequestBody参数	公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
eipName	String	否	RequestBody参数	长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
internetChargeType	String	否	RequestBody参数	公网带宽计费方式，可选参数详见internetChargeType，若不指定internetChargeType，默认付费方式同BCC，预付费默认为包年包月按带宽，后付费默认为按使用带宽计费。
purchaseCount	int	否	RequestBody参数	批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
name	String	否	RequestBody参数	虚拟机名字（可选）。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
				虚拟机主机名（可选）。默认都不指定hostname。如果指定hostname：批量时hostname

hostname	String	否	RequestBody参数	作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。仅支持小写字母、数字以及-．特殊字符，必须以字母开头，不可连续使用特殊符号，不支持特殊符号开头或结尾，长度2-64。
autoSeqSuffix	boolean	否	RequestBody参数	是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
isOpenHostnameDomain	boolean	否	RequestBody参数	是否自动生成hostname domain（可选参数） 是:true 否:false
adminPass	String	否	RequestBody参数	待指定的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()，密码需要加密传输
billing	Billing	是	RequestBody参数	订单、计费相关参数
zoneName	String	否	RequestBody参数	指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。专属实例使用专属服务器所在zone,无需指定该字段。
subnetId	String	否	RequestBody参数	指定subnet信息，为空时将使用默认子网
securityGroupId	String	否	RequestBody参数	已废弃，指定securityGroup信息，为空时将使用默认安全组
securityGroupIds	List	否	RequestBody参数	指定securityGroup信息列表，为空时将使用默认安全组
enterpriseSecurityGroupIds	List	否	RequestBody参数	指定企业安全组信息列表，不为空时，securityGroupIds无效，为空时将使用默认安全组
relationTag	boolean	否	RequestBody参数	其关联资源CDS（数据盘）、EIP、快照、快照链是否统一加标签，默认为false。
isOpenIpv6	boolean	否	RequestBody参数	待创建实例是否开启ipv6，只有当镜像和子网都支持ipv6时才可开启，true表示开启，false表示关闭，不传表示自动适配镜像和子网的ipv6支持情况
tags	List<TagModel>	否	RequestBody参数	待创建的标签列表，具体数据格式参见下述TagModel
keypairId	String	否	RequestBody参数	待创建实例所要绑定的密钥对ID
autoRe			Reques	

autoRenewTimeUnit	String	否	RequestBody参数	按月付费或者按年付费 月是"month",年是"year"
autoRenewTime	int	否	RequestBody参数	自动续费的时间 按月是1-9 按年是 1-3
cdsAutoRenew	boolean	否	RequestBody参数	【此参数废弃，cds自动续费和bcc保持一致】cds是否自动续费 是:true 否:false
aspId	String	否	RequestBody参数	自动快照策略ID
bidModel	String	否	RequestBody参数	竞价实例出价模型， 市场价: "market" 自定义："custom"。参考BidModel
bidPrice	String	否	RequestBody参数	竞价实例出价金额，单位：元/分钟。若是自定义出价，且出价金额小于市场价，则不允许创建。当bidModel='custom'时才有效。
dedicatedHostId	String	否	RequestBody参数	专属服务器id，指定虚机置放位置时指定该值。
deployId	String	否	RequestBody参数	指定实例所在的部署集id
deployIdList	List	否	RequestBody参数	指定实例所在的部署集id列表
userData	String	否	RequestBody参数	若实例满足使用实例自定义数据的限制，您可传入UserData信息。因为传输API请求时，不会加密您设置的UserData，建议不要以明文方式传入机密的信息，例如密码和私钥等。如果必须传入，建议加密后，然后以Base64的方式编码后再传入，在实例内部以同样的方式反解密。
specId	String	否	RequestBody参数	规格族
deletionProtection	int	否	RequestBody参数	实例释放保护，默认0未开启，1开启，开启后禁止手动释放虚机
isOpenHostEye	Boolean	否	RequestBody参数	是否开启主机安全，true:开启，false：关闭；不传默认开启
hosteyeType	String	否	RequestBody参数	主机安全类型，isOpenHostEye为true时生效； "enterprise"，表示开启企业版主机安全；“open”或不传，表示开启基础版主机安全
resGroupId	String	否	RequestBody参数	指定实例要绑定的资源组id
enableJ			Reques	

umboFrame	boolean	否	tBody参数	是否开启Jumbo帧，默认值false，开启:true，关闭:false。注意:只有支持Jumbo帧的套餐才能开启
enableHt	boolean	否	RequestBody参数	是否开启Ht,ebc使用，默认值true, true:开启，false:关闭
dataPartitionType	String	否	RequestBody参数	数据盘文件格式,可选值：xfs，ext4
rootPartitionType	String	否	RequestBody参数	系统盘文件格式,可选值：xfs，ext4
fileSystems	List<FileSystemModel>	否	RequestBody参数	指定实例要挂载的cfs文件系统列表
disableRootDiskSerial	Boolean	否	RequestBody参数	创建实例时是否隐藏系统盘SN,默认false。 true:隐藏 fase:不隐藏
internalIps	List	否	RequestBody参数	内网IP列表
networkPurchaseType	String	否	RequestBody参数	EIP线路类型，包含标准BGP（BGP）和增强BGP（BGP_S），默认标准BGP
ehcClusterId	String	否	RequestBody参数	创建roce实例时可选参数，若为空则使用默认EHC集群
cpuThreadConfig	String	否	RequestBody参数	调整每物理核的线程数（vCPU），本质上对应各处理器的超线程能力是否启用。 取值范围：1、2 注意： 1. 默认情况下，百度智能云实例按照默认值2配置。 2. 仅Intel第七代以上、AMD第三代以上的裸金属实例规格支持设置 CPU 线程数。
numaConfig	String	否	RequestBody参数	调整CPU的NUMA配置，对于不同处理器平台取值有不同含义。 1. Intel平台：0代表关闭NUMA特性，1代表开启NUMA特性。 2. AMD平台：主要影响NPS(Nodes Per Socket)的配置，可取值0、1、2、4、auto，分别对应NPS0、NPS1、NPS2、NPS4以及自动。 注意： 1. 默认情况下，百度智能云实例按照开启NUMA优化（Intel实例）、NPS1（AMD实例），也即默认值1配置。 2. 仅Intel第七代以上、AMD第三代以上的裸金属实例规格支持设置NUMA选项。

返回

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
instanceIds	List	虚机实例ID的集合，其中ID符合BCE规范，必须是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
warningList	List	创建虚机产生的warning信息

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.InsufficientRootDiskSize	Not enough root disk size.	400	使用镜像过大，系统盘大小不够
Instance.InvalidateRootDiskStorageType	Invalidate root disk storage type.	400	无效的系统盘介质
Instance.InstanceBeBanned	The instance is banned, contact technical support for assistance.	403	Bcc服务已被封禁
Instance.GlobalInstanceQuotaLimitExceeded	Quota exceeds limit for regional postpaid instances pool, contact technical support for further assistance.	413	后付费类型的虚机数目超出当前区域预售额度限制
Instance.InstanceQuotaLimitExceeded	Quota exceeds limit.	413	实例个数超过限制
Instance.EipQuotaLimitExceeded	The number of eip will exceed the limit.	413	合并购买时eip个数超过限制
Instance.InvalidateConfiguration	Invalidate cpu memory configuration.	400	无效的虚机配置
BadRequest	spec is invalid.	400	待创建虚拟机实例的套餐规格参数无效
Instance.MKTImageInstanceNotSupportCreateSnapshotException	Instance with mkt image not support create system snapshot.	403	云市场镜像不支持创建系统快照

示例

请求示例（通过指定实例套餐规格spec创建实例）

```
 POST /v2/instanceBySpec?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "spec": "bcc.g3.c4m12",
    "createCdsList": [
        {
            "storageType": "std1",
            "cdsSizeInGB": 5
        }
    ],
    "name": "example",
    "imageId": "m-mRjO3cjs",
    "networkCapacityInMbps": 0,
    "purchaseCount": 1,
    "billing":{
        "paymentTiming":"Prepaid",
        "reservation": {
            "reservationLength": 1
        }
    }
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "instanceIds": [
        "i-T1I3OtUO"
    ],
    "warningList": [
        "warning info"
    ]
}
```

查询实例列表

该接口用于查询所有实例的详细信息。

注意事项：不同查询字段之间是and关系，同一查询字段传入的多个值之间是or关系。

请求结构

```
 GET /v{version}/instance?marker={marker}&maxKeys={maxKeys}&internalIp={internalIp}&dedicatedHostId=
{dedicatedHostId}&zoneName={zoneName}&autoRenew={autoRenew} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
internalIp	String	否	Query参数	内网IP
dedicatedHostId	String	否	Query参数	专属服务器ID
zoneName	String	否	Query参数	可用区信息
instanceIds	String	否	Query参数	多个实例ID，英文逗号分割，最多支持100个
volumeIds	String	否	Query参数	多个磁盘ID，英文逗号分割，最多支持100个
instanceNames	String	否	Query参数	多个实例名称，英文逗号分割，精确搜索。最多支持100个
fuzzyInstanceName	String	否	Query参数	实例的名称，支持关键字模糊查询。与instanceNames参数只能2选1，不能同时传值，否则报错。
deploySetIds	String	否	Query参数	多个部署集ID，英文逗号分割，最多支持100个
securityGroupIds	String	否	Query参数	多个安全组ID，英文逗号分割，最多支持100个
paymentTiming	String	否	Query参数	支付方式（Prepaid / Postpaid）
status	String	否	Query参数	实例状态（Recycled / Running / Stopped / Stopping / Starting）
tags	String	否	Query参数	多个标签，逗号分割，格式：tagKey:tagValue 或 tagKey
vpcId	String	否	Query参数	vpcId，只能与privateIps查询参数组合使用
privateIps	String	否	Query参数	多个内网IP，英文逗号分隔，最多支持100个，必须和vpcId组合使用
ehcClusterId	String	否	Query参数	ehc集群id

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
instances	List<InstanceModel>	实例信息，由 InstanceModel 组成的集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	zone is invalid.	400	无效的可用区
NoSuchObject	subnet not exists.	404	子网不存在

请求示例

```
 GET /v2/instance?vpcId=vpc-vv3xw6d9970b&privateIps=192.168.48.3 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "marker": "",
    "isTruncated": false,
    "maxKeys": 1000,
    "instances": [
        {
            "id": "i-zadG8d4l",
            "name": "instance-696snyc6",
            "roleName": "",
            "hostname": "instance-696snyc6",
            "instanceType": "N6",
            "spec": "bcc.g5.c1m4",
            "status": "Running",
            "desc": "",
            "createdFrom": "customPurchase",
            "paymentTiming": "Prepaid",
            "createTime": "2022-10-11T06:11:42Z",
            "expireTime": "2023-01-11T06:11:42Z",
            "internalIp": "192.168.48.3",
            "publicIp": "",
            "cpuCount": 1,
            "isomerismCard": "",
            "cardCount": "0",
            "npuVideoMemory": "",
            "memoryCapacityInGB": 4,
            "localDiskSizeInGB": 0,
            "imageId": "m-zIaNc3qH",
            "placementPolicy": "unknown",
            "subnetId": "sbn-2escuever5fi",
            "vpcId": "vpc-vv3xw6d9970b",
```

```
            "vpcId": "vpc-vv3xw6d9970b",
            "zoneName": "cn-bj-a",
            "dedicatedHostId": "",
            "deletionProtection": 0,
            "deploysetList": [],
            "autoRenew": true,
            "ipv6": "",
            "resGroupInfos":[
                {
                    "groupId":"RESG-jec9gimkKaY",
                    "groupName":"test2"
                }
            ],
            "nicInfo": {
                "eniId": "eni-ramtd68yeyiq",
                "eniUuid": "8ed449ff-b2ee-4491-b117-35fb731c4370",
                "name": "eth0",
                "type": "primary",
                "subnetId": "sbn-2escuever5fi",
                "subnetType": "BCC",
                "az": "zoneA",
                "description": "",
                "deviceId": "608666b9-7156-4f47-8445-9c5497282573",
                "status": "inuse",
                "macAddress": "fa:f6:00:01:7e:ef",
                "vpcId": "vpc-vv3xw6d9970b",
                "createdTime": "Tue Oct 11 14:11:44 UTC 2022",
                "eniNum": 1,
                "eriNum": 0,
                "eriInfos": [],
                "ips": [
                    {
                        "privateIp": "192.168.48.3",
                        "eip": "null",
                        "primary": "true",
                        "eipId": "",
                        "eipAllocationId": "",
                        "eipSize": "0",
                        "eipStatus": "",
                        "eipGroupId": "",
                        "eipType": "null"
                    }
                ],
                "securityGroups": []
            },
            "eniNum": "1",
            "tags": [
                {
                    "tagKey": "test2",
                    "tagValue": "bcc"
                }
            ],
            "networkCapacityInMbps": 0
        }
    ]
}
```


查询指定实例详情

该接口用于查询指定实例的详细信息。

请求结构

```
GET /v{version}/instance/{instanceId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待查询的实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
instance	InstanceModel	返回的实例详情

请求示例

```
GET /v2/instance/i-CFQRKIT2 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "instance": {
    "id": "i-JmwRSUbR",
    "name": "instance-4nbckz0b",
    "roleName": "",
    "hostname": "instance-4nbckz0b",
    "instanceType": "N5",
    "spec": "bcc.g4.c2m8",
    "status": "Running",
    "repairStatus": "normal",
    "desc": "",
    "createdFrom": "",
    "paymentTiming": "Postpaid",
    "createTime": "2023-06-13T08:07:52Z",
    "internalIp": "172.16.16.15",
    "publicIp": "",
    "cpuCount": 2,
    "isomerismCard": "",
    "cardCount": "0",
    "npuVideoMemory": "",
    "memoryCapacityInGB": 8,
    "localDiskSizeInGB": 0,
    "imageId": "m-Z1TCx0Zv",
```

```
    "imageName": "slurm-good3-0531",
    "imageType": "custom",
    "placementPolicy": "default",
    "subnetId": "sbn-b7ekx6uj6k6m",
    "vpcId": "vpc-41avheyaawqc",
    "hostId": "",
    "switchId": "",
    "rackId": "",
    "deploysetId": "",
    "zoneName": "cn-bd-b",
    "dedicatedHostId": "",
    "osVersion": "20.04 LTS",
    "osArch": "amd64 (64bit)",
    "osName": "Ubuntu",
    "hosteyeType": "open",
    "deploysetList": [],
    "resGroupInfos":[
        {
            "groupId":"RESG-jec9gimkKaY",
            "groupName":"test2"
        }
      ],
    "nicInfo": {
      "eniId": "eni-4scnfvrstj72",
      "eniUuid": "95da812f-f27f-4456-aad9-e27cf13d3e61",
      "name": "eth0",
      "type": "primary",
      "subnetId": "sbn-b7ekx6uj6k6m",
      "subnetType": "BCC",
      "az": "zoneB",
      "description": "",
      "deviceId": "3ac2af07-c13c-4bfd-ab00-a19706033b4a",
      "status": "inuse",
      "macAddress": "fa:26:00:0e:38:26",
      "vpcId": "vpc-41avheyaawqc",
      "createdTime": "Tue Jun 13 16:07:52 UTC 2023",
      "eniNum": 0,
      "eriNum": 0,
      "eriInfos": [],
      "ips": [
        {
          "privateIp": "172.16.16.15",
          "eip": "null",
          "primary": "true",
          "eipId": "",
          "eipAllocationId": "",
          "eipSize": "0",
          "eipStatus": "",
          "eipGroupId": "",
          "eipType": "null"
        }
      ],
      "securityGroups": [
        "g-wxsaduu9uajq"
      ],
      "enterpriseSecurityGroups": []
    },
    "deletionProtection": 0,
    "ipv6": "",
    "tags": [
      {
        "tagKey": "nodeType",
```

```
          "tagValue": "compute"
        }
      ],
      "volumes": [
        {
          "isSystemVolume": true,
          "diskSizeInGB": 40,
          "volumeId": "v-ClTVjiB4"
        }
      ],
      "networkCapacityInMbps": 0
    }
  }
```

启动实例

该接口用于启动指定实例。

- 实例状态必须为Stopped，调用此接口才可以成功返回，否则提示409错误。
- 接口调用成功后实例进入Starting状态。

请求结构

```
PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待启动的实例ID
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：start

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.StartInstanceOperationForbidden	Instance not stopped when starting.	409	只有 Stopped 状态的实例才可以进行开启
Instance.StartInstanceWhenCreating	Instance creating when starting.	409	创建中的实例不可启动
Instance.StartExpiredInstance	Instance expired when starting.	409	过期 （expired）实例不可启动

请求示例

```
 PUT /v2/instance/i-YufwpQAe?start HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

启动实例（批量启动）

该接口用于批量启动实例。

- 实例状态必须全部为Stopped，调用此接口才可以成功返回，否则提示409错误。
- 接口调用成功后实例进入Starting状态。

请求结构

```
 PUT /v{version}/instance/batchAction?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceIds":["instanceId1", "instanceId2"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：start
instanceIds	List	是	RequestBody参数	实例id组成的集合，集合元素数量不超过100个

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.StartInstanceOperationForbidden	Instance not stopped when starting.	409	只有 Stopped 状态的实例才可以进行开启
Instance.StartInstanceWhenCreating	Instance creating when starting.	409	创建中的实例不可启动
Instance.StartExpiredInstance	Instance expired when starting.	409	过期 （expired）实例不可启动

请求示例

```
 PUT /v2/instance/batchAction?start HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds": ["instanceId1"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

停止实例

该接口用于停止指定实例。

- 系统后台会在实例实际Stop成功后进入“已停止”状态。
- 只有状态为Running的实例才可以进行此操作，否则提示409错误。
- 实例支持强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "forceStop":forceStop,
    "stopWithNoCharge":stopWithNoCharge
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待启动的实例ID
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：stop
forceStop	Boolean	否	RequestBody参数	指定实例是否强制关机，可选值：true、false，缺省为false
stopWithNoCharge	Boolean	否	RequestBody参数	指定实例是否关机不计费，可选值：true、false，缺省为false

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.StopInstanceOperationForbidden	Instance not running when shutting down	409	只有 Running 状态的实例才可以进行关机
Bcc.PrePayCannotStopWithNoCharge	pre pay instance can not stop with no charge.	403	预付费实例不能进行关机不计费
Bcc.Rocv1CannotStopWithNoCharge	Shutdown no charge feature is not available for instances with local disk.	403	关机不收费功能不适用于带有本地盘的实例
Bcc.InstanceTypeNotSupportException	Shutdown no charge feature is not available for this type of instance.	403	关机不收费功能不适用于该类型实例
Bcc.InstanceTypeNotSupportException	The instance is in the state of shutdown and no charge. This operation is not supported.If you need to do this, please cancel the shutdown and not charge.	403	实例正处于关机不收费状态中，不支持该操作，如需进行此操作，请先取消关机不计费

请求示例

```
 PUT /v2/instance/i-YufwpQAe?stop HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "forceStop":false,
    "stopWithNoCharge":true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

停止实例（批量停止）

该接口用于批量停止指定实例。

- 系统后台会在实例实际Stop成功后进入“已停止”状态。
- 只有状态全部为Running的实例才可以进行此操作，否则提示409错误。
- 实例支持强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。
- 只有全部实例都满足条件时，批量关机不计费才会生效，可调用查询可关机不计费的实例列表接口查询支持关机不计费的实例列表。

请求结构

```
 PUT /v{version}/instance/batchAction?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceIds": ["instanceId1","instanceId2"],
    "forceStop":forceStop,
    "stopWithNoCharge": stopWithNoCharge
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：stop
instanceIds	List	是	Request Body参数	实例id组成的集合，集合元素数量不超过100个
forceStop	Boolean	否	RequestBody参数	指定实例是否强制关机，可选值：true、false，缺省为false
stopWithNoCharge	Boolean	否	RequestBody参数	指定实例是否关机不计费，可选值：true、false，缺省为false

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.StopInstanceOperationForbidden	Instance not running when shutting down	409	只有 Running 状态的实例才可以进行关机
Bcc.PrePayCannotStopWithNoCharge	pre pay instance can not stop with no charge.	403	预付费实例不能进行关机不计费
Bcc.Rocv1CannotStopWithNoCharge	Shutdown no charge feature is not available for instances with local disk.	403	关机不收费功能不适用于带有本地盘的实例
Bcc.InstanceTypeNotSupportException	Shutdown no charge feature is not available for this type of instance.	403	关机不收费功能不适用于该类型实例
Bcc.InstanceTypeNotSupportException	The instance is in the state of shutdown and no charge. This operation is not supported.If you need to do this, please cancel the shutdown and not charge.	403	实例正处于关机不收费状态中，不支持该操作，如需进行此操作，请先取消关机不计费
Instance.StopInstanceOperationForbidden	Instance not running when shutting down	409	只有 Running 状态的实例才可以进行关机
Bcc.StopWithNoChargeForbidden	batch stop with no charge failed. + {[instanceId1,instanceId2]:errMsg}	409	批量关机不计费操作失败，{实例id：不支持的原因}

请求示例

```
 PUT /v2/instance/batchAction?stop HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceIds": ["instanceId1"],
    "forceStop":false,
    "stopWithNoCharge": true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

重启实例

该接口用于重启指定实例。

- 只有状态为Running的实例才可以进行此操作，否则提示409错误。
- 接口调用成功后实例进入Starting状态。
- 支持强制重启，强制重启等同于传统服务器的断电重启，可能丢失实例操作系统中未写入磁盘的数据。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "forceStop":forceStop
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待启动的实例ID
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：reboot
forceStop	boolean	否	Request Body参数	是否强制停止实例，可选值true,false,缺省为false

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.RebootInstanceOperationForbidden	Instance not running when rebooting.	409	只有 Running 状态的实例才可以进行重启

请求示例

```
 PUT /v2/instance/i-YufwpQAe?reboot HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "forceStop": true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

修改实例密码

该接口用于修改指定虚机的管理员密码。

- 只有Running和Stopped状态的实例才可以用调用接口，否则提示409错误。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
   "adminPass": adminPass
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	指定的实例ID
action	String	是	Query参数	对实例执行的动作，当前取值：changePass
adminPass	String	是	Request Body参数	待指定的实例管理员密码，8-32位字符，英文，数字和符号必须同时存在，符号仅限`!@#$%^*()`，密码需要加密传输，详见密码加密传输规范定义

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.PasswordOperationForbidden	Instance not stopped when changing password.	409	只有 Stopped 状态的实例才可以修改密码

请求示例

```
 PUT /v2/instance/i-YufwpQAe?changePass HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "adminPass": "c5b31add-ffe0-44b0-86eb-063109a1b6fe"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

修改实例属性

该接口用于修改指定BCC实例的属性，目前仅支持修改实例名称或网卡队列数；

- 修改实例名称时，只有 运行中（Running ）和已关机（ Stopped ）状态的实例才可以调用该接口，否则提示 409 错误；
- 修改实例网卡队列数时，只有已关机（Stopped）且不是关机不计费状态才可以调用该接口；
- 弹性裸金属EBC类实例不支持修改网卡队列数；
- 修改后的实例网卡队列数不允许大于实例规格支持的最大网卡队列数，实例规格标称网卡队列数和最大网卡队列数可通过查询实例套餐规格接口查询。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "name": InstanceName,
    "netEthQueueCount": newNetEthQueueCount
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	指定的实例ID
action	String	是	Query参数	对实例执行的动作，当前取值：modifyAttribute。
name	String	否	Request Body参数	实例名称，支持大小写字母、数字以及-_/.特殊字符，必须以字母开头，长度1-65个字符。
netEthQueueCount	Integer	否	Request Body参数	修改后的网卡队列数
enableJumboFrame	boolean	否	Request Body参数	是否开启Jumbo帧，开启:true，关闭:false。注意:只有支持Jumbo帧的套餐才能开启

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
Instance.NameInvalid	Instance name is invalid.	400	name无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
 PUT /v2/instance/i-YufwpQAe?modifyAttribute HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "name": "new_instance_name",
   "netEthQueueCount": 3
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

重装实例

该接口用于指定实例重装操作系统。

- 实例重装后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。
- EBC实例在已关机状态下不支持重装。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "imageId": imageId,
    "adminPass":"adminPass"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	指定的实例ID
action	String	是	Query参数	对实例执行的动作，当前取值：rebuild
imageId	String	是	Request Body参数	待指定的镜像ID
adminPass	String	否	Request Body参数	机器密码，密码需要加密传输，详见密码加密传输规范，必须传递adminPass、keypairId其中一个参数
keypairId	String	否	RequestBody参数	待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
userData	String	否	RequestBody参数	待重装的自定义脚本
cleanLastUserData	boolean	否	RequestBody参数	清空历史userData，默认false，即默认重装历史userData（包括gpu驱动和CFS文件系统）
isOpenHostEye	boolean	否	RequestBody参数	是否开启主机安全，默认true
sysRootSize	int	否	RequestBody参数	仅对ebc实例生效。系统盘大小，仅对ebc实例生效
isPreserveData	boolean	否	RequestBody参数	仅对ebc实例本地盘生效。是否保留数据重装，当值为true时，raidId和sysRootSize字段不生效
raidId	String	否	RequestBody参数	仅对ebc实例生效。raid配置Id，此参数在isPreserveData为false时为必填，在isPreserveData为true时不生效
dataPartitionType	String	否	RequestBody参数	仅对ebc实例生效。数据盘分区格式
rootPartitionType	String	否	RequestBody参数	仅对ebc实例生效。系统盘分区格式

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.InsufficientRootDiskSize	Not enough root disk size.	400	使用镜像过大，系统盘大小不够
Instance.TemplateInPackageRebuildException	Template in package can not rebuild.	400	镜像有关联CDS快照不能用于重装

请求示例

```
 PUT /v2/instance/i-YufwpQAe?rebuild HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "imageId": "m-mRjO3cjs",
    "adminPass":"1qaz@WSX"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

重装实例（批量重装）

该接口用于批量多个BCC实例重装操作系统。

- 实例重装后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。
- EBC实例在已关机状态下不支持重装。

请求结构

```
 PUT /v{version}/instance/rebuild HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "instanceIds": [
        "instanceId"
    ],
    "imageId": imageId,
    "adminPass":"adminPass"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceIds	List<String>	是	RequestBody参数	指定的实例ID列表，最多100个
imageId	String	是	RequestBody参数	待指定的镜像ID
adminPass	String	否	RequestBody参数	机器密码，密码需要加密传输，详见密码加密传输规范 ，必须传递adminPass、keypairId其中一个参数
keypairId	String	否	RequestBody参数	待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
userData	String	否	RequestBody参数	自定义脚本
useLastUserData	boolean	否	RequestBody参数	是否使用当前自定义脚本，默认true
isOpenHostEye	boolean	否	RequestBody参数	是否开启主机安全，默认true
sysRootSize	int	否	RequestBody参数	仅对ebc实例生效。系统盘大小，仅对ebc实例生效
isPreserveData	boolean	否	RequestBody参数	仅对ebc实例本地盘生效。是否保留数据重装。当值为true时，raidId和sysRootSize字段不生效
raidId	String	否	RequestBody参数	仅对ebc实例生效。raid配置Id，此参数在isPreserveData为false时为必填，在isPreserveData为true时不生效
dataPartitionType	String	否	RequestBody参数	仅对ebc实例生效。数据盘分区格式
rootPartitionType	String	否	RequestBody参数	仅对ebc实例生效。系统盘分区格式

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.InsufficientRootDiskSize	Not enough root disk size.	400	使用镜像过大，系统盘大小不够

请求示例

```
 PUT /v2/instance/rebuild HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceIds": [
        "i-syGfPUYO",
        "i-2EzRRGDx"
    ]
    "imageId": "m-mRjO3cjs",
    "adminPass":"1qaz@WSX"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

将实例加入安全组

该接口用于为指定BCC实例添加普通安全组，每个实例最多关联10个安全组。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "securityGroupId" : securityGroupId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	虚拟机实例ID
action	String	是	Query参数	本接口取值：bind
securityGroupId	String	是	Request Body参数	要关联到该实例上的安全组ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
SecurityGroup.BindInstanceExceededLimit	Quota exceeds limit when binding instance to security group.	413	实例绑定安全组数目超限
SecurityGroup.ExpiredInstanceForbiddenBind	Instance already expired when binding to security group.	403	过期或欠费的实例无法绑定安全组
SecurityGroup.NoSecurityGroup	No security group associated for the instance.	400	实例至少需要关联一个安全组

请求示例

```
 PUT /v2/instance/i-YufwpQAe?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "securityGroupId" : "g-nky7qeom"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

将实例移出安全组

该接口用于将BCC实例从指定的安全组中移出。

- 每个实例至少关联一个安全组，默认关联默认安全组。
- 如果实例仅属于一个安全组，尝试移出时，请求会报 403 错误。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "securityGroupId" : securityGroupId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	实例ID
action	String	是	Query参数	本接口取值：unbind
securityGroupId	String	是	Request Body参数	要从该实例取消关联的安全组ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 PUT /v2/instance/i-YufwpQAe?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "securityGroupId" : "g-nky7qeom"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询实例VNC地址

该接口用于查询指定实例的VNC地址URL。

- VNC地址一次使用后即失效。
- URL地址有效期为10分钟。

请求结构

```
 GET /v{version}/instance/{instanceId}/vnc HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	虚拟机实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
vncUrl	String	VNC 地址URL

请求示例

```
 GET /v2/instance/i-YufwpQAe/vnc HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "vncUrl": "http://10.105.97.40/vnc_auto.html?token=e4328aa7-d1ba-48be-99d1-aa4bfe51ab38"
}
```

实例续费

该接口用于对指定包年包月（预付费）实例操作续费，延长到期时长。

- 实例变配期间不能进行续费操作。
- 续费时若实例已欠费停机，续费成功后该实例将重新启动。
- 支持关联的包年包月（预付费）CDS/EIP/MKT的产品一同续费，关联的按量付费（后付费）CDS/EIP/MKT不会一同续费。
- 该接口是一个异步接口。
- 专属实例不支持续费。
- 以月为单位进行续费。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action}&relatedRenewFlag={relatedRenewFlag}&clientToken={clientToken}
HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "billing":{
        "reservation":{
            "reservationLength": reservationLength
        }
    }

}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	虚拟机实例ID
action	String	是	Query参数	本接口中该参数取值：purchaseReserved
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
relatedRenewFlag	String	否	Query参数	设置实例关联续费标识，默认为空字符串。可选标识请参照RelatedRenewFlag。**注意**：实例关联的CDS盘会一同进行续费。
billing	Billing	是	RequestBody参数	订单详情
cdsCustomPeriod	CdsCustomPeriod	否	RequestBody参数	挂载磁盘的自定义续费周期，可选大于等于（续费后BCC到期时间-CDS当前到期时间）的续费时长，范围为【1，60】个月。不传时默认与BCC续费后的到期时间保持一致。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	订单id

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
 PUT /v2/instance/i-YufwpQAe?purchaseReserved&clientToken=be31b98c-5e41-4838-9830-9be700de5a20&relatedRenewFlag=CDS_EIP_MKT HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
 "billing":{
    "reservation":{
       "reservationLength":1
    }
 }
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{"orderId": "daeeb9f0cc8a4b61aa4a7dafd58630a2"}
```

释放后付费实例(POST请求的释放)

该接口用于释放单个按量付费（后付费）云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。

请求结构

```
 POST /v{version}/instance/{instanceId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "relatedReleaseFlag": relatedReleaseFlag,
    "deleteCdsSnapshotFlag":deleteCdsSnapshotFlag,
    "bccRecycleFlag" : bccRecycleFlag,
    "deleteRelatedEnisFlag" : deleteRelatedEnisFlag,
    "cdsAttributeActive" : cdsAttributeActive
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	实例id
relatedReleaseFlag	boolean	是	RequestBody参数	是否关联释放当前时刻实例挂载的全部EIP和云磁盘数据盘，当值为true且cdsAttributeActive参数值为true时数据盘只释放有关联释放属性的磁盘，cdsAttributeActive为false时释放全部磁盘
deleteCdsSnapshotFlag	boolean	是	RequestBody参数	是否在释放云磁盘时释放云磁盘的全部快照，当cdsAttributeActive参数值为true时该值不生效，仅根据关联释放属性判断是否释放自动快照，手动快照默认保留。
deleteRelatedEnisFlag	boolean	否	RequestBody参数	实例释放时是否删除关联的ENI
bccRecycleFlag	boolean	否	RequestBody参数	实例释放时是否进入回收站
cdsAttributeActive	boolean	否	RequestBody参数	是否按照磁盘关联释放属性释放资源，默认为false，当该参数为true时将按照磁盘配置的关联释放属性确定资源是否释放，优先级高于relatedReleaseFlag和deleteCdsSnapshotFlag选项

返回头域

除公共头域外，无其它特殊头域。

返回参数

无特殊返回参数

错误码	错误描述	HTTP状态码	中文解释
Instance.ReleaseWhileCreating	Instance can not release during creating.	403	不可以释放创建中的实例
Instance.ReleaseNotPrepay	Instance can not release if not prepay.	403	不可以释放非包年包月的实例

请求示例

```
 POST /v2/instance/i-YufwpQAe  HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "relatedReleaseFlag": true,
    "deleteCdsSnapshotFlag":true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

实例变更子网

变更实例所属的子网。

- 开机状态下变更子网后会重启，关机状态下变更子网不重启。
- 仅支持同AZ变更子网，可以指定变更的普通安全组或企业安全组。

请求结构

```
 PUT /v{version}/subnet/changeSubnet HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
      "instanceId": instanceId,
      "subnetId": subnetId,
      "internalIp": internalIp,
      "securityGroupIds": [securityGroupId],
      "enterpriseSecurityGroupIds": [enterpriseSecurityGroupId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	RequestBody参数	虚拟机实例ID
subnetId	String	是	RequestBody参数	变更后的子网ID
internalIp	String	否	RequestBody参数	指定内网IP，不指定则随机生成
securityGroupIds	List	否	RequestBody参数	变更普通安全组，跨VPC变更子网时生效，不支持和企业安全组同时修改
enterpriseSecurityGroupIds	List	否	RequestBody参数	变更企业安全组，跨VPC变更子网时生效，不支持和安全组同时修改

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/subnet/changeSubnet HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId": "i-3823SVMR",
    "subnetId": "sbn-c82119xnk59i",
    "internalIp": "192.168.1.18",
    "securityGroupIds": ["g-dt6xvzpcp93m"],
    "enterpriseSecurityGroupIds": ["esg-kbnx8vg85ity"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

修改实例描述

该接口用于修改指定BCC实例的描述信息；

只有 Running 和 Stopped 状态的实例才可以调用该接口，否则提示 409 错误。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "desc": desc
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	指定的实例ID
desc	String	是	Request Body参数	实例描述信息

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
 PUT /v2/instance/i-YufwpQAe?modifyDesc HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "desc": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

计费变更-转包年包月

该接口用于BCC实例计费方式由按量付费（后付费）转为包年包月（预付费）。

请求结构

```
 POST /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "duration": duration
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：toPrepay
duration	int	是	Request Body参数	购买时长（单位：月）
relationCds	boolean	否	Request Body参数	变更关联的数据盘，已废弃，关联的按量付费（后付费）CDS需同BCC一起变更为包年包月（预付费）。
autoRenew	boolean	否	Request Body参数	实例到期后是否自动续费，取值：**true**：自动续费，**false**：不自动续费，默认值：false。
autoRenewPeriod	int	否	Request Body参数	每次自动续费的时长（单位：月）。取值范围： 1，2，3，4，5，6，7，8，9，12，24，36，默认值：1。仅当autoRenew取值为true，该参数有效。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	虚机实例ID的集合，其中ID符合BCE规范，必须是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。

请求示例

```
 POST /v2/instance/i-dnvlAmdc?toPrepay HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "duration": 2
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
 "orderId": "5c001b75e24d45639c7bb83532bcbf52"
}
```

实例解绑标签

该接口用于解绑实例上关联的标签。

- 可以同时解绑多个标签。

请求结构

```
 PUT /v{version}/instance/{instanceId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "changeTags": [
     {
      "tagKey" : tagKey,
      "tagValue": tagValue
     }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	RequestBody参数	虚拟机实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：unbind
attachRelatedResourceTag	boolean	否	RequestBody参数	关联资源（cds系统盘、数据盘、eip、快照、快照链）是否一同解绑标签。true：关联解绑；false：不关联解绑。不传默认false
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/instance/i-yect8WnR/tag?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "changeTags": [
     {
      "tagKey" : "company",
      "tagValue": "bd"
     }
   ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

实例绑定标签

该接口用于给实例绑定标签，用于分类和查找。

请求结构

```
 PUT /v{version}/instance/{instanceId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	RequestBody参数	虚拟机实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：bind
attachRelatedResourceTag	boolean	否	RequestBody参数	关联资源（cds系统盘、数据盘、eip、快照、快照链）是否一同绑定标签。true：关联绑定；false：不关联绑定。不传默认false
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/instance/i-yect8WnR/tag?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "changeTags": [
      {
       "tagKey" : "company",
       "tagValue": "bd"
      }
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

获取实例角色列表

该接口用于获取BCC实例角色列表。

- 角色的定义和功能详见https://cloud.baidu.com/doc/IAM/s/Gjxlikxcf

请求结构

```
 GET /v{version}/instance/role/list HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

无

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
roles	List<InstanceRoleModel>	实例角色列表

错误码

错误码	错误描述	HTTP状态码	中文解释
ServiceInternalError	Internal service occurs error.	500	内部错误
AccessDenied	No permission to accessor or permission error.	403	禁止操作

请求示例

```
GET http://bcc.bj.baidubce.com/v2/instance/role/list
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "roles":[{
        "roleName":"roleName"
    }]
}
```

绑定角色

该接口用于BCC实例绑定角色。

- 角色的定义和功能详见https://cloud.baidu.com/doc/IAM/s/Gjxlikxcf

请求结构

```
POST /v{version}/instance/role?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "roleName": roleName,
    "instances": [{
        "instanceId": instanceId
    }]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的操作，当前取值：bind
roleName	String	是	requestBody参数	实例绑定的角色名称
instances	List<InstancePassRoleModel>	是	requestBody参数	要绑定角色的实例id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
failInstances	List<InstancePassRoleFailModel>	实例绑定角色失败列表
instanceRoleAssociations	List<InstanceRoleAssociationModel>	实例绑定角色成功列表

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
BadRequest	instances is null or empty	400	绑定角色实例列表为空
BadRequest	roleName is null or empty	400	绑定角色名称为空
BadRequest	invalid roleName	400	角色名称无效
AccessDenied	No permission to accessor or permission error.	403	禁止操作

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/role?bind
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "roleName": "roleName",
    "instances": [{
        "instanceId": "i-123456"
    }]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "failInstances": [{
        "instanceId": "i-123456",
        "failMessage": "failMessage"
     }],
    "instanceRoleAssociations": [{
        "instanceId": "i-123456"
    }]
}
```

解绑角色

该接口用于BCC实例解绑角色。

- 角色的定义和功能详见https://cloud.baidu.com/doc/IAM/s/Gjxlikxcf

请求结构

```
 POST /v{version}/instance/role?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "roleName": roleName,
    "instances": [{
        "instanceId": instanceId
    }]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的操作，当前取值：unbind
roleName	String	是	requestBody参数	实例解绑的角色名称
instances	List<InstancePassRoleModel>	是	requestBody参数	要解绑角色的实例id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
failInstances	List<InstancePassRoleFailModel>	实例解绑角色失败列表
instanceRoleAssociations	List<InstanceRoleAssociationModel>	实例解绑角色成功列表

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
BadRequest	instances is null or empty	400	绑定角色实例列表为空
BadRequest	roleName is null or empty	400	绑定角色名称为空
BadRequest	invalid roleName	400	角色名称无效
AccessDenied	No permission to accessor or permission error.	403	禁止操作

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/role?unbind
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "roleName": "roleName",
    "instances": [{
        "instanceId": "i-123456"
    }]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "failInstances": [{
        "instanceId": "i-123456",
        "failMessage": "failMessage"
     }],
    "instanceRoleAssociations": [{
        "instanceId": "i-123456"
    }]
}
```

修改实例主机名

该接口用于修改指定BCC实例的主机名。

请求结构

```
 PUT /v{version}/instance/{instanceId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "hostname": hostname,
    "isOpenHostnameDomain": isOpenHostnameDomain,
    "reboot": reboot ,
    "isAllowHostnameRepeat": isAllowHostnameRepeat
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	指定的实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：changeHostname
hostname	String	是	RequestBody参数	实例主机名
isOpenHostnameDomain	Boolean	否	RequestBody参数	是否开启hostname domain true:是 false:否，默认false
reboot	Boolean	否	RequestBody参数	是否自动重启，默认false
isAllowHostnameRepeat	Boolean	否	RequestBody参数	是否允许hostname重复 true:是 false:否，默认false

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Illegal name format.	400	主机名无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
BadRequest	Hostname duplicate in same vpc.	400	同VPC下主机名重复

请求示例

```
 PUT /v2/instance/i-YufwpQAe?changeHostname HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "hostname": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

根据实例id查询实例列表

该接口用于根据实例ID查询实例列表。

请求结构

```
 POST /v{version}/instance/listByInstanceId?marker={marker}&maxKeys={maxKeys} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
   "instanceIds": [
      "i-xktdeMSf", "i-lyWRtII7"
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000。
instanceIds	List	是	RequestBody参数	待查询的实例id列表，最多支持100个。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
instances	List<InstanceModel>	实例信息，由 InstanceModel 组成的集合

请求示例

```
 POST /v2/instance/listByInstanceId?marker=i-IyWRtII7&maxKeys=1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds": [
        "i-xktdeMSf", "i-IyWRtII7"
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "i-xktdeMSf",
    "marker": "i-IyWRtII7",
    "maxKeys": 1,
    "isTruncated": true,
    "instances": [
        {
            "id": "i-IyWRtII7",
            "createTime": "2015-08-06T13:23:13Z",
            "name": "instance-j93wzbn1",
            "instanceType": "N1",
            "status": "Running",
            "desc": "console",
            "paymentTiming":"Postpaid",
            "expireTime": null,
            "internalIp": "192.168.6.15",
            "publicIp": "-",
            "cpuCount": 1,
            "memoryCapacityInGB": 1,
            "localDiskSizeInGB": 0,
            "networkCapacityInMbps": 1,
            "imageId": "m-3zfBY1Ku",
            "placementPolicy": "default",
    "zoneName": "cn-bj-a",
            "autoRenew":"true"
        }
    ]
}
```

释放ipv6

该接口用于BCC实例释放ipv6辅助内网IP。

请求结构

```
 POST /v{version}/instance/delIpv6 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId": instanceId,
    "ipv6Address": ipv6Address,
    "reboot": reboot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	requestBody参数	实例id
ipv6Address	String	否	requestBody参数	需要释放的ipv6地址
reboot	boolean	否	requestBody参数	是否重启，true表示重启，为空表示不重启。默认不重启false。

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码 ####

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	实例找不到或者不存在
BadRequest	instance doesn't have ipv6 address	400	实例无ipv6
Vpc.DelIpv6Exception	Vpc delete ipv6 exception.	503	vpc释放ipv6异常

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/delIpv6
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "instanceId": "i-123456",
  "ipv6Address": "2400:da00:e003:2712::200",
  "reboot": true
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

添加ipv6

该接口用于BCC实例添加ipv6辅助内网IP。

请求结构

```
 POST /v{version}/instance/addIpv6 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId": instanceId,
    "ipv6Address": ipv6Address,
    "reboot": reboot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	requestBody参数	实例id
ipv6Address	String	否	requestBody参数	ipv6地址，为空表示自动分配ipv6
reboot	boolean	否	requestBody参数	是否重启，true表示重启，为空表示不重启

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
ipv6Address	String	主网卡全量的ipv6

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	实例找不到或者不存在
BadRequest	instance has ipv6 already, please release ipv6 first	400	实例已经有ipv6，需要先释放ipv6
BadRequest	ipv6Address is invalid	400	无效的ipv6地址
Instance.GetHostEnableIpv6Exception	get host enable ipv6 exception.	400	获取虚机内核是否支持ipv6异常
Instance.VMKernelVersionTooLowException	vm kernel version is too low to support ipv6, please purchase the vm which support ipv6.	400	虚机版本过低，建议您重新购买支持IPv6的虚机
Vpc.AddIpv6Exception	Vpc add ipv6 exception.	503	vpc添加ipv6异常

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/addIpv6
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "instanceId": "i-123456",
  "ipv6Address": "2400:da00:e003:2712::285",
  "reboot": true
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "ipv6Address": "2400:da00:e003:2712::285"
}
```

查询多个实例详情

根据多个实例ID来查询多个实例详情。

请求结构

```
 POST /v{version}/instance/listByInstanceId HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "instanceIds": ["instanceId",""]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceIds	Array	是	RequestBody参数	实例id数组
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
instances	List<InstanceModel>	实例信息，由 InstanceModel 组成的集合

请求示例

```
 POST /v2/instance/listByInstanceId
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds":["i-5poIAZSW","i-q490RD6t"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
 "marker": "",
 "isTruncated": false,
 "maxKeys": 1000,
 "instances": [{
  "id": "i-cZbnhO0B",
  "serialNumber": "1060b348-b9e4-4940-baa6-fd9f300fc566",
  "hostId": "h-gHBSWl6i2Cq5rlynq2ngw",
  "rackId": "rk-10iVdnFTv5xYMmArVurhw",
  "switchId": "sw-Jew0Mc0xnljBz26wFLT4RA",
  "deploysetId": "dset-qrE9FWAR",
  "deploysetList": [{
   "deploysetId": "dset-qrE9FWAR",
   "name": "name",
   "concurrency": 1,
   "strategy": "HOST_HA"
  }],
  "name": "instance-r6337ue1-1",
  "hostname": "instance-r6337ue1-1",
  "instanceType": "E1",
  "status": "Running",
  "desc": "",
  "createdFrom": "customPurchase",
  "paymentTiming": "Postpaid",
  "createTime": "2024-06-26T11:42:01Z",
  "internalIp": "192.168.64.65",
  "publicIp": "",
  "cpuCount": 2,
  "isomerismCard": "",
  "cardCount": "0",
  "npuVideoMemory": "",
  "memoryCapacityInGB": 2,
  "localDiskSizeInGB": 0,
  "imageId": "m-H2E8PurP",
```

```
  "placementPolicy": "unknown",
  "subnetId": "sbn-yfxhv6b0sxy4",
  "vpcId": "vpc-nys8a7qhzdrs",
  "zoneName": "cn-bj-d",
  "repairStatus": "normal",
  "dedicatedHostId": "",
  "ipv6": "240c:4081:8005:ca01::14",
  "eniQuota": 2,
  "eriQuota": 0,
  "serviceComponents": {
   "hasAlive": "notSupport"
  },
  "nicInfo": {
   "eniId": "eni-kgm8hqvsp6a3",
   "eniUuid": "96f54c86-15f6-4d2b-b63b-1a94c6dfd194",
   "name": "eth0",
   "type": "primary",
   "subnetId": "058855bc-fbcb-4003-b5c4-18f19d47e68b",
   "subnetType": "BCC",
   "az": "zoneD",
   "description": "",
   "deviceId": "1060b348-b9e4-4940-baa6-fd9f300fc566",
   "status": "inuse",
   "macAddress": "fa:20:20:33:e8:8f",
   "vpcId": "3c63adcf-2b76-4a3b-82cf-04b8e36051df",
   "createdTime": "Wed Jun 26 19:42:02 UTC 2024",
   "eniNum": 0,
   "ips": [{
    "privateIp": "192.168.64.65",
    "eip": "null",
    "primary": "true",
    "eipId": "",
    "eipAllocationId": "",
    "eipSize": "0",
    "eipStatus": "",
    "eipGroupId": "",
    "eipType": "null"
   }],
   "ipv6s": [{
    "privateIp": "240c:4081:8005:ca01::14",
    "eip": "null",
    "primary": "false",
    "eipId": "",
    "eipAllocationId": "",
    "eipSize": "0",
    "eipStatus": "",
    "eipGroupId": "",
    "eipType": "null"
   }],
   "securityGroups": [
    "g-37bb45gkyuat"
   ]
  },
  "eniNum": "0",
  "tags": [{
   "tagKey": "默认项目",
   "tagValue": ""
  }],
  "networkCapacityInMbps": 0
 }]
}
```

删除实例与部署集之间的关系

该接口用于删除实例和部署集之间的关系。

请求结构

```
 POST /v{version}/instance/deployset/delRelation HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "deployId": "deployId",
  "instanceIdList": ["instanceId",""]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
deployId	String	是	RequestBody参数	部署集id
instanceIdList	String	是	RequestBody参数	实例id数组

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码	错误描述	HTTP状态码	中文解释
BadRequest	invalid deployset id	400	无效的部署集id

请求示例

```
 POST /v2/instance/deployset/delRelation
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "deployId":"dset-tTHtVKX1",
    "instanceIdList":["i-q490RD6t"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

设置实例删除保护

本接口用于设置实例释放保护。

- 已开启实例释放保护的实例无法手动释放。
- 开启释放保护不影响实例的定时释放，欠费释放以及实例过期释放。

请求结构

```
 PUT /v{version}/instance/{instanceId}/deletionProtection HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
 "deletionProtection": deletionProtection
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待设置删除保护的实例ID
deletionProtection	int	是	RequestBody参数	删除保护策略，默认0，不设置

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码	错误描述	HTTP状态码	中文解释
BadRequest	invalid DeletionProtection value	400	无效的删除保护值
NoSuchObject	The specified object is not found or resource do not exist.	404	指定的资源不存在

请求示例

```
 PUT /v2/instance/i-5polAZSW/deletionProtection HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "deletionProtection": 1
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

批量转后付费

批量转包年包月

该接口用于批量将BCC实例计费方式由按量付费（后付费）转为包年包月（预付费）。

请求结构

```
 POST /v{version}/instance/batch/charging?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
  "config": [
    {
      "instanceId": instanceId1,
      "duration": duration,
      "autoPay": autoPay,
      "cdsList": [volumeId1, volumeId2]
    },
    {
      "instanceId": instanceId2,
      "duration": duration,
      "autoPay": autoPay,
      "cdsList": [volumeId3, volumeId4]
    }
  ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的操作，当前取值：toPrepay
config	List<PrepayConfig>	是	RequestBody参数	需要批量转包年包月实例的配置，最多支持20个

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	订单ID。

请求示例

```
 POST /v2/instance/batch/charging?toPrepay HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048f***

{
  "config": [
    {
      "instanceId": "i-43TqYnng",
      "duration": 1,
      "autoPay": true,
      ""cdsList": ["v-tP2MJ3YI","v-9eLRwv8n"]
    },
    {
      "instanceId": "i-4D3Jinng",
      "duration": 1,
      "autoPay": true,
      ""cdsList": ["all"]
    }
  ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: a41fdefe-28a6-4cb5-87b5-c88caf412a1c
Date: Thu, 23 Mar 2023 06:30:58 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "orderId": "5c001b75e24d***"
}
```

查询可变配规格

该接口用于获取BCC实例可变配规格。

请求结构

```
  POST /v{version}/instance?{action} HTTP/1.1
  Host: bcc.bj.baidubce.com
  Authorization: authorization string

  {
     "spec": spec,
     "zone": zone
  }
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的操作，当前取值：resizeList
spec	String	否	RequestBody参数	实例规格。实例规格、规格族ID和实例ID有且只能传一个。
specId	String	否	RequestBody参数	规格族ID。实例规格、规格族ID和实例ID有且只能传一个。
zone	String	否	RequestBody参数	指定zone信息，命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。若传入spec或specId，则该参数为必填。
instanceIdList	List	否	RequestBody参数	实例ID列表。实例规格、规格族ID和实例ID有且只能传一个。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
specList	List	可变配规格列表

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Spec, specid and instanceid can only be queried through one type.	400	实例规格、规格族ID和实例ID有且只能传一个。
BadRequest	At least one spec, specid and instanceIdList is required.	400	实例规格、规格族ID和实例ID有且只能传一个。

请求示例

```
 POST /v2/instance?resizeList HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization:  bce-auth-v1/78532e9951254e85b1f79***

{
    "spec": "bcc.g5.c2m8",
    "zone": "cn-bj-a"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: 23 Mar 2023 06:30:58 GMT
x-bce-request-id: 5f9a3a71-bed4-45a5-97e7-d4cffbd468c9
Server: BWS

{
    "specList": [
        "bcc.c4.c12m24",
        "bcc.c4.c16m32",
        "bcc.c4.c24m48",
        "bcc.c4.c2m4"
    ]
}
```

批量释放实例

该接口用于批量释放按量付费（后付费）BCC实例。

- 释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。
- 释放的时候默认只释放实例和系统盘，用户可以选择是否关联释放当前实例挂载的EIP和数据盘。

请求结构

```
 POST /v{version}/instance/batchDelete HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "relatedReleaseFlag" :  relatedReleaseFlag,
    "deleteCdsSnapshotFlag" : deleteCdsSnapshotFlag,
    "bccRecycleFlag" : bccRecycleFlag,
    "deleteRelatedEnisFlag" : deleteRelatedEnisFlag,
    "instanceIds" : [
        instanceId
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceIds	List	是	RequestBody参数	实例id列表，列表元素数量不超过100个
relatedReleaseFlag	boolean	是	RequestBody参数	是否关联释放当前时刻，实例挂载的EIP和数据盘（只有该字段为true时deleteCdsSnapshotFlag字段才会有效，若该字段为false，deleteCdsSnapshotFlag字段的值无效）
deleteCdsSnapshotFlag	boolean	是	RequestBody参数	是否释放云磁盘快照
bccRecycleFlag	boolean	否	RequestBody参数	实例释放时是否进入回收站
deleteRelatedEnisFlag	boolean	否	RequestBody参数	实例释放时是否删除关联的ENI

返回头域

除公共头域外，无其它特殊头域。

返回参数

无特殊返回参数

错误码	错误描述	HTTP状态码	中文解释
Instance.DeleteServerFailException	Instance can not release if not postpay.	403	只能释放后付费的实例

请求示例

```
  POST /v2/instance/batchDelete HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048f**

{
  "relatedReleaseFlag": true,
  "deleteCdsSnapshotFlag": true,
  "bccRecycleFlag": true,
  "deleteRelatedEnisFlag": true,
  "instanceIds": [
    "i-YufwpQAe"
  ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1e9f9fc4-25d5-450b-9720-29fe66ddf840
Date: Mar, 22 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

批量变配实例

该接口用于批量变配实例。

请求结构

```
 PUT /v{version}/instanceBatchBySpec?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "spec": spec,
  "instanceIdList": [
    "i-7pxLG046"
  ],
  "enableJumboFrame":enableJumboFrame
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对实例执行的动作，本接口中该参数取值：resize
spec	String	是	RequestBody参数	批量变配的实例规格
instanceIdList	List	是	RequestBody参数	实例id组成的集合，最多30个
enableJumboFrame	Boolean	否	RequestBody参数	实例是否开启Jumbo帧，true:开启Jumbo帧，false:关闭Jumbo帧。默认为空，为空时：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderUuidResults	List	变配订单ID列表。

错误码

若批量操作的实例里有不支持变更至目标规格，会过滤不操作，支持变更的实例正常下发任务

错误码	错误描述	HTTP状态码	中文解释
ResizeServerError	Spec not allowed or out of stock.	400	不允许变配到这个规格
ResizeServerError	The number of resize exceeds 30.	400	一次变配的实例数量不能超过30个

请求示例

```
 PUT /v2/instanceBatchBySpec?resize HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fb****

{
  "spec": "bcc.g5.c2m8",
  "instanceIdList": [
    "i-7pxLG046"
  ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 019d4594-6133-4537-8dfd-f6fcdfc42dce
Date: Mar, 23 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "orderUuidResults": ["instanceId1"]
}
```

释放回收站实例

该接口用于释放回收站实例，清理回收站资源，相关数据全部丢失且不可恢复。

请求结构

```
 DELETE /v{version}/recycle/instance/{instanceId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
...
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待释放的虚拟机实例ID

返回头域

除公共头域外，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 DELETE /v2/recycle/instance/i-YufwpQAe HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2****
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Mar, 23 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

创建自动续费规则

该接口用于创建自动续费规则。

- 若您的资源即将到期（到期时间小于24小时），建议您先进行手动续费。
- 开通自动续费后，系统会在到期日前7天自动为其续费。
- 自动续费将按照您最近一次设置的续费周期执行。
- 实例关联的预付费CDS同步创建自动续费规则。

请求结构

```
 POST /v{version}/instance/batchCreateAutoRenewRules HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
  "renewTimeUnit":renewTimeUnit,
  "renewTime": renewTime,
  "renewEip": renewEip,
  "instanceId": instanceId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	RequestBody参数	实例Id
renewTimeUnit	String	是	RequestBody参数	续费时间，单位：month，支持1, 2, 3, 4, 5, 6, 7, 8, 9；单位：year，支持1, 2, 3
renewTime	int	是	RequestBody参数	续费时长
renewEip	boolean	否	RequestBody参数	是否合并eip自动续费，默认值为true。

返回头域

除公共头域，无其它特殊头域。

请求示例

```
 POST /v2/instance/batchCreateAutoRenewRules HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048f***

{
  "renewTimeUnit": "month",
  "renewTime": 1,
  "renewEip": false,
  "instanceId": "i-RrgyXFh4"
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 956690b1-f4b0-4452-9e50-8e75cae04333
Date: Tue, 28 Mar 2023 08:06:35 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除自动续费规则

该接口用于删除自动续费规则。

- 实例关联的预付费CDS磁盘会同时删除自动续费规则。

请求结构

```
  POST /v{version}/instance/batchDeleteAutoRenewRules HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
  "renewEip": renewEip,
  "instanceId": instanceId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	RequestBody参数	实例ID
renewEip	boolean	否	RequestBody参数	是否合并关闭eip自动续费，默认为true。

返回头域

除公共头域，无其它特殊头域。

请求示例

```
 POST /v2/instance/batchDeleteAutoRenewRules HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048f***

{
  "renewEip": false,
  "instanceId": "i-RrgyXFh4"
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: bd18c3f7-f643-4a17-b603-bd1a516d2c14
Date: Tue, 28 Mar 2023 08:06:35 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

后付费设置自动释放

该接口用于按量付费（后付费）实例设置自动释放。 **请求结构**

```
 PUT /v{version}/instance/{instanceId}?{action}&clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "releaseTime": "releaseTime"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
instanceId	String	是	URL参数	实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：autorelease
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
releaseTime	String	是	RequestBody参数	释放时间，格式yyyy-MM-dd'T'HH:mm:ss'Z'

响应头域

除公共头域外，无其它特殊头域。

响应参数 无

请求示例

```
 PUT http://bcc.bj.baidubce.com/v2/instance/i-j7vsd5e?autorelease&clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
  "releaseTime": "2023-03-27T00:00:00Z"
}
```

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

获取实例网卡信息

该接口用于获取实例网卡信息，包括主网卡和弹性网卡。

请求结构

```
GET /v{version}/eni/{instanceId}?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceId	String	是	Query参数	网卡挂载的虚机ID

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
enis	List<EniInfo>	网卡信息列表，具体参数参见 EniInfo

请求示例

```
GET http://bcc.bj.baidubce.com/v2/eni/i-ofsX9WtN?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
    "enis": [
        {
            "eniId": "eni-531s8c7f45rs",
            "name": "test",
            "vpcId": "vpc-vv3xw6d9970b",
            "subnetId": "sbn-2escuever5fi",
            "zoneName": "cn-bj-a",
            "description": "",
            "createdTime": "2022-10-20T06:31:42Z",
            "macAddress": "fa:f6:00:14:05:1e",
            "status": "inuse",
            "securityGroupIds": [
                "g-6uwn49bxhbvj"
            ],
            "privateIpSet": [
                {
                    "primary": true,
                    "privateIpAddress": "192.168.48.8"
                }
            ]
        },
        {
            "eniId": "eni-wwi3mdqp6f5j",
            "name": "eth0",
            "vpcId": "vpc-vv3xw6d9970b",
            "subnetId": "sbn-2escuever5fi",
            "zoneName": "cn-bj-a",
            "description": "",
            "createdTime": "2023-03-21T02:09:03Z",
            "macAddress": "fa:f6:00:01:83:03",
            "status": "inuse",
            "securityGroupIds": [
                "g-6uwn49bxhbvj"
            ],
            "privateIpSet": [
                {
                    "primary": true,
                    "privateIpAddress": "192.168.48.5"
                },
                {
                    "primary": false,
                    "privateIpAddress": "192.168.48.3"
                }
            ]
        }
    ]
}
```

实例变更VPC

该接口用于实例变更VPC。

- EBC实例在关机状态下不支持变更VPC。

请求结构

```
 PUT /v{version}/vpc/changeVpc?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "instanceId": instanceId,
  "subnetId": subnetId,
  "internalIp": internalIp,
  "securityGroupIds": [securityGroupId],
  "reboot": reboot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceId	String	是	RequestBody参数	虚机ID
subnetId	String	是	RequestBody参数	目标VPC的子网ID
internalIp	String	否	RequestBody参数	内网IP
securityGroupIds	List	否	RequestBody参数	安全组列表
enterpriseSecurityGroupIds	List	否	RequestBody参数	企业安全组列表，不能同时指定普通安全组和企业安全组

响应头域

除公共头域外，无其它特殊头域。

响应参数

无参数

请求示例

```
 PUT http://bcc.bj.baidubce.com/v2/vpc/changeVpc?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
  "instanceId": "i-sj7rj9Gu",
  "subnetId": "a0k74h216-b6f7-47fe-a4d6-4f3wjg7543f",
  "securityGroupIds": ["7ef30e08-4cea-550a-9d61-3201496b3dc7"]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

回收站内BCC恢复计费

该接口用于回收站内按量付费（后付费）BCC实例移出回收站，恢复计费。

请求结构

```
 POST /v{version}/instance/recovery?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization
{
  "instanceIds": [
      {
         "instanceId": "instanceId"
      },
      {
         "instanceId": "instanceId"
      }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceIds	List	是	RequestBody参数	实例ID列表，最多支持100个。

响应头域

除公共头域外，无其它特殊头域。

响应参数

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/recovery?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
  instanceIds: [
    {
       "instanceId": "i-v7f9jfdu"
    },
    {
       "instanceId": "i-9tv4qak"
    }
  ]
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询回收站实例列表

该接口用于查询处于回收站中的BCC实例列表。

请求结构

```
POST /v{version}/recycle/instance?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "marker": "marker",
  "maxKeys" : maxKeys,
  "instanceId" : "instanceId",
  "name": "name",
  "paymentTiming": "paymentTiming",
  "recycleBegin": "recycleBegin",
  "recycleEnd": "recycleEnd"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
marker	String	否	RequestBody参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	RequestBody参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000。
instanceId	String	否	RequestBody参数	虚机Id
name	String	否	RequestBody参数	虚机名称
paymentTiming	String	否	RequestBody参数	支付方式，包括包年包月（Prepaid）和按量付费（Postpaid）
recycleBegin	String	否	RequestBody参数	查询进入回收站时间大于等于该起始时间，格式yyyy-MM-ddTHH:mm:ssZ
recycleEnd	String	否	RequestBody参数	查询进入回收站时间小于等于该终止时间，格式yyyy-MM-ddTHH:mm:ssZ

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
instances	List<ResourceBo>	虚机列表信息，详见ResourceBo

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/recycle/instance?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
  "recycleBegin":"2023-03-20T00:00:00Z",
  "recycleEnd":"2023-03-22T00:00:00Z"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
    "marker": "",
    "isTruncated": false,
    "maxKeys": 1000,
    "instances": [
        {
            "id": "i-4sWEnqCC",
            "serialNumber": "a16ae78f-2d98-4037-b71f-b36a99e313ad",
            "name": "instance-g2ug3wrz",
            "recycleTime": "2023-03-20T19:57:59Z",
            "deleteTime": "2023-03-27T19:57:59Z",
            "paymentTiming": "Prepaid",
            "serviceName": "云服务器",
            "serviceType": "BCC",
            "configItem": {
                "cpu": 1,
                "memory": 1,
                "type": "N6",
                "specId": "ic5",
                "spec": "bcc.ic5.c1m1",
                "zoneName": "cn-bj-a"
            },
            "configItems": [
                "cpu:1",
                "memory:1",
                "type:N6",
                "specId:ic5",
                "spec: bcc.ic5.c1m1",
                "zoneName: cn-bj-a"
            ]
        },
        {
            "id": "i-Cygj2uN7",
```

```
            "serialNumber": "b3d58c5b-c0f6-44d4-bca7-afb7b78b2fb4",
            "name": "instance-3yce4ts0",
            "recycleTime": "2023-03-20T19:57:49Z",
            "deleteTime": "2023-03-27T19:57:50Z",
            "paymentTiming": "Prepaid",
            "serviceName": "云服务器",
            "serviceType": "BCC",
            "configItem": {
                "cpu": 1,
                "memory": 1,
                "type": "N6",
                "specId": "ic5",
                "spec": "bcc.ic5.c1m1",
                "zoneName": "cn-bj-a"
            },
            "configItems": [
                "cpu:1",
                "memory:1",
                "type:N6",
                "specId:ic5",
                "spec: bcc.ic5.c1m1",
                "zoneName: cn-bj-a"
            ]
        }
    ]
}
```

预付费实例提前释放

该接口用于释放单个包年包月（预付费）云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。

预付费释放退款规则详情请见：退款说明

请求结构

```
 POST /v{version}/instance/delete HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId": instanceId,
    "relatedReleaseFlag": relatedReleaseFlag,
    "deleteCdsSnapshotFlag":deleteCdsSnapshotFlag,
    "deleteRelatedEnisFlag" : deleteRelatedEnisFlag
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	实例id
relatedReleaseFlag	boolean	否	RequestBody参数	是否关联释放当前时刻，实例挂载的EIP和数据盘。默认值：false，不关联释放。**注意：**实例关联的预付费CDS盘会强制随实例一起释放。
deleteCdsSnapshotFlag	boolean	否	RequestBody参数	是否释放云磁盘快照。默认值：false，不关联释放
deleteRelatedEnisFlag	boolean	否	RequestBody参数	实例释放时是否删除关联的ENI。默认值：false，不关联释放

返回头域

除公共头域外，无其它特殊头域。

返回参数

参数名称	类型	描述
successResources	InstanceDeleteResultModel	删除成功的资源集合
failResources	InstanceDeleteResultModel	删除失败的资源集合
instanceRefundFlag	boolean	实例是否成功释放

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.DeletionProtection	Instance can not release with DeletionProtection.	403	不允许释放删除保护中的虚机
InsufficientCreditException	Insufficient user refund quota	400	用户退费的配额不足

请求示例

```
 POST /v2/instance/delete  HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId": "instanceId",
    "relatedReleaseFlag": true,
    "deleteCdsSnapshotFlag":true,
    "deleteRelatedEnisFlag" : true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "successResources": {
     "instanceId": "instanceId",
     "eip": "eipId",
     "insnapIds": ["insnapId1", "insnapId2"],
     "snapshotIds": ["snapshotId1", "snapshotId2"],
     "volumeIds": ["volumeId1", "volumeId2"]
    },
    "failResources": {
     "instanceId": "",
     "eip": "",
     "insnapIds": ["insnapId3"],
     "snapshotIds": ["snapshotId3"],
     "volumeIds": ["volumeId3"]
    },
    "instanceRefundFlag": true
 }
```

查询可关机不计费的实例列表

该接口用于查询所有符合关机不计费条件的实例的详细信息，以下6种实例不支持关机不计费：

1. 预付费计费实例
2. 带本地盘的实例（本地SSD型 & 大数据型）
3. 异构实例：（GPU云服务器（部分）&FPGA云服务器）
4. DCC专属实例
5. 抢占实例
6. 非运行状态的实例
7. ica2/ca2/ga2/ma2规格族实例
8. EBC裸金属实例

请求结构

```
 GET /v{version}/instance/noCharge?marker={marker}&maxKeys={maxKeys}&internalIp={internalIp}&keypairId=
{keypairId}&zoneName={zoneName} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
internalIp	String	否	Query参数	内网ip
keypairId	String	否	Query参数	密钥对id
zoneName	String	否	Query参数	可用区名称，如：cn-bj-a

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	Boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
instances	List<InstanceModel>	实例信息，由 InstanceModel 组成的集合

请求示例

```
 GET /v2/instance/noCharge?marker=i-ED2BiZR7&maxKeys=1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "marker": "i-ED2BiZR7",
    "isTruncated": true,
    "nextMarker": "i-15INKlWp",
    "maxKeys": 1,
    "instances": [
        {
            "id": "i-ED2BiZR7",
            "name": "test-name",
            "hostname": "hostname",
            "instanceType": "N6",
            "status": "Running",
            "desc": "",
            "paymentTiming": "Postpaid",
            "createTime": "2023-11-16T08:37:09Z",
            "internalIp": "192.168.0.11",
            "publicIp": "",
            "cpuCount": 1,
            "isomerismCard": "",
            "cardCount": "0",
            "memoryCapacityInGB": 4,
            "localDiskSizeInGB": 0
```

```
            "localDiskSizeInGB": 0,
            "imageId": "m-6TqSR4r0",
            "placementPolicy": "unknown",
            "subnetId": "sbn-1rg9fb0caqq3",
            "vpcId": "vpc-5mbbq0mx2a1n",
            "zoneName": "cn-bj-a",
            "dedicatedHostId": "",
            "nicInfo": {
                "eniId": "eni-mhegdfrq4337",
                "eniUuid": "712b244b-6ae7-4cb9-9668-ffa62323f449",
                "name": "eth0",
                "type": "primary",
                "subnetId": "cfb112c1-9c7f-4be3-aafb-beb94a36ac31",
                "subnetType": "BCC",
                "az": "zoneA",
                "description": "",
                "deviceId": "8d67c5ae-e1e2-4771-8aae-d3d056dada0b",
                "status": "inuse",
                "macAddress": "fa:f6:00:11:f2:d9",
                "vpcId": "ca0327d6-ddb8-4bea-a58c-abf21947c1e7",
                "createdTime": "Thu Nov 16 16:37:10 UTC 2023",
                "eniNum": 0,
                "ips": [
                    {
                        "privateIp": "192.168.0.11",
                        "eip": "null",
                        "primary": "true",
                        "eipId": "",
                        "eipAllocationId": "",
                        "eipSize": "0",
                        "eipStatus": "",
                        "eipGroupId": "",
                        "eipType": "null"
                    }
                ],
                "securityGroups": [
                    "g-rrza83kb8wkw"
                ]
            },
            "eniNum": "0",
            "tags": [
                {
                    "tagKey": "默认项目",
                    "tagValue": ""
                }
            ],
            "networkCapacityInMbps": 0
        }
    ]
}
```


批量退订预付费实例

该接口用于批量退订包年包月（预付费）实例。

- 支持释放预付费云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。
- 单次操作最多支持20个实例。
- 不支持已经过期的预付费资源释放，包含实例及其关联的CDS、EIP等。
- 预付费释放退款规则详情请见：退款说明。

请求结构

```
 POST /v{version}/instance/batchRefundResource HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceIds": ["instanceId"],
    "relatedReleaseFlag": relatedReleaseFlag,
    "deleteCdsSnapshotFlag":deleteCdsSnapshotFlag,
    "deleteRelatedEnisFlag" : deleteRelatedEnisFlag
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceIds	List	是	RequestBody参数	实例id列表，单次操作最多支持20个实例
relatedReleaseFlag	boolean	否	RequestBody参数	是否关联释放当前时刻，实例挂载的EIP和数据盘。默认值：false，不关联释放。**注意**：实例关联的预付费CDS盘会强制随实例一起释放。
deleteCdsSnapshotFlag	boolean	否	RequestBody参数	是否释放云磁盘快照。默认值：false，不关联释放
deleteRelatedEnisFlag	boolean	否	RequestBody参数	实例释放时是否删除关联的ENI。默认值：false，不关联释放

返回头域

除公共头域外，无其它特殊头域。

返回参数

参数名称	类型	描述
failedInstanceIds	List	删除失败的实例列表

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.DeletionProtection	Instance can not release with DeletionProtection.	403	不允许释放删除保护中的虚机
InsufficientCreditException	Insufficient user refund quota	400	用户退费的配额不足
BadRequest	The size of the prepay instances is limited to 20.	400	单次操作最多支持20个实例
BadRequest	Instanceid status not support delete	400	实例状态不对
Instance.ReleaseNotPrepay	This interface not support delete postpay instance	403	后付费虚机，不允许释放

请求示例

```
 POST /v2/instance/batchRefundResource  HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b***
{
    "instanceIds": ["instanceId"],
    "relatedReleaseFlag": true,
    "deleteCdsSnapshotFlag":true,
    "deleteRelatedEnisFlag" : true
}
```

返回示例

```
  HTTP/1.1 200 OK
  x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
  Date: Wed, 03 Dec 2024 06:42:19 GMT
  Content-Type: application/json;charset=UTF-8
  Server: BWS
  {
    "failedInstanceIds": []
  }
```

设置抢占实例eip自动释放

该接口用于设置EIP随抢占实例自动释放，该设置对BCC实例的主网卡和按量付费（后付费）EIP生效，即

- 实例主网卡主IP绑定的按量付费（后付费）EIP释放，不进入回收站；包年包月（预付费）EIP解绑；
- 若实例主网卡绑有辅助IP，辅助IP绑有EIP，后付费EIP释放，不进入回收站，包年包月（预付费）EIP解绑；
- 若实例绑有弹性网卡，弹性网卡卸载，弹性网卡绑定的EIP与弹性网卡不解除绑定关系。

请求结构

```
 PUT /v{version}/instance/{instanceId}/modifyRelatedDeletePolicy HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
 "isEipAutoRelatedDelete": true
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待设置的实例ID
isEipAutoRelatedDelete	Boolean	是	RequestBody参数	实例绑定的EIP是否随抢占实例关联自动释放。是：true，否：false

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码	错误描述	HTTP状态码	中文解释
OperationDenied	only bidding instance could modify the policy	403	只有抢占实例允许设置该参数
NoSuchObject	The specified object is not found or resource do not exist.	404	指定的资源不存在

请求示例

```
 PUT /v2/instance/i-5polAZSW/modifyRelatedDeletePolicy HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "isEipAutoRelatedDelete":true
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

批量增加主网卡辅助IP

接口描述

本接口用于批量增加主网卡辅助IP，需要重启虚机生效。 **请求结构**

```
 PUT /v{version}/instance/batchAddIp?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId": instanceId,
    "secondaryPrivateIpAddressCount": secondaryPrivateIpAddressCount,
    "privateIps":[
         privateIps
     ],
    "allocateMultiIpv6Addr":true
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceId	String	是	RequestBody参数	虚机ID
secondaryPrivateIpAddressCount	int	否	RequestBody参数	需要增加IPV6/IPV4的数量，与privateIps必须存在一个
privateIps	List	否	RequestBody参数	需要增加的IPV6/IPV4地址，与secondaryPrivateIpAddressCount必须存在一个
allocateMultiIpv6Addr	boolean	是	RequestBody参数	是否分配IPV6，创建IPV6必须是true

响应头域

除公共头域外，无其它特殊头域。

响应参数

无参数 **请求示例**

```
 PUT http://bcc.bj.baidubce.com/v2/instance/batchAddIp?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId": "i-DhjUXmSA",
    "privateIps": [
        "2400:da00:e003:0:219::2f",
        "2400:da00:e003:0:219::30"
    ],
    "allocateMultiIpv6Addr": true
}
或者
{
    "instanceId": "i-DhjUXmSA",
    "secondaryPrivateIpAddressCount": 2,
    "allocateMultiIpv6Addr": true
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
    "privateIps": [
        "2400:da00:e003:0:219::2f",
        "2400:da00:e003:0:219::30"
    ],
    "eniPrivateIps": null
}
```

批量删除主网卡辅助IP

该接口用于批量删除BCC实例主网卡辅助IP，需要重启BCC实例生效。 **请求结构**

```
 PUT /v{version}/instance/batchDelIp?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId": instanceId,
    "privateIps":[
         privateIps
     ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceId	String	是	RequestBody参数	虚机ID
privateIps	List	是	RequestBody参数	需要删除的IPV6/IPV4地址

响应头域

除公共头域外，无其它特殊头域。

响应参数

无参数 **请求示例**

```
 PUT http://bcc.bj.baidubce.com/v2/instance/batchDelIp?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId": "i-DhjUXmSA",
    "privateIps": [
        "2400:da00:e003:0:219::2f",
        "2400:da00:e003:0:219::30"
    ]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
    "success": true,
    "status": 200
}
```

实例变配（通过指定实例套餐规格）

该接口用于为指定BCC实例升降配，目前支持通过更新实例套餐规格对虚机的CPU以及内存进行升降配操作。

- 只有实例Running或Stopped状态时可以进行升降配操作

- 实例升降配会自动重启一次，请提前完成相关操作，避免影响业务。

请求结构

```
 PUT /v{version}/instanceBySpec/{instanceId}?{action}&clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "spec": spec,
    "enableJumboFrame": enableJumboFrame
}
```

请求头域 除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	实例ID
action	String	是	Query参数	对实例执行的操作，当前取值：resize
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
spec	String	是	Request Body参数	目标实例规格
enableJumboFrame	Boolean	否	Request Body参数	实例是否开启Jumbo帧，true:开启Jumbo帧，false:关闭Jumbo帧。默认为空，为空时：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。

返回头域 除公共头域，无其它特殊头域。

返回参数 无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
Instance.InstanceBeBanned	The instance is banned, contact technical support for assistance.	403	被封禁
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
PUT /v2/instanceBySpec/i-YufwpQAe?resize&clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fb***

{
    "spec": bcc.ic3.c2m2
}
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询实例自定义数据

获取实例用户数据信息

请求结构

```
POST /v{version}/instance/attribute/getUserdata
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId":instanceId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	URL参数	待启动的实例ID

返回参数

参数名称	类型	描述
userData	String	最近一次自定义脚本内容（使用base64编码）
instanceId	String	实例短ID

请求示例

```
POST /v2/instance/attribute/getUserdata HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId": "i-t5pmtNZI"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "userData": "aGVsbG8gd29ybGQ=",
    "instanceId": "i-t5pmtNZI"
}
```

进入救援模式

接口说明

实例进入救援模式

请求结构

```
 PUT /v{version}/instance/rescue/mode/enter?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId":"instanceId",
    "forceStop":"true",
    "password":"password"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	Request Body参数	虚拟机实例ID
forceStop	Boolean	是	Request Body参数	强制关机
password	String	是	Request Body参数	虚拟机密码

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 PUT /v2/instance/rescue/mode/enter HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId":"i-kotclGLf"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "requestId": "15dd5d7c-2ce0-4cd4-8616-ac0371cea63b"
}
```

退出救援模式

接口说明

实例退出救援模式

请求结构

```
 PUT /v{version}/instance/rescue/mode/exit?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId":"instanceId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	Request Body参数	虚拟机实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 PUT /v2/instance/rescue/mode/exit HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId":"i-kotclGLf"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "requestId": "15dd5d7c-2ce0-4cd4-8616-ac0371cea63b"
}
```

磁盘相关接口

创建CDS磁盘

- 本接口用于创建CDS磁盘。支持从快照创建的方式，但系统盘快照不能用来创建CDS磁盘。
- 请求参数中 cdsSizeInGB 和 snapshotId 必须选择其中一项来指定磁盘的大小或使用快照创建磁盘。
- 当snapshotId 参数存在且不为空时，cdsSizeInGB参数如果未指定，默认使用快照大小创建CDS磁盘。如果指定了cdsSizeInGB参数，则参数值不能小于快照大小值。
- 创建接口为异步接口，可通过查询磁盘详情接口查询磁盘状态
- 当预付费CDS挂载到已经开通自动续费的BCC时，CDS磁盘自动开通自动续费

请求结构

```
 POST /v{version}/volume?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "storageType": storageType,
  "cdsSizeInGB" : cdsSizeInGB,
  "snapshotId": snapshotId,
  "purchaseCount" : purchaseCount,
  "name": name,
  "description": description,
  "instanceId": "instanceId",
  "encryptKey": encryptKey,
  "renewTimeUnit": "renewTimeUnit",
  "renewTime": renewTime,
  "relationTag": relationTag,
  "chargeType" : chargeType,
  "deleteWithInstance" : deleteWithInstance
  "zoneName": "zoneName",
  "tags": [
    {
        "tagKey": "tagKey",
        "tagValue": "tagValue"
    }
  ],
  "resGroupId": "groupId",
  "clusterId": "clusterId",
  "autoSnapshotPolicy": {
    "autoSnapshotPolicyId": "autoSnapshotPolicyId"
  },
  "billing":{
    "paymentTiming":paymentTiming,
    "reservation":{
        "reservationLength":reservationLength,
        "reservationTimeUnit":reservationTimeUnit
      }
  }
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号

			数	
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见幂等性。
purchaseCount	int	否	RequestBody参数	批量创建的CDS磁盘的个数，必须为大于0的整数，单次创建不能超过5个。可选参数，缺省为1
name	String	否	RequestBody参数	磁盘新的名称，自定义镜像名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
description	String	否	RequestBody参数	磁盘新的描述信息
cdsSizeInGB	int	否	RequestBody参数	CDS磁盘容量，必须为大于0的整数，单位为GB，大小为5~32765GB，其中 SSD 类型云磁盘的起售容量不小于 50GB。当参数snapshotId为空或不存在时该参数必需。
storageType	StorageType	否	RequestBody参数	CDS磁盘存储类型，包括enhanced_ssd_pl1【增强型SSD_PL】，enhanced_ssd_pl2【增强型SSD_PL2】，cloud_hp1 或 premium_ssd 【通用型SSD】 和 hp1 或 ssd 【高性能云磁盘】 四种类型，默认 hp1。
chargeType	String	否	RequestBody参数	计费类型，付费方式，包括预支付（Prepaid）和后支付（Postpaid），该参数仅当instanceId非空，且对应实例类型为预付费时需要指定。如果instanceId为空，则创建后付费类型CDS；如果instanceId非空： 如果实例为预付费，需要指定chargeType；如果实例为后付费，创建后付费CDS
snapshotId	String	否	RequestBody参数	快照ID，支持从快照创建磁盘。当此参数存在时且不为空时，cdsSizeInGB参数将被忽略，此时非必需。
zoneName	String	否	RequestBody参数	指定可用区信息，默认为空，由系统自动选择。
autoSnapshotPolicy	List<AutoSnapshotPolicyModel>	否	RequestBody参数	磁盘当前配置的自动快照策略
renewTimeUnit	String	否	RequestBody参数	按月付费或者按年付费 月是"month",年是"year"。**注意：该参数已废弃**，当创建预付费CDS并挂载到已经开通自动续费的BCC时，CDS磁盘自动开通自动续费且续费周期和BCC实例保持一致。
renewTime	int	否	RequestBody参数	自动续费的时间 按月是1-9 按年是 1-3。**注意：该参数已废弃**，当创建预付费CDS并挂载到已经开通自动续费的BCC时，CDS磁盘自动开通自动续费且续费周期和BCC实例保持一致。
instanceId	String	否	RequestBody参数	创建磁盘并挂载到该虚机上。预付费CDS必须挂载预付费BCC。**注意：**当创建预付费CDS并挂载到已经开通自动续费的BCC时，CDS磁盘自动开通自动续费且续费周期和BCC实例保持一致。
encryptKey	String	否	RequestBody参数	KMS密钥ID。
relationTag	boolean	否	RequestBody参数	待创建CDS指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在

tags	List<TagModel>	否	RequestBody参数	待绑定的标签列表
clusterId	String	否	RequestBody参数	CDS专属集群ID
resGroupId	String	否	RequestBody参数	资源组ID
deleteWithInstance	boolean	否	RequestBody参数	磁盘随实例删除，计费方式为按量后付时生效，初始值为false，V2 API需要在释放实例时指定cdsAttributeActive参数为true才可生效
billing	Billing	否	RequestBody参数	创建预付费磁盘指定购买时长，默认对齐BCC到期时间，预付费磁盘的到期时间必须晚于等于BCC到期时间。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
volumeIds	List	磁盘ID，符合BCE规范，必须是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
warningList	List	挂载磁盘产生的warning信息

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.SystemSnapshotForbiddenOperation	System snapshot can not be used to create volume.	403	系统快照不允许创建磁盘

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/volume?clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "storageType": "hp1",
  "cdsSizeInGB" : 5,
  "purchaseCount" : 2,
  "name": "TestName2",
  "description": "TestDesc2",
  "chargeType": "Postpaid"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
        "volumeIds": [
            "v-NSuv0ssl",
            "v-MPO0YONO"
         ],
        "warningList": [
            "warning info"
        ]
}
```

查询磁盘列表

该接口用于查询所有的磁盘列表信息，不包含本地磁盘。

请求结构

```
 GET /v{version}/volume?marker={marker}&maxKeys={maxKeys}&instanceId={instanceId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
instanceId	String	否	Query参数	实例id，可查询实例挂载的磁盘列表（包含系统盘）。如需查询用户创建的所有磁盘信息，不填写此参数。
zoneName	String	否	Query参数	可用区信息
clusterId	String	否	Query参数	CDS专属集群ID
chargeFilter	String	否	Query参数	CDS计费类型,取值postpay/prepay
usageFilter	String	否	Query参数	CDS磁盘类型，取值system/data
name	String	否	Query参数	CDS磁盘名称

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
volumes	List<VolumeModel>	返回的磁盘信息列表

请求示例

```
 GET /v2/volume?marker=v-8CM91d5U&maxKeys=1&instanceId=i-j7IMWiyX HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Cache-Control: no-cache
Content-Length: 17527
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:38:06 GMT
Server: BWS

  {
     "nextMarker": "v-w7RKfn2d",
     "marker": "v-8CM91d5U",
     "maxKeys": 1,
     "volumes": [
      {
        "id": "v-8CM91d5U",
        "createTime": "2015-06-25T12:48:29Z",
        "expireTime": null,
        "name": "instance-0svq10tr-1",
        "diskSizeInGB": 20,
        "status": "InUse",
        "type": "Cds",
        "storageType": "hp1",
        "desc": "",
        "paymentTiming": "Postpaid",
        "attachments": [
           {
              "volumeId": "v-3zmCcxbR",
              "instanceId": "i-j7IMWiyX",
              "device": "/dev/vdb"
              "serial":"062b3f5d-b8a5-4c29-9"
           }
         ],
  "zoneName": "cn-bj-a"
        }
     ],
     "truncated": true
  }
```

查询磁盘详情

该接口用于根据磁盘ID查询单个云磁盘的详细信息。

请求结构

```
GET /v{version}/volume/{volumeId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待查询的磁盘ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
volume	VolumeModel	返回的磁盘信息

请求示例

```
GET /v2/volume/v-8CM91d5U HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "volume": {
     "id": "v-8CM91d5U",
     "createTime": "2015-06-25T12:48:29Z",
     "expireTime": null,
     "name": "instance-0svq10tr-1",
     "diskSizeInGB": 100,
     "status": "InUse",
     "type": "Cds",
     "storageType": "hp1",
     "desc": "",
     "paymentTiming":"Postpaid",
     "attachments": [
                {
                    "volumeId": "v-3zmCcxbR",
                    "instanceId": "i-j7IMWiyX",
                    "device": "/dev/vdb"
                    "serial":"062b3f5d-b8a5-4c29-9"
                }
               ],
        "regionId":"bj",
        "sourceSnapshotId":"s-7HjSrGvA",
        "snapshotNum":"2",
        "tags":[
           {
              "tagKey":"默认项目",
              "tagValue":""
           }
        ],
        "autoSnapshotPolicy":{
           "id":"asp-AwDut1hQ",
           "name":"cxy_asp_19",
           "timePoints":[
              19,
              20,
              21
           ],
           "repeatWeekdays":[
              1,
              2,
              3
           ],
           "retentionDays":3,
           "status":"active"
        },
  "zoneName": "cn-bj-a"
  }
 }
```

挂载CDS磁盘

- 该接口用于将指定磁盘挂载在指定云服务器BCC实例下。

> **注意：**
>
> cds磁盘需要挂载在与其处于相同可用区下的指定BCC实例上，否则将返回403错误。

- 只有磁盘状态为 Available 且实例状态为 Running 或 Stopped 时才允许挂载，否则调用此接口将返回 409 错误。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
  "instanceId":instanceId,
  "deleteWithInstance":deleteWithInstance,
  "deleteAutoSnapshot":deleteAutoSnapshot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	磁盘ID
action	String	是	Query参数	对磁盘的执行动作，当前取值：attach
instanceId	String	是	Request Body参数	待挂载的虚拟机实例ID
deleteWithInstance	boolean	否	Request Body参数	磁盘随实例删除，默认为false，仅后付费类型的数据盘支持配置该选项
deleteAutoSnapshot	boolean	否	Request Body参数	自动快照随磁盘删除，默认为false，所有类型的磁盘都支持配置该选项

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
volumeAttachment	VolumeAttachmentModel	磁盘挂载虚机的信息
warningList	List	挂载磁盘产生的warning信息

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.InstanceStatusCannotAttach	Instance only when it is running or stopped state can mount to it.	409	挂载磁盘，虚机状态不允许
Volume.DistNotAvailable	Volume status is not available, can not be attached to instance.	409	磁盘状态不可用
Volume.DiskAttachedQuotaExceedLimit	The attached disk number to an instance exceeds limit.	413	cds磁盘挂载超额

请求示例

```
 PUT /v2/volume/v-3zmCcxbR?attach HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "instanceId":"i-j7IMWiyX"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "volumeAttachment": {
        "volumeId": "v-3zmCcxbR",
        "instanceId": "i-j7IMWiyX",
        "device": "/dev/vdb"
        "serial":"062b3f5d-b8a5-4c29-9"
    },
    "warningList": [
        "warning info"
    ]
}
```

卸载CDS磁盘

该接口用于将指定的磁盘从BCC实例上卸载。

- BCC实例状态为Running或Stopped或者EBC实例状态为Running时，都可以执行此操作，否则将提示 409 错误。
- 如果 volumeId 的磁盘未挂载在 instanceId 的实例上，该操作失败，提示 404 错误。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
"instanceId" : instanceId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待卸载的磁盘对应的ID
action	String	是	Query参数	对磁盘的执行动作，当前取值：detach
instanceId	String	是	Request Body参数	所挂载的实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.DiskNotAttachedInstance	Volume is not mounted in this instance, please make sure and try again.	400	磁盘未绑定实例
Volume.InstanceStatusCannotDetach	Instance only when it is running or stopped state can unmount from it.	409	卸载磁盘，虚机状态不允许

请求示例

```
 PUT /v2/volume/v-bSlHQpKA?detach HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceId":"i-2fCSL4Y6"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

释放CDS磁盘（新）

该接口用于释放未挂载的CDS数据盘，系统盘不能释放。

- 用户可指定是否删除磁盘关联的快照，缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 用户可以指定是否将磁盘放入回收站。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误。
- 如果 volumeId 的磁盘不存在，将提示 404 错误。
- 与老接口功能上的区别：可以控制是否删除与磁盘关联的快照、是否进入回收站。

请求结构

```
 POST /v{version}/volume/{volumeId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
  "autoSnapshot": "on",
  "manualSnapshot": "on",
  "recycle": "on",
  "cdsAttributeActive": cdsAttributeActive
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待释放的磁盘ID
autoSnapshot	String	否	Request body参数	取值为"on"时，会删除磁盘关联的自动快照
manualSnapshot	String	否	Request body参数	取值为"on"时，会删除磁盘关联的手动快照
recycle	String	否	Request body参数	取值为"on"时，数据盘进入回收站，取值为"off"时，立即删除。默认为"on"。
cdsAttributeActive	boolean	否	Request body参数	自动快照是否按照“自动快照随磁盘释放”属性进行资源释放，默认为false，当该参数为true时将根据自动快照的关联释放属性进行释放，优先级高于autoSnapshot

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.DiskStatusCannotBeDeleted	Volume in current status can not be deleted.	409	磁盘当前状态不能被删除
Volume.SystemDiskCannotBeDeleted	System disk can not be deleted.	403	系统盘不能被删除

请求示例

```
 POST /v2/volume/v-3zmCcxbR HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: authorization-string

{
  "autoSnapshot": "on",
  "manualSnapshot": "on"
  "recycle": "on",
  "cdsAttributeActive": false
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘扩容及类型变更

该接口用于为指定磁盘变配扩容，修改CDS磁盘容量大小或磁盘类型。

- "使用中"或者"已挂载"状态时，才可以发起磁盘变配扩容操作。
- 该接口为异步接口，可通过查询磁盘详情接口查询VolumeStatus字段查看任务状态，当磁盘状态变为InUse时变更完成。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action}&clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "newCdsSizeInGB" : newCdsSizeInGB,
    "newVolumeType": newVolumeType
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	磁盘ID
action	String	是	Query参数	本接口取值：resize
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
newCdsSizeInGB	int	否	Request Body参数	新扩容CDS磁盘的容量大小，必须为大于当前CDS容量的整数，单位为GB，大小为0~32765GB的正整数。
newVolumeType	StorageType	否	Request Body参数	CDS支持对预付费和后付费云磁盘进行升配和降配。newVolumeType和newCdsSizeInGB不允许同时为空。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
warningList	List	磁盘扩容变更产生的warning信息

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.DiskIdIsEmpty	Volume id must be provided.	400	磁盘id不允许为空
Volume.DiskSizeInvalid	Disk size in invalid.	400	cds 大小不合法 400
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作
Volume.DiskQuotaExceededLimit	Number of volumes exceeds limit.	413	cds磁盘超额

请求示例

```
 PUT /v2/volume/v-3zmCcxbR?resize&clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "newCdsSizeInGB"：10,
      "newVolumeType"："hp1"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
   "warningList": [
      "warning info"
   ]
}
```

回滚磁盘

该接口用于从指定磁盘制作的快照回滚磁盘数据。

- 磁盘状态必须为 Available 才可以执行此操作。
- 指定快照ID(snapshotId) 必须是由指定磁盘ID(volumeId) 创建的快照。
- 若是回滚系统盘，实例状态必须为 Running 或 Stopped 才可以执行此操作。

> **注意：**
>
> 回滚系统盘快照，自本次快照以来的系统盘数据将全部丢失，不可恢复。

请求结构

```
 PUT /v2/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "snapshotId": snapshotId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待回滚的磁盘ID
action	String	是	Query参数	对磁盘执行的操作，当前取值：rollback
snapshotId	String	是	Request Body参数	用于回滚指定磁盘的快照ID，必须为 volumeId 所创建的快照

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
SnapshotOperationForbidden	snapshot operation forbidden.	403	快照操作 deny
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
 PUT /v2/volume/v-3zmCcxbR?rollback HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "snapshotId": "s-8CM91d5U"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘续费

该接口用于为指定磁盘续费，延长该磁盘过期时间。

- 磁盘扩缩容期间不能进行续费操作。
- 异步接口。
- 仅当磁盘到期时间早于挂载的BCC实例时允许续费到BCC到期时间

请求结构

```
 PUT /v2/volume/{volumeId}?{action}&clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	磁盘ID
action	String	是	Query参数	本接口中该参数取值：purchaseReserved
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
billing	Billing	是	RequestBody参数	自定义续费时长，单位为月。为空时表示与挂载的BCC到期时间对齐，必须保证自定义时长大于（BCC到期时间-CDS当前到期时间）否则续费失败。支持的时长为【1，60】个月
instanceId	String	否	RequestBody参数	未挂载磁盘预期挂载到的预付费BCC实例ID，磁盘未挂载状态下必传，已挂载状态下可不传或传当前挂载的实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
SnapshotOperationForbidden	snapshot operation forbidden.	403	快照操作 deny
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在
OperationDenied	Operation not available.	409	不可操作

请求示例

```
PUT /v2/volume/v-3zmCcxbR?purchaseReserved&clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘重命名

该接口用于修改指定磁盘的名称。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "name" : name
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待重命名的磁盘对应的ID
action	String	是	Query参数	对磁盘的执行动作，当前取值rename
name	String	是	Request Body参数	磁盘新的名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.DiskNameInvalid	Volume name is invalid.	400	磁盘名称不符合规则。

请求示例

```
 PUT /v2/volume/v-bSlHQpKA?rename HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "name":"volume-new-name"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

修改磁盘属性

该接口用于修改指定磁盘属性（名称、描述信息）。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "cdsName": "volume-new-name",
    "desc": "volume-new-desc",
    "deleteWithInstance": deleteWithInstance,
    "deleteAutoSnapshot": deleteAutoSnapshot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待修改磁盘ID
action	String	是	Query参数	对磁盘的执行动作，当前取值：modify
cdsName	String	否	Request Body参数	磁盘新的名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
desc	String	否	Request Body参数	磁盘新的描述信息，desc与cdsName不能同时为空
deleteWithInstance	boolean	否	Request Body参数	磁盘随实例删除，初始值为false，显式赋值后所有相关磁盘对应属性会被新值替换，仅后付费类型的数据盘支持配置该选项，V2 API需要在释放实例时指定cdsAttributeActive参数为true才可生效
deleteAutoSnapshot	boolean	否	Request Body参数	自动快照随磁盘删除，初始值为false，显式赋值后所有相关磁盘对应属性会被新值替换，所有类型的磁盘都支持配置该选项，V2 API需要在释放实例时指定cdsAttributeActive参数为true才可生效

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.DiskNameInvalid	Volume name is invalid.	400	磁盘名称不符合规则。

请求示例

```
 PUT /v2/volume/v-bSlHQpKA?modify HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "cdsName": "volume-new-name",
    "desc": "volume-new-desc"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘计费变更

描述

- 该接口用于变更磁盘计费方式，仅支持后付费转预付费、预付费转后付费两种方式。
- 预付费转后付费支持两种生效方式：到期后生效、立即生效。
- 当后付费CDS未挂载，或者挂载到的BCC非预付费类型时，禁用后转预。

请求结构

```
 PUT /v{version}/volume/{volumeId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	URL参数	待变更计费磁盘ID
action	String	是	Query参数	对磁盘的执行动作，当前取值modifyChargeType
billing	Billing	否	RequestBody参数	订单信息。**注意：该参数已废弃，** 当后付费CDS未挂载，或者挂载到的BCC非预付费类型时，禁用后转预；CDS后转预后的到期时间和其挂载的预付费BCC实例的到期时间一致。
effectiveType	String	否	RequestBody参数	转后付费生效方式，可选参数：AtOnce（立即转按量付费）、AfterExpiration（到期后转按量付费）。不传默认为到期后按量付费。

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Volume.VolumeTypeNotSupport	Volume type not support to_postpay.	403	磁盘类型不支持 "转后付费"
Volume.PaymentTypeNotSupportToPostpay	Payment type not support to_postpay	403	付费方式不支持 "转后付费"
Volume.AutoRenewNotSupportToPostpay	Auto renew instance not support to_postpay operation.	403	开通自动续费的实例不支持 "转后付费"
Volume.ExpiredResourceNotSupportToPostpay	Expired resource not support to_postpay operation.	403	已到期资源不支持 "转后付费"
Volume.VolumeTypeNotSupportToPrepay	Volume type not support to_prepay.	403	磁盘类型不支持 "转预付费"
Volume.PrepaidInstanceOperationForbidden	Postpaid instances are not supported for this operation.	409	付费方式不支持 "转预付费"
Volume.DiskStatusNotSupportBillingChange	The volume status not support billing change.	409	磁盘状态不允许计费变更

请求示例

```
 PUT /v2/volume/v-**********?modifyChargeType HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: authorization-string
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

开通自动续费

该接口用于为指定磁盘开通自动续费。

- 注意数据盘才能进行自动续费操作。
- 注意过期状态磁盘不能进行自动续费操作。
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间。

请求结构

```
 POST /v{version}/volume/autoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: ${authorization-string}

{
    "volumeId" : volumeId,
    "renewTimeUnit" : renewTimeUnit,
    "renewTime" : renewTime
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接
volumeId	String	是	RequestBody参数	磁盘ID
renewTimeUnit	String	是	RequestBody参数	按月付费或者按年付费 月是"month",年是"year"; 当前仅支持按月付费
renewTime	int	是	RequestBody参数	自动续费的时间 按月是1-9 按年是 1-3

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/volume/autoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: ${authoriazion-string}

{
    "volumeId" : "v-****",
    "renewTimeUnit" : "month",
    "renewTime" : 3
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: ${request-id}
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

取消自动续费

该接口用于为指定磁盘取消自动续费。

- 注意数据盘才能取消进行自动续费操作。
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间。

请求结构

```
 POST /v{version}/volume/cancelAutoRenew?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "volumeId" : volumeId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接
volumeId	String	是	RequestBody参数	磁盘ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/volume/cancelAutoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization-string

{
    "volumeId" : "v-sf*****"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘绑定标签

该接口用于给磁盘绑定标签，用于分类和查找。

请求结构

```
 PUT /v{version}/volume/{volumeId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ],
    "relationTag": relationTag
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	RequestBody参数	磁盘ID
action	String	是	Query参数	对磁盘的执行动作，当前取值：bind
relationTag	boolean	否	RequestBody参数	是否给关联资源绑定标签
changeTags	List<TagModel>	是	RequestBody参数	待绑定的标签列表，具体数据格式参见下述TagModel,绑定的标签如果之前不存在，将自动创建该标签

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/volume/v-y*******/tag?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48*****

{
    "changeTags": [
      {
       "tagKey" : "company",
       "tagValue": "bd"
      }
    ],
    "relationTag": false
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: d795139f-8b98-4a55-8d5e-dba7fd5b2b4b
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

磁盘解绑标签

该接口用于解绑磁盘上的标签。

请求结构

```
 PUT /v{version}/volume{volumeId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ],
    "relationTag" : relationTag
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
volumeId	String	是	RequestBody参数	磁盘ID
action	String	是	Query参数	对磁盘的执行动作，当前取值：unbind
relationTag	boolean	否	RequestBody参数	是否给关联资源解绑标签
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/volume/v-yec******/tag?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e4804****

{
    "changeTags": [
      {
       "tagKey" : "company",
       "tagValue": "bd"
      }
    ],
    "relationTag" : false
}
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 9a2dbfd2-c8bb-47e6-9110-bc02b0abe63e
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

获取可购买磁盘信息

该接口用于根据可用区获取可购买的磁盘类型、大小、配额限制。

请求结构

```
GET /v{version}/volume/disk?zoneName={zoneName}&clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
zoneName	String	是	Query参数	zone信息，zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。当传入zoneName为空串或为非法zone时，会返回全部可用区的可购买磁盘信息。

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
cdsRatio	String	单个BCC可创建的最大磁盘数
cdsTotalCapacityGB	String	磁盘总容量上限（GB）
cdsTotal	String	BCC实例个数 * cdsRatio。 **注意：该参数已经废弃，即不在按照BCC实例数量限制用户购买的CDS数量，当前返回为固定值**
cdsCreated	String	已创建磁盘个数，包括回收站内的磁盘
cdsUsedCapacityGB	String	磁盘已使用总容量（GB）
diskZoneResources	DiskZoneResources	可用区可创建的磁盘信息，具体参数可见 DiskZoneResources

请求示例

```
 GET http://bcc.bj.baidubce.com/v2/volume/disk?zoneName=cn-bj-a&clientToken=3A8d84403***********
Host: bcc.bj.baidubce.com
Authorization: authorization-string
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
    "cdsRatio": "15",
    "cdsTotalCapacityGB": "163840",
    "cdsTotal": "75",
    "cdsCreated": "0",
    "cdsUsedCapacityGB": "0",
    "diskZoneResources": [
        {
            "zoneName": "cn-bj-a",
            "diskInfos": [
                {
                    "storageType": "premium_ssd",
                    "maxDiskSize": 32765,
                    "minDiskSize": 50
                },
                {
                    "storageType": "ssd",
                    "maxDiskSize": 32765,
                    "minDiskSize": 5
                },
                {
                    "storageType": "enhanced_ssd_pl1",
                    "maxDiskSize": 32765,
                    "minDiskSize": 5
                }
            ]
        }
    ]
}
```

查询CDS价格

查询CDS价格。

请求结构

```
 POST /v{version}/volume/getPrice HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "storageType": specId,
    "cdsSizeInGB": spec,
    "paymentTiming": paymentTiming,
    "zoneName": zoneName,
    "purchaseCount": purchaseCount,
    "purchaseLength": purchaseLength
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
storageType	StorageType	是	RequestBody参数	CDS磁盘存储类型，包括SSD_Enhanced（增强型SSD），cloud_hp1 (通用型SSD)，hp1 (高性能云磁盘)和hdd (通用型HDD) 四种类型，默认 hp1。
cdsSizeInGB	int	是	RequestBody参数	CDS磁盘容量
paymentTiming	String	是	RequestBody参数	付费方式，包括Postpaid(后付费)，Prepaid(预付费)两种。
zoneName	String	是	RequestBody参数	可用区名称
purchaseCount	int	否	RequestBody参数	任意数量CDS的总价格，必须为大于0的整数，可选参数，缺省为1
purchaseLength	int	否	RequestBody参数	时长，[1,2,3,4,5,6,7,8,9,12,24,36]，单位：月

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
price	List<CdsPrices>	查询CDS对应价格信息

请求示例

```
 POST /v2/volume/getPrice HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "purchaseLength": 1,
    "paymentTiming": "Prepaid",
    "storageType":"cloud_hp1",
    "cdsSizeInGB":1000,
    "purchaseCount":1,
    "zoneName": "cn-bj-b"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "price": [
    {
      "storageType": "cloud_hp1",
      "cdsSizeInGB": 1000,
      "unit": "12month",
      "price": 8964.0
    }
  ]
}
```

镜像相关接口

创建自定义镜像

该接口用于创建自定义镜像，创建后的镜像可用于创建实例，每个用户单地域最多可拥有20个镜像。

- 支持通过实例创建和通过快照创建两种方式。
- 当通过实例创建时，只有 Running 或 Stopped 状态的实例才可以执行成功，否则会提示 409 错误。
- 仅限通过系统盘快照创建自定义镜像。
- 当通过快照创建时，只有 Available 状态的快照才可以执行成功，否则会提示 409 错误。

请求结构

```
 POST /v{version}/image?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "imageName" : imageName,
    "instanceId" : instanceId,
    "snapshotId" : snapshotId,
    "encryptKey": encryptKey,
    "relateCds": relateCds
    "detection": detection
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见幂等性。
imageName	String	是	RequestBody参数	待创建的自定义镜像名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
instanceId	String	否	RequestBody参数	当从实例创建镜像时，此参数是指用于创建镜像的实例ID。可选参数，与 snapshotId 不同时存在，同时存在时只取instanceId
snapshotId	String	否	RequestBody参数	当从快照创建镜像时，此参数是指用于创建镜像的快照ID。可选参数，与 instanceId 不同时存在，同时存在时只取instanceId
relateCds	boolean	否	RequestBody参数	是否创建包含了所有 cds 盘的大镜像，如果为true，镜像不能加密，且必须在白名单内
encryptKey	String	否	RequestBody参数	加密密钥
detection	bolean	否	RequestBody参数	镜像检测，不配置此参数时不触发检测。如果为true，镜像创建完成后自动执行镜像检测。 如果为false，镜像创建完成后不执行镜像检测。默认取值：false。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
imageId	String	已创建的镜像的ID

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.ImageNameInvalid	Image name is invalid.	400	镜像名称格式不正确
Image.ImageIdIsEmpty	Image ID cannot be empty	400	镜像ID格式不能为空
Image.ImageQuotaExceedLimit	Image number exceeds the limit.	413	镜像数目超限
Image.InstanceStatusForbidden	Instance not in running or stopped status.	409	实例状态不是运行中或者已停止，请确认并重试
Image.SnapshotStatusForbidden	Snapshot status is not Available, please confirm and retry.	409	快照状态为不可用，请确认并重试
Image.NonSystemSnapshotForbidden	Non system snapshot can not used to create image, please confirm and retry.	403	非系统快照无法创建镜像

请求示例

```
 POST /v2/image?clientToken=1849ef6b-f73c-4f5d-b88b-259ca47af849 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "imageName" : "template_example_001",
    "instanceId":"i-1849ef6b"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "imageId" : "m-nky7qeom"
}
```

查询镜像列表

该接口用于查询用户所有的镜像信息。

- 查询的镜像信息包括公共镜像、自定义镜像和服务集成镜像。
- 支持按 imageType 来过滤查询，此参数非必需，缺省为 All,即查询所有类型的镜像。
- 支持按 imageName 来过滤查询镜像，返回名称中包含该字符串的镜像。此参数非必需。

请求结构

```
 GET /v{version}/image?marker={marker}&maxKeys={maxKeys}&imageType={imageType}&imageName={imageName}
HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
imageType	String	否	Query参数	指定要查询何种类型的镜像，包括All(所有)，System(系统镜像/公共镜像)，BbcSystem(bbc系统镜像/bbc公共镜像)，Custom(自定义镜像)，Integration(服务集成镜像)，Sharing(共享镜像)，GpuBccSystem(GPU专用公共镜像)，GpuBccCustom(GPU专用自定义镜像)，FpgaBccSystem(FPGA专用公共镜像)，FpgaBccCustom(FPGA专用自定义镜像)，缺省值为All
imageName	String	否	Query参数	过滤查询镜像，返回名称中包含该字符串的镜像

返回头域

除公共头域，无其它特殊头域

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
images	List<ImageModel>	返回的镜像列表

请求示例

```
 GET /v2/image?marker=m-nky7qeom&maxKeys=1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "maxKeys":1,
    "marker":"m-nky7qeom",
    "isTruncated":true,
    "nextMarker":"m-I88RTNWG",
    "images":[
  {
      "id": "m-nky7qeom",
      "createTime": "2015-06-17T10:37:02Z",
      "name": "ubuntu-14.04.1-server-amd64-201506171832",
      "type": "System",
      "osType": "linux",
      "osVersion": "14.04.1 LTS",
      "osName": "Ubuntu",
      "osBuild": "2015061700",
      "osArch": "x86_64 (64bit)",
      "status": "Available",
      "desc": null
  }
    ]
}
```

查询镜像详情

该接口用于根据指定的镜像ID查询单个镜像的详细信息。

请求结构

```
 GET /v{version}/image/{imageId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待查询的镜像ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
image	ImageModel	返回的镜像信息

请求示例

```
 GET /v2/image/m-nky7qeom HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "image": {
        "id": "m-nky7qeom",
        "name": "ubuntu-14.04.1-server-amd64-201506171832",
        "createTime": "2015-06-17T10:37:02Z",
        "osArch": "x86_64 (64bit)",
        "osBuild": "2015061700",
        "osName": "Ubuntu",
        "osType": "linux",
        "osVersion": "14.04.1 LTS",
        "osLang": "ENG",
        "specialVersion": "",
        "type": "System",
        "status": "Available",
        "diskSize": 0,
        "minDiskGb": 10,
        "ephemeralSize": 0,
        "encrypted": false,
        "package": false,
        "desc": ""
    }
}
```

删除自定义镜像

该接口用于用户删除指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能删除。

- imageId 所指定的镜像不存在，提示404错误。
- 镜像删除后无法恢复，不能再用于创建、重置实例。

请求结构

```
 DELETE /v{version}/image/{imageId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待删除的镜像ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.UnCustomImageCannotBeDeleted	Only custom image can be deleted.	403	非自定义镜像不能删除
Image.CreatingImageCannotBeDeleted	Cannot delete image when creating.	409	创建中的镜像不能删除
Image.ImageHasBeenSharedToOthers	Image has been shared to others.	409	镜像已被共享给其他用户，无法删除

请求示例

```
 DELETE /v2/image/m-nky7qeom HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

共享自定义镜像

该接口用于用户共享自己指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能共享。

- imageId 所指定的镜像不存在，提示404错误。
- 镜像共享后，被共享的用户可以使用此镜像创建、重置实例。
- 请求参数中的account和accountId均为可选参数，但不能全部为空，当两个参数同时出现时，服务端会自动去重。

请求结构

```
 POST /v{version}/image/{imageId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "account" : "account",
   "accountId" : "accountId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待共享的镜像ID
action	String	是	Query参数	对镜像执行的动作，当前取值：share
account	String	否	Request Body参数	待共享的用户名
accountId	String	否	Request Body参数	待共享的用户ID
ucAccount	String	否	RequestBody参数	uc账号

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.UnCustomImageCannotBeShared	Only custom image can be shared.	403	非自定义镜像不能共享
Image.InvalidAccount	Account is invalid.	400	无效的用户
Image.AccountListIsEmpty	Account list is empty.	400	用户列表为空
Image.ExceedSharedToUserNumQuota	Exceed shared to user num quota.	413	超出可共享的用户数量限制
Image.CannotShareToOwner	Cannot share to owner.	403	镜像不能共享给所属用户
Image.AccountHasBeenShared	Account has been shared.	409	用户已接收到此共享镜像，不能重复共享
Image.ExceedAcceptSharedImageNumQuota	Exceed accept shared image num quota	413	超出可接受共享的镜像数量限制

请求示例

```
 POST /v2/image/m-nky*****?share HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization-string

{
   "account" : "********",
   "accountId" : "**********"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

取消共享自定义镜像

该接口用于用户取消共享指定的自定义镜像。

- imageId 所指定的镜像不存在，提示404错误。
- 镜像取消共享后，被取消共享的用户不能再使用此镜像创建、重置实例。
- 请求参数中的account和accountId均为可选参数，但不能全部为空，当两个参数同时出现时，服务端会自动去重。

请求结构

```
 POST /v{version}/image/{imageId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "account" : "account",
   "accountId" : "accountId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待取消共享的镜像ID
action	String	是	Query参数	对镜像执行的动作，当前取值：unshare
account	String	否	Request Body参数	待取消共享的用户名
accountId	String	否	Request Body参数	待取消共享的用户ID
ucAccount	String	否	RequestBody参数	uc账号

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.UnCustomImageCannotBeShared	Only custom image can be shared.	403	非自定义镜像不能共享
Image.InvalidAccount	Account is invalid.	400	无效的用户
Image.AccountListIsEmpty	Account list is empty.	400	用户列表为空

请求示例

```
 POST /v2/image/m-nky****?unshare HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization-string

{
   "accountId" : "ab1d7802c***********"
}
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询镜像已共享的用户列表

该接口用于查询自定义镜像已共享的用户列表。

- imageId 所指定的镜像不存在，提示404错误。

请求结构

```
GET /v{version}/image/{imageId}/sharedUsers HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待查询的镜像ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
users	List<SharedUser>	返回的共享用户列表

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.UnCustomImageCannotBeShared	Only custom image can be shared.	403	非自定义镜像不能共享

请求示例

```
GET /v2/image/m-nky7qeom/sharedUsers HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    users:[
        {
            account:'ceshizhanghao',
            accountId : 'ab1d7802cb9b436bafaf3361fe482ca6'
        }
    ]
}
```

根据实例ID批量查询OS信息

该接口用于根据实例ID批量查询实例的操作系统信息。

- instanceIds参数非必需，缺省返回结果空。

请求结构

```
 POST /v{version}/image/os HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "instanceIds": ["instanceId"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
instanceIds	List<String>	否	Request Body参数	实例ID列表

返回头域

除公共头域，无其他特殊头域

返回参数

参数名称	类型	描述
osInfo	List<OsModel>	返回的OS列表

请求示例

```
 POST /v2/image/os HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds": [
    "i-ZeVPsPzv",
    "i-HmGPcF2D",
    "i-FAISyeUz"
 ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
"osInfo": [
    {
        "instanceId": "i-FAISyeUz",
        "osArch": "i386 (32bit)",
        "osVersion": "14.04.1 LTS",
        "osName": "Ubuntu",
        "osType": "linux"
    },
    {
        "instanceId": "i-ZeVPsPzv",
        "osArch": "i386 (32bit)",
        "osVersion": "14.04.1 LTS",
        "osName": "Ubuntu",
        "osType": "linux"
    },
    {
        "instanceId": "i-HmGPcF2D",
        "osArch": "i386 (32bit)",
        "osVersion": "14.04.1 LTS",
        "osName": "Ubuntu",
        "osType": "linux"
    }
]
```

导入镜像

该接口用于导入单个自定义镜像。

请求结构

```
 POST /v{version}/image/import HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
osName	String	是	Request Body参数	操作系统名称
osArch	String	是	Request Body参数	操作系统位数
osType	String	是	Request Body参数	操作系统类型
osVersion	String	是	Request Body参数	操作系统版本
name	String	是	Request Body参数	镜像名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
bosUrl	String	是	Request Body参数	bos镜像地址
detection	bolean	否	RequestBody参数	镜像检测，不配置此参数时不触发检测。如果为true，镜像导入后自动执行镜像检测。如果为false，镜像导入后不执行镜像检测。默认取值：false。
generationType	String	否	Request Body参数	导入的镜像适用于BCC还是EBC，枚举值：BCC、EBC，默认BCC

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
Id	String	返回的镜像ID

请求示例

```
 POST /v2/image/import HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "osName":"Centos",
    "osArch":"32",
    "osType":"linux",
    "osVersion":"6.5",
    "name":"szth-onlineA_34_test",
    "bosUrl":"http://cloud.baidu.com/t34"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "id":"m-cxFNBt46"
}
```

镜像解绑标签

该接口用于解绑自定义镜像上关联的标签。

请求结构

```
 PUT /v{version}/image/{imageId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "changeTags": [
     {
      "tagKey" : tagKey,
      "tagValue": tagValue
     }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	RequestBody参数	镜像ID
action	String	是	Query参数	对镜像执行的操作，当前取值：unbind
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/image/m-7hz803oi/tag?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e4******

{
   "changeTags": [
     {
      "tagKey" : "company",
      "tagValue": "bd"
     }
   ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 84033835-d45e-4a02-be7f-d7335a1c6476
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

镜像绑定标签

该接口用于给自定义镜像绑定标签，用于分类和查找。

请求结构

```
 PUT /v{version}/image/{imageId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "changeTags": [
     {
      "tagKey" : tagKey,
      "tagValue": tagValue
     }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	RequestBody参数	镜像ID
action	String	是	Query参数	对镜像执行的操作，当前取值：bind
changeTags	List<TagModel>	是	RequestBody参数	待绑定的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/image/m-Q0ezqMIa/tag?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e*****
{
    "changeTags": [
      {
       "tagKey" : "company",
       "tagValue": "bd"
      }
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: f25f6c03-d473-474b-b988-a9326b8a06b6
Date: Mar, 22 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

根据实例规格查询可用公共镜像

该接口用于根据实例规格查询可用的公共镜像信息。

请求结构

```
 GET /v{version}/image/getAvailableImagesBySpec HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，默认10，最大数量不超过100
spec	String	是	Query参数	实例规格
osName	String	否	Query参数	镜像操作系统名称。例：Windows、Centos、Ubuntu等

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
images	List<SystemImageModel>	返回的公共镜像列表

请求示例

```
 GET /v2/image/getAvailableImagesBySpec?maxKeys=1&spec=bcc.ic4.c1m1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 06 Sep 2023 07:39:43 GMT
Authorization:  bce-auth-v1/8179c90f19ec49f79164110c18e8eff4/2023-09-06T07:39:43****
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 106fa628c-320c-4ddd-b3bc-579cda4b287d
Date: Wed, 06 Sep 2023 07:39:45 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "isTruncated": true,
    "marker": "m-xuP01Lfw",
    "maxKeys": 1,
    "nextMarker": "m-QtDM9yPo",
    "images": [
        {
            "imageId": "m-xuP01Lfw",
            "imageName": "16.04 LTS amd64 (64bit)-CUDA8.0",
            "osType": "linux",
            "osVersion": "7.5",
            "osArch": "x86_64 (64bit)",
            "osName": "CentOS",
            "osLang": "ENG",
            "minSizeInGiB": 20
        }
    ]
}
```

取消跨地域复制自定义镜像

该接口用于取消跨区域复制自定义镜像。

- imageId是目的地域的镜像id，若指定的镜像不存在，提示404错误。

请求结构

```
 POST /v{version}/image/{imageId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待取消复制的镜像ID
action	String	是	Query参数	对镜像执行的动作，当前取值：cancelRemoteCopy

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.ImageStatusForbidden	Image status is not dest_copying.	409	镜像不是复制中状态，无法取消

请求示例

```
 POST /v2/image/m-nky7qeom?cancelRemoteCopy HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

跨地域复制自定义镜像

该接口用于用户跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制。

- imageId 所指定的镜像不存在，提示404错误。
- 镜像复制成功后，用户可在目的地域使用此镜像创建、重置实例。

请求结构

```
 POST /v{version}/image/{imageId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "name" : "destImageName",
   "destRegion" : ["bj","gz"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
imageId	String	是	URL参数	待复制的镜像ID
action	String	是	Query参数	对镜像执行的动作，当前取值：remoteCopy
name	String	否	Request Body参数	复制镜像名
destRegion	List	是	Request Body参数	目的regionId,可以传多个

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
result	List	远程复制结果

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.RegionNotSupportCopy	Region not support copy.	409	目的region不支持复制
Image.ImageStorageFormatInvalid	Image storage format is invalid.	409	镜像存储格式不支持复制
Image.ImageNotSupportCopy	Image not support copy.	409	非自定义镜像不能复制
Image.ImageStatusForbidden	Image status is not Available.	409	镜像状态为不可用

请求示例

```
 POST /v2/image/m-nky7qeom?remoteCopy HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "name": "test-copy-image",
  "destRegion": ["bj","gz"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
   "result": [
      {
         "region": "gz",
         "imageId": "m-3ev9sySj",
         "errMsg": "success",
         "code": "success"
      },
      {
         "region": "bj",
         "imageId": "m-3ev9sySj",
         "errMsg": "success",
         "code": "success"
      }
   ]
}
```

修改自定义镜像名称

该接口用于修改自定义镜像名称。

请求结构

```
 PUT /v{version}/image/rename?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "name" : imageName,
    "imageId" : imageId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见幂等性。
name	String	是	Request Body参数	待创建的自定义镜像名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
imageId	String	是	Request Body参数	自定义镜像的ID

返回头域

除公共头域，无其它特殊头域。

错误码

错误码	错误描述	HTTP状态码	中文解释
Image.ImageNameInvalid	Image name is invalid.	400	镜像名称格式不正确
BadRequest	Bad request parameters or illegal request.	400	请求参数不合法

请求示例

```
 PUT /v2/image/rename?clientToken=1849ef6b-f73c-4f5d-b88b-259ca47af849 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "name" : "newName",
    "imageId":"m-1849ef6b"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

快照相关接口

创建快照

该接口用于从指定的磁盘创建快照，支持创建系统盘快照和CDS数据盘快照。

- 创建系统盘快照时，实例的状态必须为 Running 或 Stopped 此操作才能正常完成。
- 创建CDS云磁盘快照时，磁盘的状态需要为InUse(使用中)或 Available(待挂载)，此操作才能正常完成。
- 每块云磁盘最多可同时创建1536份快照，包括512份手动快照和1024份自动快照。

请求结构

```
 POST /v{version}/snapshot?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
 "volumeId":volumeId,
   "snapshotName": snapshotName,
   "tags": [
      {
         "tagKey": "tagKey",
         "tagValue": "tagValue"
      }
   ]
 "desc" : create snapshot,
   "retentionInDays": retentionInDays
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
volumeId	String	是	Request Body参数	用于创建快照的磁盘ID，系统盘则为实例ID
snapshotName	String	是	Request Body参数	快照名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
desc	String	否	Request Body参数	快照描述信息，选填
tags	List<TagModel>	否	Request Body参数	绑定标签信息
retentionInDays	int	否	Request Body参数	快照的保留时间，默认-1永久保留，取值范围为1-10000天

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
snapshotId	String	已创建的快照ID

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
Snapshot.SnapshotNameInvalid	snapshot name is invalid.	400	名称非法
Instance.SnapshotLimitExceeded	The number of snapshot will exceed the quota limit.	413	快照超出额度

请求示例

```
 POST /v2/snapshot?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "volumeId": "v-nky7qeom",
    "snapshotName": "example-001"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "snapshotId" : "s-6r3dam6F"
}
```

查询快照列表

该接口用于查询用户所有的快照信息，包括系统盘快照和CDS数据盘快照信息。

请求结构

```
 GET /v{version}/snapshot?marker={marker}&maxKeys={maxKeys}&volumeId={volumeId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串。
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
volumeId	String	否	Query参数	通过磁盘ID过滤，系统盘则为实例ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
snapshots	List<SnapshotModel>	返回的快照列表

错误码

无特殊返回错误码

请求示例

```
 GET /v2/snapshot?marker=s-6r3dam6F&maxKeys=2 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "s-tpxSNbqb",
    "marker": "s-6r3dam6F",
    "maxKeys": 2,
    "snapshots": [
        {
            "id": "s-6r3dam6F",
            "createTime": "2015-07-08T10:22:04Z",
            "name": "snapshot-name",
            "sizeInGB": 5,
            "status": "Available",
            "createMethod": "MANUAL",
            "volumeId": "v-lgJOjBWj",
            "desc": "snapshot-decription"
        },
        {
            "id": "s-UrGLpTYC",
            "createTime": "2015-07-08T10:21:03Z",
            "name": "snapshot-name",
            "sizeInGB": 5,
            "status": "Available",
            "createMethod": "MANUAL",
            "volumeId": "v-lgJOjBWj",
            "desc": "snapshot-decription"
        }
    ],
    "isTruncated": false
}
```

查询快照详情

该接口用于用户通过指定快照ID查询单个快照的详细信息。

请求结构

```
GET /v{version}/snapshot/{snapshotId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
snapshotId	String	是	URL参数	待查询的快照ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
snapshot	SnapshotModel	返回的快照信息

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在

请求示例

```
GET /v2/snapshot/s-6r3dam6F HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "snapshot": {
        "id": "s-6r3dam6F",
        "createTime": "2015-07-08T10:21:03Z",
        "name": "snapshot-name",
        "sizeInGB": 5,
        "status": "Available",
        "createMethod": "MANUAL",
        "volumeId": "v-lgJOjBWj",
        "desc": "snapshot-decription"
    }
}
```

删除快照

该接口用于通过快照ID删除指定快照。

- snapshotId 所指定的快照不存在时，此操作无法正常完成，将提示 404 错误。
- 只有当快照的状态为 CreatedFailed （创建失败）或 Available（可用）时，此操作才能正常完成，否则将提示 403 错误。

请求结构

```
DELETE /v{version}/snapshot/{snapshotId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
snapshotId	String	是	URL参数	待删除的快照的ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

无特殊错误码

请求示例

```
DELETE /v2/snapshot/s-6r3dam6F HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询快照链列表

该接口用于查询用户的快照链列表信息。

请求结构

```
GET /v{version}/snapshot/chain HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
orderBy	String	否	Query参数	排序属性，可选: chainId(快照链id,默认值),chainSize(快照链大小),volumeSize(磁盘大小)
order	String	否	Query参数	排序方式 可选:asc(正序,默认值), desc(倒序)
pageSize	int	否	Query参数	每页返回条数,默认值为1000
pageNo	int	否	Query参数	页数,默认值为1
volumeId	String	否	Query参数	磁盘id,该字段非空则只返回这个磁盘的快照链信息

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderBy	String	排序属性
totalCount	int	快照链总数
pageSize	int	该页容量
pageNo	int	页数
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
snapchains	List<SnapchainModel>	快照链信息，由 SnapchainModel 组成的集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在

请求示例

```
 GET /v2/snapshot/chain HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "orderBy": "chainSize",
  "totalCount": 11,
  "pageSize": 2,
  "pageNo": 1,
  "isTruncated": true,
  "snapchains": [
    {
      "status": "active",
      "chainSize": "20.0",
      "chainId": "sl-IvmGOxPc",
      "instanceId": "i-DRqFcFu8",
      "userId": "dcc4b11a97984b62baa65d4592f4d432",
      "volumeId": "v-IvmGOxPc",
      "volumeSize": 40,
      "manualSnapCount": 1,
      "createTime": "2019-12-23T14:35:07Z",
      "autoSnapCount": 0
    },
    {
      "status": "active",
      "chainSize": "0.0",
      "chainId": "sl-dqH1rplQ",
      "instanceId": "",
      "userId": "dcc4b11a97984b62baa65d4592f4d432",
      "volumeId": "v-dqH1rplQ",
      "volumeSize": 50,
      "manualSnapCount": 1,
      "createTime": "2019-12-23T11:02:16Z",
      "autoSnapCount": 0
    }
   ]
}
```

快照链解绑标签

该接口用于解绑快照链上关联的标签。

请求结构

```
 PUT /v{version}/snapshot/chain/{chainId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
chainId	String	是	RequestBody参数	快照链ID
action	String	是	Query参数	对快照链执行的操作，当前取值：unbind
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/snapshot/chain/sl-a5Yq4sTU/tag?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e480*****

{
    "changeTags": [
      {
       "tagKey" : "company",
       "tagValue": "bd"
      }
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 71a1a3da-5504-4254-88f5-f2793179e1ca
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

快照链绑定标签

该接口用于给快照链绑定标签，用于分类和查找。

请求结构

```
 PUT /v{version}/snapshot/chain/{chainId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
chainId	String	是	RequestBody参数	快照链ID
action	String	是	Query参数	对快照链执行的操作，当前取值：bind
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/snapshot/chain/sl-a5Yq4sTU/tag?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fb*****

{
   "changeTags": [
     {
      "tagKey" : "company",
      "tagValue": "bd"
     }
   ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: ed0b13b1-14ad-4b74-a76c-ba8c904d3c58
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

跨地域复制快照

该接口用于将一份磁盘快照从一个地域复制到另一个地域。

请求结构

```
 PUT /v{version}/snapshot/remote_copy/{snapshotId}?clientToken={clientToken} HTTP/1.1
Host bcc.bj.baidubce.com
Authorization: authorization string
{
   "destRegionInfos": [
      {
         "name": "test",
         "destRegion": "hkg"
      }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
version	String	是	URL参数	API版本号(v2版本支持，v3暂不支持)
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等。
snapshotId	String	是	URL参数	待复制的快照ID
destRegionInfos	List<RemoteCopyRequest>	是	requestBody	待复制到目标区域列表

返回头域

除公共头域，无其他特殊头域

返回参数

参数名称	类型	描述
result	List<RemoteCopySnapshot>	目标区域快照信息

错误码

错误码	错误描述	HTTP状态码	中文解释
OperationDenied	Operation not available	409	无效操作，包括参数异常
NoSuchObject	The specified object is not found or resource do not exist.	404	源快照不存在
snapshot.CannotCopy	napshot not support copy	403	源快照不支持跨区域复制

请求示例

```
 POST /v2/snapshot/remote_copy/s-AJcboed5?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2021-12-03T06:42:19Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "destRegionInfos": [
      {
         "name": "test",
         "destRegion": "hkg"
      }
   ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2021 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
   "result": [
      {
         "region": "hkg",
         "snapshotId": "s-qwZ8QD1R"
      }
   ]
}
```

自动快照策略相关接口

创建自动快照策略

该接口用于创建自动快照策略。

请求结构

```
 POST /v{version}/asp?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name":name,
    "timePoints":[timePoints],
    "repeatWeekdays":[repeatWeekdays],
    "retentionDays":retentionDays
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
name	String	是	RequestBody参数	自动快照策略名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
timePoints	List<String>	是	RequestBody参数	一天中做快照时间点，取值为0~23。
repeatWeekdays	List<String>	是	RequestBody参数	一周中做快照的时间，取值为0~6。
retentionDays	String	是	RequestBody参数	自动快照保留天数，取-1则永久保留。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
aspId	String	自动快照策略ID

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
Asp.NameInvalid	asp name is invalid.	400	名称非法
Asp.RetentionDaysInvalid	RetentionDays should in [1,10000]	400	保留天数非法
Asp.TimePointsInvalid	TimePoints should in [0,23]	400	执行时间点参数异常
Asp.RepeatWeekdaysInvalid	RepeatWeekdays should in [1,7]	400	执行日期参数异常

请求示例

```
 POST /v2/asp?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "name":"asp-num1",
    "timePoints":["20"],
    "repeatWeekdays":["1","5"],
    "retentionDays":"7"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "aspId" : "asp-oG6prjrl"
}
```

绑定自动快照策略

该接口用于将自动快照策略绑定到指定CDS磁盘。

请求结构

```
 PUT /v{version}/asp/{aspId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "volumeIds":[volumeId],
    "deleteAutoSnapshot":deleteAutoSnapshot
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
aspId	String	是	URL参数	自动快照策略ID
action	String	是	Query参数	对自动快照策略执行的动作，当前取值：attach
volumeIds	List<String>	是	RequestBody参数	需要绑定的磁盘ID
volumeIds	List<String>	是	RequestBody参数	需要绑定的磁盘ID
deleteAutoSnapshot	boolean	否	RequestBody参数	自动快照随磁盘删除，初始值为false，显式赋值后所有相关磁盘对应属性会被新值替换

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Asp.VolumeTypeInvalid	Volume type should be Cds or System	400	volume类型非法
Asp.VolumeStatusInvalid	Volume must be in-use	400	volume状态必须为in-use
BadRequest	missing parameter.	400	缺少参数
ServiceInternalError	Internal service occurs error.	500	内部错误
NoSuchObject	The specified object is not found or resource do not exist.	404	指定资源不存在

请求示例

```
PUT /v2/asp/asp-oG6prjrl?attach HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "volumeIds":["v-111","v-222"]
}
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

解绑自动快照策略

该接口用于解绑自动快照策略。

请求结构

```
PUT /v{version}/asp/{aspId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "volumeIds":[volumeId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
aspId	String	是	URL参数	自动快照策略ID
action	String	是	Query参数	对自动快照策略执行的动作，当前取值：detach
volumeIds	List<String>	是	RequestBody参数	需要解绑的磁盘ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	missing parameter.	400	缺少参数
ServiceInternalError	Internal service occurs error.	500	内部错误

请求示例

```
PUT /v2/asp/asp-oG6prjrl?detach HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "volumeIds":["v-111","v-222"]
}
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除自动快照策略

该接口用于删除自动快照策略。

请求结构

```
DELETE /v{version}/asp/{aspId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
aspId	String	是	URL参数	自动快照策略ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数错误
NoSuchObject	The specified object is not found or resource do not exist.	404	指定的自动快照策略不存在
ServiceInternalError	Internal service occurs error.	500	内部错误

请求示例

```
DELETE /v2/asp/asp-oG6prjrl HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询自动快照策略列表

该接口用于查询所有的自动快照策略列表信息。

请求结构

```
GET /v{version}/asp?marker={marker}&maxKeys={maxKeys}&aspName={aspName}&volumeName={volumeName} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
aspName	String	否	Query参数	自动快照策略名称
volumeName	String	否	Query参数	磁盘名称名称

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
autoSnapshotPolicys	List<AutoSnapshotPolicyModel>	返回的自动快照策略列表

错误码

无特殊返回错误码

请求示例

```
GET /v2/asp?marker=asp-oG6prjrl&maxKeys=2 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "asp-33333",
    "marker": "asp-oG6prjrl",
    "maxKeys": 2,
    "autoSnapshotPolicys": [
        {
            "id": "asp-oG6prjrl",
            "name": "snapshot-name",
            "timePoints": [20,21],
            "repeatWeekdays": [0, 6],
            "retentionDays": 7,
            "status": "active",
            "createdTime": "2015-07-08T10:22:04Z",
            "updatedTime": "2015-07-09T10:22:04Z",
            "deletedTime": "1971-01-01T00:00:01Z",
            "lastExecuteTime": "2015-07-09T10:22:04Z",
            "volumeCount": 1
        },
        {
            "id": "asp-22222",
            "name": "snapshot-name1",
            "timePoints": [23],
            "repeatWeekdays": [5],
            "retentionDays": 7,
            "status": "active",
            "createdTime": "2015-07-08T10:22:04Z",
            "updatedTime": "1971-01-01T00:00:01Z",
            "deletedTime": "1971-01-01T00:00:01Z",
            "lastExecuteTime": "1971-01-01T00:00:01Z",
            "volumeCount": 0
        }
    ],
    "isTruncated": false
}
```

查询自动快照策略详情

该接口用于查询指定自动快照策略详情。

请求结构

```
 GET /v{version}/asp/{aspId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
aspId	String	是	URL参数	自动快照策略ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
autoSnapshotPolicy	AutoSnapshotPolicyModel	返回的自动快照策略

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
NoSuchObject	The specified object is not found or resource do not exist.	404	资源不存在

请求示例

```
GET /v2/asp/asp-123 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "autoSnapshotPolicy": {
        "id": "asp-123",
        "name": "snapshot-name",
        "timePoints": [20,21],
        "repeatWeekdays": [0, 6],
        "retentionDays": 7,
        "status": "active",
        "createdTime": "2015-07-08T10:22:04Z",
        "updatedTime": "2015-07-09T10:22:04Z",
        "deletedTime": "1971-01-01T00:00:01Z",
        "lastExecuteTime": "2015-07-09T10:22:04Z",
        "volumeCount": 1
    }
}
```

自动快照策略变更

该接口用于自动快照策略的信息修改，支持修改内容：策略名称、创建时间、重复日期、保留时间。

请求结构

```
 PUT /v{version}/asp/update?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name":name,
    "timePoints":timePoints,
    "repeatWeekdays":repeatWeekdays,
    "retentionDays":retentionDays,
    "aspId":aspId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
name	String	是	RequestBody参数	自动快照策略名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
timePoints	List<String>	是	RequestBody参数	一天中做快照时间点，取值为0~23。
repeatWeekdays	List<String>	是	RequestBody参数	一周中做快照的时间，取值为0~6。
retentionDays	String	是	RequestBody参数	自动快照保留天数，取-1则永久保留。
aspId	String	是	RequestBody参数	自动策略的id

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	Bad request parameters or illegal request.	400	请求参数无效
Asp.NameInvalid	asp name is invalid.	400	名称非法
Asp.RetentionDaysInvalid	RetentionDays should in [1,10000]	400	保留天数非法
Asp.TimePointsInvalid	TimePoints should in [0,23]	400	执行时间点参数异常
Asp.RepeatWeekdaysInvalid	RepeatWeekdays should in [1,7]	400	执行日期参数异常

请求示例

```
 PUT /v2/asp/update?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "name":"asp-num1",
    "timePoints":["10","20"],
    "repeatWeekdays":["1","5"],
    "retentionDays":"7",
    "aspId":"asp-0yykRGvU"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

安全组相关接口

创建安全组

该接口用于创建安全组以及安全组规则。

- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId唯一性索引，重复记录会报409错误。
- protocol的取值（tcp|udp|icmp），默认值为空，代表all。

请求结构

```
 POST /v{version}/securityGroup?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
name	String	是	RequestBody参数	创建的安全组的名字,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
vpcId	String	否	RequestBody参数	安全组所属的vpcId。
desc	String	否	RequestBody参数	对所创建的安全组的描述信息
rules	List<SecurityGroupRuleModel>	是	RequestBody参数	创建安全组时绑定的安全组规则列表
tags	List<Tag>	否	RequestBody参数	创建安全组时绑定的标签列表

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
securityGroupId	String	已创建的安全组的ID

错误码

错误码	错误描述	HTTP状态码	中文解释
SecurityGroup.RuleRemarkTooLong	Security group description is too long.	400	安全组规则备注字段超限
BadRequest	protocol is invalid.	400	协议类型错误
SecurityGroup.RulePortOrderInvalid	The order of security group rule port is incorrect.	400	安全组规则端口顺序不正确
SecurityGroup.RulePortRangeInvalid	The value of security group rule port is exceeded.	400	安全组规则的端口数字超出允许范围
SecurityGroup.RulePortInvalid	The value of security group rule port must be number.	400	安全组规则端口非数字
SecurityGroup.RuleCIDRAddressError	Security group rule ip(cidr) address is incorrect.	400	安全组规则源（目标）地址错误
SecurityGroup.RuleDirectionError	Security group rule direction is incorrect.	400	安全组规则direction不符合
SecurityGroup.RuleNumberExceededLimit	Security groups that contain rule number exceed limit.	413	安全组所包含规则数目超限
SecurityGroup.RuleDuplicated	Security group rule is duplicated.	409	安全组规则重复

请求示例

```
 POST /v2/securityGroup?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
 "name": "test-security-group",
 "desc": "create a security group for api doc",
 "vpcId": "vpc-uxjret8rnv",
 "rules": [
  {
   "remark": "备注",
   "protocol": "tcp",
   "portRange": "1-65535",
   "direction": "ingress",
   "sourceIp": "",
   "sourceGroupId": ""
  },
  {
   "remark": "全部协议",
   "protocol": "",
   "portRange": "",
   "direction": "egress",
   "destIp": "",
   "destGroupId": ""
  }
 ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "securityGroupId": "g-nky7qeom"
}
```

查询安全组列表

- 该接口用于查询用户所有的安全组信息。

请求结构

```
 GET /v{version}/securityGroup?marker={marker}&maxKeys={maxKeys}&instanceId={instanceId}&vpcId={vpcId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
instanceId	String	否	Query参数	实例id，可查询实例关联的安全组列表。如需查询所有创建的安全组信息，不填写此参数。
vpcId	String	否	Query参数	vpc实例id，可查询实例关联的安全组列表。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
securityGroups	List<SecurityGroupModel>	安全组信息，由SecurityGroupModel组成的集合

请求示例

```
 GET /v2/securityGroup?instanceId=i-kjuinbcd HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "",
    "marker": "",
    "maxKeys": 1000,
    "securityGroups": [
        {
        "desc": "",
        "id": "g-4NxWoxeq",
        "name": "common2",
        "vpcId":"vpc-9xuevtmc6u",
        "createdTime":"2019-09-24T08:25:59Z",
        "sgVersion": 0,
        "rules": [
            {
                "destGroupId": "",
                "destIp": "all",
                "direction": "egress",
                "ethertype": "IPv4",
                "portRange": "1-65535",
                "protocol": "all",
                "remark": "bae",
                "securityGroupId": "g-4NxWoxeq",
                "securityGroupRuleId": "r-gkv8yupumvx2",
                "createdTime": "2020-07-27T13:00:52Z",
                "updatedTime": "2020-07-27T13:00:52Z"
            }
        ],
        "tags":[
          {
           "tagKey": tagKey,
           "tagValue": tagValue
          }
        ]
        }
    ],
    "isTruncated": false
}
```

授权安全组规则

- 该接口用于在安全组中授权新的安全组规则。
- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp | destIp、sourceGroupId | destGroupId六元组作为唯一性索引，若安全组中存在相同的规则，则会报409错误。

请求结构

```
 PUT /v{version}/securityGroup/{securityGroupId}?{action}&clientToken={clientToken}&sgVersion={sgVersion} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
 "rule":
  {
   "remark": "备注",
   "protocol": protocol,
   "portRange": portRange,
   "direction": direction,
   "sourceIp": sourceIp,
   "sourceGroupId": sourceGroupId
  }
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
sgVersion	long	否	Query参数	安全组版本号
action	String	是	Query参数	对安全组执行的动作，当前取值authorizeRule
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见ClientToken幂等性。
securityGroupId	String	是	URL参数	授权新安全组规则的安全组id
rule	SecurityGroupRuleModel	是	RequestBody参数	待授权安全组规则

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	protocol is invalid.	400	协议类型错误
SecurityGroup.RulePortOrderInvalid	The order of security group rule port is incorrect.	400	安全组规则端口顺序不正确
SecurityGroup.RulePortRangeInvalid	The value of security group rule port is exceeded.	400	安全组规则的端口数字超出允许范围
SecurityGroup.RulePortInvalid	The value of security group rule port must be number.	400	安全组规则端口非数字
SecurityGroup.RuleCIDRAddressError	Security group rule ip(cidr) address is incorrect.	400	安全组规则源（目标）地址错误
SecurityGroup.RuleDirectionError	Security group rule direction is incorrect.	400	安全组规则direction不符合
NoSuchObject	The specified object is not found or resource do not exist.	404	目标安全组不存在
SecurityGroup.RuleNumberExceededLimit	Security groups that contain rule number exceed limit.	413	安全组所包含规则数目超限
SecurityGroup.RuleDuplicated	Security group rule is duplicated.	409	安全组规则重复
VersionMismatch	The sgVersion is mismatch.	400	指定安全组规则版本号和当前最新版本不一致。若请求中没有携带sgVersion参数，则不会返回该错误码。

请求示例

```
 PUT /v2/securityGroup/g-nky7qeom?authorizeRule&clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
 "rule":
  {
   "remark": "备注",
   "protocol": "tcp",
   "portRange": "1-65535",
   "direction": "ingress",
   "sourceIp": "",
   "sourceGroupId": ""
  }
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

撤销安全组规则

- 该接口用于撤销安全组中的规则。

- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引。若安全组中不存在对应的规则，将报404错误。

请求结构

```
 PUT /v{version}/securityGroup/{securityGroupId}?{action}&clientToken={clientToken}&sgVersion={sgVersion} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
 "rule":
  {
   "remark": "备注",
   "protocol": protocol,
   "portRange": portRange,
   "direction": direction,
   "sourceIp": sourceIp,
   "sourceGroupId": sourceGroupId
  }
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
sgVersion	long	否	Query参数	安全组版本号，可从安全组列表获取
action	String	是	Query参数	对安全组执行的动作，当前取值revokeRule
securityGroupId	String	是	URL参数	待撤销安全组规则的安全组id
rule	SecurityGroupRuleModel	是	RequestBody参数	待撤销的安全组规则

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
SecurityGroup.RulePortOrderInvalid	The order of security group rule port is incorrect.	400	安全组规则端口顺序不正确
SecurityGroup.RulePortRangeInvalid	The value of security group rule port is exceeded.	400	安全组规则的端口数字超出允许范围
SecurityGroup.RulePortInvalid	The value of security group rule port must be number.	400	安全组规则端口非数字
NoSuchObject	The specified object is not found or resource do not exist.	404	目标安全组不存在
SecurityGroup.RuleNotExist	The Security group rule is not exist.	404	撤销的安全组规则不存在
VersionMismatch	The sgVersion is mismatch.	400	指定安全组规则版本号和当前最新版本不一致。若请求中没有携带sgVersion参数，则不会返回该错误码。

请求示例

```
 PUT /v2/securityGroup/g-nky7qeom?revokeRule&clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
 "rule":
  {
   "remark": "备注",
   "protocol": "tcp",
   "portRange": "1-65535",
   "direction": "ingress",
   "sourceIp": "",
   "sourceGroupId": ""
  }
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除安全组

该接口用于删除用户指定安全组。

请求结构

```
 DELETE /v{version}/securityGroup/{securityGroupId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
securityGroupId	String	是	URL参数	要删除的安全组ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
NoSuchObject	The specified object is not found or resource do not exist.	404	目标安全组不存在
SecurityGroup.InstancesAssociatedSecurityGroupCanNotBeDeleted	Security group can't be deleted while instance associated.	409	存在实例关联的安全组无法删除

请求示例

```
DELETE /v2/securityGroup/g-nky7qeom HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除安全组规则

该接口用于删除指定的安全组规则。

请求结构

```
DELETE /v{version}/securityGroup/rule/{securityGroupRuleId}?sgVersion={sgVersion} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
sgVersion	long	否	Query参数	安全组版本号
securityGroupRuleId	String	是	URL参数	要删除的安全组规则ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
NoSuchObject	The specified object is not found or resource do not exist.	404	目标安全组规则不存在
VersionMismatch	The sgVersion is mismatch.	400	指定安全组规则版本号和当前最新版本不一致。若请求中没有携带sgVersion参数，则不会返回该错误码。

请求示例

```
 DELETE /v2/securityGroup/rule/r-6eh9yhp8ugw3 HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

更新安全组规则

该接口用于更新安全组规则信息。

请求结构

```
 PUT /v{version}/securityGroup/rule/update?sgVersion={sgVersion}
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "securityGroupRuleId":"r-q1ek3jvwuede",
    "sourceIp":"3.3.3.9",
    "remark":"test"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
sgVersion	long	否	Query参数	安全组版本号
securityGroupRuleId	String	是	RequestBody参数	安全组规则id
remark	String	否	RequestBody参数	描述
portRange	String	否	RequestBody参数	端口范围
sourceIp	String	否	RequestBody参数	入站规则Ip，与sourceGroupId不可同时存在
sourceGroupId	String	否	RequestBody参数	入站规则安全组id，与sourceIp不可同时存在
destIp	String	否	RequestBody参数	出站规则Ip，与destGroupId不可同时存在
destGroupId	String	否	RequestBody参数	出站规则安全组id，与destIp不可同时存在
protocol	String	否	RequestBody参数	协议，暂不支持tcp/udp转icmp

返回头域

除公共头域，无其它特殊头域。

返回参数

无返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
VersionMismatch	The sgVersion is mismatch.	400	指定安全组规则版本号和当前最新版本不一致。若请求中没有携带sgVersion参数，则不会返回该错误码。
NoSuchObject	Resource not exist.	404	安全组规则不存在
InvalidProtocolException	Invalid protocol	400	协议错误，不支持转icmp
SecurityGroup.RuleCIDRAddressError	Security group rule ip(cidr) address is incorrect.	400	安全组规则源（目标）地址错误
SecurityGroup.RuleRemarkTooLong	Security group description is too long	400	描述超过长度限制
SecurityGroup.RulePortInvalid	The value of security group rule port must be number	400	安全组规则的端口数字超出允许范围
BadRequest	protocol is invalid.	400	协议类型错误

请求示例

```
 PUT /v2/securityGroup/rule/update
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2017-03-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{"remark":"test"}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Thu, 16 Mar 2017 06:29:48 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

实例关联安全组

接口说明

实例关联安全组

请求结构

```
PUT /v{version}/securitygroup/bind?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "securityGroupType" : "securityGroupType",
  "instanceIds" : ["instanceIds"],
  "securityGroupIds" : [ "securityGroupIds"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	参数类型	是否必须	参数位置	描述
instanceIds	**List<String>**	是	RequestBody	待绑定的虚机的短id列表
securityGroupIds	**List<String>**	是	RequestBody	待绑定的安全组的短id列表
securityGroupType	**String**	是	RequestBody	待绑定的安全组类型（enterprise/normal）

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	参数类型	描述
requestId	**String**	请求Id

请求示例

```
PUT /v2/securitygroup/bind HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceIds":["i-LO8XGmwh"],
    "securityGroupIds":["esg-ub0q1p72pbmh"],
    "securityGroupType":"enterprise"
}
```

响应示例

```
HTTP/1.1 200 OK
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "requestId" : "00edf3b9-6271-4fef-bf80-645c873fef3b"
}
```

实例解绑安全组

接口说明

实例解绑安全组

请求结构

```
PUT /v{version}/securitygroup/unbind?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "securityGroupType" : "securityGroupType",
  "instanceIds" : ["instanceIds"],
  "securityGroupIds" : [ "securityGroupIds"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	参数类型	是否必须	参数位置	描述
instanceIds	**List<String>**	是	RequestBody	待解绑的虚机的短id列表
securityGroupIds	**List<String>**	是	RequestBody	待解绑的安全组的短id列表
securityGroupType	**String**	是	RequestBody	待解绑的安全组类型（enterprise/normal）

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	参数类型	描述
requestId	**String**	请求Id

请求示例

```
PUT /v2/securitygroup/unbind HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds":["i-LO8XGmwh"],
    "securityGroupIds":["esg-ub0q1p72pbmh"],
    "securityGroupType":"enterprise"
}
```

响应示例

```
HTTP/1.1 200 OK
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
   "requestId": "c96efc10-9f0b-4500-9954-302033f33ce5"
}
```

实例替换安全组

接口说明

实例替换安全组

请求结构

```
PUT /v{version}/securitygroup/replace?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "securityGroupType" : "securityGroupType",
  "instanceIds" : ["instanceIds"],
  "securityGroupIds" : [ "securityGroupIds"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	参数类型	是否必须	参数位置	描述
instanceIds	List	是	RequestBody	待替换的虚机的短id列表
securityGroupType	String	是	RequestBody	目标安全组类型（企业安全组：enterprise，普通安全组：normal）
securityGroupIds	List	是	RequestBody	目标安全组的短id列表

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	参数类型	描述
requestId	**String**	请求Id

请求示例

```
PUT /v2/securitygroup/replace HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceIds":["i-LO8XGmwh","i-IeK97Gff"],
    "securityGroupIds":["esg-r8dz9yhm57kk"],
    "securityGroupType":"enterprise"
}
```

响应示例

```
HTTP/1.1 200 OK
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
  "requestId": "9142e4e7-6e21-42bb-9494-6a60fe321e09"
}
```

抢占实例相关接口

创建抢占实例

本接口用于创建抢占实例。

请求结构

```
POST /v{version}/instance/bid HTTP/1.1
Host: bcc.bj.baidubce.com
```

```
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "spec": "spec",
    "specId": "specId",
    "ephemeralDisks": [
        {
            "storageType": storageType,
            "sizeInGB": size
        },
        {
            "storageType": storageType,
            "sizeInGB": size
        }
    ],
    "adminPass": "adminPass",
    "aspId": "aspId",
    "instanceType": instanceType,
    "cpuCount": cpuCoreCount,
    "bidModel":bidModel,
    "bidPrice":bidPrice,
    "memoryCapacityInGB": memoryCapacityInGB,
    "rootDiskSizeInGb":rootDiskSizeInGb,
    "rootDiskStorageType":"rootDiskStorageType",
    "createCdsList": [
        {
            "storageType": storageType,
            "cdsSizeInGB": cdsSizeInGB,
            "snapshotId": "snapshotId"
        }
    ],
    "userData": "userData",
    "purchaseNum": purchaseNum
    "isomerismCard": "isomerismCard",
    "name": "name",
    "hostname": "hostname",
    "autoSeqSuffix": autoSeqSuffix,
    "isOpenHostnameDomain": isOpenHostnameDomain,
    "imageId": "imageId",
    "networkCapacityInMbps": networkCapacityInMbps,
    "eipName": "eipName",
    "internetChargeType": internetChargeType,
    "purchaseCount": purchaseCount,
    "zoneName": zoneName,
    "subnetId": subnetId,
    "securityGroupId": securityGroupId,
    "deletionProtection": "deletionProtection",
    "enterpriseSecurityGroupId": "enterpriseSecurityGroupId",
    "relationTag": relationTag,
    "isOpenIpv6": isOpenIpv6,
    "tags": [
      {
        "tagKey": tagKey,
        "tagValue": tagValue
      }
    ],
    "keypairId": keypairId,
    "fileSystems": {
        "fsId": "fsId",
        "mountAds": "mountAds",
        "path": "path",
        "protocol": "protocol"
```

```
    },
    "resGroupId": "resGroupId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
imageId	String	是	RequestBody参数	待创建虚拟机实例的镜像ID，可通过调用查询镜像列表接口选择获取所需镜像ID。
instanceType	String	否	RequestBody参数	待创建虚拟机实例的类型，具体可选类型参见下述InstanceType,为空时使用默认虚机类型。
cpuCount	int	是	RequestBody参数	待创建虚拟机实例的CPU核数，可选配置请参考区域机型以及可选配置。
bidModel	String	是	RequestBody参数	抢占实例出价模型， 市场价: "market" 自定义："custom"。参考BidModel
bidPrice	String	否	RequestBody参数	抢占实例出价金额，单位：元/分钟。若是自定义出价，且出价金额小于市场价，则不允许创建。当bidModel='custom'时才有效。
memoryCapacityInGB	int	是	RequestBody参数	待创建虚拟机实例的内存容量，单位GB，可选配置请参考区域机型以及可选配置。
rootDiskSizeInGb	int	否	RequestBody参数	待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
rootDiskStorageType	String	否	RequestBody参数	待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘，可指定系统盘磁盘类型可参见StorageType。
createCdsList	List<CreateCdsModel>	否	RequestBody参数	待创建的CDS磁盘列表，具体数据格式参见下述CreateCdsModel
networkCapacityInMbps	int	否	RequestBody参数	公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
internetCharge	String	否	RequestBody参	公网带宽计费方式，可选参数详见internetChargeType，若不指定internetChargeType，默

Type			参数	认为按使用带宽计费。
purchaseCount	int	否	RequestBody参数	批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
name	String	否	RequestBody参数	虚拟机名字（可选）。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
hostname	String	否	RequestBody参数	虚拟机主机名（可选）。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。仅支持小写字母、数字以及- . 特殊字符，必须以字母开头，不可连续使用特殊符号，不支持特殊符号开头或结尾，长度2-64。
autoSeqSuffix	boolean	否	RequestBody参数	是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
isOpenHostnameDomain	boolean	否	RequestBody参数	是否自动生成hostname domain（可选参数） 是:true 否:false
adminPass	String	否	RequestBody参数	待指定的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()，密码需要加密传输，详见链接
zoneName	String	否	RequestBody参数	指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。专属实例使用专属服务器所在zone,无需指定该字段。
subnetId	String	否	RequestBody参数	指定subnet信息，为空时将使用默认子网
securityGroupId	String	否	RequestBody参数	指定securityGroup信息，为空时将使用默认安全组
relationTag	boolean	否	RequestBody参数	待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
isOpenIpv6	boolean	否	RequestBody参数	待创建实例是否开启ipv6，只有当镜像和子网都支持ipv6时才可开启，true表示开启，false表示关闭，不传表示自动适配镜像和子网的ipv6支持情况
tags	List<TagModel>	否	RequestBody参数	待创建的标签列表，具体数据格式参见下述TagModel
keypairId	String	否	RequestBody参数	待创建实例所要绑定的密钥对ID

aspId	String	否	RequestBody参数	自动快照策略ID
specId	String	否	RequestBody参数	规格族
spec	String	否	RequestBody参数	实例规格
ephemeralDisks	List<EphemeralDisk>	否	RequestBody参数	DCC实例可以创建多块本地盘，需要指定磁盘类型以及大小。其他类型BCC最多只能使用一块本地盘，使用默认磁盘类型，需要指定磁盘大小。FPGA实例以及GPU实例默认使用一块本地磁盘，根据配置指定本地盘大小，具体请参考GPU型BCC可选规格配置 以及FPGA型BCC可选规格配置 。
adminPass	String	否	RequestBody参数	待指定的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()
userData	String	否	RequestBody参数	若实例满足使用实例自定义数据的限制，您可以Base64的方式编码后再传入UserData信息。因为传输API请求时，不会加密您设置的UserData，建议不要以明文方式传入机密的信息，例如密码和私钥等；如果必须传入，建议加密后在实例内部以同样的方式反解密。
isomerismCard	String	否	RequestBody参数	待创建实例所要携带的异构卡信息，具体可选信息参照，非GPU型实例无需指定此字段
fileSystems	List<FileSystemModel>	否	RequestBody参数	挂载文件存储CFS，具体参数见FileSystemModel
enterpriseSecurityGroupId	String	否	RequestBody参数	指定企业安全组
deletionProtection	int	否	RequestBody参数	实例释放保护，默认0未开启，1开启，开启后禁止手动释放虚机
eipName	String	否	RequestBody参数	长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
resGroupId	String	否	RequestBody参数	指定实例要绑定的资源组id

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
instanceIds	List<String>	虚机实例ID的集合，其中ID符合BCE规范，必须是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.InsufficientRootDiskSize	Not enough root disk size.	400	使用镜像过大，系统盘大小不够
Instance.InvalidateRootDiskStorageType	Invalidate root disk storage type.	400	无效的系统盘介质
Instance.InstanceBeBanned	The instance is banned, contact technical support for assistance.	403	Bcc服务已被封禁
Instance.GlobalInstanceQuotaLimitExceeded	Quota exceeds limit for regional postpaid instances pool, contact technical support for further assistance.	413	后付费类型的虚机数目超出当前区域预售额度限制
Instance.InstanceQuotaLimitExceeded	Quota exceeds limit.	413	实例个数超过限制
Instance.EipQuotaLimitExceeded	The number of eip will exceed the limit.	413	合并购买时eip个数超过限制
Instance.InvalidateConfiguration	Invalidate cpu memory configuration.	400	无效的虚机配置
BadRequest	bid price should be greater than market price.	400	抢占实例自定义出价应大于市场价
BadRequest	invalid purchaseNum.	400	购买数量需要大于0
BadRequest	missing bcc config.	400	缺少bcc配置
BadRequest	invalid instanceType.	400	无效的实例类型
BadRequest	bidPrice can not be null when bidModel is custom.	400	自定义出价时bidPrice不能为空
BadRequest	bidModel is invalid.	400	bidModel的值无效
BadRequest	bidPrice is invalid.	400	bidPrice的值无效
Instance.PermissionDeny	Instance permission deny. permission deny	401	无权限
RealNameQualify	No real name authentication	409	不可操作,未实名认证
Instance.RegionDenyException	You're not allowed to access this region.	403	region拒绝访问异常.
BadRequest	Not support serviceType.	400	不支持该服务类型
BadRequest	cds config error.	400	cds配置错误
Instance.InstanceCountCdsCountNotMatchException	Cds count must be multiple number of Instance count	400	创建的虚机数量和cds数量不匹配
Instance.InvalidateZoneException	Invalidate zone name.	400	无效的可用区
Instance.NotSupportInstanceTypeException	not support the instance type.	403	不支持该机型
BadRequest	Not support serviceType	400	不支持该服务类型
BadRequest	invalid deploy id	400	无效的部署集ID
BadRequest	invalid cds count	400	cds 数量应该是bcc的整数倍
BadRequest	cannot get market price	400	不能够获得市场价

请求示例

```
 POST /v2/instance/bid HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceType": "N1",
    "cpuCount": 2,
    "bidModel": "custom",
    "bidPrice": 0.01,
    "memoryCapacityInGB": 2,
    "rootDiskSizeInGb":45,
    "rootDiskStorageType":"cloud_hp1",
    "createCdsList": [
        {
            "storageType": "std1",
            "cdsSizeInGB": 5
        }
    ],
    "name": "example",
    "imageId": "m-mRjO3cjs",
    "networkCapacityInMbps": 1,
    "internetChargeType": "BANDWIDTH_POSTPAID_BY_HOUR",
    "purchaseCount": 1,
    "relationTag": true,
        "tags": [
            {
                "tagKey":"tag1",
                "tagValue":"var1"
            }
    ],
    "keypairId":"k-R0J0BEA9"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "orderId": "d1ace996973643b299ea506a80dae4da",
    "instanceIds": [
        "i-T1I3OtUO"
    ]
}
```

查询抢占实例套餐

本接口用于查询抢占实例的套餐规格。

请求结构

```
 POST /v{version}/instance/bidFlavor HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

无请求参数

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
zoneResources	List<ZoneResource>	各个可用区的套餐规格，具体数据格式参见下述ZoneResource

错误码

错误码	错误描述	HTTP状态码	中文解释
Instance.NotSupportInstanceTypeException	not support the instance type.	403	不支持该机型

请求示例

```
 POST /v2/instance/bidFlavor HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "zoneResources": [
    {
      "zoneName": "cn-bj-a",
      "bccResources": [
        {
          "instanceType": "N1",
          "flavors": [
            {
              "specId": "c1",
              "cpuCount": 1,
              "memoryCapacityInGB": 2,
              "productType": "bidding",
              "spec": "bcc.c1.c1m2"
            }
          ]
        }
      ]
    }
  ]
}
```

查询抢占实例市场价

本接口用于查询抢占实例市场价。

请求结构

```
 POST /v{version}/instance/bidPrice HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "instanceType": instanceType,
    "cpuCount": cpuCoreCount,
    "memoryCapacityInGB": memoryCapacityInGB,
    "rootDiskSizeInGb":rootDiskSizeInGb,
    "rootDiskStorageType":rootDiskStorageType,
    "createCdsList": [
        {
            "storageType": storageType,
            "cdsSizeInGB": cdsSizeInGB
        }
    ],
    "purchaseCount":purchaseCount
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceType	String	是	RequestBody参数	待创建虚拟机实例的类型，具体可选类型参见InstanceType。
cpuCount	int	是	RequestBody参数	待创建虚拟机实例的CPU核数，可选配置请参考区域机型以及可选配置。
memoryCapacityInGB	int	是	RequestBody参数	待创建虚拟机实例的内存容量，单位GB，可选配置请参考区域机型以及可选配置。
rootDiskSizeInGb	int	否	RequestBody参数	待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
rootDiskStorageType	String	否	RequestBody参数	待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘，可指定系统盘磁盘类型可参见StorageType。
createCdsList	List<CreateCdsModel>	否	RequestBody参数	待创建的CDS磁盘列表，具体数据格式参见下述CreateCdsModel
purchaseCount	int	否	RequestBody参数	批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
networkCapacityInMbps	int	否	RequestBody参数	公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
zoneName	String	否	RequestBody参数	指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。专属实例使用专属服务器所在zone,无需指定该字段。
internetChargeType	String	否	RequestBody参数	公网带宽计费方式，可选参数详见internetChargeType，若不指定internetChargeType，默认付费方式同BCC，预付费默认为包年包月按带宽，后付费默认为按使用带宽计费。

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
money	String	总价。
count	String	购买个数。
perMoney	String	单个价格。

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	invalid purchaseNum.	400	购买数量需要大于0
BadRequest	missing bcc config.	400	缺少bcc配置
BadRequest	invalid instanceType.	400	无效的实例类型
Instance.InvalidateZoneException	Invalidate zone name.	400	无效的可用区

请求示例

```
 POST /v2/instance/bidPrice HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceType": "N1",
    "cpuCount": 2,
    "memoryCapacityInGB": 2,
    "rootDiskSizeInGb":45,
    "rootDiskStorageType":"cloud_hp1",
    "createCdsList": [
        {
            "storageType": "std1",
            "cdsSizeInGB": 5
        }
    ],
    "purchaseCount": 1
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "money": "0.0011367",
    "count": "1",
    "perMoney": "0.0011367"
}
```

取消抢占实例订单

本接口用于取消抢占实例订单。

请求结构

```
 POST /v{version}/instance/cancelBidOrder HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "orderId": orderId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
orderId	String	是	RequestBody参数	订单id

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	订单Id。

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	cancelBidOrder can not is null.	400	订单id不能为空
BadRequest	query order is null.	400	订单id无效，请求结果为空

请求示例

```
 POST /v2/instance/cancelBidOrder HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "orderId": orderId
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "orderId": "fcae766149954067a0b2ae4f1d1a092d"
}
```

密钥对相关接口

创建密钥对

该接口用于创建密钥对，单个用户在一个地域默认配额500个（创建和导入的密钥对之和）。创建后的密钥对可植入BCC实例中以实现远程登录BCC实例。

请求结构

```
 POST /v{version}/keypair HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name": "keypair01",
    "description": "some description"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
name	String	是	RequestBody	待创建的密钥对名称，只支持大小写字母、数字、汉字以及-_ /.特殊字符，必须以字母开头，长度1-65。
description	String	否	RequestBody	待创建的密钥对的描述

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
keypair	KeypairModel	创建的密钥对对象

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNameInvalid	Keypair name is invalid.	400	密钥对名称格式不正确
Keypair.KeypairDescriptionInvalid	Keypair description is invalid.	400	密钥对描述格式不正确
Keypair.KeypairQuotaExceedLimit	Keypair number exceeds the limit.	413	密钥对数目超限

请求示例

```
 POST /v2/keypair HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "name": "keypair01",
    "description": "some description"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "keypair":
        {
            "keypairId": "k-y7VWq",
            "name": "keypair01",
            "description": "someDescription",
            "createTime": "2018-01-27T13:23:13Z",
            "regionId": "bj",
            "publicKey": "publicKey-publicKey"
            "privateKey":"privateKey-privateKye"
        }
}
```

导入密钥对

该接口用于用户自行导入创建密钥对。 请求参数中导入的密钥对仅支持以下加密方式： - rsa - dsa - ssh-rsa - ssh-dss - ecdsa

请求结构

```
 PUT /v{version}/keypair?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "name": "keypair01",
    "description": "some description",
    "publicKey": "your-pulicKey"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对密钥对执行的操作，当前取值：import
name	String	是	RequestBody	待创建的密钥对名称，只支持大小写字母、数字、汉字以及-_ /.特殊字符，必须以字母开头，长度1-65。
description	String	否	RequestBody	待创建的密钥对的描述
publicKey	String	是	RequestBody	公钥内容

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
keypair	KeypairModel	密钥对对象

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNameInvalid	Keypair name is invalid.	400	密钥对名称格式不正确
Keypair.KeypairDescriptionInvalid	Keypair description is invalid.	400	密钥对描述格式不正确
Keypair.KeypairQuotaExceedLimit	Keypair number exceeds the limit.	413	密钥对数目超限
Keypair.KeypairPublicKeyExist	Keypair already exist.	400	公钥已存在
Keypair.PublicKeyFormatInvalid	Public key format is invalid.	400	公钥格式不正确

请求示例

```
 POST /v2/keypair?import HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "name": "keypair01",
    "description": "some description",
    "publicKey": "your-publicKey"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "keypair":{
        "instanceCount":0,
        "name":"keypair01",
        "description":"from import",
        "createdTime":"2019-01-30T11:52:41Z",
        "publicKey":"publicKey",
        "fingerPrint":"fingerPrint",
        "regionId":"bj",
        "keypairId":"keypairId"
    }
}
```

绑定密钥对

该接口用于将指定密钥对（限单个）绑定到指定1个或多个BCC实例上。

- 目前一台BCC实例只能绑定一个密钥对，若操作的BCC实例已经绑定密钥对，则此操作将替换该BCC实例原有的密钥对。此操作仅适用于linux系统的BCC实例，且所选BCC实例必须处于运行中或已关机状态。

请求结构

```
 PUT /v{version}/keypair/{keypairId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceIds" : [instanceId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
keypairId	String	是	URL参数	待绑定的密钥对的id
action	String	是	Query参数	对密钥对执行的操作，当前取值：attach
instanceIds	List<String>	是	RequestBody参数	待绑定的虚机的短id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

无

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.UnsupportOsType	Unsupported os type, only linux supported.	403	不支持的操作系统，仅支持linux操作系统
Keypair.InstanceStatusForbidden	Instance not in running or stopped status.	409	仅能操作运行中或关机状态的虚机
Keypair.AttachAttachedKeypair	Can not attach to an attached keypair.	400	不能绑定已绑定的密钥对
Keypair.InstanceNotExist	Instance not exist.	404	虚机不存在
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在

请求示例

```
 PUT /v2/keypair/k-YufwpQAe?attach HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
   "instanceIds": ["i-ZeVPsPzv"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

查询密钥对列表

该接口用于查询密钥对列表。

请求结构

```
 GET /v{version}/keypair?marker={marker}&maxKeys={maxKeys} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
keypairs	List<KeypairModel>	密钥对结果集

错误码

无特殊返回错误码

请求示例

```
 GET /v2/keypair?marker=k-lyWRtII7&maxKeys=1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "nextMarker": "k-xktdeMSf",
    "marker": "k-lyWRtII7",
    "maxKeys": 1,
    "isTruncated": true,
    "keypairs": [
        {
            "keypairId": "k-lyWRtII7",
            "createTime": "2018-01-27T13:23:13Z",
            "name": "instance-j93wzbn1",
            "regionId":"bj",
            "instanceCount":3
        }
    ]
}
```

查询密钥对详情

该接口用于查询指定单个密钥对的详细信息。

请求结构

```
GET /v{version}/keypair/{keypairId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
keypairId	String	是	URL参数	待查询的密钥对ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
keypair	KeypairModel	返回的密钥对详情

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在

请求示例

```
GET /v2/keypair/i-YufwpQAe HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2018-01-
27T13:23:13Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "keypair":
    {
        "keypairId": "keypairId",
        "createTime": "2018-01-27T13:23:13Z",
        "name": "keypair-name",
        "description": "console",
        "instanceCount":3,
        "regionId":"bj",
        "publicKey": "publicKey"
    }
}
```

更改密钥对描述

该接口用于更改密钥对描述，若用户提供的新的描述内容为空，则删除所操作密钥对的描述。

请求结构

```
 PUT /v{version}/keypair/{keypairId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "description" : newDescription
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
action	String	是	Query参数	对密钥对执行的操作，当前取值：updateDesc
keypairId	String	是	URL参数	待更改描述的密钥对id
description	String	否	RequestBody参数	新的密钥对描述

返回头域

除公共头域，无其它特殊头域。

返回参数

无

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在
Keypair.PublicKeyFormatInvalid	Public key format is invalid.	400	密钥对描述非法

请求示例

```
 PUT /v2/keypair/k-h4NDJPhY?updateDesc HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "description": "new description"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

重命名密钥对

该接口用于重命名密钥对。

请求结构

```
 PUT /v{version}/keypair/{keypairId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name" : keypairName
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
keypairId	String	是	URL参数	待重命名的密钥对id
action	String	是	Query参数	对密钥对执行的操作，当前取值：rename
name	String	是	RequestBody	新的密钥对名称，只支持大小写字母、数字、汉字以及-_ /.特殊字符，必须以字母开头，长度1-65。

返回头域

除公共头域，无其它特殊头域。

返回参数

无

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在
Keypair.KeypairNameInvalid	Keypair name is invalid.	400	密钥对名称非法

请求示例

```
 PUT /v2/keypair/k-h4NDJPhY?rename HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "name": "newName"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

解绑密钥对

该接口用于将所选BCC实例与它们各自绑定的密钥对解绑。

- 目前一台BCC实例只能绑定一个密钥对，此操作将使所选BCC实例与其通过百度智能云控制台操作所绑定的密钥对解绑，若的BCC实例未绑定该密钥对，则不进行处理。

- 此操作仅适用于linux系统的BCC实例，且所选BCC实例必须处于运行中或已关机状态。
- 注：
  - 1）用户所选BCC实例可能并未绑定任何密钥对，则此操作对该BCC实例无任何作用；
  - 2）用户可能对所选BCC实例手动绑定了密钥对，则此操作对其手动绑定的密钥对不产生任何影响；
  - 3）用户如若此前通过百度智能云控制台操作，为所选BCC实例绑定过密钥对，且该密钥对状态正常，则此操作将从该BCC实例中删除该密钥对

请求结构

```
 PUT /v{version}/keypair/{keypairId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
   "instanceIds" : [instanceId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
keypairId	String	是	URL参数	待解绑的密钥对id
action	String	是	Query参数	对密钥对执行的操作，当前取值：detach
instanceIds	List<String>	是	RequestBody参数	待解绑的实例id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

无

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.UnsupportOsType	Unsupported os type, only linux supported.	403	不支持的操作系统，仅支持linux操作系统
Keypair.InstanceStatusForbidden	Instance not in running or stopped status.	409	仅能操作运行中或已关机状态的实例
Keypair.AttachAttachedKeypair	Can not attach to an attached keypair.	400	不能绑定已绑定的密钥对
Keypair.InstanceNotExist	Instance not exist.	404	实例不存在
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在

请求示例

```
 PUT /v2/keypair/k-h4NDJPhY?detach HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "instanceIds" : ["i-ePEyBGdM"]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除密钥对

该接口用于删除密钥对，已经绑定BCC实例的密钥对无法被删除。

请求结构

```
 DELETE /v{version}/keypair/{keypairId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
keypairId	String	是	URL参数	待删除的密钥对id

返回头域

除公共头域，无其它特殊头域。

返回参数

无

错误码

错误码	错误描述	HTTP状态码	中文解释
Keypair.KeypairNotExist	Keypair not exist.	404	密钥对不存在
Keypair.DeleteKeypairInUse	Keypair is in use.	403	已绑定的密钥对不能被删除

请求示例

```
 DELETE /v2/keypair/k-h4NDJPhY HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

部署集相关接口

创建部署集

该接口用于创建部署集。

请求结构

```
 POST /v{version}/instance/deployset/create?clientToken={clientToken}
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name" : "testname",
    "desc" : "test desc",
    "strategy": "HOST_HA"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
name	string	否	RequestBody参数	部署集名字
desc	string	否	RequestBody参数	部署集描述
concurrency	int	否	RequestBody参数	部署集并发度
strategy	string	否	RequestBody参数	部署集策略（HOST_HA:宿主机,RACK_HA:机架,TOR_HA:交换机）

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
deploySetIds	List<String>	部署集Ids

请求示例

```
 POST /v2/instance/deployset/create HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: authorization-string
{
    "name" : "testname",
    "desc" : "test desc",
    "strategy": "HOST_HA"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "deploySetIds":["dset-L******"]
}
```

查询部署集列表

该接口用于查询用户的部署集列表。

请求结构

```
 GET /v{version}/instance/deployset/list
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数 无

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
deploySets	deploySetModel	部署集列表

请求示例

```
 GET /v2/instance/deployset/list HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
"deploySets":[
   {
      "instanceCount":"2",
      "name":"bcc_tor",
      "strategy":"TOR_HA",
      "concurrency":1,
      "deploysetId":"dset-ciLqX0UY",
      "desc":"bcc-tor",
      "azIntstanceStatisList":[
         {
            "instanceTotal":5,
            "instanceCount":2,
            "bccInstanceCnt":1,
            "bbcInstanceCnt":1,
            "zoneName":"cn-bj-a"
         },
         {
            "instanceTotal":10,
            "instanceCount":0,
            "bccInstanceCnt":0,
            "bbcInstanceCnt":0,
            "zoneName":"cn-bj-b"
         }
      ]
   }
 ]
}
```

修改部署集

该接口用于修改部署集信息。

请求结构

```
 PUT /v{version}/instance/deployset/{deployId}?{action}
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
   "name": "name",
   "desc": "desc"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
deployId	String	是	URL参数	部署集id
action	String	是	Query参数	对部署集执行的操作，当前取值：modifyAttribute
name	string	否	RequestBody参数	部署集名字（name与desc至少传一个）
desc	string	否	RequestBody参数	部署集描述（name与desc至少传一个）

返回头域

除公共头域，无其它特殊头域。

返回参数

无

请求示例

```
 POST /v2/instance/deployset/dset-Ly4w*****?modifyAttribute HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: authorization-string
{
    "name" : "testname",
    "desc" : "test desc"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

删除部署集

该接口用于删除部署集。

请求结构

```
 DELETE /v{version}/instance/deployset/{deployId}
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数 无

返回头域

除公共头域，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
deployId	String	是	URL参数	部署集id

返回参数 无

请求示例

```
 DELETE /v2/instance/deployset/dset-Ly****** HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: authorization-string
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

部署集移出实例

本接口用于将部署集移出BCC实例。

请求结构

```
 POST /v{version}/instance/deployset/delRelation HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "deployId": "deployId",
    "instanceIdList": ["instanceId"]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
deployId	String	是	requestBody参数	部署集id
instanceIdList	List	是	requestBody参数	实例id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/deployset/delRelation
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "deployId": "dset-42N1llvY",
    "instanceIdList": ["i-123456"]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

调整部署集

该接口用于调整部署集。

请求结构

```
 POST /v{version}/instance/deployset/updateRelation HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceId":"instanceId",
    "deploysetIdList":[
        "deploysetId"
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
instanceId	String	是	requestBody参数	实例id
deploysetIdList	List	是	requestBody参数	要加入的部署集id列表

返回头域

除公共头域，无其它特殊头域。

请求示例

```
 POST /v2/instance/deployset/updateRelation HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "instanceId":"i-123456",
    "deploysetIdList":[
        "dset-42N1llvY",
        "dset-42N1llv1"
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

获取部署集详情

该接口用于获取部署集的详细信息。

请求结构

```
 GET /v{version}/deployset/{id} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
id	String	是	URL参数	部署集id

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
shortId	String	部署集短id
name	String	部署集名称
desc	String	部署集描述
strategy	String	部署集策略
concurrency	int	并发数
azIntstanceStatisList	List<AzIntstanceStatisDetail>	可用区实例统计列表

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	invalid id:	400	部署集ID为空
BadRequest	invalid deployset id	400	无效部署集ID

请求示例

```
 GET /v2/deployset/dset-tTHtVKX1 HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "shortId": "dset-tTHtVKX1",
    "name": "deploysetA",
    "desc": "deploysetA",
    "strategy": "HOST_HA",
    "concurrency": 1,
    "azIntstanceStatisList": []
}
```

专属集群相关接口

创建专属集群

该接口用于创建CDS专属集群。

- CDS专属集群仅支持包年包月（预付费），不支持按量付费（后付费）。预付费购买时长为6个月、1年、2年、3年、5年。
- 集群支持的最小容量为85TB，最大支持容量1015TB，购买步长为10TB。

- 创建后不可更改集群的地域和可用区。因此用户需要提前规划云资源所属的地域和可用区。
- CDS专属集群由于容量较大，供应链需要预留一定准备时间，若有需求可以联系客户经理。

请求结构

```
 POST /v{version}/volume/cluster HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "zoneName": zoneName,
    "clusterName": clusterName,
    "clusterSizeInGB": clusterSizeInGB,
    "storageType": storageType,
    "purchaseCount": purchaseCount,
    "billing": {
        "paymentTiming": paymentTiming,
        "reservation": {
            "reservationLength": reservation,
            "reservationTimeUnit": reservationTimeUnit
        }
    }
    "tags": [
    {
        "tagKey": tagkey,
        "tagValue": tagvalue
    }
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号。
zoneName	String	否	RequestBody参数	指定可用区信息，默认为空，由系统自动选择。
clusterName	String	是	RequestBody参数	创建CDS专属集群的名字。
clusterSizeInGB	String	是	RequestBody参数	集群总容量单位GB，最小容量85TB，最大容量1015TB，购买步长10TB。
storageType	int	是	RequestBody参数	待创建专属集群类型。
purchaseCount	int	否	RequestBody参数	创建CDS专属集群的数量，必须大于0的证书，缺省值为1
billing	Billing	是	RequestBody参数	订单、计费相关参数
tags	List<TagModel>	否	RequestBody参数	待创建的标签列表，具体数据格式参见TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
clusterIds	List	专属集群ID的集合，其中ID符合BCE规范，必须是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。

请求示例

```
 POST /v{version}/volume/cluster HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/******

{
"zoneName": "cn-bj-b",
"clusterName": "",
"clusterSizeInGB": 97280,
"storageType": "hp1",
"purchaseCount": 1,
"billing": {
    "paymentTiming": "Prepaid",
    "reservation": {
        "reservationLength": 6,
        "reservationTimeUnit": "Month"
    }
"tags": [
    {
        "tagKey": "key1",
        "tagValue": "value1"
    },
    {
        "tagKey": "key2",
        "tagValue": "value2"
    }
 ]

}
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Mar, 23 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "clusterIds": [
    "DC-1ns4****"
  ]
}
```

查询专属集群列表

该接口用于查询专属集群列表信息。

请求结构

```
GET /v{version}/volume/cluster HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
clusterName	String	否	Query参数	查询的专属集群名称
zoneName	String	否	Query参数	可用区信息

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页。
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现。
maxKeys	int	每页包含的最大数量
result	List<VolumeClusterModel>	返回的磁盘信息列表

请求示例

```
GET /v2/volume/cluster HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/*****
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b9ed9312-bddf-4d86-95cc-ae1f121fa787
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "result": [
        {
            "clusterId": "DC-aax07twB",
            "clusterName": "dbsc-5pEZ5oI2",
            "createdTime": "2022-08-18T11:44:14Z",
            "expiredTime": "2023-02-18T11:45:05Z",
            "status": "preparing",
            "logicalZone": "zoneB",
            "productType": "prepay",
            "clusterType": "hp1",
            "totalCapacity": 97280,
            "usedCapacity": 0,
            "availableCapacity": 97280,
            "expandingCapacity": 0,
            "createdVolumeNum": 0,
            "enableAutoRenew": false
        }
    ],
    "marker": "",
    "isTruncated": false,
    "nextMarker": null,
    "maxKeys": 1000
}
```

查询专属集群详情

该接口用于根据专属集群ID查询单个专属集群的详细信息。

请求结构

```
 GET /v{version}/volume/cluster/{clusterId} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	URL参数	专属集群ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
clusterId	String	专属集群ID
clusterName	String	专属集群名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65
createdTime	String	创建日期，符合BCE日期规范
expiredTime	String	过期时间，符合BCE日期规范。
status	String	专属集群状态
logicalZone	String	专属集群类型
productType	String	专属集群付费类型
clusterType	String	专属集群类型
totalCapacity	int	专属集群总容量
usedCapacity	int	专属集群已使用容量
availableCapacity	int	专属集群可用容量
expandingCapacity	int	专属集群扩展容量
createdVolumeNum	int	由专属集群创建的CDS数量
affiliatedCDSNumber	List	由专属集群创建的CDS列表
enableAutoRenew	boolean	是否开启自动付费
tags	List<TagModel>	已绑定的标签列表

请求示例

```
 GET /v2/volume/cluster/DC-bsVqFe6j HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/*****
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 0cf8ea06-92be-47df-bd5f-34cab433a079
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "clusterId": "DC-bsVqFe6j",
    "clusterName": "dbsc",
    "createdTime": "2022-04-13T07:36:25Z",
    "expiredTime": "2022-10-13T07:36:59Z",
    "status": "preparing",
    "logicalZone": "zoneA",
    "productType": "prepay",
    "clusterType": "hdd",
    "totalCapacity": 97280,
    "usedCapacity": 0,
    "availableCapacity": 97280,
    "expandingCapacity": 0,
    "createdVolumeNum": 2,
    "affiliatedCDSNumber": [
        "v-MmbIhBqv",
        "v-siPBYIT2"
    ],
    "enableAutoRenew": false，
    "tags": [
          {
              "tagKey": "key1",
              "tagValue": "value1"
          }
    ]
}
```

变配专属集群

该接口用于为指定专属集群变配扩容，修改专属集群容量大小。

请求结构

```
 PUT /v{version}/volume/{clusterId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "newClusterSizeInGB" : newClusterSizeInGB,
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	URL参数	磁盘ID
action	String	是	Query参数	对专属集群执行的操作，当前取值：resize
newClusterSizeInGB	int	是	Request Body参数	新扩容专属集群容量大小，必须为大于当前专属集群容量的整数，单位为GB，大小为87040~1039360的正整数。

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Cluster.ResizeParamInvalid	cluster does not exist.	400	专属集群id不存在
Cluster.ClusterSizeInvalid	cluster newClusterSizeInGB error.	400	专属集群容量大小不合法

请求示例

```
  PUT /v2/volume/cluster/DC-g4MhBMut?resize HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/*****

{
  "newClusterSizeInGB" : 107520
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 864388da-ff1f-488b-be35-cfce5d9e762d
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

专属集群续费

该接口用于为指定专属集群续费，延长该专属集群到期时间。

请求结构

```
 PUT /v2/volume/cluster/{clusterId}?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "billing":{
      "reservation":{
         "reservationLength": reservationLength,
         "reservationTimeUnit": reservationTimeUnit
      }
   }
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	URL参数	磁盘ID
action	String	是	Query参数	对专属集群执行的操作，当前取值：purchaseReserved
billing	Billing	是	RequestBody参数	订单信息

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Cluster.OrderParamInvalid	request parameter cannot be empty.	400	请求参数不能为空
BadRequest	request parameter error.	400	请求参数无效

请求示例

```
 PUT /v2/volume/cluster/DC-g4MhBMut?purchaseReserved HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882****

{
 "billing":{
    "reservation":{
       "reservationLength":1,
       "reservationTimeUnit":"Month"
    }
 }
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: c24591e8-85f6-4e21-9099-0aa1614abc49
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

开通专属集群自动续费

该接口用于为指定专属集群开通自动续费。

请求结构

```
 POST /v2/volume/cluster/autoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "clusterId" : clusterId,
   "renewTimeUnit" : renewTimeUnit,
   "renewTime" : renewTime
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	RequestBody参数	专属集群ID
renewTimeUnit	String	是	RequestBody参数	按月付费或者按年付费 月是"month",年是"year"
renewTime	int	是	RequestBody参数	自动续费的时间 按月是1-9 按年是 1-3

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/volume/cluster/autoRenew  HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634d****

{
    "clusterId": "DC-g4MhBMut",
    "renewTimeUnit": "month",
    "renewTime": 6
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b859fd4b-56eb-4b79-a9ea-956d41d9b80f
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

取消专属集群自动续费

该接口用于为指定专属集群取消自动续费。

请求结构

```
 POST /v{version}/volume/cluster/cancelAutoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "clusterId" : clusterId,
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	RequestBody参数	专属集群ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/volume/cluster/cancelAutoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634d***
{
    "clusterId": "DC-g4MhBMut"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Mar, 21 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

专属集群解绑标签

该接口用于解绑专属集群已绑定的标签。

- 可以同时解绑多个标签。

请求结构

```
 PUT /v{version}/volume/{clusterId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
   "changeTags": [
     {
      "tagKey" : tagKey,
      "tagValue": tagValue
     }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	RequestBody参数	专属集群ID
action	String	是	Query参数	对磁盘执行的操作，当前取值：unbind
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT /v2/volume/cluster/DC-8KMWo6T7/tag?unbind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "changeTags": [
      {
       "tagKey" : "key1",
       "tagValue": "value1"
      }
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 121443a7-4ad5-451d-3415-71cb844c2350
Date: Wed, 09 Dec 2023 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

专属集群绑定标签

该接口用于给专属集群绑定标签，用于资源管理。

请求结构

```
 PUT /v{version}/volume/cluster/{clusterId}/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string

{
    "changeTags": [
      {
       "tagKey" : tagKey,
       "tagValue": tagValue
      }
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clusterId	String	是	RequestBody参数	专属集群ID
action	String	是	Query参数	对专属集群执行的操作，当前取值：bind
changeTags	List<TagModel>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见下述TagModel

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数。

请求示例

```
 PUT bcc.bj.baidubce.com/v2/volume/cluster/DC-8KMWo6T7/tag?bind HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "changeTags": [
      {
       "tagKey" : "key1",
       "tagValue": "value1"
      }
    ]
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 121432a7-4ad5-453d-9215-71cb234c0a50
Date: Wed, 09 Dec 2023 08:42:09 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

预留实例券相关接口

创建预留实例券

该接口用于创建预留实例券。

请求结构

```
 POST /v{version}/instance/reserved/create HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceName": reservedInstanceName,
    "scope": scope,
    "zoneName": zoneName,
    "spec":spec,
    "offeringType":offeringType,
    "instanceCount":instanceCount,
    "reservedInstanceCount":reservedInstanceCount,
    "reservedInstanceTime":reservedInstanceTime,
    "reservedInstanceTimeUnit":reservedInstanceTimeUnit,
    "autoRenewTimeUnit":autoRenewTimeUnit,
    "autoRenewTime":autoRenewTime,
    "autoRenew":autoRenew,
    "effectiveTime":effectiveTime,
    "tags": [
       "tagKey":tagKey,
       "tagValue":tagValue
    ],
    "ticketId":ticketId,
    "ehcClusterId":ehcClusterId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述

version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
reservedInstanceName	String	否	requestBody参数	规则：支持大小写字母、数字、中文以及-_ /.特殊字符，必须以中文或字母开头，长度1-65，可重复；客户未命名的情况下自动命名：reservedInstance-${随机生成}，同样遵循以上命名规则
scope	String	否	requestBody参数	实例券的可用范围；默认：AZ
zoneName	String	是	requestBody参数	可用区，例如：cn-bj-a
spec	String	是	requestBody参数	实例规格，例如：bcc.g4.c2m8
offeringType	String	是	requestBody参数	默认值全预付：FullyPrepay
instanceCount	String	否	requestBody参数	实例数量，默认为1，不可变更
reservedInstanceCount	String	否	requestBody参数	预留实例券数量，默认1，不能超过配额，默认配额20，若需要更多配额，请联系工单（可链接）申请。
reservedInstanceTime	String	是	requestBody参数	预留实例券购买时长，支持3，6，9，12，24，36个月
reservedInstanceTimeUnit	String	否	requestBody参数	预留实例券购买时长单位，默认为month，不可变更
autoRenew	Boolean	否	requestBody参数	自动续费开关，默认为false
autoRenewTimeUnit	String	否	requestBody参数	预留实例券自动续费时长单位,默认为month，不可变更
autoRenewTime	String	否	requestBody参数	预留实例券自动续费时长，支持3，6，9，12，24，36个月,autoRenew为true时，必选且必须与reservedInstanceTime一致
effectiveTime	String	否	requestBody参数	生效时间，默认立即生效；自定义生效时间，最长不能超过1天。且只能设置为15min为间隔的整点（比如：10:00、10:15、10:30、10:45、11:00）
tags	List<TagModel>	否	RequestBody参数	待创建的标签列表，具体数据格式参见下述TagModel

ticketId	String	否	requestBody参数	代金券id
ehcClusterId	String	否	requestBody参数	创建roce预留实例券时可选参数，若为空则使用默认EHC集群

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	创建预留实例券的订单号
reservedInstanceIds	Array	预留实例券的id集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/create
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceName": "test",
  "scope": "AZ",
  "zoneName": "cn-bj-a",
  "spec": "bcc.g4.c2m8",
  "offeringType": "FullyPrepay",
  "instanceCount": 1,
  "reservedInstanceCount": 1,
  "reservedInstanceTime": 12,
  "reservedInstanceTimeUnit": "month",
  "autoRenewTimeUnit": "month",
  "autoRenewTime": 12,
  "autoRenew": true,
  "effectiveTime": "2023-05-22T21:00:00Z",
  "tags": [
     {
        "tagKey": "test",
        "tagValue": "test"
     }
  ],
  "ticketId": ""
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "reservedInstanceIds": [
      "test_f0c549b9614f"
    ],
    "orderId": "orderId"
}
```

查询预留实例券

该接口用于查询预留实例券列表信息。

请求结构

```
 POST /v{version}/instance/reserved/list?marker={marker}&maxKeys={maxKeys} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [
      reservedInstanceId
    ],
    "reservedInstanceName": reservedInstanceName,
    "zoneName": zoneName,
    "reservedInstanceStatus":reservedInstanceStatus,
    "offeringType":offeringType,
    "spec":spec,
    "osType":osType,
    "sortKey":sortKey,
    "sortDir":sortDir
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
reservedInstanceIds	Array	否	requestBody参数	预留实例券的id集合
reservedInstanceName	String	否	requestBody参数	实例券的名称
zoneName	String	否	requestBody参数	可用区，例如：cn-bj-a
reservedInstanceStatus	String	否	requestBody参数	预留实例券的状态
spec	String	否	requestBody参数	实例规格，例如：bcc.g4.c2m8
offeringType	String	否	requestBody参数	FullyPrepay：全预付
osType	String	否	requestBody参数	支持的镜像类型，all/linux/windows
instanceId	String	否	requestBody参数	抵扣实例ID
instandeName	String	否	requestBody参数	实例名称
isDeduct	Boolean	否	requestBody参数	是否有抵扣实例：true/false
ehcClusterId	String	否	requestBody参数	ehc集群id
sortKey	String	否	requestBody参数	排序字段：osType/instanceCount/effectiveTime/expireTime
sortDir	String	否	requestBody参数	排序方式：desc/asc

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
totalCount	Integer	预留实例券的总数量
marker	String	标记查询的起始位置
maxKeys	Integer	每页包含的最大数量
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
isTruncated	bool	true表示后面还有数据，false表示已经是最后一页
reservedInstances	List<ReservedInstanceInfo>	预留实例券信息，由 ReservedInstanceInfo 组成的集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/list
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceName": "test",
  "zoneName": "cn-bj-a",
  "spec": "bcc.g4.c2m8",
  "offeringType": "FullyPrepay"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "totalCount": 1,
    "marker": null,
    "maxKeys": 1000,
    "nextMarker": null,
    "isTruncated": true,
    "reservedInstances": [
        {
            "autoRenew": false,
            "renewTimeUnit": null,
            "renewTime": null,
            "nextRenewTime": null,
            "reservedInstanceId": "r-xxcxcx",
            "reservedInstanceUuid": "dfadfadfadfafafada",
            "reservedInstanceName": "test",
            "scope": "AZ",
            "zoneName": "cn-bj-a",
            "spec": "bcc.g4.c2m8",
            "reservedType": "BCC",
            "offeringType": "FullyPrepay",
            "osType": "all",
            "reservedInstanceStatus": "Active",
            "instanceCount": 1,
            "effectiveTime": "2023-05-18T06:33:34Z",
            "expireTime": "2023-06-18T06:34:57Z",
            "tags": null
        }
    ]
}
```

调整预留实例券

该接口用于调整预留实例券。

- 不支持同时调整可用区和预留实例券名称。

请求结构

```
 PUT /v{version}/instance/reserved/modify HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "reservedInstances": [
    {
        "reservedInstanceId": "reservedInstanceId",
        "zoneName": "zoneName",
        "ehcClusterId": "ehcClusterId"
    }
  ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
reservedInstanceId	String	是	requestBody参数	要调整的预留实例券id
zoneName	String	否	requestBody参数	要调整的目标可用区，例如：cn-bj-b。不支持同时修改reservedInstanceName
reservedInstanceName	String	否	requestBody参数	要调整的实例券名称；规则：支持大小写字母、数字、中文以及-_ /.特殊字符，必须以中文或字母开头，长度1-65，可重复；客户未命名的情况下自动命名：reservedInstance-${随机生成}，同样遵循以上命名规则。不支持同时修改zoneName
ehcClusterId	String	否	requestBody参数	变更roce预留实例券时可选参数，若为空则使用默认EHC集群

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
modifyReservedInstanceOrders	Array	调整预留实例券的返回集合
reservedInstanceId	String	预留实例券的id
orderId	String	预留实例券变更的订单号

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 PUT http://bcc.bj.baidubce.com/v2/instance/reserved/modify
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstances": [
    {
        "reservedInstanceId": "r-yeeVFWyn",
        "zoneName": "cn-bj-b"
    }
  ]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS


{
    "modifyReservedInstanceOrders": [
      {
        "reservedInstanceId": "test_b0e3d7387d0b",
        "orderId": "test_5cf7663b10de"
      }
    ]
}
```

续费预留实例券

该接口用于续费预留实例券。

请求结构

```
 POST /v{version}/instance/reserved/renew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [
       reservedInstanceId
    ],
    "reservedInstanceTime": reservedInstanceTime,
    "reservedInstanceTimeUnit": reservedInstanceTimeUnit,
    "autoRenewTimeUnit":autoRenewTimeUnit,
    "autoRenewTime":autoRenewTime,
    "autoRenew":autoRenew
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
reservedInstanceIds	Array	是	requestBody参数	实例券id集合
reservedInstanceTime	String	是	requestBody参数	预留实例券续费时长，支持3，6，9，12，24，36个月，必须与购买时长保持一致
reservedInstanceTimeUnit	String	否	requestBody参数	预留实例券购买时长单位，默认为month，不可变更
autoRenew	Boolean	否	requestBody参数	自动续费开关，默认为false
autoRenewTimeUnit	String	否	requestBody参数	预留实例券自动续费时长单位，默认为month，不可变更
autoRenewTime	String	否	requestBody参数	预留实例券自动续费时长，支持3，6，9，12，24，36个月，autoRenew为true时，必选且必须与reservedInstanceTime一致

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
orderId	String	创建预留实例券的订单号
reservedInstanceIds	Array	预留实例券的id集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/renew
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": [
      "r-test"
  ],
  "reservedInstanceTime": 12,
  "reservedInstanceTimeUnit": "month",
  "autoRenewTimeUnit": "month",
  "autoRenewTime": 12,
  "autoRenew": true
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2023 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "reservedInstanceIds": [
      "test_f0c549b9614f"
    ],
    "orderId": "orderId"
}
```

预留实例券转出列表

该接口用于查询预留实例券转出列表。

请求结构

```
 POST /v{version}/instance/reserved/transfer/out/list HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [reservedInstanceId],
    "transferRecordIds": [transferRecordId],
    "spec": spec,
    "status": status
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
reservedInstanceIds	List	否	requestBody参数	通过预留实例券id列表过滤
transferRecordIds	List	否	requestBody参数	通过预留实例券转移记录id列表过滤
spec	String	否	requestBody参数	通过实例规格过滤
status	String	否	requestBody参数	通过转移记录状态过滤

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
totalCount	int	预留实例券转移记录总数
transferRecords	List<TransferOutRecord>	预留实例券转移记录详情

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/out/list
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": ["r-OQJhJY5Z"],
  "transferRecordIds": ["t-EbYIxxgG"],
  "spec": "bcc.g3.c1m1",
  "status": "timeout"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
"totalCount": 1,
"transferRecords": [{
 "transferRecordId": "t-EbYIxxgG",
 "recipientUserId": "xxxx**********************yyyy",
 "status": "timeout",
 "reservedInstanceInfo": {
  "reservedInstanceId": "r-OQJhJY5Z",
  "reservedInstanceName": "reservedInstance-qGRTVDmq",
  "scope": "AZ",
  "zoneName": "cn-bj-a",
  "spec": "bcc.g3.c1m1",
  "reservedType": "BCC",
  "offeringType": "PostPay",
  "osType": "all",
  "reservedInstanceStatus": "Active",
  "instanceCount": 1,
  "effectiveTime": "2024-03-22T10:07:04Z",
  "expireTime": "2024-04-22T10:18:26Z",
  "autoRenew": false,
  "renewTimeUnit": "",
  "renewTime": 0,
  "nextRenewTime": ""
 },
 "applicationTime": "2024-03-26T07:30:48Z",
 "expireTime": "2024-03-26T08:30:48Z",
 "endTime": "2024-03-26T08:30:48Z"
}]
}
```

拒绝预留实例券转移

该接口用于拒绝预留实例券转移（仅适用转入列表中的转移记录）。

请求结构

```
 POST /v{version}/instance/reserved/transfer/refuse HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "transferRecordIds": [transferRecordId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
transferRecordIds	List	是	requestBody参数	要拒绝的预留实例券转移记录id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
-	void	无返回

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/refuse
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "transferRecordIds": ["t-tgQYk4Rx"]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

发起预留实例券转移

该接口用于发起预留实例券转移。

请求结构

```
 POST /v{version}/instance/reserved/transfer/create HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [reservedInstanceId],
    "recipientAccountId": recipientAccountId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
reservedInstanceIds	List	是	requestBody参数	要转移的预留实例券id列表
recipientAccountId	String	是	requestBody参数	接收人的账号id

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
-	List	预留实例券转移记录id列表

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/create
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": ["r-yeeVFWyn"],
  "recipientAccountId": "recipientAccountId"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

["t-tgQYk4Rx"]
```

接受预留实例券转移

该接口用于接受预留实例券转移（仅适用转入列表中的转移记录）。

请求结构

```
 POST /v{version}/instance/reserved/transfer/accept HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "transferRecordId": transferRecordId,
    "ehcClusterId": ehcClusterId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
transferRecordId	String	是	requestBody参数	要接受的预留实例券转移记录id
ehcClusterId	String	否	requestBody参数	接收roce预留实例券时可选参数，若为空则使用默认EHC集群

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
-	void	无返回

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/accept
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "transferRecordId": "t-tgQYk4Rx"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

预留实例券转入列表

该接口用于查询预留实例券转入列表。

请求结构

```
 POST /v{version}/instance/reserved/transfer/in/list HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [reservedInstanceId],
    "transferRecordIds": [transferRecordId],
    "spec": spec,
    "status": status
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
reservedInstanceIds	List	否	requestBody参数	通过预留实例券id列表过滤
transferRecordIds	List	否	requestBody参数	通过预留实例券转移记录id列表过滤
spec	String	否	requestBody参数	通过实例规格过滤
status	String	否	requestBody参数	通过转移记录状态过滤

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
totalCount	int	预留实例券转移记录总数
transferRecords	List<TransferInRecord>	预留实例券转移记录详情

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/in/list
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": ["r-OQJhJY5Z"],
  "transferRecordIds": ["t-EbYIxxgG"],
  "spec": "bcc.g3.c1m1",
  "status": "timeout"
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
"totalCount": 1,
"transferRecords": [{
 "transferRecordId": "t-EbYIxxgG",
 "grantorUserId": "xxxx**********yyyy",
 "status": "timeout",
 "reservedInstanceInfo": {
  "reservedInstanceId": "r-OQJhJY5Z",
  "reservedInstanceName": "reservedInstance-qGRTVDmq",
  "scope": "AZ",
  "zoneName": "cn-bj-a",
  "spec": "bcc.g3.c1m1",
  "reservedType": "BCC",
  "offeringType": "PostPay",
  "osType": "all",
  "reservedInstanceStatus": "Active",
  "instanceCount": 1,
  "effectiveTime": "2024-03-22T10:07:04Z",
  "expireTime": "2024-04-22T10:18:26Z",
  "autoRenew": false,
  "renewTimeUnit": "",
  "renewTime": 0,
  "nextRenewTime": ""
 },
 "applicationTime": "2024-03-26T07:30:48Z",
 "expireTime": "2024-03-26T08:30:48Z",
 "endTime": "2024-03-26T08:30:48Z"
}]
}
```

撤销预留实例券转移

该接口用于撤销预留实例券转移(仅适用转出列表中的转移记录)。

请求结构

```
 POST /v{version}/instance/reserved/transfer/revoke HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "transferRecordIds": [transferRecordId]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见链接。
transferRecordIds	List	是	requestBody参数	要撤销的预留实例券id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
-	void	无返回

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/transfer/revoke
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "transferRecordIds": ["t-tgQYk4Rx"]
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

预留实例券绑定标签

该接口用于给预留实例券绑定标签，支持同时给多个预留实例券绑定多个标签，每次请求预留实例券最多100个。 **请求结构**

```
 PUT /v{version}/bcc/reserved/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
// 预留实例券ID列表，单次最多100个
"reservedInstanceIds": ["r-HrztSVk0", "r-9lUk0lnl"],
"changeTags": [

   {
      "tagKey":"openapi-1",
      "tagValue":"test"
   },
   {
      "tagKey":"openapi-2",
      "tagValue":"test"
   }
]
```

}

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
action	String	是	Query参数	对实例执行的操作，当前取值：bind。
reservedInstanceIds	List<String>	是	RequestBody参数	预留实例券ID列表，单次最多100个。
changeTags	List<TagModel>	是	RequestBody参数	待绑定的标签列表，具体数据格式参见TagModel。

返回头域 除公共头域外，无其它特殊头域。

返回参数

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/bcc/reserved/tag?bind
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
   // 预留实例券ID列表，单次最多100个
   "reservedInstanceIds": ["r-HrztSVk0", "r-9lUk0lnI"],
   "changeTags": [
      {
         "tagKey":"key1",
         "tagValue":"value1"
      },
      {
         "tagKey":"key2",
         "tagValue":"value2"
      }
   ]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

预留实例券解绑标签

该接口用于解绑预留实例券已绑定的标签，可以同时解绑多个预留实例券的多个标签，每次请求预留实例券最多100个。

请求结构

```
 PUT /v{version}/bcc/reserved/tag?{action} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
   // 预留实例券ID列表，单次最多100个
   "reservedInstanceIds": ["r-HrztSVk0", "r-9lUk0lnI"],
   "changeTags": [
      {
         "tagKey":"key1",
         "tagValue":"value1"
      },
      {
         "tagKey":"key2",
         "tagValue":"value2"
      }
   ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
action	String	是	Query参数	对实例执行的操作，当前取值：unbind 。
reservedInstanceIds	List<String>	是	RequestBody参数	预留实例券ID列表，单次最多100个。
changeTags	List<Tag>	是	RequestBody参数	待解绑的标签列表，具体数据格式参见Tag。

返回头域

除公共头域外，无其它特殊头域。

返回参数 无特殊返回参数。

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/bcc/reserved/tag?unbind
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
   // 预留实例券ID列表，单次最多100个
   "reservedInstanceIds": ["r-HrztSVk0", "r-9lUk0lnI"],
   "changeTags": [
      {
         "tagKey":"key1",
         "tagValue":"value1"
      },
      {
         "tagKey":"key2",
         "tagValue":"value2"
      }
   ]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
```

预留实例券关闭自动续费规则

该接口用于关闭自动续费预留实例券。

请求结构

```
 POST /v{version}/instance/reserved/cancelAutoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [
      reservedInstanceId
    ]
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
reservedInstanceIds	List	是	requestBody参数	实例券id集合

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
reservedInstanceIds	Array	预留实例券的id集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
NoSuchObject	Auto-renewal failed: reservedInstanceIds not fount: [reservedInstanceId]	404	[reservedInstanceId]不存在

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/cancelAutoRenew
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": [
     "r-xxx"
  ],
  "autoRenewTimeUnit": "month",
  "autoRenewTime": 24
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2023 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "reservedInstanceIds": [
       "r-xxx"
    ]
}
```

预留实例券创建自动续费规则

该接口用于开通自动续费预留实例券。

- 若您的资源即将到期（到期时间小于24小时），建议您先进行手动续费。
- 开通自动续费后，系统会在到期日前7天自动为其续费。
- 自动续费将按照您最近一次设置的续费周期执行。

请求结构

```
 POST /v{version}/instance/reserved/autoRenew HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "reservedInstanceIds": [
        reservedInstanceId
    ],
    "autoRenewTimeUnit":autoRenewTimeUnit,
    "autoRenewTime":autoRenewTime
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
reservedInstanceIds	Array	是	requestBody参数	实例券id集合
autoRenewTimeUnit	String	否	requestBody参数	预留实例券自动续费时长单位，默认为month，不可变更
autoRenewTime	String	否	requestBody参数	预留实例券自动续费时长，支持3，6，9，12，24，36个月，必须与购买时长一致（该时长可通过预留实例券列表reservedInstanceTime字段）

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
reservedInstanceIds	Array	预留实例券的id集合

错误码

错误码	错误描述	HTTP状态码	中文解释
BadRequest	request is invalid	400	请求参数无效
NoSuchObject	Auto-renewal failed: reservedInstanceIds not fount: [reservedInstanceId]	400	[reservedInstanceId]不存在

请求示例

```
 POST http://bcc.bj.baidubce.com/v2/instance/reserved/autoRenew
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
  "reservedInstanceIds": [
      "r-xxx"
  ],
  "autoRenewTimeUnit": "month",
  "autoRenewTime": 24
}
```

返回示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2023 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS

{
    "reservedInstanceIds": [
        "r-xxx"
    ]
}
```

维修平台相关接口

维修任务相关接口

维修平台为全局服务，API的服务域名为：bbc.bj.baidubce.com

查询维修任务列表

查询所有未关闭的维修任务

请求结构

```
GET /v{version}/task?marker={marker}&maxKeys={maxKeys}&startTime={startTime}&endTime={endTime} HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
instanceId	String	否	Query参数	故障实例的ID
internalIp	String	否	Query参数	故障实例的内网IP
status	String	否	Request Body参数	任务状态
timeRangeType	String	否	Request Body参数	时间节点类型，当前支持类型：errStart、newTask、authorizeTask、unAuthorizeTask
timeRangeStart	String	否	Request Body参数	timeRangeType类型开始时间，例如： 2021-06-29 19:00:00。若为空则只按照timeRangeEnd限制
timeRangeEnd	String	否	Request Body参数	timeRangeType类型结束时间，例如： 2021-06-29 19:00:00。若为空则只按照timeRangeStart限制

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
RepairTask	List<TaskModel>	维修任务信息，由 TaskModel 组成的集合

错误码

无特殊返回错误码

请求示例

```
GET /v2/task?marker=task-lyWRtII7&maxKeys=1&startTime=2021-06-29 19:00:00&endTime=2021-07-07 10:26:35
HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "task-xktdeMSf",
    "marker": "task-lyWRtII7",
    "maxKeys": 1,
    "isTruncated": true,
    "RepairTask": [
        {
            "taskId": "task-lyWRtII7",
            "instanceId": "i-j93wzbn1",
            "internalIp": "192.168.1.1",
            "status": "processing",
            "errResult": "test error"
        }
    ]
}
```

维修任务详情

查询维修任务详情

请求结构

```
GET /v{version}/task/{taskId} HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	URL参数	维修任务ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
RepairTaskDetail	RepairTaskDetail	维修任务详细信息

错误码

无特殊返回错误码

请求示例

```
 GET /v2/task/task-tv9xpdWi HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "status": "processing",
    "end": "end",
    "floatingIp": "",
    "instanceId": "i-1XeaPYPQ",
    "region": "su",
    "internalIp": "",
    "serverStatus": "repairing",
    "taskId": "task-L3IKPZCI",
    "errResult": "test error",
    "instanceName": "MS_PQ"
}
```

授权维修任务

用户授权对维修任务进行维修

请求结构

```
 POST /v{version}/task/authorize HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "taskId": "taskId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	Request Body参数	维修任务ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Rp.IllegalArguments	Bad request parameters or illegal request.	400	请求参数无效
Rp.UnexpectedStatus	Task current status cannot be authorized.	409	任务当前状态不可授权

请求示例

```
 POST /v2/task/authorize HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "taskId": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

暂不处理维修任务

用户暂不处理维修任务，后续还可以再授权维修

请求结构

```
 POST /v{version}/task/unauthorize HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "taskId": "taskId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	Request Body参数	维修任务ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Rp.IllegalArguments	Bad request parameters or illegal request.	400	请求参数无效
Rp.UnexpectedStatus	Task current status cannot be unauthorized	409	任务当前状态不可暂不处理

请求示例

```
 POST /v2/task/unauthorize HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "taskId": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

确认故障恢复

确认故障已修复，维修任务关闭

请求结构

```
 POST /v{version}/task/confirm HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
   "taskId": "taskId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	Request Body参数	维修任务ID

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Rp.IllegalArguments	Bad request parameters or illegal request.	400	请求参数无效
Rp.UnexpectedStatus	Task current status cannot be confirmed	409	任务当前状态不可确认故障已修复

请求示例

```
 POST /v2/task/confirm HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "taskId": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

确认故障未恢复

维修任务完成后，用户可以确认故障是否修复。如果确认故障未修复，用户可以填入新的故障名称，维修任务重新开始处理

请求结构

```
 POST /v{version}/task/disconfirm HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "taskId": "taskId",
    "newErrResult":"newErrResult"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	Request Body参数	维修任务ID
newErrResult	String	是	Request Body参数	新的故障名称，可自定义故障问题

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

错误码	错误描述	HTTP状态码	中文解释
Rp.IllegalArguments	Bad request parameters or illegal request.	400	请求参数无效
Rp.UnexpectedStatus	Task current status cannot be disconfirmed	409	任务当前状态不可确认故障未修复

请求示例

```
 POST /v2/task/disconfirm HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
   "taskId": "example",
 "newErrResult": "test error"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

任务操作记录

任务的操作记录

请求结构

```
 POST /v{version}/task/record HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
   "taskId": "taskId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
taskId	String	是	Request Body参数	维修任务ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
RepairRecord	OperationRecord	维修任务操作记录

错误码

错误码	错误描述	HTTP状态码	中文解释
Rp.IllegalArguments	Bad request parameters or illegal request.	400	请求参数无效

请求示例

```
 POST /v2/task/record HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "taskId": "example"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "RepairRecord": [
    {
      "operator": "192.168.1.1",
      "operateTime": "2020-07-29T14:11:52Z",
      "name": "预授权维修任务"
    },
    {
      "operator": "系统",
      "operateTime": "2020-07-29T14:11:52Z",
      "name": "新建维修任务"
    }
  ]
}
```

预授权相关接口

维修平台为全局服务 API的服务域名为：bbc.bj.baidubce.com

查询预授权规则列表

查询用户的预授权规则列表

请求结构

```
 GET /v{version}/rule?marker={marker}&maxKeys=1&ruleId={ruleId}&ruleName={ruleName} HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
ruleId	String	否	Query参数	预授权规则 ID
ruleName	String	否	Request Body参数	预授权规则名称

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
RuleList	List<RuleModel>	预授权规则信息，由 RuleModel 组成的集合

错误码

无特殊返回错误码

请求示例

```
 GET /v2/rule?marker=rule-uKxDO5cr&maxKeys=1 HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 {
   "marker": "",
   "isTruncated": false,
   "maxKeys": 1000,
   "RuleList": [
      {
         "ruleId": "rule-uKxDO5cr",
         "ruleName": "q",
         "tagCount": 1,
         "associateBbcNum": 13,
         "errorBbcNum": 0,
         "limit": 10000,
         "status": "disabled"
      }
   ]
}

# 维修任务详情
```

预授权规则详情

请求结构

```
GET /v{version}/rule/{ruleId} HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ruleId	String	是	URL参数	预授权规则ID

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
RuleDetail	RuleDetail	预授权规则详细信息

错误码

无特殊返回错误码

请求示例

```
GET /v2/rule/rule-uKxDO5cr HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "ruleId": "rule-uKxDO5cr",
    "ruleName": "q",
    "tagCount": 1,
    "tags": [
        {
            "tagKey": "默认项目",
            "tagValue": ""
        }
    ],
    "associateBbcNum": 13,
    "errorBbcNum": 0,
    "limit": 10000,
    "status": "disabled"
}
```

创建预授权规则

创建预授权规则

请求结构

```
 POST /v{version}/rule/create HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "ruleName":"ruleName",
    "limit":limit,
    "enabled":enabled
    "tagStr":"tagStr"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ruleName	String	是	Request Body参数	预授权规则名称
limit	int	是	Request Body参数	预授权规则上限
enabled	int	否	Request Body参数	是否开启规则，0：不开启，1：开启。默认为0
tagStr	String	是	Request Body参数	关联的tag，tagKey:tagValue的格式

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
ruleId	String	预授权规则id

错误码

无特殊返回错误码

请求示例

```
 POST /v2/rule/create HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "ruleName":"test-rule",
    "limit":10,
    "tagStr":"tag1:value1"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "ruleId": "rule-xxx"
}
```

删除预授权规则

删除预授权规则，只能是被禁用的规则

请求结构

```
 POST /v{version}/rule/delete HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "ruleId":"ruleId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ruleId	String	是	Request Body参数	预授权规则id

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

无特殊返回错误码

请求示例

```
 PUT /v2/rule/delete HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "ruleId":"test-rule"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

启用预授权规则

启用预授权规则，只能是被禁用的规则

请求结构

```
 POST /v{version}/rule/enable HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "ruleId":"ruleId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ruleId	String	是	Request Body参数	预授权规则id

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

无特殊返回错误码

请求示例

```
 PUT /v2/rule/enable HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "ruleId":"test-rule"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

禁用预授权规则

禁用预授权规则，只能是被启用的规则

请求结构

```
 POST /v{version}/rule/disable HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string

{
    "ruleId":"ruleId"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ruleId	String	是	Request Body参数	预授权规则id

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

错误码

无特殊返回错误码

请求示例

```
 PUT /v2/rule/disable HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "ruleId":"test-rule"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
```

维修记录相关接口

维修平台为全局服务 API的服务域名为：bbc.bj.baidubce.com

维修记录（已关闭的维修任务）

查询维修记录（已关闭的维修任务）

请求结构

```
 GET /v{version}/closedTask?marker={marker}&maxKeys={maxKeys}&startTime={startTime}&endTime={endTime}
HTTP/1.1
Host: bbc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
marker	String	否	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串
maxKeys	int	否	Query参数	每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
instanceId	String	否	Query参数	故障实例的ID
serviceType	String	否	Query参数	故障实例的类型
taskId	String	否	Query参数	维修任务ID
timeRangeType	String	否	Request Body参数	时间节点类型，当前支持类型：errStart、newTask、authorizeTask、unAuthorizeTask
timeRangeStart	String	否	Request Body参数	timeRangeType类型开始时间，例如： 2021-06-29 19:00:00。若为空则只按照timeRangeEnd限制
timeRangeEnd	String	否	Request Body参数	timeRangeType类型结束时间，例如： 2021-06-29 19:00:00。若为空则只按照timeRangeStart限制

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	int	每页包含的最大数量
RepairTask	List<ClosedTaskModel>	维修任务信息，由 TaskModel 组成的集合

错误码

无特殊返回错误码

请求示例

```
 GET /v2/closedTask?marker=task-tv9xpdWi&maxKeys=1&startTime=2021-06-29 19:00:00&endTime=2021-07-07 10:26:35 HTTP/1.1
Host: bbc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "nextMarker": "task-xktdeMSf",
    "marker": "task-tv9xpdWi",
    "maxKeys": 1,
    "isTruncated": true,
    "RepairTask": [
     {
         "instanceId": "i-V541twDm",
       "errResult": "test error",
          "createTime": "2020-07-30T14:10:39Z",
       "taskId": "task-tv9xpdWi",
       "endTime": "2020-07-30T17:03:28Z"
     }
    ]
}
```

地域及可用区相关接口

获取域名列表接口

该接口用于获取指定地域或者全部地域的BCC(或CDS、预留实例券)域名列表。

统一使用全局域名bcc.baidubce.com

请求结构

```
 POST /v{version}/region/describeRegions?clientToken={clientToken} HTTP/1.1
Host: bcc.baidubce.com
Authorization: authorization string
{
    "region":region
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
clientToken	String	否	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串。
region	String	否	RequestBody参数	指定地域获取对应的域名，不传默认获取全部地域域名列表

返回头域

除公共头域，无其它特殊头域。

响应参数

名称	类型	描述
regions	List<RegionModel>	地域信息

请求示例(获取全部的域名列表)

```
 POST /v2/region/describeRegions?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "regions": [
        {
            "regionId": "bj",
            "regionName": "华北-北京",
            "regionEndpoint": "bcc.bj.baidubce.com"
        },{
            "regionId": "su",
            "regionName": "华东-苏州",
            "regionEndpoint": "bcc.su.baidubce.com"
        }
    ]
}
```

请求示例(获取指定region的域名列表)

```
 POST /v2/region/describeRegions?clientToken=be31b98c-5e41-4838-9830-9be700de5a20 HTTP/1.1
Host: bcc.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de

{
    "region": "bj"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS
{
    "regions": [
        {
            "regionId": "bj",
            "regionName": "华北-北京",
            "regionEndpoint": "bcc.bj.baidubce.com"
        }
    ]
}
```

查询可用区列表

该接口用于查询当前地域（region）下的可用区（zone）的列表信息。

请求结构

```
 GET /v{version}/zone HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
zones	List<ZoneModel>	可用区列表信息

请求示例

```
 GET /v2/zone HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "zones": [
        {
            "zoneName": "cn-bj-a"
        },
        {
            "zoneName": "cn-bj-b"
        }
    ]
}
```

其他接口

查询实例套餐价格

该接口用于查询实例规格对应的价格。

请求结构

```
 POST /v{version}/instance/price HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "specId": specId,
    "spec": spec,
    "paymentTiming": paymentTiming,
    "zoneName": zoneName,
    "purchaseCount": purchaseCount,
    "purchaseLength": purchaseLength
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
specId	String	是	RequestBody参数	实例规格族
spec	String	是	RequestBody参数	实例套餐规格
paymentTiming	String	是	RequestBody参数	付费方式，包括Postpaid(后付费)，Prepaid(预付费)两种。
zoneName	String	是	RequestBody参数	可用区名称
purchaseCount	int	否	RequestBody参数	查询在指定实例套餐规格下，任意数量实例的总价格，必须为大于0的整数，可选参数，缺省为1
purchaseLength	int	是	RequestBody参数	时长，[1,2,3,4,5,6,7,8,9,12,24,36]，单位：月

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
price	List<SpecIdPrices>	实例套餐规格对应价格信息

请求示例

```
 POST /v2/instance/price HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "purchaseCount": 1,
    "purchaseLength": 1,
    "specId": "lgn1",
    "paymentTiming": "Prepaid",
    "zoneName": "cn-bj-b"
}
或
{
    "purchaseCount": 1,
    "purchaseLength": 1,
    "spec": "bcc.lgn1.c12m40.1p40",
    "paymentTiming": "Prepaid",
    "zoneName": "cn-bj-b"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
  "price": [
    {
      "specId": "lgn1",
      "specPrices": [
        {
          "spec": "bcc.lgn1.c12m40.1p40",
          "specPrice": "4017.0",
          "tradePrice": "3213.6",
          "status": "available"
        }
      ]
    },
    {
      "specId": "lgn1",
      "specPrices": [
        {
          "spec": "bcc.lgn1.c24m80.2p40",
          "specPrice": "8034.0",
          "tradePrice": "6427.2",
          "status": "available"
        }
      ]
    },
    {
      "specId": "lgn1",
      "specPrices": [
        {
          "spec": "bcc.lgn1.c48m160.4p40",
          "specPrice": "16068.0",
          "tradePrice": "12854.4",
          "status": "available"
        }
      ]
    }
  ]
}
或
{
  "price": [
    {
      "specId": "lgn1",
      "specPrices": [
        {
          "spec": "bcc.lgn1.c12m40.1p40",
          "specPrice": "4017.0",
          "tradePrice": "3213.6",
          "status": "available"
        }
      ]
    }
  ]
}
```

查询实例套餐规格(新)

该接口用于查询实例规格列表信息。创建实例时建议使用参数(spec)指定需要的机型以及配置。

请求结构

```
GET /v{version}/instance/flavorSpec HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
zoneName	String	否	Query参数	可用区名称
specs	String	否	Query参数	待查询套餐规格列表，多个套餐规格用逗号分割，如：bcc.g4.c1m4,bcc.g5.c2m8
specIds	String	否	Query参数	待查询规格族列表，多个规格族用逗号分割，如：g4,g5

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
zoneResources	List<ZoneResourceDetailSpec>	各可用区下可用实例资源套餐规格列表信息

请求示例

```
GET /v2/instance/flavorSpec?specs=bcc.g5.c2m8,bcc.hgn5IB.c116m978.8A100-80g&zoneName=cn-bj-a HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

返回示例

```
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "zoneResources": [
    {
        "zoneName": "cn-bj-a",
        "ebcResources":
        {
            "customFlavors": [],
            "flavorGroups": []
        },
        "bccResources":
        {
            "flavorGroups": [
            {
                "groupId": "gpuHGN5",
                "flavors": [
                {
                    "specId": "hgn5IB",
                    "cpuCount": 116
```

```
            "cpuCount": 116,
            "memoryCapacityInGB": 978,
            "ephemeralDiskCount": 1,
            "gpuCardType": "nTeslaA100-80G",
            "gpuCardCount": 8,
            "cpuModel": "Intel Xeon Platinum 8350C",
            "cpuGHz": "2.6",
            "eniQuota": 8,
            "eriQuota": 0,
            "rdmaType": "IB",
            "rdmaNetCardCount": 4,
            "rdmaNetBandwidth": 100,
            "systemDiskType": [
                "hp1",
                "cloud_hp1",
                "SSD_Enhanced"
            ],
            "dataDiskType": [
                "local"
            ],
            "enableJumboFrame": false,
            "nicIpv4Quota": 40,
            "nicIpv6Quota": 40,
            "networkBandwidth": "200",
            "networkPackage": "5000",
            "netEthQueueCount": "32",
            "netEthMaxQueueCount": "32",
            "productType": "Prepaid",
            "spec": "bcc.hgn5IB.c116m978.8A100-80g"
        }]
    },
    {
        "groupId": "common",
        "flavors": [
        {
            "specId": "g5",
            "cpuCount": 2,
            "memoryCapacityInGB": 8,
            "ephemeralDiskCount": 0,
            "cpuModel": "Intel Xeon Platinum 8350C",
            "cpuGHz": "2.6",
            "eniQuota": 2,
            "eriQuota": 0,
            "systemDiskType": [
                "hp1",
                "cloud_hp1",
                "SSD_Enhanced"
            ],
            "dataDiskType": [
                "hp1",
                "cloud_hp1",
                "SSD_Enhanced"
            ],
            "enableJumboFrame": false,
            "nicIpv4Quota": 8,
            "nicIpv6Quota": 8,
            "networkBandwidth": "1.5",
            "networkPackage": "40",
            "netEthQueueCount": "2",
            "netEthMaxQueueCount": "2",
            "productType": "Prepaid",
            "spec": "bcc.g5.c2m8"
        }]
```

```
        }]
      }
    }]
}
```

通过套餐规格查询可用区

该接口用于通过实例规格查询可用区列表。

请求结构

```
 GET /v{version}/instance/flavorZones?clientToken={clientToken} HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
  "instanceType": instanceType,
  "productType": productType,
  "spec": spec,
  "specId": specId
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
version	String	是	URL参数	API版本号。
clientToken	String	是	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串，详见 幂等性。
instanceType	String	否	Query参数	实例类型，具体可选类型参见 InstanceType
productType	String	否	Query参数	实例规格支持的付费方式
spec	String	否	Query参数	实例规格，产品类型为ebc（也即实例规格前缀为ebc/ehc的实例规格）暂不支持本接口
specId	String	否	Query参数	规格族

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
zoneNames	List<String>	可用区列表

请求示例

```
 GET http://bcc.bj.baidubce.com/v1/instance/flavorZones?spec=bcc.g5.c1m4&clientToken=3A8d84403d-a94b-40ce-9d6a-f6225dd26676
Host: bcc.bj.baidubce.com
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2015 03:28:11 GMT
x-bce-request-id: d8752367-38e8-45e4-b4c7-e53be3137ce5
Server: BWS
{
  {
    "zoneNames": [
        "cn-bj-a"
    ]
  }
}
```

附录

Model对象定义

InstanceModel

参数名称	类型	描述
id	String	实例ID，符合BCE规范，是一个定长字符串，且只允许包含大小写字母、数字、连字号(-)和下划线(_)。
name	String	实例名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
hostname	String	实例主机名,仅支持小写字母、数字以及- .特殊字符，必须以字母开头，不可连续使用特殊符号，不支持特殊符号开头或结尾，长度2-64。
instanceType	InstanceType	实例类型，具体可选类型参见InstanceType，DCC（专属服务器）该字段为空
status	InstanceStatus	实例状态
repairStatus	String	实例维修状态，包含normal（正常），wait_authorize（待授权维修），wait_repair（待发单维修），repairing（维修中）四种状态 。
desc	String	实例描述信息
paymentTiming	String	付费方式，付费方式，包括Postpaid(按量付费)，Prepaid(包年包月付费)两种。
createTime	String	创建时间
expireTime	String	过期时间
internalIp	String	内网IP
publicIp	String	外网IP
cpuCount	int	CPU(Core)个数
gpuCard	String	实例所携带的GPU卡信息，具体信息参照GpuType
fpgaCard	String	实例所携带的FPGA卡信息，具体信息参照FpgaType
cardCount	int	实例所携带的GPU卡或FPGA卡或异构卡数量，仅在gpuCard或fpgaCard或isomerismCard字段不为空时返回该字段
memoryCapacityInGB	int	内存容量，单位为GB
npuVideoMemory	int	npu显存大小
localDiskSizeInGB	int	本地磁盘大小（不含系统盘，系统盘为免费赠送），单位为GB
imageId	String	镜像ID

imageId	String	镜像ID
imageName	String	镜像名称
imageType	String	镜像类型
serviceComponents	Map<String, String>	服务组件，Map<组件名称,组件状态>
networkCapacityInMbps	int	公网带宽，单位为Mb
placementPolicy	String	实例置放策略，取值default、dedicatedHost。
zoneName	String	可用区信息
subnetId	String	子网ID
vpcId	String	VPC ID
autoRenew	Boolean	是否自动续费，是：true，否：false
enableJumboFrame	Boolean	是否开启Jumbo帧，开启：true，关闭：false
keypairId	String	密钥对ID
keypairName	String	密钥对名称
dedicatedHostId	String	专属服务器ID
ipv6	String	ipv6地址
tags	List<Tag>	标签信息
deploysetId	String	部署集ID
deploysetList	List<DeploySetModel>	部署集信息列表
osVersion	String	操作系统版本
osArch	String	操作系统架构
osName	String	操作系统名称
hosteyeType	String	hosteye版本，enterprise企业版，open基础版，为空或无该字段表示不绑定hosteye
nicInfo	NicInfo	网卡信息，具体参数参见 NicInfo
deletionProtection	int	是否开启删除保护，1开启，0没开
volumes	List<VolumeModel>	磁盘信息，具体参数参见VolumeModel
isEipAutoRelatedDelete	Boolean	实例绑定的EIP是否随抢占实例关联自动释放，是：true，否：false

NicInfo

参数名称	类型	描述
eniId	String	网卡ID
eniUuid	String	网卡长ID
name	String	网卡名称
type	String	网卡类型，primary为主网卡
subnetId	String	子网ID
subnetType	String	子网类型
az	String	可用区信息
description	string	描述
deviceId	string	虚机长ID
status	string	网卡状态
macAddress	string	物理地址
vpcId	string	VPC ID
createdTime	string	创建时间
eniNum	int	网卡数量
eriNum	int	rdma套餐的eri网卡数量
eriInfos	EriInfo	eri网卡信息
ips	List<IpInfo>	辅助IP和主IP信息
ipv6s	List<IpInfo>	ipv6信息
securityGroups	List	安全组短ID列表（主网卡+弹性网卡的安全组）
enterpriseSecurityGroups	List	企业安全组短ID列表（主网卡+弹性网卡的企业安全组）

IpInfo

参数名称	类型	描述
privateIp	String	内网IP地址
eip	String	公网IP地址
primary	String	是否为主IP
eipId	String	绑定的eip 长ID
eipAllocationId	String	eip 短ID
eipSize	String	eip带宽峰值
eipStatus	String	eip状态
eipGroupId	String	共享带宽组ID
eipType	String	eip类型，shared表示共享带宽，normal表示普通eip

VolumeModel

参数名称	类型	描述
id	String	磁盘ID（磁盘详情、磁盘列表接口返回）
name	String	磁盘名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。（磁盘详情、磁盘列表接口返回）
diskSizeInGB	int	磁盘大小，单位是GB（实例详情、磁盘详情、磁盘列表接口返回）
paymentT	String	付费方式，付费方式，包括Postpaid(按量付费)，Prepaid(包年包月)两种。（磁盘列表、磁盘详情

iming	String	接口返回）
createTime	String	创建日期，符合BCE日期规范（磁盘列表、磁盘详情接口返回）
expireTime	String	过期时间（磁盘列表、磁盘详情接口返回）
status	VolumeStatus	磁盘状态 （磁盘列表、磁盘详情接口返回）
type	VolumeType	磁盘类型 （磁盘列表、磁盘详情接口返回）
storageType	StorageType	CDS磁盘存储类型，包括SSD_Enhanced (增强型SSD)，cloud_hp1（通用型SSD），hp1 (高性能云磁盘)和hdd (通用型HDD) 共四种磁盘类型，默认 hp1。（磁盘列表、磁盘详情接口返回）
description	string	描述信息（磁盘列表接口返回）
desc	String	描述信息 （磁盘详情接口返回）
attachments	List<VolumeAttachmentModel>	挂载设备信息列表，磁盘未挂载时该值为空。 （磁盘列表、磁盘详情接口返回）
zoneName	String	可用区信息 （磁盘列表、磁盘详情接口返回）
regionId	String	所在region（磁盘详情接口返回）
snapshotNum	String	磁盘当前具有的快照数量（磁盘详情接口返回）
sourceSnapshotId	String	创建磁盘所用的快照id（磁盘详情接口返回）
autoSnapshotPolicy	AutoSnapshotPolicyModel	磁盘当前配置的快照策略（磁盘详情接口返回）
tags	List<TagModel>	磁盘当前配置的标签（磁盘详情接口返回）
resGroupInfos	List<GroupInfo>	磁盘当前绑定的资源组（磁盘详情接口返回）
encrypted	boolean	是否加密 （磁盘列表、磁盘详情接口返回）
enableAutoRenew	boolean	是否自动续费（磁盘列表、磁盘详情接口返回）
autoRenewTime	int	自动续费时间（磁盘列表、磁盘详情接口返回）
isSystemVolume	boolean	是否为系统盘（实例详情、磁盘列表、磁盘详情接口返回）
clusterId	string	CDS专属集群ID（磁盘列表、磁盘详情接口返回）
volumeId	string	磁盘ID（实例详情接口返回）
cdsExtralo	int	额外性能（磁盘详情、磁盘列表接口返回）
deleteWithInstance	boolean	磁盘随实例删除，仅后付费类型的数据盘返回
deleteAutoSnapshot	boolean	自动快照随磁盘删除，任何类型的磁盘都会返回

GroupInfo

参数名称	类型	描述
groupId	String	资源组id
groupName	String	资源组名称

VolumeClusterModel

参数名称	类型	描述
clusterId	String	专属集群ID
clusterName	String	专属集群名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65
createdTime	String	创建日期，符合BCE日期规范
expiredTime	String	过期时间，符合BCE日期规范。
status	String	专属集群状态
logicalZone	String	专属集群类型
productType	String	专属集群付费类型
clusterType	String	专属集群类型
totalCapacity	int	专属集群总容量
usedCapacity	int	专属集群已使用容量
availableCapacity	int	专属集群可用容量
expandingCapacity	int	专属集群扩展容量
createdVolumeNum	int	由专属集群创建的CDS数量
affiliatedCDSNumber	List	由专属集群创建的CDS列表
enableAutoRenew	boolean	是否开启自动付费

DiskZoneResources

参数名称	类型	描述
zoneName	String	zone信息，zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
diskInfos	List<DiskInfo>	可用区可创建的磁盘信息，具体参数参见DiskInfo

DiskInfo

参数名称	类型	描述
storageType	String	磁盘存储类别
maxDiskSize	int	最大可创建磁盘容量
minDiskSize	int	最小创建磁盘容量

EniInfo

参数名称	类型	描述
eniId	String	网卡ID
name	String	网卡名称
vpcId	String	VPC ID
subnetId	String	子网ID
zoneName	String	区域
descritption	String	描述
createdTime	String	创建时间
macAddress	String	物理地址
status	String	状态
securityGroupIds	List	绑定的安全组列表
privateIpSet	List<IpAddress>	IP地址信息，具体参数参见 IpAddress

IpAddress

参数名称	类型	描述
primary	boolean	是否为主IP
publicIpAddress	String	公网IP
privateIpAddress	String	内网IP
ipv6Address	String	IPV6地址

EphemeralDisk

参数名称	类型	描述
storageType	StorageType	磁盘存储类别
sizeInGB	int	磁盘总容量
freeSizeInGB	int	可用容量，GB

ZoneModel

参数名称	类型	描述
zoneName	String	可用区信息

TagModel

参数名称	类型	描述
tagKey	String	标签键
tagValue	String	标签值

ResourceBo

参数名称	类型	描述
id	String	实例ID
serialNumber	String	实例长ID
name	String	实例名称
recycleTime	String	进入回收站的时间
deleteTime	String	从回收站删除的时间
paymentTiming	String	付费类型 prepay/postpay
serviceName	String	资源名称，这里为"云服务器"
serviceType	String	资源类型，这里为"BCC"
configItem	ConfigItem	实例配置信息，具体信息参照 ConfigItem
configItems	List	实例配置列表

RegionModel

参数名称	类型	描述
regionId	String	地域ID，例：bj
regionName	String	地域名称，例：华北-北京
regionEndpoint	String	地域对应的接入地址（Endpoint）

ConfigItem

参数名称	类型	描述
cpu	int	cpu个数
memory	int	内存大小
type	String	实例类型，具体可选类型参见InstanceType
specId	String	实例规格类型
spec	String	实例规格
zoneName	String	可用区名称

CreateCdsModel

参数名称	类型	是否必需	描述
cdsSizeInGB	int	是	CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
storageType	StorageType	否	CDS磁盘存储类型，默认是hp1(高性能云磁盘)。
snapshotId	String	否	快照ID，当通过快照创建磁盘时，此属性有效，不能小于快照大小
deleteWithInstance	boolean	否	磁盘随实例删除，计费方式为按量后付时生效，初始值为false，V2 API需要在释放实例时指定cdsAttributeActive参数为true才可生效
encryptKey	String	否	加密密钥

VolumeAttachmentModel

参数名称	类型	描述
volumeId	String	磁盘ID
instanceId	String	实例ID
device	String	挂载设备路径（该参数即将停止使用，为提高代码的兼容性，建议您尽量不要使用该参数）
serial	String	磁盘序列号

SnapshotModel

参数名称	类型	描述
id	String	快照ID
name	String	快照名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
sizeInGB	int	快照大小，单位是GB
createTime	String	快照创建时间，符合BCE规范的日期格式
status	SnapshotStatus	快照状态
createMethod	String	快照创建方式，包括手动创建，计划创建
volumeId	String	磁盘ID或实例ID，为空表示相关磁盘已释放
desc	String	快照描述信息
expireTime	String	快照到期时间，符合BCE规范的日期格式（list返回的简化版model无此参数）
package	boolean	是否为大镜像，默认为false
templateId	String	自定义镜像id
insnapId	String	实例快照id
encrypted	boolean	是否加密过
tags	List<Tag>	快照绑定的标签列表

ImageModel

参数名称	类型	描述
id	String	镜像ID
name	String	镜像名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
type	ImageType	镜像类型
osType	String	操作系统类型
osVersion	String	操作系统版本
osArch	String	操作系统位数
osName	String	操作系统名称
osBuild	String	镜像操作系统的构建时间
osLang	String	操作系统语言（CHS中文版，ENG英文版）
createTime	String	镜像的创建时间，符合BCE规范的日期格式
status	ImageStatus	镜像状态
desc	String	镜像描述信息
specialVersion	String	操作系统特殊版本信息，financial_safty代表金融加固版本
diskSize	int	已废弃
minDiskGb	int	所需磁盘最小容量
ephemeralSize	int	已废弃
shareToUserNumLimit	int	可共享的用户数量限制（镜像列表接口不支持此字段）
sharedToUserNum	int	已共享的用户数量（镜像列表接口不支持此字段）
package	boolean	是否是大镜像，默认为false
encrypted	boolean	是否为加密镜像
snapshot	List<SnapshotModel>	作为一个大镜像所附带的数据盘快照

SecurityGroupModel

参数名称	类型	描述
id	String	安全组ID
name	String	名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
desc	String	描述
vpcId	String	私有网络VPC ID
createdTime	String	安全组创建时间
sgVersion	long	安全组版本号
rules	List<SecurityGroupRuleModel>	安全组规则
tags	List<Tag>	安全组绑定的标签列表

SecurityGroupRuleModel

参数名称	类型	描述	是否必须
remark	String	备注	否
direction	String	入站/出站，取值ingress或egress。	是
ethertype	String	网络类型，取值IPv4或IPv6。值为空时表示默认取值IPv4。	否
portRange	String	端口范围，可以指定80等单个端口，值为空时默认取值1-65535。	否
protocol	String	协议类型，tcp、udp或icmp，值为空时默认取值all。	否
sourceGroupId	String	源安全组ID	否
sourceIp	String	源IP地址，与sourceGroupId不能同时设定值。	否
destGroupId	String	目的安全组ID	否
destIp	String	目的IP地址，与destGroupId不能同时设定值。	否
securityGroupId	String	安全组ID	否
securityGroupRuleId	String	安全组规则ID	否
createdTime	String	安全组规则创建时间	否
updatedTime	String	安全组规则修改时间	否

SnapchainModel

参数名称	类型	描述
status	String	快照链状态 可取值active,expired,clear
chainSize	String	快照链容量,单位GB
chainId	String	快照链id
instanceId	String	虚机id
userId	String	用户id
volumeId	String	磁盘id
volumeSize	int	磁盘大小,单位GB
manualSnapCount	int	手动快照个数
autoSnapCount	int	自动快照个数
createTime	String	创建时间

ZoneInstanceType

参数名称	类型	描述
zoneName	String	zone信息,zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
instanceTypes	List<String>	可用机型

ZoneResource

参数名称	类型	描述
zoneName	String	可用区名称
bccResources	List<BccBidResources>	bcc资源

BccBidResources

参数名称	类型	描述
instanceType	String	实例类型
flavors	List<BccBidFlavors>	套餐列表，当instanceType对应的竞价实例的套餐售罄时flavors返回为空。

BccBidFlavors

参数名称	类型	描述
specId	String	实例规格类型
cpuCount	int	cpu核数
memoryCapacityInGB	int	内存大小
productType	ProductType	产品类型
spec	String	实例规格

InstanceTypeModel

参数名称	类型	描述
type	String	规格分类
name	String	规范代码
cpuCount	int	CPU（Core）个数
memorySizeInGB	int	内存（GB）
localDiskSizeInGB	int	本地磁盘（GB）

KeypairModel

参数名称	类型	描述
keypairId	String	密钥对id
name	String	密钥对名称
description	String	密钥对描述
createdTime	Datetime	密钥对创建时间
publicKey	String	公钥内容
privateKey	String	私钥内容
instanceCount	Integer	密钥对绑定的虚机数目
regionId	String	密钥对所在的地域id
fingerPrint	String	公钥指纹

OsModel

参数名称	类型	描述
instanceId	String	实例ID
osArch	String	操作系统位数
osName	String	操作系统名称
osVersion	String	操作系统版本
osType	String	操作系统类型
osLang	String	操作系统语言（CHS中文版，ENG英文版）
specialVersion	String	操作系统特殊版本信息，financial_safty代表金融加固版本

SharedUser

参数名称	类型	描述
account	String	共享的用户名
accountId	String	共享的用户ID
ucAccount	String	uc账号

AutoSnapshotPolicyModel

参数名称	类型	描述
id	String	自动快照策略ID
name	String	自动快照策略名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
timePoints	List<int>	指定自动快照的创建时间点。最小单位为小时，从 00:00~23:00 共 24 个时间点可选，参数为 0~23 的数字，如：1 代表在 01:00 时间点。可以选定多个时间点。 传递参数为一个带有格式的 Json Array：[0, 1, … 23]，最多 24 个时间点，用半角逗号字符隔开。
repeatWeekdays	List<int>	指定自动快照的重复日期。选定周一到周日中需要创建快照的日期，参数为0~6 的数字，如：0 表示周一。允许选择多个日期。 传递参数为一个带有格式的 Json Array：[0，1…6]。
status	String	快照状态，有active(运行)、deleted（删除）、paused（暂停）三种状态
retentionDays	int	指定自动快照的保留时间，单位为天。 -1：永久保存 1~65536：指定保存天数。
createdTime	String	自动快照策略的创建时间，符合BCE规范的日期格式 （自该字段起，及之后字段，在volume的接口中没有返回）
updatedTime	String	自动快照策略的最近更新时间，符合BCE规范的日期格式
deletedTime	String	自动快照策略的删除时间，符合BCE规范的日期格式
lastExecuteTime	String	自动快照策略的最后执行时间，符合BCE规范的日期格式
volumeCount	int	关联磁盘数量

InstancePassRoleModel

参数名称	类型	描述
instanceId	String	实例ID

InstancePassRoleFailModel

参数名称	类型	描述
instanceId	String	实例ID
failMessage	String	失败信息

InstanceRoleAssociationModel

参数名称	类型	描述
instanceId	String	实例ID

InstanceRoleModel

参数名称	类型	描述
roleName	String	角色名称

ZoneResourceDetailSpec

参数名称	类型	描述
zoneName	String	可用区
bccResources	BccResources	BCC实例资源信息
ebcResources	BccResources	EBC实例资源信息

BccResources

参数名称	类型	描述
flavorGroups	List<FlavorGroup>	实例套餐规格信息

FileSystemModel

参数名称	类型	描述
fsId	String	cfs文件系统ID
mountAds	String	挂载目标的地址
path	String	挂载目录
protocol	String	cfs文件系统的协议类型，可选值为：nfs，smb

FlavorGroup

参数名称	类型	描述
groupId	String	实例套餐规格族
flavors	List<BccFlavor>	实例套餐规格

BccFlavor

参数名称	类型	描述
cpuCount	int	cpu数量
memoryCapacityInGB	int	内存容量（单位：GB）
ephemeralDiskInGb	int	本地数据盘容量（单位：GB）
ephemeralDiskCount	int	本地数据盘数量
ephemeralDiskType	String	本地数据盘类型
gpuCardType	String	gpu卡类型
gpuCardCount	int	gpu卡数量
fpgaCardType	String	fpga卡类型
fpgaCardCount	int	fpga卡数量
productType	String	支持计费类型（PrePaid：包年包月；PostPaid：按量付费；both：包年包月/按量付费）
spec	String	实例套餐规格
specId	String	实例套餐规格类型
enableJumboFrame	Boolean	实例套餐是否支持开启Jumbo帧，开启:true，关闭:false
cpuModel	String	处理器型号
cpuGHz	String	处理器主频
networkBandwidth	String	内网带宽(Gbps)
networkPackage	String	网络收发包
netEthQueueCount	String	套餐网卡队列数
netEthMaxQueueCount	String	套餐网卡最大支持的队列数
eniQuota	int	ENI最大数量（配额）
eriQuota	int	ERI最大数量（配额）
rdmaType	String	rdma类型，RoCE或IB
rdmaNetCardCount	int	rdma网卡数量
rdmaNetBandwidth	int	rdma网卡带宽(Gbps)
systemDiskType	StorageType	套餐支持的系统盘磁盘类型
dataDiskType	StorageType	套餐支持的数据盘磁盘类型
nicIpv4Quota	int	单网卡IPv4地址数量（配额）
nicIpv6Quota	int	单网卡IPv6地址数量（配额）
volumeCount	int	CDS数量

SpecIdPrices

参数名称	类型	描述
specId	String	实例规格类型
specPrices	List<SpecPrices>	实例套餐规格信息

SpecPrices

参数名称	类型	描述
spec	String	实例套餐规格
status	String	实例套餐规格状态
specPrice	double	实例套餐规格对应价格
tradePrice	double	实例最终价，即优惠后订单实付价格

AsGroup

参数名称	类型	描述
groupId	String	伸缩组ID，符合BCE规范，是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
groupName	String	伸缩组名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
region	String	伸缩组所在地域
status	AsGroupStatus	伸缩组状态
vpcId	String	私有网络VpcId
nodeNum	int	伸缩组下节点数量
createTime	String	创建时间
zoneInfo	ZoneInfo	可用区及子网信息
groupConfig	GroupConfig	伸缩组规格
blbId	String	负载均衡Id

ZoneInfo

参数名称	类型	描述
zone	String	可用区
subnetId	String	子网ID

GroupConfig

参数名称	类型	描述
minNodeNum	int	最小节点数量
maxNodeNum	int	最大节点数量
cooldownInSec	int	冷却时间（单位：秒）

VpcInfo

参数名称	类型	描述
vpcId	String	私有网络VpcId
vpcName	String	私有网络VPC名称

NodeModel

参数名称	类型	描述
instanceId	String	实例ID
instanceName	String	实例名称
floatingIp	String	浮动IP
internalIp	String	内网IP
status	String	节点状态
payment	String	付费方式
cpuCount	int	cpu数量
memoryCapacityInGB	int	内存大小
instanceType	String	实例类型
sysDiskInGB	int	系统盘大小
subnetType	String	子网类型
isProtected	boolean	是否受保护节点
createTime	String	创建时间
nodeType	String	节点类型
eip	AsEipModel	eip信息

AsEipModel

参数名称	类型	描述
bandwidthInMbps	int	最大带宽
address	String	公网IP
eipStatus	String	EIP状态
eipAllocationId	String	弹性公网IP-实例ID

DeploySetModel

参数名称	类型	描述
deploysetId	String	部署集id
name	String	部署集名称
desc	String	部署集描述
concurrency	int	部署集并发数
strategy	String	部署集策略，HOST_HA：宿主机；RACK_HA：机架；TOR_HA：交换机；POD_RQ：集群
azIntstanceStatisList	List<AzIntstanceStatisDetail>	可用区实例数量统计列表

AzIntstanceStatisDetail

参数名称	类型	描述
zoneName	String	可用区名称
instanceCount	String	部署集关联的实例数量
bccInstanceCnt	String	部署集关联的BCC实例数量
bbcInstanceCnt	String	部署集关联的BBC实例数量
instanceTotal	int	当前部署集strategy类型下指定可用区配额
instanceIds	List	部署集关联的实例列表，查询部署集详情返回，查询部署集列表不返回
bccInstanceIds	List	部署集关联的BCC实例ID列表，查询部署集详情返回，查询部署集列表不返回
bbcInstanceIds	List	部署集关联的BBC实例数量，查询部署集详情返回，查询部署集列表不返回

PrepayConfig

参数名称	类型	描述
instanceId	String	实例ID
duration	int	购买时长（单位：月）
cdsList	List	（该参数已废弃，bcc挂载的按量付费CDS数据盘必须一起转包年包月） 变更关联的数据盘列表，默认为空，all表示关联的全部数据盘（例"cdsList":["all"]，若仅变更部分关联数据盘，传具体的数据盘id）
autoRenew	boolean	实例到期后是否自动续费，取值：**true**：自动续费，**false**：不自动续费，默认值：false。
autoRenewPeriod	int	每次自动续费的时长（单位：月）。取值范围：1，2，3，4，5，6，7，8，9，12，24，36，默认值：1。仅当autoRenew取值为true，该参数有效。
autoPay	boolean	是否自动支付，默认true，表示自动支付

PostpayConfig

参数名称	类型	描述
instanceId	String	实例ID
cdsList	List	（该参数已废弃，bcc挂载的包年包月CDS数据盘必须一起转按量付费）变更关联的数据盘列表，默认为空，all表示关联的全部数据盘（例"cdsList":["all"]，若仅变更部分关联数据盘，传具体的数据盘id）。
effectiveType	String	生效方式，可选参数：AtOnce（立即转按量付费）、AfterExpiration（到期后转按量付费）。不传默认为到期后按量付费。批量操作该参数必须保持一致。

InstanceDeleteResultModel

参数名称	类型	描述
instanceId	String	实例id
eip	String	eip
insnapIds	List	实例快照id列表
snapshotIds	List	快照id列表
volumeIds	List	磁盘id列表

类型编码定义

VolumeType

编码	描述
System	系统盘
Ephemeral	本地磁盘
Cds	CDS云磁盘

StorageType

编码	描述	可创建数据盘	可创建系统盘
enhanced_ssd_pl1 或 SSD_Enhanced	增强型SSD_PL1	是	是
enhanced_ssd_pl2	增强型SSD_PL2	是	是
enhanced_ssd_pl3	增强型SSD_PL3	是	是
cloud_hp1 或 premium_ssd	通用型SSD	是	是
hp1 或 ssd	高性能云磁盘	是	是
hdd	通用型HDD (该类型已在逐步下线中)	是	否
local	本地盘	否	否
sata	上一代云磁盘，创建DCC子实例专用	-	-
local_ssd	数据存储于本地SSD盘，延时较低，IO和吞吐能力较好，适用于中大型关系型数据库业务	-	-
local_hdd 或 std1	本地HDD磁盘，适合大规模数据存储的场景	-	-
local_nvme 或 nvme	本地NVME磁盘,数据存储于本地NVME SSD盘，超高性能，适用于时延敏感型业务	-	-

ImageType

编码	描述
System	系统镜像/公共镜像
Custom	自定义镜像
Integration	服务集成镜像

GpuType

编码	描述
V100-32	NVIDIA Tesla V100-32G
V100-16	NVIDIA Tesla V100-16G
P40	NVIDIA Tesla P40
P4	NVIDIA Tesla P4
K40	NVIDIA Tesla K40
DLCard	NVIDIA 深度学习开发卡

FpgaType

编码	描述
KU115	xilinx ku115

InternetChargeType

编码	描述
BANDWIDTH_PREPAID	包年包月按带宽结算
TRAFFIC_POSTPAID_BY_HOUR	流量按小时后付费
BANDWIDTH_POSTPAID_BY_HOUR	带宽按小时后付费

BidModel

编码	描述
market	按照市场价出价
custom	自定义出价

DeployStrategy

编码	描述
HOST_HA	宿主机隔离
RACK_HA	机架隔离
TOR_HA	交换机隔离

SystemImageModel

	参数名称	类型
imageId	String	镜像ID
imageName	String	镜像名称
osType	String	操作系统类型
osVersion	String	操作系统版本
osArch	String	操作系统位数
osName	String	操作系统名称
osLang	String	操作系统语言（CHS中文版，ENG英文版）
minSizeInGiB	String	支持系统盘最小容量（单位：GB）

状态编码定义

ProductType

编码	描述
Prepaid	包年包月（预付费）
Postpaid	按量付费（后付费）

InstanceStatus

编码	描述
Starting	启动中
Running	运行中
Stopping	停止中
Stopped	已停止
Recycled	处于回收站中
Deleted	已释放，该状态为内部状态，API无法查询
Scaling	扩展中
Expired	已过期或欠费
Error	错误
SnapshotProcessing	快照操作中
ImageProcessing	镜像操作中
Recharging	续费中
VolumeResizing	磁盘扩容中
BillingChanging	计费变更中
ChangeSubnet	子网变更中
ChangeVpc	VPC变更中
AttachingPort	挂载弹性网卡中
DetachingPort	卸载弹性网卡中
Moving	迁移中，实例操作变更可用区/跨AZ迁移时的状态

VolumeStatus

编码	描述
Creating	创建中。卷创建是异步操作，Creating表示系统已收到创建请求，并分配相应卷ID，但尚未创建完成
Available	待挂载。卷已创建完成，状态正常，且未attach到任何虚拟机实例
Attaching	挂载中。卷attach是异步操作，该状态表示正在进行attach操作
NotAvailable	临时不可用状态
InUse	使用中
Detaching	卸载中。卷detach是异步操作，该状态表示正在进行detach操作
Deleting	删除中
Deleted	已删除，该状态为内部状态，API无法查询
Scaling	扩展中
Expired	已过期或欠费
Error	错误
SnapshotProcessing	快照操作中
ImageProcessing	镜像操作中
Recharging	续费中

RemoteCopyRequest

编码	类型	描述
name	String	快照名称
destRegion	String	待复制快照的目标区域

RemoteCopySnapshot

编码	类型	描述
region	String	目标区域
snapshotId	String	成功复制到目标区域快照的快照ID

SnapshotStatus

编码	描述
Creating	创建中
CreatedFailed	创建失败
Available	可用
NotAvailable	不可用

ImageStatus

编码	描述
Creating	创建中
CreatedFailed	创建失败
Available	可用
NotAvailable	不可用
Error	错误

RelatedRenewFlag

编码	描述
CDS	只对BCC实例关联的包年包月（预付费）CDS进行续费
EIP	只对BCC实例关联的包年包月（预付费）EIP进行续费
MKT	只对BCC实例关联的包年包月（预付费）MKT进行续费
CDS_EIP	只对BCC实例关联的包年包月（预付费）CDS、EIP进行续费
CDS_MKT	只对BCC实例关联的包年包月（预付费）CDS、MKT进行续费
EIP_MKT	只对BCC实例关联的包年包月（预付费）EIP、MKT进行续费
CDS_EIP_MKT	只对BCC实例关联的包年包月（预付费）CDS、EIP、MKT进行续费

AsGroupStatus

编码	描述
CREATING	创建中
RUNNING	运行中
SCALING_UP	扩容中
SCALING_DOWN	缩容中
ATTACHING_NODE	移入节点中
DETACHING_NODE	移出节点中
DELETING	删除中
BINDING_BLB	绑定blb中
UNBINDING_BLB	解绑blb中
COOLDOWN	冷却中
PAUSE	挂起中
DELETED	已删除

区域机型以及可选配置

InstanceType

编码	描述	新机型名称
N1	普通型BCC实例	通用型g1、计算型c1、密集计算型ic1、内存型m1
N2	普通型 II BCC实例	通用型g2、计算型c2、密集计算型ic2、内存型m2
N3	普通型 III BCC实例	通用型g3、计算型c3、密集计算型ic3、内存型m3
N4	网络增强型BCC实例	通用网络增强型g3ne、计算网络增强型c3ne、内存网络增强型m3ne
N5	普通型IV BCC实例	通用型g4、密集计算型ic4、计算型c4、内存型m4
N6	普通型V BCC实例	通用型g5、密集计算型ic5、计算型c5、内存型m5
C1	计算优化型实例	高主频计算型hcc1、高主频通用型hcg1
C2	计算优化 II 型实例	高主频计算型hcc2、高主频通用型hcg2
S1	存储优化型实例	本地SSD型l1
G1	GPU型实例	---
F1	FPGA型实例	---

GpuType

编码	描述
V100-32	NVIDIA Tesla V100-32G
V100-16	NVIDIA Tesla V100-16G
P40	NVIDIA Tesla P40
P4	NVIDIA Tesla P4
K40	NVIDIA Tesla K40
DLCard	NVIDIA 深度学习开发卡

区域可支持实例类型（随产品规划而变化，请以控制台购买页为准）

区域	支持创建实例类型
北京(bj)	普通型BCC(N1)，可选配置参考普通型BCC(N1)可选规格配置 普通型 II BCC(N2)，可选配置参考普通型 II BCC(N2)可选规格配置 普通型 III BCC(N3)，可选配置参考普通型 III BCC(N3)可选规格配置 计算优化型实例(C1)，可选配置参考计算优化型实例(C1)可选规格配置 存储优化型实例(S1)，可选配置参考存储优化型实例(S1)可选规格配置 GPU型实例(G1)，可选配置参考GPU型实例(G1)可选规格配置 FPGA型实例(F1)，可选配置参考FPGA型实例(F1)可选规格配置
广州(gz)	普通型BCC(N1)，可选配置参考普通型BCC(N1)可选规格配置 普通型 II BCC(N2)，可选配置参考普通型 II BCC(N2)可选规格配置 普通型 III BCC(N3)，可选配置参考普通型 III BCC(N3)可选规格配置 计算优化型实例(C1)，可选配置参考计算优化型实例(C1)可选规格配置 存储优化型实例(S1)，可选配置参考存储优化型实例(S1)可选规格配置 GPU型实例(G1)，可选配置参考GPU型实例(G1)可选规格配置
苏州(su)	普通型 II BCC(N2)，可选配置参考普通型 II BCC(N2)可选规格配置 普通型 III BCC(N3)，可选配置参考普通型 III BCC(N3)可选规格配置 计算优化型实例(C1)，可选配置参考计算优化型实例(C1)可选规格配置 存储优化型实例(S1)，可选配置参考存储优化型实例(S1)可选规格配置 GPU型实例(G1)，可选配置参考GPU型实例(G1)可选规格配置

普通型BCC可选规格配置

包年包月（预付费）可选规格配置(CPU核数, 内存GB)	按量付费（后付费）可选规格配置(CPU核数, 内存GB)
[1, 1], [1, 2], [1, 4], [1, 8], [2, 2], [2, 4], [2, 8], [2, 12], [2, 16], [4, 4], [4, 8], [4, 12], [4, 16], [4, 32], [8, 8], [8, 12], [8, 16], [8, 24], [8, 32], [8, 64], [12, 12], [12, 16], [12, 24], [12, 32], [12, 48], [12, 64], [16, 16], [16, 24], [16, 32], [16, 48], [16, 64]	同包年包月（预付费）可选规格配置(CPU核数, 内存GB)

普通型BCC II 可选规格配置

包年包月（预付费）可选规格配置(CPU核数, 内存GB)	按量付费（后付费）可选规格配置(CPU核数, 内存GB)
[1, 1], [1, 2], [1, 4], [1, 8], [2, 2], [2, 4], [2, 8], [2, 12], [2, 16], [4, 4], [4, 8], [4, 12], [4, 16], [4, 32], [8, 8], [8, 12], [8, 16], [8, 24], [8, 32], [8, 64], [12, 12], [12, 16], [12, 24], [12, 32], [12, 48], [12, 64], [16, 16], [16, 24], [16, 32], [16, 48], [16, 64], [16, 128], [32, 64], [32, 128], [48, 96], [48, 128], [48, 192]	[1, 1], [1, 2], [1, 4], [1, 8], [2, 2], [2, 4], [2, 8], [2, 12], [2, 16], [4, 4], [4, 8], [4, 12], [4, 16], [4, 32], [8, 8], [8, 12], [8, 16], [8, 24], [8, 32], [8, 64], [12, 12], [12, 16], [12, 24], [12, 32], [12, 48], [12, 64], [16, 16], [16, 24], [16, 32], [16, 48], [16, 64], [16, 128]

普通型BCC III 可选规格配置

包年包月（预付费）可选规格配置(CPU核数, 内存GB)	按量付费（后付费）可选规格配置(CPU核数, 内存GB)
[1, 1], [1, 2], [1, 4], [1, 8], [2, 2], [2, 4], [2, 8], [2, 12], [2, 16], [4, 4], [4, 8], [4, 12], [4, 16], [4, 32], [8, 8], [8, 12], [8, 16], [8, 24], [8, 32], [8, 64], [12, 12], [12, 16], [12, 24], [12, 32], [12, 48], [12, 64], [16, 16], [16, 24], [16, 32], [16, 48], [16, 64], [16, 128], [24, 48], [24, 96], [24, 128], [32, 64], [32, 128], [32, 256], [48, 96], [48, 128], [48, 192], [48, 256], [64, 128], [64, 256], [96, 256]	同包年包月（预付费）可选规格配置(CPU核数, 内存GB)

计算优化型实例可选规格配置

包年包月（预付费）可选规格配置(CPU核数, 内存GB)	按量付费（后付费）可选规格配置(CPU核数, 内存GB)
[4, 8], [4, 16], [4, 32], [8, 16], [8, 32], [16, 32], [16, 64], [24, 96]	同包年包月（预付费）可选规格配置(CPU核数, 内存GB)

存储优化型实例可选规格配置

预付费可选规格配置(CPU核数, 内存GB)	后付费可选规格配置(CPU核数, 内存GB)
[1, 1], [1, 2], [1, 4], [1, 8], [2, 2], [2, 4], [2, 8], [2, 12], [2, 16], [4, 4], [4, 8], [4, 12], [4, 16], [4, 32], [8, 8], [8, 12], [8, 16], [8, 24], [8, 32], [8, 64], [12, 12], [12, 16], [12, 24], [12, 32], [12, 48], [12, 64], [16, 16], [16, 24], [16, 32], [16, 48], [16, 64]	同预付费可选规格配置(CPU核数, 内存GB)

GPU型BCC可选规格配置

预付费可选规格配置(GPU卡类型,GPU卡数量,CPU核数, 内存GB,本地磁盘大小GB)	后付费可选规格配置(GPU卡类型,GPU卡数量,CPU核数, 内存GB,本地磁盘大小GB)
[P4,1,12,40,450],[P4,2,24,80,900],[P4,4,48,160,1800], [P40,1,12,40,450],[P40,2,24,80,900], [P40,4,48,160,1800], [K40,1,6,40,450],[K40,2,12,80,400],[K40,4,24,160,800], [DLCard,1,6,40,450],[DLCard,2,12,80,400], [DLCard,4,24,160,800]	同预付费可选规格配置(GPU卡类型,GPU卡数量,CPU核数, 内存GB,本地磁盘大小GB)

FPGA型BCC可选规格配置

预付费可选规格配置(FPGA卡类型,FPGA卡数量,CPU核数, 内存GB,本地磁盘大小GB)	后付费可选规格配置(FPGA卡类型,FPGA卡数量,CPU核数, 内存GB,本地磁盘大小GB)
[KU115,1,16,64,450]	同预付费可选规格配置(FPGA卡类型,FPGA数量,CPU核数, 内存GB,本地磁盘大小GB)

实例套餐规格定义（已废弃，不建议使用）

规格分类	规格代码	CPU(Core)	内存（GB）
Tiny	bcc.t1.tiny	1	1
General	bcc.g1.tiny	1	2
General	bcc.g2.tiny	1	4
General	bcc.g3.tiny	1	8
General	bcc.g1.small	2	2
General	bcc.g2.small	2	4
General	bcc.g3.small	2	8
General	bcc.g4.small	2	12
General	bcc.g1.medium	4	4
General	bcc.g2.medium	4	8
Memory	bcc.m1.medium	4	12
Memory	bcc.m2.medium	4	16
Memory	bcc.m1.large	8	16
Memory	bcc.m2.large	8	24
Memory	bcc.m3.large	8	32
cpu	bcc.c1.large	8	8
cpu	bcc.c2.large	8	12
cpu	bcc.c1.xlarge	12	12
cpu	bcc.c2.xlarge	12	16
cpu	bcc.c3.xlarge	12	24
cpu	bcc.c4.xlarge	12	32
cpu	bcc.c5.xlarge	12	48
cpu	bcc.c1.2xlarge	16	16
cpu	bcc.c2.2xlarge	16	24
cpu	bcc.c3.2xlarge	16	32
cpu	bcc.c4.2xlarge	16	48
cpu	bcc.c5.2xlarge	16	64

订单信息定义

Billing

状态	类型	描述
paymentTiming	String	付费方式，包括预支付（Prepaid）、后支付（Postpaid）和抢占实例（bidding）
reservation	Reservation	保留信息，支付方式为后支付时不需要设置，预支付时必须设置

Reservation

状态	类型	描述
reservationLength	int	时长，[1,2,3,4,5,6,7,8,9,12,24,36]
reservationTimeUnit	String	时间单位，Month，当前仅支持按月

TransferInRecord

状态	类型	描述
transferRecordId	String	券转移记录id
grantorUserId	String	转让人账号(脱敏处理)
status	String	券转移记录状态
reservedInstanceInfo	ReservedInstanceInfo	预留实例券详情
applicationTime	String	券转移发起时间
expireTime	String	券转移过期时间
endTime	String	券转移结束时间

TransferOutRecord

状态	类型	描述
transferRecordId	String	券转移记录id
recipientUserId	String	接收人账号(脱敏处理)
status	String	券转移记录状态
reservedInstanceInfo	ReservedInstanceInfo	预留实例券详情
applicationTime	String	券转移发起时间
expireTime	String	券转移过期时间
endTime	String	券转移结束时间

ReservedInstanceInfo

状态	类型	描述
reservedInstanceId	String	预留实例券id
reservedInstanceUuId	String	预留实例券长id
reservedInstanceName	String	预留实例券名称
scope	String	预留实例券生效范围
zoneName	String	预留实例券可用区
spec	String	预留实例券实例规格
reservedType	String	预留实例券类型
offeringType	String	预留实例券付费方式
osType	String	预留实例券镜像类型
reservedInstanceStatus	String	预留实例券状态
instanceCount	int	预留实例券实例数量
instanceId	String	抵扣实例ID
instanceName	String	抵扣实例名称
effectiveTime	String	预留实例券生效时间
expireTime	String	预留实例券过期时间
autoRenew	String	预留实例券是否开启自动续费
renewTimeUnit	String	预留实例券自动续费单位
renewTime	String	预留实例券续费时长
nextRenewTime	String	预留实例券下次自动续费时间
ehcClusterId	String	预留实例券所在ehc集群id

TaskModel

状态	类型	描述
taskId	String	任务ID，符合BCE规范，是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
instanceId	String	实例ID
internalIp	String	实例内网IP
status	String	任务状态
errResult	String	故障描述
createTime	String	创建时间
enableHotplug	Boolean	是否支持热维修，true表示支持，空表示不支持
isSystemDiskRecoverable	Boolean	系统盘数据是否可修复，true表示可修复，空表示不可修复
relocationTime	int	冷迁预估时长，空表示无此信息
retainDataDisk	Boolean	是否保留数据盘数据，true表示保留，空表示不保留
errStartTime	String	故障发生时间
newTaskTime	String	创建维修任务时间
authorizeTaskTime	String	授权维修任务时间
unAuthorizeTaskTime	String	未授权维修任务时间
simCompletionTime	String	维修人员维修完成时间
checkTaskTime	String	验收维修任务时间
firstConfirmedTime	String	首次验收维修任务或验收未完成时间

ClosedTaskModel

状态	类型	描述
taskId	String	任务ID，符合BCE规范，是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
instanceId	String	实例ID
internalIp	String	实例内网IP
errResult	String	故障描述
startTime	String	任务开始时间
endTime	String	任务结束时间
errStartTime	String	故障发生时间
newTaskTime	String	创建维修任务时间
authorizeTaskTime	String	授权维修任务时间
unAuthorizeTaskTime	String	未授权维修任务时间
simCompletionTime	String	维修人员维修完成时间
checkTaskTime	String	验收维修任务时间
firstConfirmedTime	String	首次验收维修任务或验收未完成时间

RepairTaskDetail

状态	类型	描述
taskId	String	任务ID，符合BCE规范，是一个定长字符串，且只允许包含大小写字母、数字、连字号（-）和下划线（_）。
instanceId	String	实例ID
instanceName	String	实例名称
errResult	String	故障描述
status	String	任务状态
serverStatus	String	实例状态
region	String	实例所在region
internalIp	String	实例内网ip
floatingIp	String	实例floating ip
errStartTime	String	故障发生时间
newTaskTime	String	创建维修任务时间
authorizeTaskTime	String	授权维修任务时间
unAuthorizeTaskTime	String	未授权维修任务时间
simCompletionTime	String	维修人员维修完成时间
checkTaskTime	String	验收维修任务时间
firstConfirmedTime	String	首次验收维修任务或验收未完成时间

OperationRecord

状态	类型	描述
name	String	操作名
operator	String	操作人
operateTime	String	操作时间

RuleModel

状态	类型	描述
ruleId	String	操作名
ruleName	String	操作人
tagCount	int	关联的标签数量
limit	int	预授权上限
status	String	规则状态，enabled/disabled

RuleDetail

状态	类型	描述
ruleId	String	操作名
ruleName	String	操作人
tagCount	int	关联的标签数量
limit	int	预授权上限
status	String	规则状态，enabled/disabled
tags	List<Tag>	关联标签列表

CdsCustomPeriod

状态	类型	描述
volumeId	String	磁盘ID（磁盘详情、磁盘列表接口返回）
period	int	续费时长，单位为月，范围为【1，60】。需保证续费后磁盘的到期时间等于或晚于BCC到期时间，否则续费失败。

集群管理相关接口

EHC集群修改

修改弹性高性能计算集群（Elastic Hyper-Computing Cluster，简称EHC）集群名和集群描述

请求结构

```
 POST /v{version}/instance/ehc/cluster/modify HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "ehcClusterId":"ehc-Djasd3pU",
    "name" : "name",
    "description" : "description"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ehcClusterId	String	是	RequestBody参数	需要修改的EHC集群ID
name	String	否	RequestBody参数	EHC集群名，不能与description同为空
description	String	否	RequestBody参数	EHC集群描述，不能与name同为空

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/instance/ehc/cluster/modify HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "ehcClusterId":"ehc-Djasd3pU",
    "name":"modifyName",
    "description":"modifyDescription"
}
```

EHC集群删除

删除无实例以及预留实例券的弹性高性能计算集群（Elastic Hyper-Computing Cluster，简称EHC）

请求结构

```
 POST /v{version}/instance/ehc/cluster/delete HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "ehcClusterIdList":ehcClusterIdList
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ehcClusterIdList	List<String>	是	RequestBody参数	删除的EHC集群id列表

返回头域

除公共头域，无其它特殊头域。

返回参数

无特殊返回参数

请求示例

```
 POST /v2/instance/ehc/cluster/delete HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "ehcClusterIdList":["ehc-Djasd3pU"]
}
```

EHC集群列表

查询弹性高性能计算集群（Elastic Hyper-Computing Cluster，简称EHC）集群列表信息

请求结构

```
 POST /{version}/instance/ehc/cluster/list HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "ehcClusterIdList" : ehcClusterIdList,
    "nameList" : nameList,
    "zoneName": "zoneName"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
ehcClusterIdList	List<String>	否	RequestBody参数	EHC集群的ID列表
nameList	List<String>	否	RequestBody参数	EHC name 列表
zoneName	String	否	RequestBody参数	EHC集群的zoneName

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
totalCount	Integer	EHC集群总数
ehcClusters	List<EhcCluster>	EHC集群信息，由 EhcCluster 组成的集合

EhcCluster

参数名称	类型	描述
ehcClusterId	String	EHC集群id
name	String	EHC集群name
description	String	EHC集群描述
zoneName	String	可用区信息
createdTime	String	创建时间
instanceIds	List<String>	集群下实例id列表
reservedInstanceIds	List<String>	集群下预留实例券id列表

请求示例

```
 POST /v2/instance/ehc/cluster/list HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
   "ehcClusterIdList":["ehc-t32CTrV0", "ehc-2Wt1bUNf"],
   "nameList":["ehc-c-2"],
   "zoneName":"cn-bj-c"
}
```

返回示例

```
 HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
Date: Wed, 03 Dec 2014 06:42:19 GMT
Content-Type: application/json;charset=UTF-8
Server: BWS

{
    "totalCount": 1,
    "ehcClusters": [
        {
            "ehcClusterId": "ehc-2Wt1bUNf",
            "name": "ehc-c-2",
            "description": "",
            "zoneName": "cn-bj-c",
            "createdTime": "2024-06-25T10:28:02Z",
            "instanceIds": [],
            "reservedInstanceIds": []
        }
    ]
}
```

EHC集群创建

创建弹性高性能计算集群（Elastic Hyper-Computing Cluster，简称EHC）

请求结构

```
 POST /v{version}/instance/ehc/cluster/create HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "name" : "name",
    "description" : "description",
    "zoneName": "zoneName"
}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必需	参数位置	描述
version	String	是	URL参数	API版本号
name	String	是	RequestBody参数	EHC集群名
zoneName	String	是	RequestBody参数	指定zone信息，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。专属实例使用专属服务器所在zone,无需指定该字段。
description	String	是	RequestBody参数	EHC集群描述

返回头域

除公共头域，无其它特殊头域。

返回参数

参数名称	类型	描述
ehcClusterId	String	返回EHC集群ID

请求示例

```
 POST /v2/instance/ehc/cluster/create HTTP/1.1
Host: bcc.bj.baidubce.com
ContentType: application/json
Authorization: bce-auth-v1/f81d3b34e48048fbb2634dc7882d7e21/2015-08-
11T04:17:29Z/3600/host/74c506f68c65e26c633bfa104c863fffac5190fdec1ec24b7c03eb5d67d2e1de
{
    "name":"ehc-name",
    "description":"description",
    "zoneName":"cn-bj-c"
}
```

返回示例

```
  HTTP/1.1 200 OK
  x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
  Date: Wed, 03 Dec 2014 06:42:19 GMT
  Content-Type: application/json;charset=UTF-8
  Server: BWS

  {
     "ehcClusterId": "ehc-Djasd3pU"
  }
```

Java-SDK

概述

本文档主要介绍云服务器BCC Java SDK的安装和使用。在使用本文档前，您需要先了解云服务器BCC的一些基本知识，若您还不了解云服务器BCC，可以参考产品描述和入门指南。

安装SDK工具包

运行环境

Java SDK工具包可在jdk1.7、jdk1.8环境下运行。

方式一：使用Maven安装

在Maven的pom.xml文件中添加bce-java-sdk的依赖：

```
<dependency>
  <groupId>com.baidubce</groupId>
  <artifactId>bce-java-sdk</artifactId>
  <version>{version}</version>
</dependency>
```

其中，`{version}`为版本号，可以在SDK下载页面找到。

方式二：直接使用JAR包安装

步骤如下：

1. 下载最新版Java SDK的压缩工具包。
2. 将下载的`bce-java-sdk-{version}.zip`解压后，复制到工程文件夹中。

3. 在Eclipse右键“工程 -> Properties -> Java Build Path -> Add JARs”。
4. 添加SDK工具包lib/bce-java-sdk-{version}.jar和第三方依赖工具包third-party/*.jar。

其中，{version}为版本号，可以在SDK下载页面找到。

SDK目录结构

```
com.baidubce
├── auth                                //BCE签名相关类
├── http                                //BCE的Http通信相关类
├── internal                            //SDK内部类
├── model                               //BCE公用model类
├── services
│   └── bcc                             //BCC服务相关类
│       ├── model                       //BCC内部model，如Request或Response
│       ├── BccClient.class             //BCC客户端入口类
│       └── BccClientConfiguration.class    //针对BCC特有的HttpClient的配置
├── util                                //BCE公用工具类
├── BceClientConfiguration.class            //对BCE的HttpClient的配置
├── BceClientException.class                //BCE客户端的异常类
├── BceServiceException.class               //与BCE服务端交互后的异常类
├── ErrorCode.class                     //BCE通用的错误码
└── Region.class                        //BCE提供服务的区域
```

快速入门

1. 初始化一个BccClient。

   BccClient是与BCC服务交互的客户端，BCC Java SDK的BCC操作都是通过BccClient完成的。用户可以参考新建BccClient，完成初始化客户端的操作。

2. 创建实例。

   BCC实例是百度智能云上的云服务器操作实例，用户可以参考创建实例来完成创建实例的操作。

3. 释放实例。

   对于已经到期的BCC实例或者是按量付费（后付费）类型的BCC实例，用户可以手动对其进行释放，用户可以参考释放实例。

4. 查询实例列表

   用户创建BCC实例后，可以参考查询实例列表来查看全部的BCC实例。

5. 查询实例详情。

   户创建BCC实例后，可以参考查询实例详情来查看全部的BCC实例。

BccClient

BccClient是BCC服务的客户端，为开发者与BCC服务进行交互提供了一系列的方法，包括实例、磁盘、镜像、快照、安全组、可用区等。

新建BccClient

通过AK/SK方式访问BCC

用户可以参考如下代码新建一个BccClient：

```
 public class Sample {
public static void main(String[] args) {
    String ACCESS_KEY_ID = <your-access-key-id>;                // 用户的Access Key ID
    String SECRET_ACCESS_KEY = <your-secret-access-key>;        // 用户的Secret Access Key

    // 初始化一个BccClient
    BccClientConfiguration config = new BccClientConfiguration();
    config.setCredentials(new DefaultBceCredentials(ACCESS_KEY_ID, SECRET_ACCESS_KEY));
    BccClient client = new BccClient(config);
    }
}
```

在上面代码中，变量ACCESS_KEY_ID与SECRET_ACCESS_KEY是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。 其中ACCESS_KEY_ID对应控制台中的“Access Key ID”，SECRET_ACCESS_KEY对应控制台中的“Access Key Secret”，获取方式请参考《操作指南 获取ACCESSKEY》。

上面的方式使用默认域名作为BCC的服务地址，如果用户需要自己指定域名，可以通过传入ENDPOINT参数来指定。

```
 String ACCESS_KEY_ID = <your-access-key-id>;                // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>;        // 用户的Secret Access Key
String ENDPOINT = <domain-name>;                            // 用户自己指定的域名

BccClientConfiguration config = new BccClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ACCESS_KEY_ID,SECRET_ACCESS_KEY));
config.setEndpoint(ENDPOINT);
BccClient client = new BccClient(config);
```

> **注意：** ENDPOINT参数只能用指定的包含区域的域名来进行定义，不指定时默认为北京区域http://bcc.bj.baidubce.com，百度智能云目前开放了多区域支持，更多请参考区域选择说明 和 服务域名。

配置HTTPS协议访问BCC

BCC支持HTTPS传输协议，您可以通过如下两种方式在BCC Java SDK中使用HTTPS访问BCC服务：

- 在endpoint中指明https:

```
 String endpoint = "http://bcc.bj.baidubce.com";
String ak = "ak";
String sk = "sk";
BccClientConfiguration config = new BccClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
BccClient client = new BccClient(config);
```

- 通过调用setProtocol方法设置https协议:

```
 String endpoint = "bcc.bj.baidubce.com"; // endpoint中不包含protocol
String ak = "ak";
String sk = "sk";
BccClientConfiguration config = new BccClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(ENDPOINT);
config.setProtocol(Protocol.HTTPS); // 如果不指明, 则使用http
BccClient client = new BccClient(config);
```

> 注意：如果在endpoint中指明了protocol, 则endpoint中的生效, 另外单独再调用setProtocol()不起作用。

```
String endpoint = "http://bcc.bj.baidubce.com";
String ak = "ak";
String sk = "sk";
BccClientConfiguration config = new BccClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(ENDPOINT);
config.setProtocol(Protocol.HTTPS); // endpoint中已经指明，此为无效操作，对http也是如此
BccClient client = new BccClient(config);
```

配置BccClient

如果用户需要配置BccClient的一些细节的参数，可以在构造BccClient的时候传入BccClientConfiguration对象。BccClientConfiguration是BCC服务的配置类，可以为客户端配置代理，最大连接数等参数。

使用代理

下面一段代码可以让客户端使用代理访问BCC服务：

```
String ACCESS_KEY_ID = <your-access-key-id>;                // 用户的Access Key ID
String SECRET_ACCESS_KEY = <your-secret-access-key>;        // 用户的Secret Access Key
String ENDPOINT = <domain-name>;                            // 用户自己指定的域名

// 创建BccClientConfiguration实例
BccClientConfiguration config = new BccClientConfiguration();

// 配置代理为本地8080端口
config.setProxyHost("127.0.0.1");
config.setProxyPort(8080);

// 创建BCC客户端
config.setCredentials(new DefaultBceCredentials(ACCESS_KEY_ID,SECRET_ACCESS_KEY));
config.setEndpoint(ENDPOINT);
BccClient client = new BccClient(config);
```

使用上面的代码段，客户端的所有操作都会通过127.0.0.1地址的8080端口做代理执行。 对于有用户验证的代理，可以通过下面的代码段配置用户名和密码：

```
// 创建BccClientConfiguration实例
BccClientConfiguration config = new BccClientConfiguration();

// 配置代理为本地8080端口
config.setProxyHost("127.0.0.1");
config.setProxyPort(8080);

//设置用户名和密码
config.setProxyUsername(<username>);                        //用户名
config.setProxyPassword(<password>);                        //密码
```

设置网络参数

用户可以用BccClientConfiguration对基本网络参数进行设置：

```
BccClientConfiguration config = new BccClientConfiguration();

// 设置HTTP最大连接数为10
config.setMaxConnections(10);

// 设置TCP连接超时为5000毫秒
config.setConnectionTimeout(5000);

// 设置Socket传输数据超时的时间为2000毫秒
config.setSocketTimeout(2000);
```

参数说明

通过BccClientConfiguration能指定的所有参数如下表所示：

参数	说明
UserAgent用户代理，指HTTP的User-Agent头	
Protocol	连接协议类型
ProxyDomain	访问NTLM验证的代理服务器的Windows域名
ProxyHost	代理服务器主机地址
ProxyPort	代理服务器端口
ProxyUsername	代理服务器验证的用户名
ProxyPassword	代理服务器验证的密码
ProxyPreemptiveAuthenticationEnabled	是否设置用户代理认证
ProxyWorkstation	NTLM代理服务器的Windows工作站名称
LocalAddress	本地地址
ConnectionTimeoutInMillis	建立连接的超时时间（单位：毫秒）
SocketTimeoutInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）
MaxConnections	允许打开的最大HTTP连接数
RetryPolicy	连接重试策略
SocketBufferSizeInBytes	Socket缓冲区大小
StreamBufferSize	流文件缓冲区大小

BCC实例

BCC实例是百度智能云上的云服务器操作实例，也是计费、权限控制等高级功能的管理实体。

创建实例（通过指定实例套餐规格）

使用以下代码可以创建BCC实例，包括普通型BCC、存储优化型BCC、计算优化型BCC、大数据机型BCC、GPU机型BCC、FPGA机型BCC：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.CreateCdsModel;
import com.baidubce.services.bcc.model.Reservation;
import com.baidubce.services.bcc.model.StorageType;
import com.baidubce.services.bcc.model.instance.CreateInstanceRequest;
import com.baidubce.services.bcc.model.instance.CreateInstanceResponse;
```

```
import java.util.ArrayList;
import java.util.List;

import static com.baidubce.services.bcc.model.instance.FpgaCardType.KU115;
import static com.baidubce.services.bcc.model.instance.GpuCardType.P4;

public class Main {
    public static void main(String[] args) {
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        BccClient client = new BccClient(bccClientConfiguration);

        CreateInstanceRequest instanceRequest = new CreateInstanceRequest();
        // 镜像ID
        instanceRequest.setImageId("m-***");
        // 待创建虚拟机实例的类型
        instanceRequest.setSpec("bcc.g5.c1m4");
        // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        instanceRequest.setRootDiskSizeInGb(20);
        // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        instanceRequest.setRootDiskStorageType("enhanced_ssd_pl1");
        // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        instanceRequest.setZoneName("cn-bj-a");
        // 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        instanceRequest.setName("instance-***");
        List<CreateCdsModel> createCdsList = new ArrayList<CreateCdsModel>();
        createCdsList.add(new CreateCdsModel()
                .withCdsSizeInGB(50)
                .withStorageType("enhanced_ssd_pl1"));
        // 待创建的CDS磁盘列表
        instanceRequest.setCreateCdsList(createCdsList);
        // 公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
        instanceRequest.setNetworkCapacityInMbps(10);
        // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
        instanceRequest.setPurchaseCount(1);
        // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        instanceRequest.setHostname("hostn***");
        // 是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
        instanceRequest.setAutoSeqSuffix(true);
        // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        instanceRequest.setAdminPass("***");
        // 订单、计费相关参数
        instanceRequest.setBilling(new Billing()
                .withPaymentTiming("Postpaid"));
        // 指定subnet信息，为空时将使用默认子网
        instanceRequest.setSubnetId("sbn-***");
        // 指定securityGroup信息，为空时将使用默认安全组
        instanceRequest.setSecurityGroupId("g-***");
        // 待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
        instanceRequest.setRelationTag(true);
        // 自动快照策略ID
        instanceRequest.setAspId("asp-***");
```

```
        // 指定实例所在的部署集id
        instanceRequest.setDeployId("dset-***");

        CreateInstanceResponse instance = client.createInstance(instanceRequest);
        System.out.println(instance.getInstanceIds());
    }
}
```

1. 创建BCC请求是一个异步请求，返回200表明订单生成，后续可以通过查询返回的实例id信息了解BCC虚机的创建进度。
2. 本接口用于创建一个或多个同配虚拟机实例。
3. 创建实例需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。
4. 创建计费方式为后付费的实例需要账户现金余额+通用代金券大于100；预付费方式的实例则需要账户现金余额大于等于实例费用。
5. 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
6. 缺省情形下，一个实例最多只能挂载5个云磁盘。
7. 创建CDS磁盘和临时数据盘时，磁盘容量大小限制为5的倍数。
8. 创建实例支持创建和添加临时数据盘，但不支持单独创建或添加临时数据盘。
9. 临时数据盘不支持挂载、卸载、删除。
10. 普通实例的临时数据盘最大不能超过500G。
11. 指定子网和安全组创建，要求子网和安全组必须同时指定或同时不指定，同时指定的子网和安全组必须同属于一个VPC，都不指定会使用默认子网和默认安全组。
12. 指定公网IP带宽创建，计费方式为按照带宽计费。
13. 创建接口为异步创建，可通过查询实例详情接口查询实例状态
14. 每个实例最多只能购买一块临时数据盘。
15. 实例的临时数据盘默认只有hp1类型。
16. 创建存储优化型实例必须购买临时数据盘，通过ephemeralDisks指定临时盘数据盘大小，默认nvme类型数据盘，无需指定。
17. 创建请求详细使用请参考BCC API 文档创建实例（通过指定实例套餐规格）)

创建实例（已废弃，建议使用“创建实例（通过指定实例套餐规格）”接口创建）

使用以下代码可以创建BCC实例，包括专属实例、普通型Ⅰ型实例、普通型Ⅱ型实例、存储优化型BCC、计算优化型BCC：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.CreateCdsModel;
import com.baidubce.services.bcc.model.Reservation;
import com.baidubce.services.bcc.model.StorageType;
```

```
import com.baidubce.services.bcc.model.instance.CreateInstanceRequest;
import com.baidubce.services.bcc.model.instance.CreateInstanceResponse;

import java.util.ArrayList;
import java.util.List;

import static com.baidubce.services.bcc.model.instance.FpgaCardType.KU115;
import static com.baidubce.services.bcc.model.instance.GpuCardType.P4;

public class Main {
    public static void main(String[] args) {
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        BccClient client = new BccClient(bccClientConfiguration);

        CreateInstanceRequest instanceRequest = new CreateInstanceRequest();
        // 镜像ID
        instanceRequest.setImageId("m-***");
        // 待创建虚拟机实例的类型
        instanceRequest.setInstanceType("N6");
        // 待创建虚拟机实例的CPU核数
        instanceRequest.setCpuCount(1);
        // 待创建虚拟机实例的内存容量，单位GB
        instanceRequest.setMemoryCapacityInGB(4);
        // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        instanceRequest.setRootDiskSizeInGb(20);
        // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        instanceRequest.setRootDiskStorageType("enhanced_ssd_pl1");
        // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        instanceRequest.setZoneName("cn-bj-a");
        // 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        instanceRequest.setName("instance-***");
        List<CreateCdsModel> createCdsList = new ArrayList<CreateCdsModel>();
        createCdsList.add(new CreateCdsModel()
                .withCdsSizeInGB(50)
                .withStorageType("enhanced_ssd_pl1"));
        // 待创建的CDS磁盘列表
        instanceRequest.setCreateCdsList(createCdsList);
        // 公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
        instanceRequest.setNetworkCapacityInMbps(10);
        // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
        instanceRequest.setPurchaseCount(1);
        // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        instanceRequest.setHostname("hostn***");
        // 是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
        instanceRequest.setAutoSeqSuffix(true);
        // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        instanceRequest.setAdminPass("***");
        // 订单、计费相关参数
        instanceRequest.setBilling(new Billing()
                .withPaymentTiming("Postpaid"));
```

```
        // 指定subnet信息，为空时将使用默认子网
        instanceRequest.setSubnetId("sbn-***");
        // 指定securityGroup信息，为空时将使用默认安全组
        instanceRequest.setSecurityGroupId("g-***");
        // 待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
        instanceRequest.setRelationTag(true);
        // 自动快照策略ID
        instanceRequest.setAspId("asp-***");
        // 指定实例所在的部署集id
        instanceRequest.setDeployId("dset-***");

        CreateInstanceResponse instance = client.createInstance(instanceRequest);
        System.out.println(instance.getInstanceIds());
    }
}
```

1. 创建BCC请求是一个异步请求，返回200表明订单生成，后续可以通过查询返回的实例id信息了解BCC虚机的创建进度。
2. 本接口用于创建一个或多个同配虚拟机实例。
3. 创建实例需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。
4. 创建计费方式为后付费的实例需要账户现金余额+通用代金券大于100；预付费方式的实例则需要账户现金余额大于等于实例费用。
5. 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
6. 缺省情形下，一个实例最多只能挂载5个云磁盘。
7. 创建CDS磁盘和临时数据盘时，磁盘容量大小限制为5的倍数。
8. 创建实例支持创建和添加临时数据盘，但不支持单独创建或添加临时数据盘。
9. 临时数据盘不支持挂载、卸载、删除。
10. 普通实例的临时数据盘最大不能超过500G。
11. 指定子网和安全组创建，要求子网和安全组必须同时指定或同时不指定，同时指定的子网和安全组必须同属于一个VPC，都不指定会使用默认子网和默认安全组。
12. 指定公网IP带宽创建，计费方式为按照带宽计费。
13. 创建接口为异步创建，可通过查询实例详情接口查询实例状态
14. 可通过该接口指定专属服务器创建实例，专属实例不参与计费。专属实例只能通过ephemeralDisks创建临时盘并指定磁盘类型。
15. 每个实例最多只能购买一块临时数据盘。
16. 实例的临时数据盘默认只有hp1类型。
17. 通过instanceType字段指定需要创建的虚机类型，目前API支持创建的虚机类型参见下述InstanceType。参数(instanceType，cpuCount，memoryCapacityInGB)可以确定需要的机型以及配置。
18. 创建存储优化型实例必须购买临时数据盘，通过ephemeralDisks指定临时盘数据盘大小，默认nvme类型数据盘，无需指定。
19. 创建请求详细使用请参考BCC API 文档创建实例

20. 创建FPGA BCC虚机需要使用指定的(CPU、内存、本地数据盘、FPGA卡类型以及专用镜像), 详细请参考BCC API 文档FPGA型BCC可选规格配置

21. 创建GPU BCC虚机需要使用指定的(CPU、内存、本地数据盘、GPU卡类型), 详细请参考BCC API 文档GPU型BCC可选规格配置

创建竞价实例

使用以下代码可以创建BCC竞价实例，包括普通型 I 型实例、普通型 III 型实例：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.CreateCdsModel;
import com.baidubce.services.bcc.model.instance.CreateInstanceRequest;
import com.baidubce.services.bcc.model.instance.CreateInstanceResponse;

import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        BccClient client = new BccClient(bccClientConfiguration);

        CreateInstanceRequest instanceRequest = new CreateInstanceRequest();
        // 镜像ID
        instanceRequest.setImageId("m-***");
        // 待创建虚拟机实例的类型
        instanceRequest.setSpec("bcc.g5.c1m4");
        // 抢占实例出价模型， 市场价: "market" 自定义："custom"。
        instanceRequest.setBidModel("market");
        // 待创建虚拟机实例的CPU核数
        instanceRequest.setCpuCount(1);
        // 待创建虚拟机实例的内存容量，单位GB
        instanceRequest.setMemoryCapacityInGB(4);
        // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        instanceRequest.setRootDiskSizeInGb(20);
        // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        instanceRequest.setRootDiskStorageType("enhanced_ssd_pl1");
        // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        instanceRequest.setZoneName("cn-bj-a");
        // 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        instanceRequest.setName("instance-***");
        List<CreateCdsModel> createCdsList = new ArrayList<CreateCdsModel>();
        createCdsList.add(new CreateCdsModel()
```

```
                .withCdsSizeInGB(50)
                .withStorageType("enhanced_ssd_pl1"));
        // 待创建的CDS磁盘列表
        instanceRequest.setCreateCdsList(createCdsList);
        // 公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
        instanceRequest.setNetworkCapacityInMbps(10);
        // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
        instanceRequest.setPurchaseCount(1);
        // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        instanceRequest.setHostname("host***");
        // 是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
        instanceRequest.setAutoSeqSuffix(true);
        // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        instanceRequest.setAdminPass("***");
        // 订单、计费相关参数
        instanceRequest.setBilling(new Billing()
                .withPaymentTiming("Postpaid"));
        // 指定subnet信息，为空时将使用默认子网
        instanceRequest.setSubnetId("sbn-***");
        // 指定securityGroup信息，为空时将使用默认安全组
        instanceRequest.setSecurityGroupId("g-***");
        // 待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
        instanceRequest.setRelationTag(true);
        // 自动快照策略ID
        instanceRequest.setAspId("asp-***");
        // 指定实例所在的部署集id
        instanceRequest.setDeployId("dset-***");

        CreateInstanceResponse instance = client.createBidInstance(instanceRequest);
        System.out.println(instance.getInstanceIds());
    }
}
```

1. 创建BCC请求是一个异步请求，返回200表明订单生成，后续可以通过查询返回的竞价实例id信息了解BCC虚机的创建进度。
2. 本接口用于创建一个或多个同配虚拟机竞价实例。
3. 创建竞价实例需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。
4. 创建计费方式为后付费的竞价实例需要账户现金余额+通用代金券大于100；预付费方式的竞价实例则需要账户现金余额大于等于竞价实例费用。
5. 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
6. 缺省情形下，一个实例最多只能挂载5个云磁盘。
7. 创建CDS磁盘和临时数据盘时，磁盘容量大小限制为5的倍数。
8. 普通实例的临时数据盘最大不能超过500G。
9. 指定子网和安全组创建，要求子网和安全组必须同时指定或同时不指定，同时指定的子网和安全组必须同属于一个VPC，都不指定会使用默认子网和默认安全组。
10. 指定公网IP带宽创建，计费方式为按照带宽计费。
11. 创建接口为异步创建，可通过查询实例详情接口查询实例状态）

查询竞价实例套餐

以下代码可以查询BCC虚机竞价套餐

```
public static void listBidFlavor(BccClient bccClient) {
    // 执行查询竞价实例套餐操作
    ListBccBidFlavorResponse listBccBidFlavorResponse = bccClient.listBidFlavor();

    for(ZoneResource zoneResource: listBccBidFlavorResponse.getZoneResources()) {
        System.out.println(zoneResource.getZoneName());
    }
}
```

查询竞价实例市场价

以下代码可以查询竞价实例市场价

```
public static void getBidInstancePrice(BccClient bccClient) {
    GetBidInstancePriceRequest getBidInstancePriceRequest = new GetBidInstancePriceRequest();
    // 设置待查询的竞价实例类型为普通型BCC Ⅰ型
    getBidInstancePriceRequest.setInstanceType(InstanceType.N1.name());
    // 设置待查询的竞价实例类型为普通型BCC Ⅲ型
    getBidInstancePriceRequest.setInstanceType(InstanceType.N3.name());
    // 设置待查询的竞价实例的CPU核数
    getBidInstancePriceRequest.setCpuCount(cpuCount);
    // 设置待查询的竞价实例的内存大小GB
    getBidInstancePriceRequest.setMemoryCapacityInGB(memoryCapacityInGB);
    // 设置待查询的竞价实例的系统盘大小
    getBidInstancePriceRequest.setRootDiskSizeInGb(rootDiskSizeInGb);
    // 设置待查询的竞价实例的系统盘介质
    getBidInstancePriceRequest.setRootDiskStorageType(rootDiskStorageType);

    List<CreateCdsModel> createCdsModelList = new ArrayList<CreateCdsModel>();
    // 新建空白的CDS数据盘，类型为高性能型
    createCdsModelList.add(new CreateCdsModel().withStorageType(StorageType.hp1).withCdsSizeInGB(5));
    // 设置待查询的竞价实例的CDS磁盘列表
    getBidInstancePriceRequest.setCreateCdsList(createCdsList);
    // 设置待查询的竞价实例的购买个数
    getBidInstancePriceRequest.setPurchaseCount(purchaseCount);

    // 执行查询竞价实例市场价的操作
    GetBidInstancePriceResponse getBidInstancePriceResponse =
bccClient.getBidInstancePrice(getBidInstancePriceRequest);

    // 获取待查询的竞价实例总价
    System.out.println(getBidInstancePriceResponse.getMoney());
    // 获取待查询的竞价实例购买个数
    System.out.println(getBidInstancePriceResponse.getCount());
    // 获取待查询的竞价实例的单个价格
    System.out.println(getBidInstancePriceResponse.getPerMoney());
}
```

取消竞价实例订单

以下代码可以取消竞价实例订单

```
    public static void cancelBidOrder(BccClient bccClient) {
        CancelBidOrderRequest cancelBidOrderRequest = new CancelBidOrderRequest();
        // 设置待取消的竞价实例订单ID
        cancelBidOrderRequest.setOrderId();
        // 执行取消竞价实例订单操作
        CancelBidOrderResponse cancelBidOrderResponse = client.cancelBidOrder(cancelBidOrderRequest);
        cancelBidOrderResponse.getOrderId();
    }
```

查询实例列表

以下代码可以查询BCC虚机实例列表,支持通过内网ip、专属服务器id、可用区名称进行筛选:

```
    import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.InstanceModel;
import com.baidubce.services.bcc.model.instance.ListInstancesRequest;
import com.baidubce.services.bcc.model.instance.ListInstancesResponse;

public class ListInstances {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListInstancesRequest listInstancesRequest = new ListInstancesRequest();
        // 批量获取列表的查询的起始位置，是一个由系统生成的字符串
        listInstancesRequest.setMarker("i-***");
        // 每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
        listInstancesRequest.setMaxKeys(1);
        // 内网IP
        listInstancesRequest.setInternalIp("192.168.***.***");
        // 专属服务器ID
        listInstancesRequest.setDedicatedHostId("d-***");
        // 可用区信息
        listInstancesRequest.setZoneName("cn-bj-d");
        // 多个标签，逗号分割，格式：tagKey:tagValue 或 tagKey
        listInstancesRequest.setKeypairId("***");
        ListInstancesResponse listInstancesResponse = client.listInstances(listInstancesRequest);

        for (InstanceModel instanceModel : listInstancesResponse.getInstances()) {
            System.out.println(instanceModel);
        }
    }
}
```

查询可关机不计费的实例列表

使用以下代码可以查询可关机不计费的实例列表，支持通过内网ip、密钥对id、可用区名称进行筛选:

```
  public static void getInstanceNoChargeList(BccClient bccClient) {
    ListGetInstanceNoChargeRequest listGetInstanceNoChargeRequest = listGetInstanceNoChargeRequest();
    // 设置分页标志
    listGetInstanceNoChargeRequest.setMarker(instanceId);
    // 设置分页返回数据大小，默认为1000，最大数量不超过1000
    listGetInstanceNoChargeRequest.setMaxKeys(maxKeys);
    // 通过internalIp进行查询
    listGetInstanceNoChargeRequest.setInternalIp(internalIp);
    // 通过keypairId进行查询
    listGetInstanceNoChargeRequest.setKeypairId(keypairId);
    // 通过可用区名称进行查询
    listGetInstanceNoChargeRequest.setZoneName(zoneName);
    // 执行查询可关机不计费的实例列表的接口
    ListInstancesResponse response = bccClient.getInstanceNoChargeList(listGetInstanceNoChargeRequest);

    for(InstanceModel instanceModel: response.getInstances()){
        System.out.println(instanceModel.getId());
    }
}
```

查询实例详情

使用以下代码可以查询指定BCC虚机的详细信息：

```
  public static void getInstanceDetail(BccClient bccClient) {
    // 查询指定BCC虚机的详细信息
    InstanceModel instanceModel = bccClient.getInstance(instanceId).getInstance();
    System.out.println(instanceModel.getName());
}
```

根据实例ID查询实例列表

```
  import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListInstanceByIdsRequest;
import com.baidubce.services.bcc.model.instance.ListInstancesResponse;
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration().withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 根据实例ID查询实例列表
        ListInstanceByIdsRequest request = new ListInstanceByIdsRequest()
                .withInstanceIds(Arrays.asList("i-sZcdKAZR", "i-CH47zOoW"));
        ListInstancesResponse response = client.listInstanceByIds(request);
        System.out.println(response);
    }
}
```

启动实例

使用以下代码可以启动指定BCC虚机实例：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;

public class StartInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 设置BCC虚机id
        String instanceId = "i-***";
        // 执行修改BCC操作
        client.startInstance(instanceId);
    }
}
```

> 接口调用成功后实例进入 Starting 状态。 实例状态必须为 Stopped，调用此接口才可以成功返回，否则提示409错误。

停止实例

使用以下代码可以停止指定BCC虚机实例：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.StopInstanceRequest;

public class StopInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        StopInstanceRequest stopInstanceRequest = new StopInstanceRequest();
        // 设置BCC虚机id
        stopInstanceRequest.setInstanceId("i-***");
        // 设置是否强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据
        stopInstanceRequest.setForceStop(Boolean.FALSE);
        // 设置是否关机不计费，TRUE为关机不计费，FALSE为关机计费。注意：只有白名单用户才可以实行关机不计费
        stopInstanceRequest.setStopWithNoCharge(Boolean.FALSE);
        // 执行BCC虚机停止操作
        client.stopInstance(stopInstanceRequest);
    }
}
```

- 系统后台会在实例实际 Stop 成功后进入“已停止”状态。
- 只有状态为 Running 的实例才可以进行此操作，否则提示 409 错误。
- 实例支持强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。

批量停止实例

使用以下代码可以批量停止指定BCC虚机实例：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import java.util.Arrays;
import com.baidubce.services.bcc.model.instance.BatchStopInstanceRequest;

public class StopInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        BatchStopInstanceRequest stopInstanceRequest = new BatchStopInstanceRequest();
        // 设置BCC虚机ids，列表元素不超过100个
        stopInstanceRequest.setInstanceIds(Arrays.asList("i-KEmEve**","i-0nPI9W**"));
        // 设置是否强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据
        stopInstanceRequest.setForceStop(Boolean.FALSE);
        // 设置是否关机不计费，TRUE为关机不计费，FALSE为关机计费。注意：只有白名单用户才可以实行关机不计费
        stopInstanceRequest.setStopWithNoCharge(Boolean.FALSE);
        // 执行BCC虚机停止操作
        client.batchStopInstance(stopInstanceRequest);
    }
}
```

- 系统后台会在实例实际 Stop 成功后进入“已停止”状态。
- 只有状态为 Running 的实例才可以进行此操作，否则提示 409 错误。
- 实例支持强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。

重启实例

使用以下代码可以重启指定BCC虚机实例：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.RebootInstanceRequest;

public class RebootInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 设置BCC虚机id
        String instanceId = "i-***";
        // 执行修改BCC操作
        client.rebootInstance(new RebootInstanceRequest().withInstanceId(instanceId));
    }
}
```

- 只有状态为 Running 的实例才可以进行此操作，否则提示 409 错误。
- 接口调用成功后实例进入 Starting 状态。
- 支持强制重启，强制重启等同于传统服务器的断电重启，可能丢失实例操作系统中未写入磁盘的数据。

修改实例密码

使用以下代码可以修改指定虚机的管理员密码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ModifyInstancePasswordRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ModifyInstancePasswordRequest modifyInstancePasswordRequest = new ModifyInstancePasswordRequest();
        // 设置BCC虚机id
        modifyInstancePasswordRequest.setInstanceId("i-***");
        // 设置BCC虚机新密码
        modifyInstancePasswordRequest.setAdminPass("***");

        bccClient.modifyInstancePassword(modifyInstancePasswordRequest);
    }
}
```

> 只有 Running 和 Stopped 状态的实例才可以用调用接口，否则提示 409 错误。

修改实例属性

修改实例属性

使用以下代码可以修改指定虚机的属性，目前只支持修改BCC虚机的名称属性:

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ModifyInstanceAttributesRequest;

public class ModifyAttributeInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 设置BCC虚机id
        String instanceId = "i-***";
        // 设置该虚机的描述
        String newName = "newInstanceName";
        // 执行修改BCC操作
        client.modifyInstanceAttributes(
                new ModifyInstanceAttributesRequest()
                        .withInstanceId(instanceId)
                        .withName(newName)
        );
    }
}
```

修改实例描述

使用以下代码可以修改特定实例描述信息：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ModifyInstanceDescRequest;

public class ModifyDescInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 设置BCC虚机id
        String instanceId = "i-***";
        // 设置该虚机的描述
        String newDesc = "modify desc";
        // 执行修改BCC操作
        client.modifyInstanceDesc(
                new ModifyInstanceDescRequest()
                        .withInstanceId(instanceId)
                        .withDesc(newDesc)
        );
    }
}
```

> 只有 Running 和 Stopped 状态的实例才可以调用该接口，否则提示 409 错误。

重装实例

使用以下代码可以使用镜像重建指定BCC虚机实例:

```
package com.baidubce.services.bcc;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.RebuildInstanceRequest;

public class RebuildInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        RebuildInstanceRequest req = new RebuildInstanceRequest();
        // 指定的实例ID
        req.setInstanceId("i-***");
        // 指定的镜像ID
        req.setImageId("m-***");
        // 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())，密码需要加密传输
        req.setAdminPass("***");
        // 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
        req.setKeypairId("k-***");

        client.rebuildInstance(req);
    }
}
```

> 实例重装后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

重装实例(批量重装)

使用以下代码可以使用镜像批量重建指定BCC虚机实例:

```
package com.baidubce.services.bcc;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.RebuildBatchInstanceRequest;

import java.util.ArrayList;
import java.util.List;

public class BatchRebuildInstances {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        RebuildBatchInstanceRequest req = new RebuildBatchInstanceRequest();
        List<String> instanceIds = new ArrayList<String>();
        instanceIds.add("i-***");
        instanceIds.add("i-***");

        // 指定的实例ID列表，最多100个
        req.setInstanceIds(instanceIds);
        // 指定的镜像ID
        req.setImageId("m-***");
        // 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())，密码需要加密传输
        req.setAdminPass("***");
        // 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
        req.setKeypairId("k-***");

        client.rebuildBatchInstance(req);
    }
}
```

> 实例重装后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

计费变更-转预付费

以下代码用于将实例计费模式由后付费转预付费

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ChangeToPrepaidRequest;
import com.baidubce.services.bcc.model.instance.ChangeToPrepaidResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ChangeToPrepaidRequest changeToPrepaidRequest = new ChangeToPrepaidRequest();
        changeToPrepaidRequest.setInstanceId("i-IC39****"); // 虚机ID
        changeToPrepaidRequest.setDuration(2);  // 预付费时长，单位为月
        ChangeToPrepaidResponse response = client.changeToPrepaid(changeToPrepaidRequest);
        System.out.println(response.getOrderId()); // 订单ID
    }
}
```

释放实例

对于后付费Postpaid以及预付费Prepaid过期的BCC虚机实例，可以使用以下代码将其释放:

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        client.releaseInstance("i-eRnq****");   // 待释放实例ID
    }
}
```

- 释放单个云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。

- 只有付费类型为Postpaid或者付费类型为Prepaid且已过期的实例才可以释放。
- 实例释放后，已挂载的CDS磁盘自动卸载，，基于此CDS磁盘的快照会保留。
- 实例释放后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

释放实例(POST请求的释放)

- 释放单个云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。
- 释放的时候默认只释放实例和系统盘，用户可以选择是否关联释放当前时刻，实例挂载的eip+数据盘（只能统一释放完或者不释放。而不是挂载的数据盘释放一个或者多个）是否一起释放。

```
 package com.baidubce.services.bcc;
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ReleaseInstanceByPostRequest;
public class ReleaseInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        ReleaseInstanceByPostRequest req = new ReleaseInstanceByPostRequest();
        // 实例id
        req.setInstanceId("i-***");
        // 是否关联释放当前时刻，实例挂载的EIP和数据盘
        req.setRelatedReleaseFlag(true);
        // 是否释放云磁盘快照
        req.setDeleteCdsSnapshotFlag(true);
        client.releaseInstanceByPost(req);
        }
    }
```

批量释放实例

- 批量释放云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。
- 释放的时候默认只释放实例和系统盘，用户可以选择是否关联释放当前时刻，实例挂载的eip+数据盘（只能统一释放完或者不释放。而不是挂载的数据盘释放一个或者多个）是否一起释放。

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ReleaseMultipleInstancesRequest;
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);
        ReleaseMultipleInstancesRequest releaseMultipleInstancesRequest = new ReleaseMultipleInstancesRequest();
        // 设置待释放的实例id列表，列表元素不超过100个
        List<String> instanceIds = new ArrayList<String>();
        instanceIds.add("i-***");
        instanceIds.add("i-***");
        releaseMultipleInstancesRequest.setInstanceIds(instanceIds);
        // 是否关联释放当前时刻，实例挂载的EIP和数据盘
        releaseMultipleInstancesRequest.setRelatedReleaseFlag(true);
        // 是否释放云磁盘快照
        releaseMultipleInstancesRequest.setDeleteCdsSnapshotFlag(true);
        // 实例释放时是否进入回收站
        releaseMultipleInstancesRequest.setBccRecycleFlag(true);
        // 实例释放时是否删除关联的ENI
        releaseMultipleInstancesRequest.setDeleteRelatedEnisFlag(true);
        bccClient.releaseMultipleInstanceByPost(releaseMultipleInstancesRequest);
    }
}
```

释放预付费实例(POST)

释放预付费云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。

释放的时候默认只释放云服务器实例和系统盘，用户可以选择是否关联释放云服务器实例挂载的EIP和数据盘，关联的EIP和数据盘需全部释放或全部不释放，不支持选择部分释放，部分不释放。

当选择关联释放EIP和数据盘，云磁盘快照可选择一同关联释放，若数据盘不关联释放，云磁盘快照也不支持关联释放。

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ReleasePrepaidInstanceRequest;
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);
        ReleasePrepaidInstanceRequest releasePrepaidInstanceRequest = new ReleasePrepaidInstanceRequest();
        // 设置待释放的实例id列表
        releasePrepaidInstanceRequest.setInstanceId("i-***");
        // 是否关联释放当前实例挂载的EIP和数据盘(只有该字段为true时 deleteCdsSnapshotFlag字段才会有效，若该字段为false,deleteCdsSnapshotFlag字段的值无效)
        releasePrepaidInstanceRequest.setRelatedReleaseFlag(true);
        // 是否释放云磁盘快照
        releasePrepaidInstanceRequest.setDeleteCdsSnapshotFlag(true);
        // 实例释放时是否删除关联的ENI
        releasePrepaidInstanceRequest.setDeleteRelatedEnisFlag(true);
        bccClient.releasePrepaidInstanceByPost(releasePrepaidInstanceRequest);
    }
}
```

实例扩缩容

支持对BCC虚机的CPU以及内存进行扩缩容操作，具体可参考以下代码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ResizeInstanceRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ResizeInstanceRequest resizeInstanceRequest = new ResizeInstanceRequest();
        resizeInstanceRequest.setInstanceId("i-IC39****");  //待变配实例ID
        resizeInstanceRequest.setCpuCount(4);   //CPU核数
        resizeInstanceRequest.setMemoryCapacityInGB(16);  //内存大小,单位GB
        resizeInstanceRequest.setEnableJumboFrame(false); //设置是否开启Jumbo帧。true表示开启Jumbo帧，false表示关闭Jumbo帧。默认为空，为空表示：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
        client.resizeInstance(resizeInstanceRequest);
    }
}
```

- 实例计费方式为预付费时，不能进行缩容操作
- 实例计费方式为后付费时，可弹性扩缩容
- 只有实例Running或Stopped状态时可以进行扩缩容操作
- 实例扩缩容之后会重启一次
- 异步接口，可通过查询实例详情接口查看扩缩容状态是否完成
- 专属实例可以通过指定的cpu、内存以及临时盘大小，专属实例临时盘大小只支持扩容而不支持缩容，具体请参考API文档 实例扩缩容

实例扩缩容（通过指定实例套餐规格）

为指定虚机扩缩容，目前支持通过更新实例套餐规格对虚机的CPU以及内存进行扩缩容操作。

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.ResizeInstanceRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ResizeInstanceRequest resizeInstanceRequest = new ResizeInstanceRequest();
        // 设置BCC虚机id
        resizeInstanceRequest.setInstanceId("i-***");
        // 设置新的套餐规格
        resizeInstanceRequest.setSpec("bcc.g5.c4m16");
        // 设置是否开启Jumbo帧。true表示开启Jumbo帧，false表示关闭Jumbo帧。默认为空，为空表示：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
        resizeInstanceRequest.setEnableJumboFrame(false);
        bccClient.resizeInstanceBySpec(resizeInstanceRequest);
    }
}
```

- 只有实例Running或Stopped状态时可以进行扩缩容操作
- 实例扩缩容之后会重启一次
- 异步接口，可通过查询实例详情接口查看扩缩容状态是否完成

将实例加入安全组

以下代码可以将指定BCC虚机实例加入到指定安全组内：

```
public static void bindInstanceToSecurityGroup(BccClient bccClient) {
    BindSecurityGroupRequest bindSecurityGroupRequest = new BindSecurityGroupRequest();
    // 设置安全组id
    bindSecurityGroupRequest.setSecurityGroupId(securityGroupId);
    // 设置BCC虚机实例id
    bindSecurityGroupRequest.setInstanceId(instanceId);
    // 执行将实例加入安全组操作
    bccClient.bindInstanceToSecurityGroup(bindSecurityGroupRequest);
}
```

每个实例最多关联10个安全组。

将实例移出安全组

以下代码可以将指定实例移出指定安全组中：

```
 public static void unbindInstanceFromSecurityGroup(BccClient bccClient) {
   UnbindSecurityGroupRequest unbindSecurityGroupRequest = new UnbindSecurityGroupRequest();
   // 设置安全组id
   unbindSecurityGroupRequest.setSecurityGroupId(securityGroupId);
   // 设置BCC虚机实例id
   unbindSecurityGroupRequest.setInstanceId(instanceId);
   // 执行将实例移出安全组操作
   bccClient.unbindInstanceFromSecurityGroup(unbindSecurityGroupRequest);
}
```

- 每个实例至少关联一个安全组，默认关联默认安全组。
- 如果实例仅属于一个安全组，尝试移出时，请求会报 403 错。

查询实例VNC地址

使用以下代码可以查询指定BCC虚机实例的VNC URL:

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.GetInstanceVncResponse;

public class GetInstanceVnc {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // i-*** 为虚拟机实例ID
        GetInstanceVncResponse response = client.getInstanceVnc("i-***");
        System.out.println(response.getVncUrl());
    }
}
```

- VNC地址一次使用后即失效
- URL地址有效期为10分钟

实例续费

对BCC虚机的续费操作，可以延长过期时长，以下代码可以对磁盘进行续费：

```
package com.baidubce.services.bcc;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.Reservation;
import com.baidubce.services.bcc.model.instance.PurchaseReservedInstanceRequeset;
import com.baidubce.services.bcc.model.instance.PurchaseReservedInstanceResponse;

public class PurchaseReservedInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        PurchaseReservedInstanceRequeset req = new PurchaseReservedInstanceRequeset();
        // 虚拟机实例ID
        req.setInstanceId("i-***");
        // 幂等性Token，是一个长度不超过64位的ASCII字符串
        req.setClientToken("random-uuid");
        // 订单详情
        req.setBilling(new Billing()
                // 续费方式，本接口针对预付费实例进行续费
                .withPaymentTiming("Prepaid")
                // 续费时长和单位，续费单位为月
                .withReservation(new Reservation()
                // 续费时长
                .withReservationLength(1))
        );
        // 设置实例关联续费标识，默认为空字符串
        req.setRelatedRenewFlag("CDS");

        PurchaseReservedInstanceResponse res = client.purchaseReservedInstance(req);
        System.out.println(res);
    }
}
```

- BCC虚机实例扩缩容期间不能进行续费操作。
- 续费时若实例已欠费停机，续费成功后有个BCC虚机实例启动的过程。
- 该接口是一个异步接口。
- 专属实例不支持续费。

实例绑定标签

以下代码用于实例绑定标签

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.TagModel;
import com.baidubce.services.bcc.model.instance.BindTagsRequest;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        BindTagsRequest bindTagsRequest = new BindTagsRequest();
        // 设置想要绑定的标签
        List<TagModel> changeTags = new ArrayList<TagModel>();
        TagModel tagModel = new TagModel().withTagKey("Key***").withTagValue("Value***");
        changeTags.add(tagModel);
        bindTagsRequest.setChangeTags(changeTags);
        // 设置对应的实例ID
        bindTagsRequest.setInstanceId("i-***");

        bccClient.bindInstanceToTags(bindTagsRequest);
    }
}
```

- 给实例绑定标签，用于分类和查找。

实例解绑标签

以下代码用于实例解绑标签

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.TagModel;
import com.baidubce.services.bcc.model.instance.UnbindTagsRequest;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        UnbindTagsRequest unbindTagsRequest = new UnbindTagsRequest();
        // 设置想要绑定的标签
        List<TagModel> changeTags = new ArrayList<TagModel>();
        TagModel tagModel = new TagModel().withTagKey("Key***").withTagValue("Value***");
        changeTags.add(tagModel);
        unbindTagsRequest.setChangeTags(changeTags);
        // 设置对应的实例ID
        unbindTagsRequest.setInstanceId("i-***");

        bccClient.unbindInstanceFromTags(unbindTagsRequest);
    }
}
```

- 解绑实例上定义的标签。
- 可以同时解绑多个标签。

实例变更子网

```
package com.baidubce.services.bcc;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ChangeInstanceSubnetRequest;

public class ChangeSubnet {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new
BccClientConfiguration().withProtocol(Protocol.HTTP).withCredentials(new DefaultBceCredentials(ak,
sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ChangeInstanceSubnetRequest req = new ChangeInstanceSubnetRequest();
        // 虚拟机实例ID
        req.setInstanceId("i-***");
        // 变更后的子网ID
        req.setSubnetId("sbn-***");
        // 变更后是否重启
        req.setReboot(true);
        // 变更普通安全组，仅变更vpc时生效,不支持和EnterpriseSecurityGroupIds同时设置
        req.setSecurityGroupIds(Collections.singletonList("g-***"));
        // 变更企业安全组，仅变更vpc时生效,不支持和SecurityGroupIds同时设置
        req.setEnterpriseSecurityGroupIds(Collections.singletonList("esg-***"));
        client.updateInstanceSubnet(req);
    }
}
```

- 变更实例所属的子网。 变更子网后默认自动重启，用户选择是否执行该操作。 变更子网的范围目前仅支持在同AZ下变更子网，不支持跨AZ或跨VPC变更子网，如果从普通子网变更至NAT专属子网请先手动解绑EIP

查询实例套餐规格

查询实例资源套餐规格列表信息。创建虚机时建议使用参数(spec)指定需要的机型以及配置：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.flavor.ListFlavorSpecRequest;
import com.baidubce.services.bcc.model.flavor.ListBccFlavorSpecResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ListFlavorSpecRequest listFlavorSpecRequest = new ListFlavorSpecRequest();
        // 设置可用区
        listFlavorSpecRequest.setZoneName("cn-bj-d");

        ListBccFlavorSpecResponse response = bccClient.listFlavorSpec(listFlavorSpecRequest);
        System.out.println(response);
    }
}
```

- 默认获取当前地域下所有可用区的实例套餐规格，可通过设置可用区获取指定可用区下的实例套餐规格。

查询实例套餐价格

查询实例资源套餐规格对应的价格。：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.BccPriceRequest;
import com.baidubce.services.bcc.model.instance.BccPriceResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        BccPriceRequest bccPriceRequest = new BccPriceRequest();
        // 设置实例规格族
        bccPriceRequest.setSpecId("g5");
        // 设置实例套餐规格
        bccPriceRequest.setSpec("bcc.g5.c2m8");
        // 设置可用区
        bccPriceRequest.setZoneName("cn-bj-d");
        // 设置计费方式
        bccPriceRequest.setPaymentTiming("prepay");
        // 设置购买时长（单位：月）
        bccPriceRequest.setPurchaseLength(2);
        // 设置购买数量
        bccPriceRequest.setPurchaseCount(2);

        BccPriceResponse response = bccClient.getPriceBySpec(bccPriceRequest);
        System.out.println(response);
    }
}
```

- 设置实例规格族会获取当前规格族下所有套餐的价格
- 设置实例套餐规格获取当前套餐的价格
- 付费方式，后付费：Postpaid，预付费：Prepaid

查询可变配规格

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListAvailableResizeSpecRequest;
import com.baidubce.services.bcc.model.instance.ListAvailableResizeSpecResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListAvailableResizeSpecRequest request = new ListAvailableResizeSpecRequest()
                .withSpec("bcc.ic4.c1m1").withZone("cn-bj-a");
        ListAvailableResizeSpecResponse response = client.listAvailableResizeSpec(request);
        System.out.println(response);
    }
}
```

获取实例网卡信息

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListInstanceEnisRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 设置实例ID
        ListInstanceEnisRequest request = new ListInstanceEnisRequest().withInstanceId("i-CH47****");
        System.out.println(client.listRecycleInstance());
    }
}
```

实例变更vpc

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.model.AbstractBceResponse;
import com.baidubce.services.bcc.model.instance.ChangeVpcRequest;
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 变更vpc
        ChangeVpcRequest request = new ChangeVpcRequest()
                .withInstanceId("i-sZcd****")
                .withSubnetId("sbn-3rgdxbbn****")
                .withSecurityGroupIds(Arrays.asList("g-63vdycz*****"));
        AbstractBceResponse response = client.changeVpc(request);
        System.out.println(response.getMetadata().getBceRequestId());
    }
}
```

- 开机状态下实例重启，关机状态下实例不重启

批量转包年包月

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchChangeToPrepaidRequest;
import com.baidubce.services.bcc.model.instance.BatchChangeToPrepaidResponse;
import com.baidubce.services.bcc.model.instance.ChangeToPrepaidRequest;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 指定计费变更的虚机ID和包年包月时长（单位：月），虚机个数不超过100个
        BatchChangeToPrepaidRequest request = new BatchChangeToPrepaidRequest();
        List<ChangeToPrepaidRequest> configList = new ArrayList<>();
        configList.add(new ChangeToPrepaidRequest().withInstanceId("i-sZcdKAZR").withDuration(1));
        configList.add(new ChangeToPrepaidRequest().withInstanceId("i-CH47zOoW").withDuration(1));
        request.setConfig(configList);

        BatchChangeToPrepaidResponse response = client.batchChangeToPrepaid(request);
        System.out.println(response);
    }
}
```

查询回收站实例列表

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListInstancesResponse;
import com.baidubce.services.bcc.model.instance.ListRecycleInstanceRequest;
import java.util.Calendar;
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 查询指定时间范围内进入回收站的后付费实例
        Date today = new Date();
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(today);
        calendar.add(Calendar.DATE, -1);
        Date lastDay = calendar.getTime();
        ListRecycleInstanceRequest request = new ListRecycleInstanceRequest()
                .withRecycleBegin(lastDay)
                .withRecycleEnd(today);
                .withPaymentTiming(PaymentTiming.Postpaid)
        ListInstancesResponse response = client.listRecycleInstance(request);
        System.out.println(response);
    }
}
```

释放回收站实例

使用以下代码可以从回收站释放指定实例

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListInstancesResponse;
import com.baidubce.services.bcc.model.instance.ListRecycleInstanceRequest;
import java.util.Calendar;
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        client.deleteRecycledInstance("i-***");
    }
}
```

删除实例与部署集之间的关系

使用以下代码可以删除实例与部署集之间的关系

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.ListInstancesResponse;
import com.baidubce.services.bcc.model.instance.ListRecycleInstanceRequest;
import java.util.Calendar;
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        DeleteInstanceDeploysetRequest deleteInstanceDeploysetRequest = new DeleteInstanceDeploysetRequest();
        deleteInstanceDeploysetRequest
                .withDeployId("dset-NXyUzFou")
                .withInstanceIdList(Collections.singletonList("i-pKq2Bnhf"));
        client.deleteInstanceDeploySet(deleteInstanceDeploysetRequest);
    }
}
```

添加Ipv6

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.InstanceIpv6Request;
import com.baidubce.services.bcc.model.instance.InstanceAddIpv6Response;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 添加ipv6
        InstanceIpv6Request request = new InstanceIpv6Request();
        request.setInstanceId("i-0nPl9WFJ");
        request.setIpv6Address("2400:da00:e003:0:41c:4100:0:5");
        request.setReboot(false);

        InstanceAddIpv6Response response = client.addIpv6(request);
        System.out.println(response);
    }
}
```

释放Ipv6

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.InstanceIpv6Request;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 释放ipv6
        InstanceIpv6Request request = new InstanceIpv6Request();
        request.setInstanceId("i-0nPl9WFJ");
        request.setIpv6Address("2400:da00:e003:0:41c:4100:0:5");
        request.setReboot(false);
        client.delIpv6(request);
    }
}
```

批量添加主网卡辅助IP

使用以下代码可以为BCC批量增加主网卡辅助IP

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchAddIpRequest;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 为主网卡分配ipv6，IP个数不超过100个
        BatchAddIpRequest request = new BatchAddIpRequest();
        request.setInstanceId("i-udU7HS0a");
        request.setAllocateMultiIpv6Addr(true);
        request.setSecondaryPrivateIpAddressCount(2);
        client.batchAddIp(request);
    }
}
```

批量删除主网卡辅助IP

使用以下代码可以为BCC批量删除主网卡辅助IP

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchDeleteIpRequest;

import java.util.Collections;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 为主网卡释放ipv6，IP个数不超过100个
        InstanceIpv6Request request = new InstanceIpv6Request();
        request.setInstanceId("i-0nPl9WFJ");
        request.setPrivateIps(Collections.singletonList("2400:da00:e003:0:41c:4100:0:5"));
        client.batchDeleteIp(request);
    }
}
```

创建自动续费规则

使用以下代码可以为BCC创建自动续费规则

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchDeleteIpRequest;

import java.util.Collections;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建自动续费规则
        client.createAutoRenewRule(new BccAutoRenewRequest()
                    .withInstanceId("i-dI2s537H")
                    .withRenewTimeUnit("month")
                    .withRenewTime(1)
                    .withRenewEip(true)
                    .withRenewCds(true));
    }
}
```

删除自动续费规则

使用以下代码可以为BCC删除自动续费规则

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchDeleteIpRequest;

import java.util.Collections;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 删除自动续费规则
        client.deleteAutoRenewRule(new BccAutoRenewRequest()
                    .withInstanceId("i-dI2s537H")
                    .withRenewEip(true)
                    .withRenewCds(true));
    }
}
```

查询实例自定义数据

使用以下代码可以查询实例自定义数据

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.instance.BatchDeleteIpRequest;

import java.util.Collections;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 设置请求体以及发送请求
        GetInstanceUserDataAttrRequest request = new GetInstanceUserDataAttrRequest();
        request.withInstanceId("i-b34ycow2");
        GetInstanceUserDataAttrResponse response = client.getInstanceUserDataAttribute(request);
        toJsonPrettyString("getInstanceUserDataAttribute", response);

    }
}
```

磁盘

创建CDS磁盘

支持新建空白CDS磁盘或者从CDS数据盘快照创建CDS磁盘，参考以下代码可以创建CDS磁盘：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.StorageType;
import com.baidubce.services.bcc.model.volume.CreateVolumeRequest;
import com.baidubce.services.bcc.model.volume.CreateVolumeResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        CreateVolumeRequest request = new CreateVolumeRequest()

                .withChargeType("Postpaid") // 计费方式
                .withCdsSizeInGB(5) // 磁盘大小
                .withStorageType(StorageType.hdd.name()) // 存储类型
                .withPurchaseCount(2) // 购买数量
                .withZoneName("cn-bj-c"); // 可用区
        CreateVolumeResponse response = client.createVolume(request);
        System.out.println(response.getVolumeIds());
    }
}
```

从快照创建CDS磁盘

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.Reservation;
import com.baidubce.services.bcc.model.StorageType;
import com.baidubce.services.bcc.model.volume.CreateVolumeRequest;
import com.baidubce.services.bcc.model.volume.CreateVolumeResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        CreateVolumeRequest request = new CreateVolumeRequest()

                .withChargeType("Postpaid") // 计费方式
                        .withReservation(new Reservation().withReservationLength(2).withReservationTimeUnit("month")))
                .withSnapshotId("s-4CMA****")  //快照ID
                .withPurchaseCount(1)     // 创建数量
                .withZoneName("cn-bj-a"); // 可用区;
        CreateVolumeResponse response = client.createVolume(request);
        System.out.println(response.getVolumeIds());
    }
}
```

> 创建CDS磁盘接口为异步接口，可通过查询磁盘详情接口查询磁盘状态，详细接口使用请参考BCC API 文档查询磁盘详情

查询磁盘列表

以下代码可以查询所有的磁盘列表，不包含临时数据盘，支持分页查询以及通过云磁盘所挂载的BCC实例ID进行过滤筛选:

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.ListVolumesRequest;
import com.baidubce.services.bcc.model.volume.ListVolumesResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListVolumesRequest listVolumesRequest = new ListVolumesRequest()
                .withZoneName("cn-bj-c") // 可用区
                .withInstanceId("i-IC39****") // 实例ID
                .withMarker("v-ePrp****")   // 起始卷ID
                .withMaxKeys(10);   // 最大返回数量
        ListVolumesResponse response = client.listVolumes(listVolumesRequest);
        System.out.println(response);
    }
}
```

查询磁盘详情

通过磁盘ID可以获取对应磁盘的详细信息，以下代码可以查询磁盘详情：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.GetVolumeRequest;
import com.baidubce.services.bcc.model.volume.GetVolumeResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        GetVolumeRequest getVolumeRequest = new GetVolumeRequest()
                .withVolumeId("v-ePrp****");  // 指定volume id
        GetVolumeResponse response = client.getVolume(getVolumeRequest);
        System.out.println(response);
    }
}
```

挂载CDS磁盘

可以将未挂载的云磁盘挂载在对应的BCC实例上，以下代码将一个CDS云磁盘挂载在对应的BCC实例上:

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.instance.StopInstanceRequest;
import com.baidubce.services.bcc.model.volume.AttachVolumeRequest;

public class AttachVolume {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        AttachVolumeRequest attachVolumeRequest = new AttachVolumeRequest();
        // 设置需要挂载的磁盘id
        attachVolumeRequest.setVolumeId("v-***");
        // 设置BCC虚机id
        attachVolumeRequest.setInstanceId("i-***");
        // 打印挂载路径
        System.out.println(client.attachVolume(attachVolumeRequest).getVolumeAttachment().getDevice());
    }
}
```

- CDS磁盘需要挂载在与其处在相同zone下的BCC实例上，否则将返回403错误。
- 只有磁盘状态为 Available 且实例状态为 Running 或 Stopped 时才允许挂载，否则调用此接口将返回 409 错误。

卸载CDS磁盘

可以将已挂载的磁盘从对应的BCC实例上卸载下来，以下代码可以支持卸载CDS磁盘:

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.DetachVolumeRequest;

public class DetachVolume {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk)).withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        DetachVolumeRequest detachVolumeRequest = new DetachVolumeRequest();
        // 设置待卸载的磁盘对应的ID
        detachVolumeRequest.setVolumeId("v-***");
        // 设置所挂载的实例ID
        detachVolumeRequest.setInstanceId("i-***");
        // 执行卸载操作
        client.detachVolume(detachVolumeRequest);


    }
}
```

- 只有实例状态为 Running 或 Stopped 时，磁盘才可以执行此操作，否则将提示 409 错误。
- 如果 volumeId 的磁盘不挂载在 instanceId 的实例上，该操作失败，提示 404 错误。

释放CDS磁盘

如下代码可以释放未挂载的CDS磁盘：

- 可以指定是否删除磁盘关联的快照，缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 可以指定是否将磁盘放入回收站，缺省情况下默认进入回收站。

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.ReleaseVolumeRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 释放磁盘
        // 请求参数
        ReleaseVolumeRequest request = new ReleaseVolumeRequest();
        // 磁盘ID
        request.setVolumeId("v-***");
        // 是否删除磁盘关联的自动快照
        request.setAutoSnapshot("on");
        // 是否删除磁盘关联的手动快照
        request.setManualSnapshot("on");
        // 是否将磁盘放入回收站，为off时直接释放，不进入回收站
        request.setRecycle("off");
        client.releaseVolume(request);
    }
}
```

- 已挂载的CDS磁盘不能释放，系统盘不能释放。
- 磁盘释放后不可恢复。缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误。

修改磁盘属性

可以使用以下代码修改指定磁盘名称、描述信息：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.ModifyCdsAttrRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ModifyCdsAttrRequest modifyCdsAttributeRequest = new ModifyCdsAttrRequest;
        // 设置磁盘id
        modifyCdsAttributeRequest.setCdsId("v-dj5Q****");
        // 设置新的磁盘名称
        modifyCdsAttributeRequest.setCdsName("volumeName");
        // 设置磁盘新的描述
        modifyCdsAttributeRequest.setDesc("volumeDesc");
        client.modifyCdsAttribute(modifyCdsAttributeRequest);
    }
}
```

磁盘计费变更

可以使用以下代码变更磁盘计费方式，仅支持按量付费（后付费）转包年包月（预付费）、包年包月（预付费）转按量付费（后付费）两种方式。变更为包年包月（预付费）需要指定购买时长。

> 通过以下代码可将包年包月（预付费）磁盘变更为按量付费（后付费）：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 将预付费磁盘变更为后付费磁盘
        // 请求参数
        ModifyVolumeChargeRequest request = new ModifyVolumeChargeRequest();
        // 磁盘ID
        request.setVolumeId("v-***");
        // 计费方式，当前取值为postpay, 可不填
        request.setBillingMethod("postpay");
        client.modifyVolumeChargeType(request);
    }
}
```

通过以下代码可以将按量付费（后付费）磁盘变更为包年包月（预付费）：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 将后付费磁盘变更为预付费磁盘
        // 请求参数
        ModifyVolumeChargeRequest request = new ModifyVolumeChargeRequest();
        // 磁盘ID
        request.setVolumeId("v-***");
        // 计费方式，当前取值为prepay, 可不填
        request.setBillingMethod("prepay");
        // 变更为prepay时，指定预付费时长，当前支持付费单位为月，取值范围为1-9
        request.setReservationLength(1);
        client.modifyVolumeChargeType(request);
    }
}
```

磁盘扩容及变配

使用以下代码可以对磁盘进行扩大容量或变更磁盘类型操作：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.ResizeVolumeRequest;

import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求变更磁盘类型和大小
        ResizeVolumeRequest request = new ResizeVolumeRequest();
        // 磁盘ID
        request.setVolumeId("v-***");
        // 磁盘大小, 单位为GB
        request.setNewCdsSizeInGB(50);
        // 磁盘类型, 取值范围见https://cloud.baidu.com/doc/BCC/s/6jwvyo0q2#storagetype
        request.setNewVolumeType("cloud_hp1");
        client.resizeVolume(request);
    }
}
```

> 磁盘只能进行扩大容量，不支持缩小容量。 只有Available状态的磁盘，才能进行扩容操作。 磁盘扩容是一个异步接口，可通过查询磁盘详情接口查询磁盘扩容状态。

回滚磁盘

可以使用指定磁盘自身的快照回滚磁盘内容，使用以下代码可以对磁盘进行回滚：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.RollbackVolumeRequest;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        RollbackVolumeRequest rollbackVolumeRequest = new RollbackVolumeRequest();
        // 设置磁盘id
        rollbackVolumeRequest.setVolumeId("v-zBms****");
        // 设置磁盘的快照id
        rollbackVolumeRequest.setSnapshotId("s-HkbJ****");
        // 执行回滚操作
        client.rollbackVolume(rollbackVolumeRequest);
    }
}
```

- 磁盘状态必须为 Available 才可以执行回滚磁盘操作。
- 指定快照id必须是由该磁盘ID创建的快照。
- 若是回滚系统盘，实例状态必须为 Running 或 Stopped 才可以执行此操作。
- 回滚系统盘快照，自本次快照以来的系统盘数据将全部丢失，不可恢复。

磁盘续费

对磁盘的续费操作，可以延长过期时长，以下代码可以对磁盘进行续费：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.Billing;
import com.baidubce.services.bcc.model.Reservation;
import com.baidubce.services.bcc.model.volume.PurchaseReservedVolumeRequest;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        PurchaseReservedVolumeRequest purchaseReservedVolumeRequest = new PurchaseReservedVolumeRequest();
        // 设置磁盘id
        purchaseReservedVolumeRequest.setVolumeId("v-dj5Q****");
        // 设置billing信息
        Billing billing = new Billing();
        // 设置续费时长，单位目前只支持'月'为单位
        billing.setReservation(new Reservation().withReservationLength(1));
        billing.setPaymentTiming("Prepaid");
        // 设置billing信息
        purchaseReservedVolumeRequest.setBilling(billing);
        // 执行续费磁盘操作
        client.purchaseReservedVolume(purchaseReservedVolumeRequest);
    }
}
```

磁盘开通自动续费

如下代码可以为指定磁盘开通自动续费：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.AutoRenewVolumeRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        AutoRenewVolumeRequest req = new AutoRenewVolumeRequest();
        // 磁盘ID
        req.setVolumeId("v-***");
        // 续费时间单位，当前支持取值为：month
        req.setRenewTimeUnit("month");
        // 续费时间，以month为单位时，取值为1-9
        req.setRenewTime(3);

        client.autoRenewVolume(req);
    }
}

}
```

- 注意数据盘才能进行自动续费操作
- 注意过期状态磁盘不能进行自动续费操作
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间

磁盘取消自动续费

如下代码可以为指定磁盘取消自动续费

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.CancelAutoRenewVolumeRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        CancelAutoRenewVolumeRequest req = new CancelAutoRenewVolumeRequest();
        req.setVolumeId("v-***");              // 磁盘ID
        client.cancelAutoRenewVolume(req);
    }
}
```

- 注意数据盘才能进行取消自动续费操作
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间

磁盘重命名

如下代码可以用于修改指定磁盘名称：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.volume.RenameVolumeRequest;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        RenameVolumeRequest renameVolumeRequest = new RenameVolumeRequest();
        // 待重命名的磁盘ID
        renameVolumeRequest.setVolumeId("v-1YgO****");
        // 磁盘新的名称，自定义镜像名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
        renameVolumeRequest.setName("newName");

        client.renameVolume(renameVolumeRequest);
    }
}
```

镜像

创建自定义镜像

支持通过实例创建和通过快照创建两种方式。参考以下代码可以创建一个自定义镜像：

```
 // 通过实例创建自定义镜像
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.CreateImageRequest;
import com.baidubce.services.bcc.model.volume.AttachVolumeRequest;

public class CreateImageFromInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        CreateImageRequest createImageRequest = new CreateImageRequest();
        // 设置自定义镜像名称
        createImageRequest.setImageName("image04");
        // 当从快照创建镜像时，此参数是指用于创建镜像的快照ID。可选参数，与 snapshotId 不同时存在，同时存在时只取instanceId
        createImageRequest.setSnapshotId("m-***");
        // 当从实例创建镜像时，此参数是指用于创建镜像的实例ID。可选参数，与 snapshotId 不同时存在，同时存在时只取instanceId
        createImageRequest.setInstanceId("i-***");
        // 设置自定义镜像是否关联实例CDS磁盘,relateCds默认为false，为true时表示关联
        createImageRequest.setRelateCds(true);
        System.out.println(client.createImage(createImageRequest).getImageId());
    }
}
```

> 注意，创建自定义镜像，默认配额20个每账号。

查询镜像列表

如下代码可以查询有权限的镜像列表:

```
package com.baidu.bce.image;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.ListImagesRequest;
import com.baidubce.services.bcc.model.image.ListImagesResponse;

public class ListImages {

    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListImagesRequest request = new ListImagesRequest();
        // 镜像类型
        request.withImageType("ALL");
        ListImagesResponse response = client.listImages(request);
        System.out.println(response);
    }
}
```

> 具体的镜像类型可详细参考BCC API文档查询镜像列表

查询镜像详情

如下代码可以查询镜像详细信息：

```
package com.baidu.bce.image;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.GetImageRequest;
import com.baidubce.services.bcc.model.image.GetImageResponse;

public class GetImage {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 镜像ID
        String imageId = "m-***";
        GetImageRequest request = new GetImageRequest();
        request.setImageId(imageId);

        GetImageResponse response = client.getImage(request);
        System.out.println(response);
    }
}
```

删除自定义镜像

如下代码可以删除一个自定义镜像：

```
package com.baidu.bce.image;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.DeleteImageRequest;

public class DeleteImage {

    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 镜像ID
        String imageId = "m-***";
        DeleteImageRequest request = new DeleteImageRequest();
        request.setImageId(imageId);

        client.deleteImage(request);
    }
}
```

跨地域复制自定义镜像

如下代码用于用户跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制

regions如北京"bj",广州"gz",苏州"su"，可多选：

```
package com.baidu.bce.image;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.RemoteCopyImageRequest;

import java.util.Arrays;
import java.util.List;

public class RemoteCopyImage {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 镜像ID
        String imageId = "m-***";

        // 镜像名称
        String name = "copied-image";

        // 目的region
        List<String> destinations = Arrays.asList("hkg");
        RemoteCopyImageRequest request = new RemoteCopyImageRequest();
        request.setImageId(imageId);
        request.setDestRegion(destinations);
        request.setName(name);

        client.remoteCopyImage(request);
    }
}
```

取消跨地域复制自定义镜像

用于取消跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制：

```
package com.baidu.bce.image;

import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.CancelRemoteCopyImageRequest;

public class CancelRemoteCopyImage {

    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 镜像ID
        String imageId = "m-***";

        CancelRemoteCopyImageRequest request = new CancelRemoteCopyImageRequest();
        request.setImageId(imageId);

        client.cancelRemoteCopyImage(request);
    }
}
```

共享自定义镜像

如下代码可以共享用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能共享：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.ShareImageRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求将镜像共享给其他账号id
        ShareImageRequest shareRequest = new ShareImageRequest();
        // 共享账号ID
        shareRequest.setAccountId("***");
        // 镜像ID
        shareRequest.setImageId("m-***");

        client.shareImage(shareRequest);
    }
}
```

取消共享自定义镜像

如下代码可以取消共享用户自己的指定的自定义镜像：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.UnShareImageRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 取消将镜像共享给某个账户id
        UnShareImageRequest unShareRequest = new UnShareImageRequest();
        // 账号ID
        unShareRequest.setAccountId("***");
        // 镜像ID
        unShareRequest.setImageId("m-***");

        client.unShareImage(unShareRequest);
    }
}
```

查询镜像已共享用户列表

如下代码可以查询镜像已共享的用户列表：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.CreateImageRequest;
import com.baidubce.services.bcc.model.image.ListSharedUserResponse;

public class ListSharedUser {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 镜像id
        String imageId = "m-***";
        ListSharedUserResponse listSharedUserResponse = client.listSharedUser(imageId);
        System.out.println(listSharedUserResponse.getUsers());
    }
}
```

根据实例短ID批量查询OS信息(新增)

如下代码可以根据实例ID批量查询OS信息：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.image.ListOsRequest;
import com.baidubce.services.bcc.model.image.ListOsResponse;
import com.baidubce.services.bcc.model.image.ListSharedUserResponse;

import java.util.ArrayList;
import java.util.List;

public class listOs {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListOsRequest listOsRequest = new ListOsRequest();

        List<String> ids = new ArrayList<String>();

        // 设置需要查询的实例ID
        ids.add("i-***");
        // 设置需要查询的实例ID
        ids.add("i-***");

        listOsRequest.setInstanceIds(ids);
        ListOsResponse listOsResponse = client.listOs(listOsRequest);
        System.out.println(listOsResponse.getOsInfo());
    }
}
```

根据实例规格查询可用公共镜像

如下代码可以根据实例规格查询可用公共镜像：

```
public static void getAvailableImagesBySpec(BccClient bccClient) {
    GetAvailableImagesBySpecRequest request = new GetAvailableImagesBySpecRequest();
    request.setSpec("bcc.g4.c1m1");
    bccClient.getAvailableImagesBySpec(request);
}
```

快照

创建快照

支持创建系统盘快照和CDS磁盘快照，参考如下代码：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.snapshot.CreateSnapshotRequest;

public class CreateSnapshot {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        CreateSnapshotRequest createSnapshotRequest = new CreateSnapshotRequest();
        // 指定磁盘id
        createSnapshotRequest.setVolumeId("v-***");
        // 指定快照名称
        createSnapshotRequest.setSnapshotName("snapshotName");
        // 设置快照描述信息
        createSnapshotRequest.setDesc("desc");
        // 打印返回信息
        System.out.println(client.createSnapshot(createSnapshotRequest));
    }
}
```

查询快照列表

如下代码可以查询快照列表：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.snapshot.ListSnapshotsRequest;

public class ListSnapshot {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListSnapshotsRequest listSnapshotsRequest = new ListSnapshotsRequest();
        // 设置分页标志
        listSnapshotsRequest.setMarker("marker");
        // 设置分页返回数据大小
        listSnapshotsRequest.setMaxKeys(100);
        // 过滤从指定的磁盘查询快照
        listSnapshotsRequest.setVolumeId("v-***");
        System.out.println(client.listSnapshots(listSnapshotsRequest));
    }
}
```

查询快照详情

如下代码可以查询快照详情：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;

public class GetSnapshot {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 快照id
        String snapshotId = "s-***";
        System.out.println(client.getSnapshot(snapshotId));
    }
}
```

删除快照

如下代码可以删除一个快照：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;

public class DeleteSnap {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 快照id
        String snapshotId = "s-***";
        client.deleteSnapshot(snapshotId);
    }
}
```

查询快照链列表

如下代码可以用于查询用户的快照链列表信息：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.model.snapshot.ListSnapchainRequest;
import com.baidubce.services.bcc.model.snapshot.ListSnapchainResponse;

public class ListSnapchain {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        ListSnapchainRequest request = new ListSnapchainRequest();

        // 排序属性，可选: chainId(快照链id,默认值),chainSize(快照链大小),volumeSize(磁盘大小)
        request.setOrderBy("chainId");
        // 排序方式，可选:asc(正序,默认值), desc(倒序)
        request.setOrder("asc");
        // 页数,可选：默认值为1
        request.setPageNo(1);
        // 每页容量，可选：默认值为1000
        request.setPageSize(100);
        // 磁盘id,可选：该字段非空则只返回这个磁盘的快照链信息
        request.setVolumeId("v-***");

        ListSnapchainResponse response = client.listSnapchain(request);
        System.out.println(response);
    }
}
```

自动快照策略

创建自动快照策略

创建自动快照策略，参考如下代码：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.CreateAspRequest;
import com.baidubce.services.bcc.model.asp.CreateAspResponse;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        CreateAspRequest createAspRequest = new CreateAspRequest();
        // 设置快照名
        createAspRequest.setName("test-asp-name");
        // 设置一天中做快照的时间点，取值为0~23，0为午夜12点
        // 例如设置做快照的时间点为凌晨2点和下午2点：
        List<Integer> timePoints = new ArrayList<Integer>();
        timePoints.add(2);
        timePoints.add(14);
        createAspRequest.setTimePoints(timePoints);
        // 设置一周中做快照的时间，取值为0~6，0代表周日，1~6代表周一到周六
        // 例如设置做快照的时间为礼拜五：
        List<Integer> repeatWeekdays = new ArrayList<Integer>();
        repeatWeekdays.add(5);
        createAspRequest.setRepeatWeekdays(repeatWeekdays);
        // 设置自动快照保留天数，取-1则永久保留
        createAspRequest.setRetentionDays("7");

        CreateAspResponse response = bccClient.createAsp(createAspRequest);
        System.out.println(response);
    }
}
```

绑定自动快照策略

绑定自动快照策略，参考如下代码：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.AttachAspRequest;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        AttachAspRequest attachAspRequest = new AttachAspRequest();
        // 设置需要绑定的自动快照策略ID
        attachAspRequest.setAspId("asp-***");
        // 设置需要绑定的磁盘ID
        List<String> volumeIds = new ArrayList<String>();
        volumeIds.add("v-***");
        volumeIds.add("v-***");
        attachAspRequest.setVolumeIds(volumeIds);

        bccClient.attachAsp(attachAspRequest);
    }
}
```

解绑自动快照策略

解绑自动快照策略，参考如下代码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.DetachAspRequest;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        DetachAspRequest detachAspRequest = new DetachAspRequest();
        // 设置需要解绑的自动快照策略ID
        detachAspRequest.setAspId("asp-***");
        // 设置需要解绑的磁盘ID
        List<String> volumeIds = new ArrayList<String>();
        volumeIds.add("v-***");
        volumeIds.add("v-***");
        detachAspRequest.setVolumeIds(volumeIds);

        bccClient.detachAsp(detachAspRequest);
    }
}
```

删除自动快照策略

删除自动快照策略，参考如下代码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.DeleteAspRequest;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        DeleteAspRequest deleteAspRequest = new DeleteAspRequest();
        // 设置需要删除的自动快照策略ID
        deleteAspRequest.setAspId("asp-***");

        bccClient.deleteAsp(deleteAspRequest);
    }
}
```

查询自动快照策略列表

查询自动快照策略列表，参考如下代码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.AutoSnapshotPolicyModel;
import com.baidubce.services.bcc.model.asp.ListAspsRequest;
import com.baidubce.services.bcc.model.asp.ListAspsResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ListAspsRequest listAspsRequest = new ListAspsRequest();
        // 设置需要查找的自动快照策略名称
        listAspsRequest.setAspName("test_stratery");
        // 设置需要查找的自动快照策略ID
        listAspsRequest.setMarker("asp-***");
        // 设置返回数据大小
        listAspsRequest.setMaxKeys(10);

        ListAspsResponse listAspsResponse = bccClient.listAsps(listAspsRequest);
        for (AutoSnapshotPolicyModel asp : listAspsResponse.getAutoSnapshotPolicys()) {
            System.out.println(asp);
        }
    }
}
```

查询自动快照策略详情

查询自动快照策略详情，参考如下代码：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.GetAspRequest;
import com.baidubce.services.bcc.model.asp.GetAspResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        GetAspRequest getAspRequest = new GetAspRequest();
        // 设置需要查找的自动快照策略ID
        getAspRequest.setAspId("asp-***");

        GetAspResponse getAspResponse = bccClient.getAsp(getAspRequest);
        System.out.println(getAspResponse);
    }
}
```

自动快照策略变更

自动快照策略修改，参考如下代码。支持修改的内容：策略名称、创建时间、重复日期、保留时间。

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.asp.UpdateAspRequest;
import java.util.ArrayList;
import java.util.List;

public class UpdateAsp {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        String aspId = "asp-osTov***";
        String name= "changeDown";
        List<String> timePoints = new ArrayList<String>(){{add("12"); add("22");}};
        List<String>  repeatWeekdays = new ArrayList<String>(){{add("1"); add("5");}};
        String retentionDays = "5";
        UpdateAspRequest updateAspRequest = new UpdateAspRequest();
        // 必填，自动策略的id
        updateAspRequest.setAspId(aspId);
        // 必填，自动快照策略名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
        updateAspRequest.setName(name);
        // 必填，一天中做快照时间点，取值为0~23。
        updateAspRequest.setTimePoints(timePoints);
        // 必填，一周中做快照的时间，取值为0~6。
        updateAspRequest.setRepeatWeekdays(repeatWeekdays);
        // 必填，自动快照保留天数，取-1则永久保留。
        updateAspRequest.setRetentionDays(retentionDays);

        client.updateAsp(updateAspRequest);
    }
}
```

安全组

查询安全组列表

以下代码可以查询安全组列表：

```
    public static void listSecurityGroup(BccClient bccClient) {
        ListSecurityGroupsRequest listSecurityGroupsRequest = new ListSecurityGroupsRequest();
        // 设置分页标志
        listSecurityGroupsRequest.setMarker(securityGroupId);
        // 设置分页返回数据大小
        listSecurityGroupsRequest.setMaxKeys(maxKey);
        // 设置筛选的实例Bcc实例id
        listSecurityGroupsRequest.setInstanceId(instanceId);
        // 设计筛选的vpcId
        listSecurityGroupsRequest.setVpcId(vpcId);
        // 执行获取安全组列表操作
        ListSecurityGroupsResponse listSecurityGroupsResponse = bccClient.listSecurityGroups(listSecurityGroupsRequest);

        for (SecurityGroupModel securityGroupModel : listSecurityGroupsResponse.getSecurityGroups()) {
            System.out.println(securityGroupModel.getName());
        }
    }
```

创建安全组

以下代码可以创建一个安全组：

```
public static void createSecurityGroup(BccClient bccClient) {
  CreateSecurityGroupRequest createSecurityGroupRequest = new CreateSecurityGroupRequest();
  // 设置安全组名字
  createSecurityGroupRequest.setName(newName);
  // 设置安全组描述
  createSecurityGroupRequest.setDesc(newDescription);
  List<SecurityGroupRuleModel> rules = new ArrayList<SecurityGroupRuleModel>();
  rules.add(new SecurityGroupRuleModel()
        // 设置备注
        .withRemark(securityGroupRemark)
        // 设置协议类型
        .withProtocol(securityGroupProtocol)
        // 设置端口范围，默认空时为1-65535，可以指定80等单个端口
        .withPortRange(securityGroupPortRange)
        // 设置入站/出站，取值ingress或egress，必选参数
        .withDirection(securityGroupDirection)
        // 设置源IP地址，与sourceGroupId不能同时设定值
        .withSourceIp(securityGroupSourceIp)
        // 设置源安全组ID
        .withSourceGroupId(securityGroupSourceGroupId));

  rules.add(new SecurityGroupRuleModel()
        // 设置备注
        .withRemark(securityGroupRemark)
        // 设置协议类型
        .withProtocol(securityGroupProtocol)
        // 设置端口范围，默认空时为1-65535，可以指定80等单个端口
        .withPortRange(securityGroupPortRange)
        // 设置入站/出站，取值ingress或egress，必选参数
        .withDirection(securityGroupDirection)
        // 设置目的IP地址，与destGroupId不能同时设定值
        .withDestIp(securityGroupDestIp)
        // 设置目的安全组ID
        .withDestGroupId(securityGroupDestGroupId));
  // 设置安全组规则
  createSecurityGroupRequest.setRules(rules);

  List<TagModel> listTags = new ArrayList<TagModel>();
  // 设置想要绑定的标签
  TagModel tagModel = new TagModel().withTagKey("test007").withTagValue("test007");
  listTags.add(tagModel);
  // 设置标签列表
  createSecurityGroupRequest.setTags(listTags);

  // 创建安全组
  System.out.println(bccClient.createSecurityGroup(createSecurityGroupRequest).getSecurityGroupId());
    }
```

同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，重复记录会报409错误。 protocol的取值（tcp|udp|icmp），默认值为空，代表all。 具体的创建安全组规则接口描述BCC API 文档创建安全组。

删除安全组

以下代码可以删除指定的安全组:

```
public static void deleteSecurityGroup(BccClient bccClient) {
    // 删除安全组id为securityGroupId的安全组
    bccClient.deleteSecurityGroup(securityGroupId);
}
```

授权安全组规则

使用以下代码可以在指定安全组中添加授权安全组规则:

```
public static void authorizeSecurityGroupRule(BccClient bccClient) {
    SecurityGroupRuleOperateRequest securityGroupRuleOperateRequest = new SecurityGroupRuleOperateRequest();

    securityGroupRuleOperateRequest.setSecurityGroupId(securityGroupId);
    SecurityGroupRuleModel securityGroupRuleModel = new SecurityGroupRuleModel()
            // 设置备注
            .withRemark(securityGroupRemark)
            // 设置协议类型
            .withProtocol(securityGroupProtocol)
            // 设置端口范围，默认空时为1-65535，可以指定80等单个端口
            .withPortRange(securityGroupPortRange)
            // 设置入站/出站，取值ingress或egress，必选参数
            .withDirection(securityGroupDirection)
            // 设置源IP地址，与sourceGroupId不能同时设定值
            .withSourceIp(securityGroupSourceIp)
            // 设置源安全组ID
            .withSourceGroupId(securityGroupSourceGroupId);

    // 设置安全组规则
    securityGroupRuleOperateRequest.setRule(securityGroupRuleModel);
    // 执行授权安全组规则
    bccClient.authorizeSecurityGroupRule(securityGroupRuleOperateRequest);
}
```

> - 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，重复记录会报409错误。
> - 具体的接口描述BCC API 文档授权安全组规则。

撤销安全组规则

使用以下代码可以在指定安全组中撤销指定安全组规则授权:

```
public static void revokeSecurityGroupRule(BccClient bccClient) {
    SecurityGroupRuleOperateRequest securityGroupRuleOperateRequest = new SecurityGroupRuleOperateRequest();
    // 设置安全组规则id
    securityGroupRuleOperateRequest.setSecurityGroupId(securityGroupId);

    SecurityGroupRuleModel securityGroupRuleModel = new SecurityGroupRuleModel()
            // 设置备注
            .withRemark(securityGroupRemark)
            // 设置协议类型
            .withProtocol(securityGroupProtocol)
            // 设置端口范围，默认空时为1-65535，可以指定80等单个端口
            .withPortRange(securityGroupPortRange)
            // 设置入站/出站，取值ingress或egress，必选参数
            .withDirection(securityGroupDirection)
            // 设置源IP地址，与sourceGroupId不能同时设定值
            .withSourceIp(securityGroupSourceIp)
            // 设置源安全组ID
            .withSourceGroupId(securityGroupSourceGroupId);
    // 执行撤销安全组规则
    bccClient.revokeSecurityGroupRule(securityGroupRuleOperateRequest);
}
```

- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，重复记录将报404错误。
- 具体的接口描述BCC API 文档撤销安全组规则。

部署集

查询部署集列表

如下代码可以查询用户的部署集列表：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.deployset.ListDeploySetResponse;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求查询部署集列表
        ListDeploySetResponse deploySets = client.listDeploySet();
        System.out.println(deploySets);
    }
}
```

创建部署集

如下代码可以创建部署集：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.deployset.CreateDeploySetRequest;
import com.baidubce.services.bcc.model.deployset.CreateDeploySetResponse;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求创建部署集
        CreateDeploySetRequest request = new CreateDeploySetRequest();
        // 部署集名称
        request.setName("your_deploy_set_name");
        // 部署集描述
        request.setDesc("your deploy set description");
        // 部署集策略，默认为HOST_HA
        request.setStrategy("HOST_HA");
        // 部署集并发数，默认为1
        request.setConcurrency(6);
        CreateDeploySetResponse deploySet = client.createDeploySet(request);
        System.out.println(deploySet);
    }
}
```

删除部署集

如下代码可以删除部署集：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.deployset.DeleteDeploySetRequest;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求删除部署集
        DeleteDeploySetRequest request = new DeleteDeploySetRequest();
        // 部署集id
        request.setDeployId("dset-***");
        client.deleteDeploySet(request);
    }
}
```

修改部署集

如下代码可以修改部署集属性：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.deployset.UpdateDeploySetRequest;


import java.util.ArrayList;
import java.util.List;


public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        // 请求修改部署集属性
        UpdateDeploySetRequest request = new UpdateDeploySetRequest();
        // 部署集id
        request.setDeployId("dset-***");
        // 部署集名称
        request.setName("modify_deploy_name");
        // 部署集描述
        request.setDesc("modify description");
        client.updateDeploySet(request);
    }
}
```

密钥对

创建密钥对

如下代码可以创建密钥对，单个用户在单region默认配额500个（创建和导入的密钥对之和）创建后的密钥对可植入实例以实现远程登录实例：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairCreateRequest;
import com.baidubce.services.bcc.model.keypair.KeypairModel;

public class CreateKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        String keyName = "dcc";
        String keyDes = "用于dcc";

        KeypairCreateRequest keypairCreateRequest = new KeypairCreateRequest();
        // 设置密钥名
        keypairCreateRequest.setName(keyName);
        // 设置密钥描述
        keypairCreateRequest.setDescription(keyDes);
        KeypairModel keypairModel = client.createKeypair(keypairCreateRequest);
        System.out.println(keypairModel.getPublicKey());
    }
}
```

导入密钥对

如下代码可以导入创建密钥对：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairImportRequest;
import com.baidubce.services.bcc.model.keypair.KeypairModel;

public class ImportKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        KeypairImportRequest requst = new KeypairImportRequest();
        // 创建密钥对名称
        requst.setName("keypairFromImport");
        // 待创建的密钥对的描述,可选参数
        requst.setDescription("self desc....");
        // 待创建的密钥对的公钥
        requst.setPublicKey("ssh-***");
        KeypairModel keypairModel = client.importKeypair(requst);
        System.out.println(keypairModel.getPublicKey());
    }
}
```

查询密钥对列表

如下代码可以查询密钥对列表：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairListRequest;
import com.baidubce.services.bcc.model.keypair.KeypairListResponse;

public class ListKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建请求体
        KeypairListRequest keypairListRequest = new KeypairListRequest();
        // 每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
        keypairListRequest.setMaxKeys(1);
        // 批量获取列表的查询的起始位置，是一个由系统生成的字符串
        keypairListRequest.setMarker("k-gXZ7LlbS");
        KeypairListResponse keypairListResponse = client.listKeypair(keypairListRequest);
        System.out.println(keypairListResponse.getKeypairs());
    }
}
```

查询密钥对详情

如下代码可以查询单个密钥对的详细信息：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairDetailRequest;
import com.baidubce.services.bcc.model.keypair.KeypairModel;

public class KeypairDetail {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建请求体
        KeypairDetailRequest keypairDetailRequest = new KeypairDetailRequest();
        // 设置被查询密钥对id
        keypairDetailRequest.setKeypairId("k-***");
        // 查询密钥对详情
        KeypairModel keypairModel = client.keypairDetail(keypairDetailRequest);
        System.out.println(keypairModel.getPublicKey());
    }
}
```

绑定密钥对

如下代码可以将所选密钥对（限单个）绑定到所选BCC实例（支持多台），目前一台BCC实例只能绑定一个密钥对：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairAttachRequest;
import java.util.ArrayList;
import java.util.List;

public class AttachKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);

        List<String> instances = new ArrayList<String>();
        // 设置被绑定实例id，可多个
        instances.add("i-***");
        KeypairAttachRequest keypairAttachRequest = new KeypairAttachRequest();
        // 设置被绑定密钥对id
        keypairAttachRequest.setKeypairId("k-***");
        keypairAttachRequest.setInstanceIds(instances);
        client.attachKeypair(keypairAttachRequest);
    }
}
```

解绑密钥对

如下实例可以将所选BCC实例与它们各自绑定的密钥对解绑，此操作仅适用于Linux系统的BCC实例，且所选BCC实例必须处于运行中或关机状态：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairDetachRequest;

import java.util.ArrayList;
import java.util.List;

public class DetachKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建请求体
        KeypairDetachRequest keypairDetachRequest = new KeypairDetachRequest();
        // 被绑密钥ID
        keypairDetachRequest.setKeypairId("k-***");
        List<String> a = new ArrayList<String>();
        a.add("i-***");
        a.add("i-***");
        // 实例
        keypairDetachRequest.setInstanceIds(a);
        // 发起解绑
        client.detachKeypair(keypairDetachRequest);
    }
}
```

删除密钥对

如下代码可以删除密钥对，已经绑定BCC实例的密钥对无法被删除：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairDeleteRequest;

public class DeleteKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 构建请求体
        KeypairDeleteRequest keypairDeleteRequest = new KeypairDeleteRequest();
        // 被删密钥ID
        keypairDeleteRequest.setKeypairId("k-***");
        // 发起删除请求
        client.deleteKeypair(keypairDeleteRequest);
    }
}
```

重命名密钥对

如下代码可以重命名密钥对：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairRenameRequest;

public class RenameKeypair {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建请求体
        KeypairRenameRequest keypairRenameRequest = new KeypairRenameRequest();
        // 待重命名的密钥对id
        keypairRenameRequest.setKeypairId("k-****");
        // 新的密钥对名称
        keypairRenameRequest.setName("Name2");
        // 发起更新秘钥对名称
        client.renameKeypair(keypairRenameRequest);
    }
}
```

更改密钥对描述

如下代码可以更改密钥对描述，若用户提供的新的描述内容为空，则删除密钥对之前的描述。

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.keypair.KeypairDetailRequest;
import com.baidubce.services.bcc.model.keypair.KeypairModel;
import com.baidubce.services.bcc.model.keypair.KeypairUpdateDescRequest;

public class UpdateKeypairDescription {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "http://bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient client = new BccClient(bccClientConfiguration);
        // 创建请求体
        KeypairUpdateDescRequest keypairUpdateDescRequest = new KeypairUpdateDescRequest();
        // 待更改描述的密钥对id
        keypairUpdateDescRequest.setKeypairId("k-***");
        // 新的密钥对描述
        keypairUpdateDescRequest.setDescription("desc");
        // 更新keypair的描述信息
        client.updateKeypairDescription(keypairUpdateDescRequest);
    }
}
```

预留实例券

创建预留实例券

使用以下代码给指定创建预留实例券：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.TagModel;
import com.baidubce.services.bcc.model.reversed.CreateReservedInstanceResponse;
import com.baidubce.services.bcc.model.reversed.CreateReservedInstancesRequest;
import lombok.extern.slf4j.Slf4j;

import java.util.Collections;


public class ExampleCreateReservedInstance {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        CreateReservedInstancesRequest request = new CreateReservedInstancesRequest();
        request.setReservedInstanceName("bj-test");
        request.setScope("AZ");
        request.setZoneName("cn-bj-a");
        request.setSpec("bcc.c5.c2m8");
        request.setOfferingType("FullyPrepay");
        request.setInstanceCount(1);
        request.setReservedInstanceCount(1);
        request.setReservedInstanceTime(1);
        request.setReservedInstanceTimeUnit("month");
        request.setAutoRenewTimeUnit("month");
        request.setAutoRenewTime(1);
        request.setAutoRenew(true);

        TagModel tagModel = new TagModel();
        tagModel.setTagKey("testKey");
        tagModel.setTagValue("testValue");
        request.setTags(Collections.singletonList(tagModel));

        CreateReservedInstanceResponse response = bccClient.createReservedInstances(request);
    }
}
```

修改预留实例券

使用以下代码给指定修改预留实例券属性：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.reversed.ModifyReservedInstancesResponse;
import com.baidubce.services.bcc.model.reversed.ModifyReservedInstancesRequest;
import com.baidubce.services.bcc.model.reversed.ModifyReservedInstancesRequest.ReservedInstance;
import lombok.extern.slf4j.Slf4j;

import java.util.Collections;

public class ExampleModifyReservedInstances {

    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ModifyReservedInstancesRequest request = new ModifyReservedInstancesRequest();

        // 修改名称
        ReservedInstance modifyName = new ReservedInstance();
        modifyName.setReservedInstanceId("r-qt4ltfUP");
        modifyName.setReservedInstanceName("update-reserved-instance");
        request.setReservedInstances(Collections.singletonList(modifyName));

        ModifyReservedInstancesResponse response = bccClient.modifyReservedInstances(request);
    }
}
```

绑定标签

使用以下代码给指定预留实例券绑定标签：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.TagModel;
import com.baidubce.services.bcc.model.reversed.ReservedTagsRequest;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ExampleBindReservedToTags {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ReservedTagsRequest reservedTagsRequest = new ReservedTagsRequest();
        // 标签列表
        List<TagModel> changeTags = new ArrayList<TagModel>();
        TagModel tagModel = new TagModel().withTagKey("Key").withTagValue("Value");
        changeTags.add(tagModel);
        reservedTagsRequest.setChangeTags(changeTags);
        // 预留实例券ID列表，，最多支持100个
        reservedTagsRequest.withInstanceIds(Arrays.asList("r-oFpMXKhv", "r-HrztSVk0"));

        bccClient.bindReservedInstanceToTags(reservedTagsRequest);
    }
}
```

解绑标签

使用以下代码从指定预留实例券中解绑标签：

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.TagModel;
import com.baidubce.services.bcc.model.reversed.ReservedTagsRequest;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ExampleUnbindReservedFromTags {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ReservedTagsRequest reservedTagsRequest = new ReservedTagsRequest();
        // 标签列表
        List<TagModel> changeTags = new ArrayList<TagModel>();
        TagModel tagModel = new TagModel().withTagKey("Key").withTagValue("Value");
        changeTags.add(tagModel);
        reservedTagsRequest.setChangeTags(changeTags);
        // 预留实例券ID列表，最多支持100个
        reservedTagsRequest.withInstanceIds(Arrays.asList("r-oFpMXKhv", "r-HrztSVk0"));

        bccClient.unbindReservedInstanceFromTags(reservedTagsRequest);
    }
}
```

地域

获取域名列表接口

使用以下代码可以获取全部地域或者指定地域的BCC(或CDS、预留实例券)域名列表：

```
 import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.region.DescribeRegionsRequest;
import com.baidubce.services.bcc.model.region.DescribeRegionsResponse;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        String endpoint = "bcc.baidubce.com";
        String ak = "ak";
        String sk = "sk";

        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);

        BccClient client = new BccClient(bccClientConfiguration);

        DescribeRegionsRequest request = new DescribeRegionsRequest();
        // 设置地域，若为空则返回所有地域的域名列表
        request.setRegion("");

        DescribeRegionsResponse response = client.describeRegions(request);
        System.out.println(response.getRegions());
    }
}
```

可用区

查询可用区列表

如下代码可以查询当前region下的可用区（zone）列表信息:

```
import com.baidubce.BceClientConfiguration;
import com.baidubce.Protocol;
import com.baidubce.auth.DefaultBceCredentials;
import com.baidubce.services.bcc.BccClient;
import com.baidubce.services.bcc.BccClientConfiguration;
import com.baidubce.services.bcc.model.zone.ZoneModel;
import com.baidubce.services.bcc.model.zone.ListZonesResponse;

public class Main {
    public static void main(String[] args) {
        // 设置您的ak、sk和要访问的endpoint
        String endpoint = "bcc.bj.baidubce.com";
        String ak = "ak";
        String sk = "sk";
        // 设置默认配置
        BceClientConfiguration bccClientConfiguration = new BccClientConfiguration()
                .withProtocol(Protocol.HTTP)
                .withCredentials(new DefaultBceCredentials(ak, sk))
                .withEndpoint(endpoint);
        // 创建bcc client
        BccClient bccClient = new BccClient(bccClientConfiguration);

        ListZonesResponse listZonesResponse = bccClient.listZones();
        for (ZoneModel zoneModel : listZonesResponse.getZones()) {
            System.out.println(zoneModel.getZoneName());
        }
    }
}
```

日志

BCC Java SDK发布版本中增加了logback作为slf4j的实现，如果用户没有自己的实现可以直接用，如果工程中有其他的如log4j则可以替代。

默认日志

如果用户使用默认的logback，则需要配置logback.xml到classpath中。如果没有这个配置文件，日志级别默认为DEBUG。

```
<configuration>
  <property name="LOG_HOME" value="./log/"/>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <!-- encoders are assigned the type
         ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>

  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <FileNamePattern>${LOG_HOME}/BccUnitTest.%d{yyyy-MM-dd}.log</FileNamePattern>
      <MaxHistory>30</MaxHistory>
    </rollingPolicy>
    <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
    </encoder>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <MaxFileSize>10MB</MaxFileSize>
    </triggeringPolicy>
  </appender>

  <root level="info">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="FILE"/>
  </root>
</configuration>
```

自有日志模块

若用户使用自己的日志实现模块，比如项目依赖于Maven，则可以类似下面的配置到pom.xml中来去除logback。

```
 <?xml version="1.0" encoding="utf-8"?>

<dependency>
  <groupId>com.baidubce</groupId>
  <artifactId>bce-java-sdk</artifactId>
  <version>${bce.sdk.version}</version>
  <exclusions>
    <exclusion>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
    </exclusion>
    <exclusion>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-core</artifactId>
    </exclusion>
    <exclusion>
      <groupId>org.slf4j</groupId>
      <artifactId>jcl-over-slf4j</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

异常处理

BCC异常提示有如下两种方式：

异常方法	说明
BceClientException	客户端异常
BceServerException	服务器异常

用户可以使用try来获取某个事件所产生的异常，例如：

```
 String instanceId = 'instanceId';
try {
    bccClient.getInstance(instanceId);
} catch (BceServiceException bce){
    System.out.println(bce.getMessage());
} catch ( BceClientException bce){
    System.out.println( bce.getMessage());
}
```

版本变更记录

v0.10.31

- Java SDK开发包[2018-05-18] 版本号 0.10.31
  - 首次发布
    - 支持BCC创建实例、查询实例列表、查询实例详情、启动实例、停止实例、重启实例、修改实例密码、修改实例属性、重装实例、释放实例、实例升降配、将实例加入/移出安全组、查询实例VNC地址、实例续费。
    - 磁盘创建、列表、详情、挂载、卸载、释放、扩容、回滚、续费接口。
    - 自定义镜像、列表、详情、删除、共享、取消共享、查询镜像已共享列表。
    - 快照创建、列表、详情、删除接口；安全组创建、列表、 删除、授权、撤销接口。
    - 增加对GPU、FPGA虚机的支持。

v0.10.52

更新时间：2019.4.10

- BCC实例：新支持修改实例描述操作
- 磁盘：释放CDS磁盘现已支持删除磁盘关联的快照；新支持修改磁盘属性、磁盘计费变更
- 镜像：新支持跨区域复制自定义镜像、取消跨区域复制自定义镜像、共享自定义镜像、取消共享自定义镜像、查询镜像已共享用户列表、根据实例ID批量查询OS信息
- 新增自动快照策略相关操作，包括创建、绑定、解绑、删除自动快照策略等
- 新增密钥对相关操作，包括创建、导入密钥对，查看密钥对详情等

v0.10.97

更新时间：2020.03.03

- BCC实例：新增获取实例规格列表，通过实例规格询价，通过实例规格创建实例接口。

Python-SDK

概述

本文档主要介绍云服务器BCC Python SDK的安装和使用。在使用本文档前，您需要先了解云服务器BCC的一些基本知识，并已开通了云服务器BCC服务。若您还不了解云服务器BCC，可以参考产品描述和入门指南。

安装SDK工具包

环境准备

1. 运行环境

   Python SDK支持在Python2.7和Python3的环境下运行。

2. 安装pycrypto依赖

   安装SDK之前，需要先执行命令`pip install pycrypto`安装pycrypto依赖。
   如果安装失败，请执行`pip install pycryptodome`

下载和安装

方式一：通过pip安装

您可以通过pip安装的方式将百度智能云Python SDK安装到您的环境中。 联网状态下，在命令行中执行如下命令：

```
pip install bce-python-sdk
```

即可将Python SDK安装到本地。

方式二：将源码包下载到本地后进行安装

1. 在开发者资源中心下载Python SDK压缩工具包。
2. 命令行移动到压缩包所在路径，执行如下命令（version替换为包名称中的版本号）：

   ```
   pip install bce-python-sdk-version.zip
   ```

   即可将Python SDK安装到本地。

您也可以解压压缩包后执行如下命令（version替换为包名称中的版本号）：

```
cd bce-python-sdk-version

python setup.py install
```

来完成SDK的安装。

SDK目录结构

```
com.baidubce
├─── auth                              //BCE签名相关类
├─── http                              //BCE的Http通信相关类
├─── internal                          //SDK内部类
├─── model                             //BCE公用model类
├─── services
│    └── bcc                          //BCC服务相关类
│        ├── model                    //BCC内部model，如Request或Response
│        ├── BccClient.class          //BCC客户端入口类
│        └── BccClientConfiguration.class    //针对BCC特有的HttpClient的配置
├─── util                              //BCE公用工具类
├─── BceClientConfiguration.class      //对BCE的HttpClient的配置
├─── BceClientException.class          //BCE客户端的异常类
├─── BceServiceException.class         //与BCE服务端交互后的异常类
├─── ErrorCode.class                   //BCE通用的错误码
└─── Region.class      //BCE提供服务的区域
```

卸载SDK

预期卸载 SDK 时，使用pip卸载“bce-python-sdk”即可。

初始化

确认Endpoint

用户若要自定义域名，可以通过传入ENDPOINT参数来指定。区域的概念请参考区域选择说明。

在开始SDK使用之前，需要您先确定好要在哪个地域进行操作，从而在配置BccClient时将地域对应的Endpoint做为参数填入。不指定时默认为北京地域，Endpoint为http://bcc.bj.baidubce.com

BCC目前支持的域名请参考服务域名。

获取密钥

要使用百度BCC，您需要拥有一个有效的 AK（Access Key ID）和SK(Secret Access Key)用来进行签名认证。AK/SK是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。

可以通过如下步骤获得并了解您的AK/SK信息：

注册百度智能云账号

创建AK/SK

获取到密钥后，也需要在配置BccClient时做为参数填入。SDK集成了鉴权认证机制，您不需要关心鉴权背后复杂的运算方法，只需要将AK/SK按要求填入对应的位置，SDK将自动为您完成鉴权相关的工作。

配置文件

```
 #从Python SDK导入BCC配置管理模块以及安全认证模块
import baidubce
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
#导入BCC相关模块
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model
from baidubce.services.bcc import gpu_card_type
from baidubce.services.bcc import fpga_card_type
from baidubce.services.bcc.bcc_model import EphemeralDisk
```

BccClient

BccClient是BCC服务的客户端，为开发者与BCC服务进行交互提供了一系列的方法，包括实例、磁盘、镜像、快照、安全组、可用区等。

通过AK/SK方式访问BCC

用户可以参考如下代码新建一个BccClient：

```
class Sample():
  def setUp(self):
      AK = 'your-access-key-id'  # 用户的Access Key ID
      SK = 'your-secret-access-key'  # 用户的Secret Access Key

      # 初始化一个BccClient
      config = BceClientConfiguration(credentials=BceCredentials(AK, SK))
      client = bcc_client.BccClient(config)
```

在上面代码中，变量AK与SK是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。

其中AK对应控制台中的“Access Key ID”，SK对应控制台中的“Access Key Secret”，获取方式请参考获取AK/SK。

上面的方式使用默认域名（北京区域）作为BCC的服务地址，如果用户需要自己指定域名，可以通过传入endpoint参数来指定。

```
 AK = 'your-access-key-id'  # 用户的Access Key ID
SK = 'your-secret-access-key'  # 用户的Secret Access Key
HOST = 'domain-name'  # 用户自己指定的域名，域名选择可参考下方"Region列表"

#### 初始化一个BccClient
config = BceClientConfiguration(credentials=BceCredentials(AK, SK), endpoint=HOST)
client = bcc_client.BccClient(config)
```

注意： endpoint参数只能用指定的包含区域的域名来进行定义，不指定时默认为北京区域http://bcc.bj.baidubce.com。BCC支持多区域选择，具体参照服务域名。

实例

BCC实例是百度智能云上的云服务器操作实例，也是计费、权限控制等高级功能的管理实体。

创建实例（通过指定实例套餐规格）

使用以下代码可以通过规格参数创建BCC实例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    # billing = bcc_model.Billing(paymentTiming='Postpaid', reservationLength=2, reservationTimeUnit='Month')
    response = client.create_instance_by_spec(
        spec='bcc.g5.c1m4', # 实例规格
        image_id='m-***', # 镜像ID
        root_disk_size_in_gb=20, # 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        root_disk_storage_type='enhanced_ssd_pl1', # 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        create_cds_list=[ # 待创建的CDS磁盘列表
            bcc_model.CreateCdsModel(
                cdsSizeInGB=50, # CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
              # CDS磁盘存储类型，默认是hp1(高性能云磁盘)。 storageType='enhanced_ssd_pl1')],
        eip_name='eip***', # 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
        internet_charge_type='BANDWIDTH_PREPAID', # 公网带宽计费方式，若不指定internetChargeType，默认为按使用带宽计费。
        purchase_count=1, # 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1。
        name='instance-***', # 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        hostname='hostn***', # 虚拟机主机名。。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        admin_pass='***', # 待指定的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        zone_name='cn-bj-a', # 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        subnet_id='sbn-***', # 指定subnet信息，为空时将使用默认子网
        security_group_id='g-***', # 指定securityGroup信息，为空时将使用默认安全组
        asp_id='asp-***', # 自动快照策略ID
        tags=[bcc_model.TagModel('key***', 'value***')], # 待创建虚机的标签列表
    )
    print response
```

- 必传参数：
  - 规格名称（例如 bcc.g4.c2m8）
  - 镜像ID
- 可选参数：参考普通创建实例接口

创建实例（已废弃，建议使用创建实例（通过指定实例套餐规格）接口创建）

使用以下代码可以创建BCC实例，包括专属实例、普通型Ⅰ 型实例、普通型ⅠⅠ型实例、普通型ⅠⅡ型实例、GPU实例、计算优化I型实例、计算优化II型实例、存储优化I型实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    # billing = bcc_model.Billing(paymentTiming='Postpaid', reservationLength=2, reservationTimeUnit='Month')
    response = client.create_instance(
        instance_type='N6', # 待创建虚拟机实例的类型
        cpu_count=1, # 待创建虚拟机实例的CPU核数
        memory_capacity_in_gb=4, # 待创建虚拟机实例的内存容量，单位GB
        image_id='m-***', # 镜像ID
        root_disk_size_in_gb=20, # 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        root_disk_storage_type='enhanced_ssd_pl1', # 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        create_cds_list=[ # 待创建的CDS磁盘列表
            bcc_model.CreateCdsModel(
                cdsSizeInGB=50, # CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
                storageType='enhanced_ssd_pl1')], # CDS磁盘存储类型，默认是hp1(高性能云磁盘)。
        eip_name='eip***', # 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
        internet_charge_type='BANDWIDTH_PREPAID', # 公网带宽计费方式，若不指定internetChargeType，默认为按使用带宽计费。
        purchase_count=1, # 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1。
        name='instance-***', # 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        hostname='hostn***', # 虚拟机主机名。。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        admin_pass='***', # 待指定的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        zone_name='cn-bj-a', # 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        subnet_id='sbn-***', # 指定subnet信息，为空时将使用默认子网
        security_group_id='g-***', # 指定securityGroup信息，为空时将使用默认安全组
        asp_id='asp-***', # 自动快照策略ID
        tags=[bcc_model.TagModel('key***', 'value***')], # 待创建虚机的标签列表
    )
    print response
```

实例类型、GPU类型定义列表请参考列表

在专属服务器上创建实例（支持对明文密码加密）

使用以下代码可以在特定服务器中创建BCC实例：

```
    def create_instance_from_dedicated_host_with_encrypted_password(self):
        #如果用户未指定client_token参数,用uuid4生成一个随机字符串给client_token
        generate_client_token = str(uuid.uuid4())
        client_token = generate_client_token()

        #设置镜像id
        image_id = 'your-choose-image-id'

        #设置专属服务器id
        dedicated_host_id = 'your-choose-dedicated-host-id'

        #设置实例名
        instance_name = 'your-choose-instance-name' + client_token

        #设置实例管理员密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())
        admin_pass = 'your-admin-pass'

        #指定subnet信息，为空时将使用默认子网
        subnet_id = 'your-choose-subnet-id'

        #指定security_group_id信息，为空时将使用默认安全组
        security_group_id = 'your-choose-security-group-id'

        #DCC专属服务器实例可以创建多块本地盘，需要指定磁盘类型及大小

        #其它类型只能用一块本地盘，使用默认类型，只需指定大小

        #设置创建本地磁盘大小
        ephemeral_disk = EphemeralDisk(6144)

        #设置本地盘类型为默认类型
        ephemeral_disks = [ephemeral_disk.__dict__, ephemeral_disk.__dict__]
        self.assertEqual(
            type(self.client.create_instance_from_dedicated_host_with_encrypted_password(1, #设置创建BCC的CPU核数
                                                  2, #设置创建BCC的内存大小GB
                                                  image_id, #设置创建BCC的镜像
                                                  dedicated_host_id, #设置创建BCC的专属服务器
                                                 purchase_count=1,# 批量创建BCC实例时使用，必须为大于0的整数
                                                  name=instance_name,# 设置创建BCC的名称
                                                  admin_pass=admin_pass,# 设置实例管理员密码
                                                 subnet_id=subnet_id,# 指定subnet信息，为空时将使用默认子网
                                                  security_group_id=security_group_id,# 指定securityGroup信息，为空时将使
用默认安全组
                                                  client_token=client_token
                                                  ephemeral_disks) #设置创建BCC的本地盘
            baidubce.bce_response.BceResponse)
```

在专属服务器上创建实例

使用以下代码可以在特定服务器中创建BCC实例：

```
    def create_instance_from_dedicated_host(self):
        #如果用户未指定client_token参数,用uuid4生成一个随机字符串给client_token
        generate_client_token = str(uuid.uuid4())
        client_token = generate_client_token()

        #设置镜像id
        image_id = 'your-choose-image-id'

        #设置专属服务器id
        dedicated_host_id = 'your-choose-dedicated-host-id'

        #设置实例名
        instance_name = 'your-choose-instance-name' + client_token

        #设置实例管理员密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())
        admin_pass = 'your-admin-pass'

        #指定subnet信息，为空时将使用默认子网
        subnet_id = 'your-choose-subnet-id'

        #指定security_group_id信息，为空时将使用默认安全组
        security_group_id = 'your-choose-security-group-id'

        #DCC专属服务器实例可以创建多块本地盘，需要指定磁盘类型及大小

        #其它类型只能用一块本地盘，使用默认类型，只需指定大小

        #设置创建本地磁盘大小
        ephemeral_disk = EphemeralDisk(6144)

        #设置本地盘类型为默认类型
        ephemeral_disks = [ephemeral_disk.__dict__, ephemeral_disk.__dict__]
        self.assertEqual(
            type(self.client.create_instance_from_dedicated_host(1, #设置创建BCC的CPU核数
                                                    2, #设置创建BCC的内存大小GB
                                                    image_id, #设置创建BCC的镜像
                                                    dedicated_host_id, #设置创建BCC的专属服务器
                                                   purchase_count=1,# 批量创建BCC实例时使用，必须为大于0的整数
                                                    name=instance_name,# 设置创建BCC的名称
                                                    admin_pass=admin_pass,# 设置实例管理员密码
                                                   subnet_id=subnet_id,# 指定subnet信息，为空时将使用默认子网
                                                    security_group_id=security_group_id,# 指定securityGroup信息，为空时将使
用默认安全组
                                                    client_token=client_token
                                                    ephemeral_disks) #设置创建BCC的本地盘
            baidubce.bce_response.BceResponse)
```

查询实例列表

以下代码可以查询BCC虚机实例列表,支持通过内网ip、专属服务器id、可用区名称进行筛选:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    `AK = `b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.list_instances(marker='i-***', # 批量获取列表的查询的起始位置，是一个由系统生成的字符串
                                max_keys=1, # 每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
                                internal_ip='192.168.***.***', # 内网IP
                                dedicated_host_id='d-***', # 专属服务器ID
                                zone_name='cn-bj-d', # 可用区信息
                                instance_ids='i-***', # 多个实例ID，英文逗号分割，最多支持100个
                                instance_names='***', # 多个实例名称，英文逗号分割，最多支持100个
                                cds_ids='v-***', # 多个cds磁盘ID，英文逗号分割，最多支持100个
                                deployset_ids='dset-***', # 多个部署集ID，英文逗号分割，最多支持100个
                                security_group_ids='g-***', # 多个安全组ID，英文逗号分割，最多支持100个
                                payment_timing='Postpaid', # 支付方式（Prepaid / Postpaid）
                                status='Running', # 实例状态（Recycled / Running / Stopped / Stopping / Starting）
                                tags='***', # 多个标签，逗号分割，格式：tagKey:tagValue 或 tagKey
                                vpc_id='vpc-***', # vpcId，只能与privateIps查询参数组合使用
                                private_ips='192.168.***.***', # 多个内网IP，英文逗号分隔，最多支持100个，必须和vpcId组合
使用
                                )
    print(res)
```

查询实例详情

使用以下代码可以查询指定BCC虚机的详细信息：

```
    def get_instance(self):

        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'
        #设置是否返回创建失败信息,True表示返回创建失败信息,默认为False
        contains_failed = True
        self.assertEqual(
            type(self.client.get_instance(instance_id, contains_failed=contains_failed)),
            baidubce.bce_response.BceResponse)
```

根据实例ID批量查询实例列表

以下代码可以根据实例ID批量查询实例列表

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_instance_by_instance_ids(instance_ids=['i-***', 'i-***', 'i-***'], # 待查询的实例id列表
                                                marker='123', # 批量获取列表的查询的起始位置
                                                max_keys=1000, # 每页包含的最大数量
                                                )
    print(resp)
```

启动实例

使用以下代码可以启动指定BCC虚机实例：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置你要操作的instance_id
    instance_id = 'i-***'
    resp = client.start_instance(instance_id)
    print(resp)
```

停止实例

使用以下代码可以停止指定BCC虚机实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.stop_instance(
        instance_id='i-***',                # 待停止的实例ID
        force_stop=False,                       # 指定实例是否强制关机，可选值：True、False，缺省为false
        stopWithNoCharge=True                       # 指定实例是否关机不计费，可选值：true、false，缺省为false
    )
    print(resp)
```

重启实例

使用以下代码可以重启指定BCC虚机实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置你要操作的instance_id
    instance_id = 'i-***'
    # 重启实例
    resp = client.reboot_instance(instance_id)
    print(resp)
```

修改实例密码

使用以下代码可以修改指定BCC虚机实例的管理员密码：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置你要操作的instance_id
    instance_id = 'i-***'
    # 设置新的密码
    admin_pass = '***

    resp = client.modify_instance_password(instance_id, admin_pass)
    print(resp)
```

修改实例属性

使用以下代码可以修改指定BCC虚机实例的名称或网卡队列数：

```
def modify_instance_attributes(self):

    #设置你要操作的instance_id
    instance_id = 'your-choose-instance-id'
    name = 'name_modify'
    neteth_queuecount = 3  # 新的网卡队列数
    self.assertEqual(
        type(self.client.modify_instance_attributes(instance_id,
                                                    name=name,
                                                    neteth_queuecount= neteth_queuecount)),
        baidubce.bce_response.BceResponse)
```

重装实例

使用以下代码可以使用镜像重建指定BCC虚机实例:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.rebuild_instance(instance_id="i-***",  # 指定的实例ID
                                  image_id="m-***",  # 指定的镜像ID
                                  admin_pass="***",  # 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅
限!@#$%^*())，密码需要加密传输
                                  key_pair_id="k-***",  # 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个
参数
                                  )
    print(res)
```

重装实例（批量）

使用以下代码批量重装实例:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.batch_rebuild_instances(instance_ids=["i-***", "i-***"],  # 指定的实例ID列表，最多100个
                                         image_id="m-***",  # 指定的镜像ID
                                         admin_pass="***",  # 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅
限!@#$%^*())，密码需要加密传输
                                         keypair_id="k-***",  # 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中
一个参数
                                         )
    print(res)
```

释放实例

对于后付费Postpaid以及预付费Prepaid过期的BCC虚机实例，可以使用以下代码将其释放:

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.release_instance(instance_id='i-sxb2****') # 释放实例ID
    print (resp)
```

实例扩缩容

支持对BCC虚机的CPU以及内存进行扩缩容操作，具体可参考以下代码：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.resize_instance(instance_id="i-sxb2****",  # 实例id
                                  cpu_count=4,  # cpu核数
                                  memory_capacity_in_gb=16, # 内存大小
                                  live_resize=True, # 是否是在线变配
                                  enable_jumbo_frame=False) # 是否开启Jumbo帧，True开启Jumbo帧，Flase关闭Jumbo帧。默认为空，为空表示：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；
                                                            # 从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
    print (resp)
```

将实例加入安全组

以下代码可以将指定BCC虚机实例加入到指定安全组内：

```
def bind_instance_to_security_group(self):

    #指定security_group_id信息
    security_group_id = 'your-choose-security-group-id'

    #设置你要操作的instance_id
    instance_id = 'your-choose-instance-id'
    self.assertEqual(type(self.client.bind_instance_to_security_group(instance_id, security_group_id)),
        baidubce.bce_response.BceResponse)
```

将实例移出安全组

以下代码可以将指定实例移出指定安全组中：

```
    def unbind_instance_from_security_group(self):

        #指定security_group_id信息
        security_group_id = 'your-choose-security-group-id'

        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'
        self.assertEqual(
            type(self.client.unbind_instance_from_security_group(instance_id, security_group_id)),
            baidubce.bce_response.BceResponse)
```

绑定实例到标签

以下代码可以将指定BCC虚机实例绑定指定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 输入标签信息，支持一次绑定多个标签
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey01',tagValue='TestValue01')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey02',tagValue='TestValue02')
    instance_tag_list = []
    instance_tag_list.append(instance_tag1)
    instance_tag_list.append(instance_tag2)
    # 设置你要操作的instance_id
    instance_id = 'i-***'

    # 执行绑定绑定实例到标签的操作
    resp = client.bind_instance_to_tags(instance_id=instance_id, tags=instance_tag_list)
    print(resp)
```

解绑实例标签

以下代码可以将指定BCC虚机实例解绑指定标签：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 输入标签信息，支持一次绑定多个标签
    instance_tag1 = bcc_model.TagModel(tagKey='Key***',tagValue='Value***')
    instance_tag2 = bcc_model.TagModel(tagKey='Key***',tagValue='Value***')
    instance_tag_list = []
    instance_tag_list.append(instance_tag1)
    instance_tag_list.append(instance_tag2)
    # 设置你要操作的instance_id
    instance_id = 'i-***'

    # 执行解绑实例标签的操作
    resp = client.unbind_instance_from_tags(instance_id=instance_id, tags=instance_tag_list)
    print(resp)
```

查询实例地址

使用以下代码可以查询指定BCC虚机实例的VNC URL:

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    print(client.get_instance_vnc("i-***"))  # i-*** 为虚拟机实例ID
```

实例续费

对BCC虚机的续费操作，可以延长过期时长，以下代码可以对磁盘进行续费：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.purchase_reserved_instance(instance_id='i-***',  # 虚拟机实例ID
                                            billing=bcc_model.Billing('Prepaid', 1),  # 订单详情，本接口针对预付费实例进行续费，续费
单位为月
                                            related_renew_flag='CDS_EIP',  # 设置实例关联续费标识，默认为空字符串
                                            client_token='random-uuid'  # 幂等性Token，是一个长度不超过64位的ASCII字符串
                                            )
    print(res)
```

开通自动续费

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_create_auto_renew_rules(
        instance_id='i-***', # 实例ID
        renew_time=2, # 续费时长
        renew_time_unit='year', # 续费时间单位，month/year
        renew_eip=True,  # 是否合并开启eip自动续费，默认值为True
        renew_cds=True,  # 是否合并开启cds自动续费，默认值为True
    )

    print(resp)
```

删除自动续费

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_delete_auto_renew_rules(
        instance_id='i-***', # 实例id
        renew_eip=True, # 是否合并关闭eip自动续费，默认值为True
        renew_cds=True  # 是否合并关闭cds自动续费，默认值为True
    )
    print(resp)
```

实例变更子网

如下代码可以变更实例的子网：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.instance_change_subnet(instance_id="i-***",  # 虚拟机实例ID
                                        subnet_id="sbn-***",  # 变更后的子网ID
                                        internal_ip="10.59.***.***",  # 指定内网IP，不指定则随机生成
                                        reboot=True,  # 变更后是否重启
                                        client_token='random-uuid'  # 幂等性Token，是一个长度不超过64位的ASCII字符串
                                        )
    print(res)
```

批量添加主网卡辅助IP

如下代码可以批量添加主网卡辅助IP：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    private_ips = ["192.168.***.***", "192.168.***.***"]
    res = client.batch_add_ip(instance_id="i-***",  # 实例ID
                              private_ips=private_ips,  # 辅助IP，最多100个，和SecondaryPrivateIpAddressCount不可同时使用
                              # secondary_private_ip_address_count=2  # 自动分配IP数量，和PrivateIps不可同时使用
                              allocate_multi_ipv6_addr=True  # 是否分配IPV6，创建IPV6必须是True
                              )
    print(res)
```

批量删除网卡辅助IP

如下代码可以批量删除网卡辅助IP：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    private_ips = ["192.168.***.***", "192.168.***.***"]
    res = client.batch_delete_ip(instance_id="i-***",  # 实例ID
                              private_ips=private_ips  # 辅助IP，最多100个
                               )
    print(res)
```

释放实例（POST）

使用以下代码释放单个后付费云服务器实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.release_instance_with_related_resources(instance_id="i-***",  # 实例id
                                                         related_release_flag=True,  # 是否关联释放当前时刻，实例挂载的EIP和数据盘
                                                         delete_cds_snapshot_flag=True,  # 是否释放云磁盘快照
                                                         delete_related_enis_flag=True,  # 实例释放时是否删除关联的ENI
                                                         bcc_recycle_flag=True  # 实例释放时是否进入回收站
                                                         )
    print(res)
```

释放预付费实例（POST）

释放预付费云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。 释放的时候默认只释放云服务器实例和系统盘，用户可以选择是否关联释放云服务器实例挂载的EIP和数据盘，关联的EIP和数据盘需全部释放或全部不释放，不支持选择部分释放，部分不释放。 当选择关联释放EIP和数据盘，云磁盘快照可选择一同关联释放，若数据盘不关联释放，云磁盘快照也不支持关联释放。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.release_prepaid_instance_with_related_resources(instance_id="i-***",  # 实例id
                                                         related_release_flag=True,  # 是否关联释放当前实例挂载的EIP和数据盘(只有该字段为true时 delete_cds_snapshot_flag字段才会有效，若该字段为false,delete_cds_snapshot_flag字段的值无效)
                                                         delete_cds_snapshot_flag=True,  # 是否释放云磁盘快照
                                                         delete_related_enis_flag=True,  # 实例释放时是否删除关联的ENI)
    print(res)
```

获取实例用户数据信息（POST）

如下代码可获取实例用户数据信息

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.get_instance_user_data(instance_id='i-b34ycow2')
    print(res)
```

磁盘

直接创建磁盘

使用以下代码可以创建磁盘：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 标签列表
    tags = [bcc_model.TagModel("example", "createCds")]

    # 自动快照策略
    auto_snapshot_policy = bcc_model.AutoSnapshotPolicyModel(name="aspExample",     # 自动快照策略名称
                                                             timePoints=[1, 2, 3],  # 自动快照创建时间点, 参数为0~23的列表
                                                             repeatWeekdays=[1, 5], # 自动快照重复日期, 参数为0~6的列表
                                                             retentionDays=30)     # 自动快照的保留时间，单位为天。 -1：永久保存
1~65536：指定保存天数

    resp = client.create_volume_with_cds_size(cds_size_in_gb=5,

                                              charge_type='Postpaid', # 计费方式
                                              zone_name='cn-bj-a', # 可用区名称
                                              instance_id='', # 创建完成后待挂载的实例ID
                                              encrypt_key='c6c0c0b3-****-****-****-ca6d5751504f', # 加密密钥ID
                                              name='test-name', # 磁盘名称
                                              description='desc', # 磁盘描述
                                              renew_time_unit='month', # 续费时长单位,支持:month(月) / year(年)
                                              renew_time=2, # 续费时长,
                                              relation_tag=True,  # 关联快照和快照链是否统一加标签
                                              auto_snapshot_policy=auto_snapshot_policy, # 自动快照策略
                                              tags=tags) # 绑定标签
    print (resp)
```

从快照创建CDS磁盘

使用以下代码可以从快照创建磁盘：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
#### 设置您的ak、sk和要访问的地域
HOST = b'http://bcc.bj.baidubce.com'
AK = b'ak'
SK = b'sk'
#### 设置默认配置
config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                endpoint=HOST)
#### 创建bcc client
client = bcc_client.BccClient(config)


resp = client.create_volume_with_snapshot_id(snapshot_id="s-er2Q****",  # 快照ID
                                             charge_type='Postpaid', # 计费方式
                                             cluster_id="DC-jh5q****",  # 专属集群ID
                                             purchase_count=2)  # 购买数量
print (resp)
```

查询磁盘列表

使用以下代码查询所有的磁盘列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
#### 设置您的ak、sk和要访问的地域
HOST = b'http://bcc.bj.baidubce.com'
AK = b'ak'
SK = b'sk'
#### 设置默认配置
config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                endpoint=HOST)
#### 创建bcc client
client = bcc_client.BccClient(config)

resp = client.list_volumes(instance_id="i-cVh8****",  # 磁盘挂载的实例ID
                           zone_name="cn-bj-c",  # 可用区
                           marker="v-3uIJ****",  # 起始卷ID
                           max_keys=10, # 返回列表的最大数量
                           cluster_id="DC-rWKx****") # CDS集群id
print (resp)
```

查询磁盘详情

使用以下代码可以根据磁盘ID查询单个磁盘的详细信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.get_volume(volume_id='v-3uIJ****') # 指定磁盘ID
    print (resp)
```

挂载CDS磁盘

使用以下代码将磁盘挂载在指定的BCC实例下，注意cds磁盘需要挂载在与其处在相同可用区下的BCC实例上。

只有磁盘状态为 Available 且实例状态为 Running 或 Stopped 时才允许挂载。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置要挂载的磁盘ID
    volume_id = 'v-***'
    # 设置要挂载的实例ID
    instance_id = 'i-***'
    resp = client.attach_volume(volume_id, instance_id)
    print(resp)
```

卸载CDS磁盘

使用以下代码将指定的磁盘从实例中卸载。

只有实例状态为 Running 或 Stopped 时，磁盘才可以执行此操作。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置要卸载的磁盘ID
    volume_id = 'v-***'
    # 设置所挂载的实例ID
    instance_id = 'i-***'
    resp = client.detach_volume(volume_id, instance_id)
    print(resp)
```

释放CDS磁盘

使用以下代码释放未挂载的CDS磁盘，系统盘不能释放。

磁盘释放后不可恢复。默认情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。

只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作：

```
    def release_volume(self):
        #设置要释放的磁盘ID
        volume_id = 'your-choose-volume-id'
        self.assertEqual(
            type(self.client.release_volume(volume_id)),
            baidubce.bce_response.BceResponse)
```

释放cds磁盘(新)

使用以下代码可以实现释放cds磁盘：

> - 该接口用于释放未挂载的CDS磁盘，系统盘不能释放。
> - 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误。
> - 如果 volumeId 的磁盘不存在，将提示 404 错误。
> - 可以控制是否释放关联的自动/手动快照、是否进入回收站。

```
#### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"

    # 请求释放cds盘
    response = client.release_volume_new(
        # 磁盘ID
        volume_id=volume_id,
        # 是否删除磁盘关联的自动快照
        # auto_snapshot="on",
        # 是否删除磁盘关联的手动快照
        # manual_snapshot="on",
        # 是否将磁盘回收至回收站，为off时直接释放，不进入回收站
        # recycle="off")
    print(response)
```

磁盘扩容

使用以下代码对磁盘进行扩大容量：

- 磁盘只能进行扩容操作，不支持缩容。
- 只有Available状态的磁盘，才能进行扩容操作。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作。

```
#### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"
    # 请求扩容磁盘
    response = client.resize_volume(
        # 磁盘ID
        volume_id=volume_id,
        # 扩容后磁盘大小, 单位为GB
        # new_cds_size=50,
        # 扩容后磁盘类型, 可取值见：https://cloud.baidu.com/doc/BCC/s/6jwvyo0q2#storagetype
        # new_volume_type="cloud_hp1")
    print(response)
```

回滚磁盘

使用以下代码用指定磁盘自身的快照回滚磁盘内容：

- 磁盘状态必须为 Available 才可以执行此操作。
- 指定 snapshotId 必须是由 volumeId 创建的快照。
- 若是回滚系统盘，实例状态必须为 Running 或 Stopped 才可以执行此操作。
- 回滚系统盘快照，自本次快照以来的系统盘数据将全部丢失，不可恢复。

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
#### 设置您的ak、sk和要访问的地域
HOST = b'http://bcc.bj.baidubce.com'
AK = b'ak'
SK = b'sk'
#### 设置默认配置
config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                endpoint=HOST)
#### 创建bcc client
client = bcc_client.BccClient(config)
volume_id = 'v-vG0z****'
#### 用于回滚指定磁盘的快照ID，必须为 volumeId 所创建的快照
snapshot_id = 's-HkbJ****'
resp = client.rollback_volume(volume_id, snapshot_id)
print (resp)
```

磁盘续费

使用以下代码可以实现磁盘续费：

磁盘扩缩容期间不能进行续费操作。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 待续费磁盘ID
    volume_id = 'v-PGE6****'
    # 设置续费时长,目前只支持续费单位为月
    billing = bcc_model.Billing('Prepaid', 2)
    resp = client.purchase_reserved_volume(volume_id=volume_id, billing=billing)
    print (resp)
```

镜像

通过实例创建自定义镜像

使用以下代码用于创建自定义镜像，默认每个账号每个地域配额20个，创建后的镜像可用于创建实例。

只有 Running 或 Stopped 状态的实例才可以执行成功。

仅限通过系统盘快照创建自定义镜像：

```
 import uuid
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    # 创建的镜像名称
    image_name = 'instanceImage***'
    # 创建的实例id
    instance_id = 'i-***'
    # 快照id
    snapshot_id = 's-B1Cc6Dho'
    resp = client.create_image_from_instance_id(image_name,
                                    instance_id=instance_id,
                                    client_token=client_token)
    print(resp)
```

通过快照创建自定义镜像

通过快照创建时，只有 Available 状态的快照才可以执行成功：

```
 import uuid

from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    # 创建的镜像名称
    image_name = 'instanceImage***'
    # 创建的实例id
    instance_id = 'i-***'
    # 快照id
    snapshot_id = 'm-***'

    resp = client.create_image_from_snapshot_id(image_name,
                                    snapshot_id=snapshot_id,
                                    client_token=client_token)
    print(resp)
```

查询镜像列表

使用如下代码可以查询用户所有的镜像信息：

- 查询的镜像信息中包括公共镜像、自定义镜像和服务集成镜像。
- 支持按 imageType 来过滤查询，此参数非必需，未设置时默认为 All,即查询所有类型的镜像：

```
def list_images(self):
    #指定要查询何种类型的镜像
    #All(所有)
    #System(系统镜像/公共镜像)
    #Custom(自定义镜像)
    #Integration(服务集成镜像)
    #Sharing(共享镜像)
    #GpuBccSystem(GPU专用公共镜像)
    #GpuBccCustom(GPU专用自定义镜像)
    #FpgaBccSystem(FPGA专用公共镜像)
    #FpgaBccCustom(FPGA专用自定义镜像)
    image_type = 'All'
    #批量获取列表的查询的起始位置
    marker = 'your-marker'
    #每页包含的最大数量
    max_keys = 100
    self.assertEqual(
        type(self.client.list_images(image_type=image_type
                                     marker=marker
                                     max_keys=max_keys)),
        baidubce.bce_response.BceResponse)
```

查询镜像详情

使用如下代码根据指定镜像ID查询单个镜像的详细信息：

```
def get_image(self):
    #待查询镜像ID
    image_id='your-choose-image-id'
    self.assertEqual(
        type(self.client.get_image(image_id)),
        baidubce.bce_response.BceResponse)
```

删除自定义镜像

使用如下代码删除用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能删除。

镜像删除后无法恢复，不能再用于创建、重置实例：

```
def delete_image(self):
    #待删除镜像ID
    image_id='your-choose-image-id'
    self.assertEqual(
        type(self.client.delete_image(image_id)),
        baidubce.bce_response.BceResponse)
```

根据实例规格查询可用公共镜像

使用以下代码可以根据实例规格查询可用公共镜像：

```
def test_get_available_images_by_spec(self):
  """
  test case for get_available_images_by_spec
  """
  resp = self.client.get_available_images_by_spec(spec='bcc.ic4.c1m1', os_name='Centos')
```

快照

创建快照

用于从指定的磁盘创建快照，支持创建系统盘快照和CDS磁盘快照。

创建系统盘快照时，实例的状态必须为 Running 或 Stopped 此操作才能正常完成。

创建CDS磁盘快照时，磁盘的状态需要为InUse或 Available 此操作才能正常完成。

可创建快照数量的配额取决于账号下拥有的磁盘总数，当前最多为磁盘数量乘以8：

```
 import uuid

from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    # 设置快照名称
    snapshot_name = 'snapshot***'
    # 创建快照的磁盘ID
    volume_id = 'v-***'
    resp = client.create_snapshot(volume_id, snapshot_name, client_token=client_token)
    print(resp)
```

查询快照列表

使用如下代码查询所有的快照信息，包括系统盘快照和数据盘快照信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http:/bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 创建快照的磁盘ID
    volume_id = 'v-***'
    # 批量获取列表的查询的起始位置
    marker = "your-marker"
    # 每页包含的最大数量
    max_keys = 100
    resp = client.list_snapshots(marker=marker, max_keys=max_keys, volume_id=volume_id)
    print(resp)
```

查询快照详情

使用如下代码可以通过用户的指定的快照ID查询单个快照的详细信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 快照ID
    snapshot_id = 's-***'
    resp = client.get_snapshot(snapshot_id)
    print(resp)
```

删除快照

使用如下的代码可以通过指定快照ID删除快照：

只有当快照的状态为 CreatedFailed 或 Available时，操作才能正常完成：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 待删除快照ID
    snapshot_id = 's-***'
    resp = client.delete_snapshot(snapshot_id=snapshot_id)
    print(resp)
```

安全组

创建安全组

如下代码可以创建安全组以及安全组规则：

```
 def create_security_group(self):
    client_token = generate_client_token()

    #设置安全组名称
    security_group_name = 'your_security_group_name' + client_token

    #设置安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel('rule_' + client_token,# 设置备注
                                            'ingress',# 设置入站/出站，取值ingress或egress，必选参数
                                            portRange='1-65535',# 设置端口范围，默认空时为1-65535，可以指定80等单个端口
                                            protocol='tcp',# 设置协议类型
                                            sourceGroupId='',# 设置源安全组ID
                                            sourceIp='')# 设置源IP地址，与sourceGroupId不能同时设定值
     security_group_rule_list = []
     security_group_rule_list.append(security_group_rule)

     #vpc的Id，空的时候，表示默认VPC的id
     security_group_rule_vpc_id = 'your-vpc-id'

     #设置安全组描述

     security_group_desc = 'your_security_group_desc'
     self.assertEqual(              type(self.client.create_security_group(name=security_group_name,
rules=security_group_rule_list,
                                       vpc_id=security_group_rule_vpc_id,
                                       desc=security_group_desc,
                                       client_token=client_token)),

         baidubce.bce_response.BceResponse)
```

查询安全组列表

如下代码可以查询用户的所有安全组信息：

```
 def list_security_groups(self):

    #批量获取列表的查询的起始位置
    marker = 'your-marker'

    #每页包含的最大数量
    maxKeys = 100

    #实例id，可查询实例关联的安全组列表
    instance_id = 'your-instance-id'

    #vpcid，可查询实例关联的安全组列表
    vpc_id = 'your-vpc-id'
    self.assertEqual(              type(self.client.list_security_groups(instance_id=instance_id,
                                       vpc_id=vpc_id,
                                       marker=marker,
                                       max_keys=max_keys)),
        baidubce.bce_response.BceResponse)
```

删除安全组

如下代码可以删除某个安全组：

```
def delete_security_group(self):

    #要删除的安全组ID
    security_group_id = 'your-security-group-id'
    self.assertEqual(
        type(self.client.delete_security_group(security_group_id)),
        baidubce.bce_response.BceResponse)
```

授权安全组规则

如下代码可以在安全组中授权新的安全组规则：

```
def authorize_security_group_rule(self):

    client_token = generate_client_token()

    #设置安全组ID
    security_group_id = 'your-security-group-id'

    #设置安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel(direction='ingress',# 设置入站/出站，取值ingress或
egress，必选参数
                                                           portRange='80-90',# 设置端口范围，默认空时为1-65535，可以指定80等单个端
口
                                                           protocol='tcp')# 设置协议类型
    self.assertEqual(
        type(self.client.authorize_security_group_rule(security_group_id=security_group_id,
                                                       rule=security_group_rule,
                                                       client_token=client_token)),
        baidubce.bce_response.BceResponse)
```

撤销安全组规则

如下代码可以撤销安全组中的规则：

```
def revoke_security_group_rule(self):

    client_token = generate_client_token()

    #待撤销安全组ID
    security_group_id = 'your-security-group-id'

    #待撤销安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel(direction='ingress',# 设置入站/出站，取值ingress或
egress，必选参数
                                                           portRange='80-90',# 设置端口范围，默认空时为1-65535，可以指定80等单个端
口
                                                           protocol='tcp')# 设置协议类型
    self.assertEqual(
        type(self.client.revoke_security_group_rule(security_group_id=security_group_id ,
                                                    security_group_rule,
                                                    client_token=client_token)),

        baidubce.bce_response.BceResponse)
```

地域

获取域名列表接口

使用以下代码可以获取全部地域或者指定地域的BCC(或CDS、预留实例券)域名列表：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    response = client.describe_regions(
        region='', # 待查询的地域，为空时代表查询全部地域
    )
    print response
```

预留实例券

绑定标签

使用以下代码给指定预留实例券绑定标签

```
from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)
    bcc_client = BccClient(bcc_sample_conf.config)

    # 预留实例券ID，最多支持100个
    reserved_instance_ids = ['r-Qyycx1SX']
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey02',
                                       tagValue='TestValue02')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey03',
                                       tagValue='TestValue03')
    # 待绑定tag列表
    instance_tags = [instance_tag1, instance_tag2]
    bcc_client.bind_reserved_instance_to_tags(reserved_instance_ids=reserved_instance_ids, tags=instance_tags)
```

解绑标签

使用以下代码从指定预留实例券中解绑标签

```
 from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)

    bcc_client = BccClient(bcc_sample_conf.config)

    # 预留实例券ID，最多支持100个
    reserved_instance_ids = ['r-Qyycx1SX']
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey02',
                                       tagValue='TestValue02')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey03',
                                       tagValue='TestValue03')
    # 待解绑tag列表
    instance_tags = [instance_tag1, instance_tag2]
    bcc_client.unbind_reserved_instance_from_tags(reserved_instance_ids=reserved_instance_ids, tags=instance_tags)
```

版本变更记录

v0.8.27

BCC python SDK第一次发布，可以利用SDK对实例、安全组、快照、磁盘、镜像等进行操作。

Python3-SDK

概述

本文档主要介绍云服务器BCC Python3 SDK的安装和使用。在使用本文档前，您需要先了解云服务器BCC的一些基本知识，若您还不了解云服务器BCC，可以参考产品描述和入门指南。

相较于BCC Python2 SDK，主要改动在于:

实例部分新增『修改实例描述』『绑定实例到标签』『从标签解绑实例』

磁盘部分新增『修改磁盘属性』『修改磁盘计费方式』

镜像部分新增『跨区域复制镜像』『取消跨区域复制镜像』『共享镜像』『取消共享镜像』『列出OS信息』

新增自动快照策略接口，包括『创建自动快照策略』『绑定自动快照策略』『解绑自动快照策略』『删除自动快照策略』『查询自动快照策略列表』『查询自动快照策略详情』

新增密钥对接口，包括『创建密钥对』『导入密钥对』『查询密钥对列表』『查询密钥对详情』『绑定密钥对』『解绑密钥对』『删除密钥对』『重命名密钥对』『更改密钥对描述』

安装SDK工具包

环境准备

1. 运行环境

   Python SDK工具包支持在Python 2.7 和Python 3 的环境运行。

2. 安装pycrypto依赖

   安装SDK之前，需要先执行命令`pip install pycrypto`安装pycrypto依赖。 如果安装失败，请执行`pip install pycryptodome`

下载和安装

方式一：通过pip安装

您可以通过pip安装的方式将百度智能云Python SDK安装到您的环境中。 联网状态下，在命令行中执行如下命令：

```
pip install bce-python-sdk
```

即可将Python SDK安装到本地。

方式二：将源码包下载到本地后进行安装

1. 在开发者资源中心下载Python SDK压缩工具包。
2. 命令行移动到压缩包所在路径，执行如下命令（version替换为包名称中的版本号）：

   ```
   pip install bce-python-sdk-version.zip
   ```

   即可将Python SDK安装到本地。

方式三：使用setup.py install文件进行安装

您也可以解压压缩包后执行如下命令（version替换为包名称中的版本号）：

```
cd bce-python-sdk-version

python setup.py install
```

卸载SDK

卸载 SDK 时，执行命令`pip uninstall bce-python-sdk`即可完成卸载。

初始化

确认Endpoint

用户若要自定义域名，可以通过传入Endpoint参数来指定。地域的概念请参考地域选择说明。

在开始SDK使用之前，需要您先确定好要在哪个地域进行操作，从而在配置BccClient时将地域对应的Endpoint做为参数填入。不指定时默认为北京地域，Endpoint为http://bcc.bj.baidubce.com

BCC目前支持的域名请参考服务域名。

获取密钥

要使用百度BCC，您需要拥有一个有效的 AK（Access Key ID）和SK(Secret Access Key)用来进行签名认证。AK/SK是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。

可以通过如下步骤获得并了解您的AK/SK信息：

注册百度智能云账号

创建AK/SK

获取到密钥后，也需要在配置BccClient时做为参数填入。SDK集成了鉴权认证机制，您不需要关心鉴权背后复杂的运算方法，只需要将AK/SK按要求填入对应的位置，SDK将自动为您完成鉴权相关的工作。

配置文件

```
#从Python SDK导入BCC配置管理模块以及安全认证模块
import baidubce
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
#导入BCC相关模块
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_model import EphemeralDisk
```

BccClient

BccClient是BCC服务的客户端，为开发者与BCC服务进行交互提供了一系列的方法，包括实例、磁盘、镜像、快照、安全组、可用区等。

通过AK/SK方式访问BCC

用户可以参考如下代码新建一个BccClient：

```
class  sample_default_host():

  def setUp(self):

      AK = 'your-access-key-id'              # 用户Access Key ID

      SK = 'your-secret-access-key'          #用户的Secret Access Key

      # 初始化一个BccClient

      config = BceClientConfiguration(credentials=BceCredentials(AK, SK))

      client = bcc_client.BccClient(config)
```

在上面代码中，变量AK与SK是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。

其中AK对应控制台中的“Access Key ID”，SK对应控制台中的“Access Key Secret”，获取方式请参考获取AK/SK。

上面的方式使用默认域名（北京区域）作为BCC的服务地址，如果用户需要自己指定域名，可以通过传入endpoint参数来指定。

```
class  sample_customize_host():

  def setUp(self):

      AK = 'your-access-key-id'              # 用户Access Key ID

      SK = 'your-secret-access-key'          # 用户Secret Access Key

      HOST = 'domain-name'                   # 用户自己指定的域名，域名选择可参考下方[Region列表]

      # 初始化一个BccClient

      config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint = HOST)

      client = bcc_client.BccClient(config)
```

注意： endpoint参数只能用指定的包含区域的域名来进行定义，不指定时默认为北京区域http://bcc.bj.baidubce.com

BCC支持的区域请查看Region列表

实例

创建实例（通过指定实例套餐规格）

使用以下代码可以通过规格参数创建BCC实例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    response = client.create_instance_by_spec(
        spec='bcc.g5.c1m4', # 实例规格
        image_id='m-***', # 镜像ID
        root_disk_size_in_gb=20, # 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        root_disk_storage_type='enhanced_ssd_pl1', # 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        create_cds_list=[ # 待创建的CDS磁盘列表
            bcc_model.CreateCdsModel(
                cdsSizeInGB=50, # CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
                storageType='enhanced_ssd_pl1')], # CDS磁盘存储类型，默认是hp1(高性能云磁盘)。
        eip_name='eip***', # 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
        internet_charge_type='BANDWIDTH_PREPAID', # 公网带宽计费方式，若不指定internetChargeType，默认为按使用带宽计费。
        purchase_count=1, # 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1。
        name='instance-***', # 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        hostname='hostn***', # 虚拟机主机名。。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
        admin_pass='***', # 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        zone_name='cn-bj-a', # 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        subnet_id='sbn-***', # 指定subnet信息，为空时将使用默认子网
        security_group_id='g-***', # 指定securityGroup信息，为空时将使用默认安全组
        asp_id='asp-***', # 自动快照策略ID
        tags=[bcc_model.TagModel('key***', 'value***')], # 待创建虚机的标签列表
    )
    print(response)
```

- 必传参数：
  - 规格名称（例如 bcc.g4.c1m1）
  - 镜像ID
- 可选参数：参考普通创建实例接口

创建实例（已废弃，建议使用“创建实例（通过指定实例套餐规格）”）

使用以下代码可以创建BCC实例，包括专属实例、普通型Ⅰ型实例、普通型ⅠⅠ型实例、普通型ⅠⅡ型实例、GPU实例、计算优化I型实例、计算优化II型实例、存储优化I型实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    response = client.create_instance(
        instance_type='N6', # 待创建虚拟机实例的类型
        cpu_count=1, # 待创建虚拟机实例的CPU核数
        memory_capacity_in_gb=4, # 待创建虚拟机实例的内存容量，单位GB
        image_id='m-***', # 镜像ID
        root_disk_size_in_gb=20, # 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        root_disk_storage_type='enhanced_ssd_pl1', # 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        create_cds_list=[  # 待创建的CDS磁盘列表
            bcc_model.CreateCdsModel(
                cdsSizeInGB=50, # CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
                storageType='enhanced_ssd_pl1')], # CDS磁盘存储类型，默认是hp1(高性能云磁盘)。
        eip_name='eip***', # 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
        internet_charge_type='BANDWIDTH_PREPAID', # 公网带宽计费方式，若不指定internetChargeType，默认为按使用带宽计费。
        purchase_count=1, # 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1。
        name='instance-***', # 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        hostname='hostn***', # 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        admin_pass='***', # 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        zone_name='cn-bj-a', # 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        subnet_id='sbn-***', # 指定subnet信息，为空时将使用默认子网
        security_group_id='g-***', # 指定securityGroup信息，为空时将使用默认安全组
        asp_id='asp-***', # 自动快照策略ID
        tags=[bcc_model.TagModel('key***', 'value***')], # 待创建虚机的标签列表
    )
    print(response)
```

类型编码请参考实例类型、磁盘类型、GPU类型

创建专属服务器的子实例

使用以下代码可以在特定服务器中创建BCC实例：

```
    def create_instance_from_dedicated_host(self):
        #如果用户未指定client_token参数,用uuid4生成一个随机字符串给client_token
        generate_client_token = str(uuid.uuid4())
        client_token = generate_client_token()

        #设置镜像id
        image_id = 'your-choose-image-id'

        #设置专属服务器id
        dedicated_host_id = 'your-choose-dedicated-host-id'

        #设置实例名
        instance_name = 'your-choose-instance-name' + client_token

        #设置实例管理员密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())
        admin_pass = 'your-admin-pass'

        #指定subnet信息，为空时将使用默认子网
        subnet_id = 'your-choose-subnet-id'

        #指定security_group_id信息，为空时将使用默认安全组
        security_group_id = 'your-choose-security-group-id'

        #DCC专属服务器实例可以创建多块本地盘，需要指定磁盘类型及大小
        #其它类型只能用一块本地盘，使用默认类型，只需指定大小

        #设置创建本地磁盘大小
        ephemeral_disk = EphemeralDisk(6144)

        #设置本地盘类型为默认类型
        ephemeral_disks = [ephemeral_disk.__dict__, ephemeral_disk.__dict__]

        self.assertEqual(
            type(self.client.create_instance_from_dedicated_host(1, #设置创建BCC的CPU核数
                                                 2, #设置创建BCC的内存大小GB
                                                 image_id, #设置创建BCC的镜像
                                                 dedicated_host_id, #设置创建BCC的专属服务器
                                                 purchase_count=1,# 批量创建BCC实例时使用，必须为大于0的整数
                                                 name=instance_name,# 设置创建BCC的名称
                                                 admin_pass=admin_pass,# 设置实例管理员密码
                                                 subnet_id=subnet_id,# 指定subnet信息，为空时将使用默认子网
                                                 security_group_id=security_group_id,# 指定securityGroup信息，为空时将使
用默认安全组
                                                 client_token=client_token
                                                 ephemeral_disks) #设置创建BCC的本地盘

            baidubce.bce_response.BceResponse)
```

创建实例在专属服务器（支持对明文密码加密）

使用以下代码可以在特定服务器中创建BCC实例：

```
  def create_instance_from_dedicated_host_with_encrypted_password(self):
        #如果用户未指定client_token参数,用uuid4生成一个随机字符串给client_token
        generate_client_token = str(uuid.uuid4())
        client_token = generate_client_token()

        #设置镜像id
        image_id = 'your-choose-image-id'

        #设置专属服务器id
        dedicated_host_id = 'your-choose-dedicated-host-id'

        #设置实例名
        instance_name = 'your-choose-instance-name' + client_token

        #设置实例管理员密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())
        admin_pass = 'your-admin-pass'

        #指定subnet信息，为空时将使用默认子网
        subnet_id = 'your-choose-subnet-id'

        #指定security_group_id信息，为空时将使用默认安全组
        security_group_id = 'your-choose-security-group-id'

        #DCC专属服务器实例可以创建多块本地盘，需要指定磁盘类型及大小
        #其它类型只能用一块本地盘，使用默认类型，只需指定大小

        #设置创建本地磁盘大小
        ephemeral_disk = EphemeralDisk(6144)

        #设置本地盘类型为默认类型
        ephemeral_disks = [ephemeral_disk.__dict__, ephemeral_disk.__dict__]

        self.assertEqual(
            type(self.client.create_instance_from_dedicated_host_with_encrypted_password(1, #设置创建BCC的CPU核数
                                                2, #设置创建BCC的内存大小GB
                                                image_id, #设置创建BCC的镜像
                                                dedicated_host_id, #设置创建BCC的专属服务器
                                                purchase_count=1,# 批量创建BCC实例时使用，必须为大于0的整数
                                                name=instance_name,# 设置创建BCC的名称
                                                admin_pass=admin_pass,# 设置实例管理员密码
                                                subnet_id=subnet_id,# 指定subnet信息，为空时将使用默认子网
                                                security_group_id=security_group_id,# 指定securityGroup信息，为空时将使
用默认安全组
                                                client_token=client_token
                                                ephemeral_disks) #设置创建BCC的本地盘

            baidubce.bce_response.BceResponse)
```

查询实例列表

以下代码可以查询BCC虚机实例列表,支持通过内网ip、专属服务器id、可用区名称进行筛选:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.list_instances(marker='i-***', # 批量获取列表的查询的起始位置，是一个由系统生成的字符串
                                max_keys=1, # 每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
                                internal_ip='192.168.***.***', # 内网IP
                                dedicated_host_id='d-***', # 专属服务器ID
                                zone_name='cn-bj-d', # 可用区信息
                                instance_ids='i-***', # 多个实例ID，英文逗号分割，最多支持100个
                                instance_names='***', # 多个实例名称，英文逗号分割，最多支持100个
                                cds_ids='v-***', # 多个cds磁盘ID，英文逗号分割，最多支持100个
                                deployset_ids='dset-***', # 多个部署集ID，英文逗号分割，最多支持100个
                                security_group_ids='g-***', # 多个安全组ID，英文逗号分割，最多支持100个
                                payment_timing='Postpaid', # 支付方式（Prepaid / Postpaid）
                                status='Running', # 实例状态（Recycled / Running / Stopped / Stopping / Starting）
                                tags='***', # 多个标签，逗号分割，格式：tagKey:tagValue 或 tagKey
                                vpc_id='vpc-***', # vpcId，只能与privateIps查询参数组合使用
                                private_ips='192.168.***.***', # 多个内网IP，英文逗号分隔，最多支持100个，必须和vpcId组合
使用
                                )
    print(res)
```

查询实例详情

使用以下代码可以查询指定BCC虚机的详细信息：

```
    def get_instance(self):
        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'
        #设置是否返回创建失败信息,True表示返回创建失败信息,默认为False
        contains_failed = True
        self.assertEqual(
            type(self.client.get_instance(instance_id, contains_failed)),

            baidubce.bce_response.BceResponse)
```

启动实例

使用以下代码可以启动指定BCC虚机实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置你要操作的instance_id
    instance_id = 'i-***'
    resp = client.start_instance(instance_id)
    print(resp)
```

停止实例

使用以下代码可以停止指定BCC虚机实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.stop_instance(
        instance_id='i-***', # 待停止的实例ID
        force_stop=False, # 指定实例是否强制关机，可选值：True、False，缺省为false
        stopWithNoCharge=True # 指定实例是否关机不计费，可选值：true、false，缺省为false
    )
    print(resp)
```

重启实例

使用以下代码可以重启指定BCC虚机实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置你要操作的instance_id
    instance_id = 'i-***'
    # 重启实例
    resp = client.reboot_instance(instance_id)
    print(resp)
```

修改实例密码

使用以下代码可以修改指定BCC虚机实例的管理员密码：

```
 from baidubce.auth.bce_credentials import BceCredentials
    from baidubce.bce_client_configuration import BceClientConfiguration
    from baidubce.services.bcc import bcc_client

    if __name__ == '__main__':
        # 设置您的ak、sk和要访问的地域
        HOST = b'http://bcc.bj.baidubce.com'
        AK = b'ak'
        SK = b'sk'
        # 设置默认配置
        config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                        endpoint=HOST)
        # 创建bcc client
        client = bcc_client.BccClient(config)

        # 设置你要操作的instance_id
        instance_id = 'i-***'
        # 设置新的密码
        admin_pass = '***'
        # 执行修改实例密码的操作
        resp = client.modify_instance_password(instance_id, admin_pass)
        print(resp)
```

修改实例属性

使用以下代码可以修改指定BCC虚机实例的名称或网卡队列数：

```
    def modify_instance_attributes(self):

        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'
        name = 'name_modify'
        neteth_queuecount = 3  # 新的网卡队列数
        self.assertEqual(
            type(self.client.modify_instance_attributes(instance_id,
                                                        name=name,
                                                        neteth_queuecount= neteth_queuecount)),
            baidubce.bce_response.BceResponse)
```

修改实例描述(新)

使用以下代码可以修改指定BCC虚机实例的描述：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置你要操作的instance_id
    instance_id = 'i-***'
    # 修改instance的描述
    desc = 'your_new_description'
    resp = client.modify_instance_desc(instance_id, desc)
    print(resp)
```

重装实例

使用以下代码可以使用镜像重建指定BCC虚机实例:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.rebuild_instance(instance_id="i-***", # 指定的实例ID
                                  image_id="m-***", # 指定的镜像ID
                                  admin_pass="***", # 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅
限!@#$%^*())，密码需要加密传输
                                  key_pair_id="k-***", # 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个
参数
                                  )
    print(res)
```

释放实例

对于后付费Postpaid以及预付费Prepaid过期的BCC虚机实例，可以使用以下代码将其释放:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.release_instance(instance_id='i-sxb2****') # 释放实例ID
    print (resp)
```

实例扩缩容

支持对BCC虚机的CPU以及内存进行扩容操作，具体可参考以下代码：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.resize_instance(instance_id="i-sxb2****",  # 实例id
                                  cpu_count=4,  # cpu核数
                                  memory_capacity_in_gb=16, # 内存大小
                                  live_resize=True, # 是否是在线变配
                                  enable_jumbo_frame=False) # 是否开启Jumbo帧，True开启Jumbo帧，Flase关闭Jumbo帧。默认为空，为空表示：变配到不支持Jumbo帧的规格时，不开启Jumbo帧；
                                                            # 从不支持变配到支持的规格时，不开启Jumbo帧；从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
    print (resp)
```

将实例加入安全组

以下代码可以将指定BCC虚机实例加入到指定安全组内：

```
    def bind_instance_to_security_group(self):

        #指定security_group_id信息
        security_group_id = 'your-choose-security-group-id'

        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'

        self.assertEqual(
            type(self.client.bind_instance_to_security_group(instance_id,security_group_id)),

            baidubce.bce_response.BceResponse)
```

将实例移出安全组

以下代码可以将指定实例移出指定安全组中：

```
    def unbind_instance_from_security_group(self):

        #指定security_group_id信息
        security_group_id = 'your-choose-security-group-id'

        #设置你要操作的instance_id
        instance_id = 'your-choose-instance-id'

        self.assertEqual(

            type(self.client.unbind_instance_from_security_group(instance_id,security_group_id)),

            baidubce.bce_response.BceResponse)
```

绑定实例到标签(新)

以下代码可以将指定BCC虚机实例绑定指定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 输入标签信息，支持一次绑定多个标签
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey01',tagValue='TestValue01')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey02',tagValue='TestValue02')
    instance_tag_list = []
    instance_tag_list.append(instance_tag1)
    instance_tag_list.append(instance_tag2)
    # 设置你要操作的instance_id
    instance_id = 'i-***'

    # 执行绑定绑定实例到标签的操作
    resp = client.bind_instance_to_tags(instance_id=instance_id, tags=instance_tag_list)
    print(resp)
```

解绑实例标签(新)

以下代码可以将指定BCC虚机实例解绑指定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 输入标签信息，支持一次绑定多个标签
    instance_tag1 = bcc_model.TagModel(tagKey='Key***',tagValue='Value***')
    instance_tag2 = bcc_model.TagModel(tagKey='Key***',tagValue='Value***')
    instance_tag_list = []
    instance_tag_list.append(instance_tag1)
    instance_tag_list.append(instance_tag2)
    # 设置你要操作的instance_id
    instance_id = 'i-***'

    # 执行解绑实例标签的操作
    resp = client.unbind_instance_from_tags(instance_id=instance_id, tags=instance_tag_list)
    print(resp)
```

查询实例地址

使用以下代码可以查询指定BCC虚机实例的VNC URL:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    print(client.get_instance_vnc("i-***"))  # i-*** 为虚拟机实例ID
```

实例续费

对BCC虚机的续费操作，可以延长过期时长，以下代码可以对磁盘进行续费：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.purchase_reserved_instance(instance_id='i-***',  # 虚拟机实例ID
                                            billing=bcc_model.Billing('Prepaid', 1),  # 订单详情，本接口针对预付费实例进行续费，续费单位为月
                                            related_renew_flag='CDS_EIP',  # 设置实例关联续费标识，默认为空字符串
                                            client_token='random-uuid'  # 幂等性Token，是一个长度不超过64位的ASCII字符串
                                            )
    print(res)
```

创建抢占实例

使用以下代码可以创建BCC抢占实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)

    response = client.create_instance_of_bid(
        bid_model='market', # 抢占实例出价模型， 市场价: "market" 自定义："custom"。
        instance_type='N6', # 待创建虚拟机实例的类型
        cpu_count=1, # 待创建虚拟机实例的CPU核数
        memory_capacity_in_gb=4, # 待创建虚拟机实例的内存容量，单位GB
        image_id='m-***', # 镜像ID
        root_disk_size_in_gb=20, # 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
        root_disk_storage_type='enhanced_ssd_pl1', # 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
        create_cds_list=[ # 待创建的CDS磁盘列表
            bcc_model.CreateCdsModel(
                cdsSizeInGB=50, # CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G
                storageType='enhanced_ssd_pl1')], # CDS磁盘存储类型，默认是hp1(高性能云磁盘)。
        eip_name='eipN***', # 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
        internet_charge_type='BANDWIDTH_PREPAID', # 公网带宽计费方式，若不指定internetChargeType，默认为按使用带宽计费。
        purchase_count=1, # 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1。
        name='instance-***', # 虚拟机名字。默认都不指定name。如果指定name：批量时name作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        hostname='host***', # 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
        admin_pass='***', # 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
        zone_name='cn-bj-a', # 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
        subnet_id='sbn-***', # 指定subnet信息，为空时将使用默认子网
        security_group_id='g-***', # 指定securityGroup信息，为空时将使用默认安全组
        asp_id='asp-***', # 自动快照策略ID
        tags=[bcc_model.TagModel('key***', 'value***')], # 待创建虚机的标签列表
    )
    print(response)
```

查询回收站实例列表

使用以下代码可以查询回收站实例列表

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_recycled_instances(payment_timing="postpaid", recycle_begin='2023-11-21T20:00:00Z')
    print(resp)
```

定时释放 (限定后付费实例)

使用以下代码可以设置后付费实例定时释放，到达预设时间后自动释放bcc，自动释放时间可查询实例详情ReleaseTime。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.auto_release_instance(
        instance_id="i-***",  # 实例ID
        release_time='2023-12-15T14:20:00Z')  # 释放时间，格式yyyy-MM-dd'T'HH:mm:ss'Z'
    print(resp)
```

查询关联部署集的实例详情

使用以下代码可以查询关联了部署集的实例详情

```
 def test_get_instance_with_deploy_set(self):

        instance_id = "i-oUXBvdIx"
        self.assertEqual(
            type(self.client.get_instance_with_deploy_set(instance_id)),
            baidubce.bce_response.BceResponse)
        print(self.client.get_instance(instance_id))
```

查询关联部署集和失败相关字段实例详情

使用以下代码可以查询关联部署集和失败相关字段实例详情

```
def test_get_instance_with_deploy_set_and_failed(self):
    """
    test case for get_instance
    """
    instance_id = "i-oUXBvdIx"
    self.assertEqual(
        type(self.client.get_instance_with_deploy_set_and_failed(instance_id)),
        baidubce.bce_response.BceResponse)
    print(self.client.get_instance(instance_id))
```

删除实例与部署集之间的关系

使用以下代码可以删除实例与部署集之间的关系

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.del_instance_deploy(instance_id_list=['i-***', 'i-***'],     # 实例ID列表
                                      deploy_set_id='dset-***'                  # 部署集ID
                                      )
```

修改实例主机名

使用以下代码可以修改实例主机名

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.modify_instance_hostname(instance_id="i-***",        # 指定的实例ID
                                           hostname="new-hostname",         # 实例主机名
                                           reboot=True,                # 是否自动重启
                                           is_open_hostname_domain=True,    # 是否开启hostname domain
                                           )
```

后付费资源从回收站恢复计费

使用以下代码可以将后付费资源从回收站恢复计费

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 需要从回收站恢复计费的实例id
    instance_ids = ["i-***", "i-***"]
    resp = client.recovery_instances(instance_ids)
    print(resp)
```

查询竞价实例市场价

使用以下代码可以查询竞价实例市场价

```
def test_get_bid_instance_price(self):
        """
        test case for get_bid_instance_price
        """
        resp = self.client.get_bid_instance_price(instance_type='N3', cpu_count=1, memory_cap_in_gb=1,
                                                  root_disk_size_in_gb=20)
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

查询竞价实例套餐

使用以下代码可以查询竞价实例套餐

```
def test_list_bid_flavor(self):
        """
        test case for list_bid_flavor
        """
        resp = self.client.list_bid_flavor()
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

释放保护

使用以下代码可以为BCC实例设置释放保护，实例当前设置可查询实例详情DeletionProtection，默认0不保护，1释放保护中:

```
def test_modify_deletion_protection(self):
    """
    test case for modify_deletion_protection
    """
    resp = self.client.modify_deletion_protection("i-XS7Db00e", 1)
    self.assertEqual(
        type(resp),
        baidubce.bce_response.BceResponse)
    if resp is not None and resp.content is not None:
        print(json.loads(resp.content.decode('utf-8')))
    else:
        print(resp)
```

查询实例套餐价格

通过以下代码可以查询实例资源套餐规格对应的价格

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.get_price_by_spec(spec_id="g5",               # 实例规格族id
                                    spec="bcc.g5.c2m8",         # 实例套餐规格
                                    payment_timing="prepay",    # 付费方式，包括Postpaid(后付费)，Prepaid(预付费)两种
                                    zone_name="cn-bj-d",        # 可用区名称
                                    purchase_num=2,             # 实例数量
                                    purchase_length=2           # 购买时长，单位：月
                                    )
    print(resp)
```

实例变更子网

如下代码可以变更实例的子网

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http:/bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.instance_change_subnet(instance_id="i-***",  # 虚拟机实例ID
                                        subnet_id="sbn-***",  # 变更后的子网ID
                                        internal_ip="10.59.***.***",  # 指定内网IP，不指定则随机生成
                                        reboot=True,  # 变更后是否重启
                                        security_group_ids=["g-p8x***"],  # 普通安全组，不支持和企业安全组同时指定，仅变更VPC时
生效
                                        enterprise_security_group_ids=["esg-***"],  # 企业安全组，不支持和普通安全组同时指定，仅
变更VPC时生效
                                        client_token='random-uuid'  # 幂等性Token，是一个长度不超过64位的ASCII字符串
                                        )
    print(res)
```

实例变更VPC

如下代码可以变更实例的VPC

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    instance_id = "i-oUXB****"      # 实例id
    subnet_id = "sbn-5k3wawcr****"  # 子网id
    internal_ip = "192.168.*.*"    # 内网ip
    resp = client.instance_change_vpc(instance_id=instance_id, subnet_id=subnet_id, internal_ip=internal_ip)
    print (resp)
```

查询实例绑定的弹性网卡列表

使用以下代码可以查询实例绑定的弹性网卡列表

```
    def test_list_instance_enis(self):
        """
        test case for list_instance_enis
        """
        resp = self.client.list_instance_enis(instance_id="i-oUXBvdIx")
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

查询实例套餐规格(新)

通过以下代码可以查询实例资源套餐规格列表信息

```
    from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_flavor_spec(zone_name='cn-bj-d')
    print(resp)
```

变配实例（通过实例套餐规格）

使用以下代码可以选择CPU,MemoryCapacityInGB,EphemeralDisks变配指定BCC实例:

```
    def test_resize_instance_by_spec(self):
        """
        test case for resize_instance_by_spec
        """
        resp = self.client.resize_instance_by_spec(instance_id="i-oUXBvdIx", spec='bcc.ic1.c1m1')
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

重装实例（批量）

使用以下代码批量重装实例:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.batch_rebuild_instances(instance_ids=["i-***", "i-***"], # 指定的实例ID列表，最多100个
                                         image_id="m-***", # 指定的镜像ID
                                         admin_pass="***", # 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())，密码需要加密传输
                                         keypair_id="k-***", # 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
                                         )
    print(res)
```

计费变更-转预付费

使用以下代码对实例计费变更-转预付费:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.change_to_prepaid(instance_id="i-sxb2****", # 实例ID
                                    duration=2,  # 续费时长，单位为月
                                    relation_cds=True) # 是否关联CDS转预付费
    print (resp)
```

查询可关机不计费的实例列表

查询可关机不计费的BCC实例列表：

```
    def test_list_instance_no_charge(self):
        """
        test case for list_instance_no_charge
        """
        resp = self.client.list_instance_no_charge(keypair_id='k-Mk1c8QPE', zone_name='cn-bj-a')
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

取消竞价实例订单

通过以下代码可以取消竞价实例订单

```
    def test_cancel_bid_order(self):
        """
        test case for cancel_bid_order
        """
        resp = self.client.cancel_bid_order(order_id='test_id')
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

开通自动续费（包含关联产品）

自动续费仅限预付费产品

```
    from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_create_auto_renew_rules(
        instance_id='i-***', # 实例ID
        renew_eip=True,  # 是否合并eip自动续费，默认值为True
        renew_cds=True,  # 是否合并cds自动续费，默认值为True
        renew_time=2, # 续费时长
        renew_time_unit='year' # 续费时间单位，month/year
    )

    print(resp)
```

关闭自动续费（包含关联产品）

自动续费仅限预付费产品

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_delete_auto_renew_rules(
        instance_id='i-***', # 实例id
        renew_eip=True, # 是否合并关闭eip自动续费，默认值为True
        renew_cds=True  # 是否合并关闭cds自动续费，默认值为True
    )
    print(resp)
```

释放回收站实例

回收站实例7天后自动释放，清理回收站资源，可以使用以下代码将其释放:

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 实例ID
    instanceId = "i-***"
    # 请求从回收站中删除实例
    response = client.delete_recycled_instance(instance_id=instanceId)
    print(response)
```

根据实例ID批量查询实例列表

以下代码可以根据实例ID批量查询实例列表

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_instance_by_instance_ids(instance_ids=['i-***', 'i-***', 'i-***'], # 待查询的实例id列表
                                            marker='123', # 批量获取列表的查询的起始位置
                                            max_keys=1000, # 每页包含的最大数量
                                            )
    print(resp)
```

批量删除实例

以下代码可以批量删除实例，同时也可以选择删除其关联的资源

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_delete_instance_with_related_resource(instance_ids=['i-***'], # 实例ID，最多100个
                                                  related_release_flag=True, # 删除实例时是否同时删除关联的资源
                                                  )
    print(resp)
```

批量重启实例

使用以下代码可以批量重启实例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 实例ID，最多100个
    resp = client.batch_start_instance(instance_ids=['i-***', 'i-***'])
    print(resp)
```

批量停止实例

使用以下代码可以批量停止实例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.batch_stop_instance(
        instance_ids=['i-***', 'i-***'], # 实例id组成的集合，最多100个
        force_stop=False, # 指定实例是否强制关机，可选值：True、False
        stop_with_no_charge=True # 是否关机计费
    )
    print(resp)
```

批量变配实例

使用以下代码可以批量变配实例

```
    from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 执行批量变配实例的操作
    resp = client.batch_resize_instance(instance_ids=['i-***'], # 实例id组成的集合，最多30个
                                        spec='bcc.g5.c8m32', # 批量变配的实例规格
                                        subnet_id='sbn-***', # 子网id
                                        logical_zone='cn-bj-d', # 可用区
                                        internal_ip_v4='ipv4'，内网IP
                                        )
    print(resp)
```

查询可以变配的实例规格

使用以下代码可以查询可以变配的实例规格

```
    from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 执行查询可以变配的实例规格的操作
    resp = client.list_available_resize_specs(instance_ids=['i-***'], # 实例ID。实例ID、实例规格和规格族ID有且只能传一个
                                              # spec='bcc.g5.c2m8', # 实例规格
                                              # spec_id='subnet_id', # 规格族ID
                                              # logical_zone='zone_name', # 实例所在可用区。若传入spec或specId，则该参数为必填。
                                              )
    print(resp)
```

批量转预付费

使用以下代码可以将实例批量转预付费，若需要对实例关联的cds变更，则必须在CdsList中传入所有需要转为预付费的cdsId，若所有cds均变更计费传入all。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc.bcc_model import PayTimingChangeReqModel

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    req1 = PayTimingChangeReqModel('i-***', cdsList=['all'], autoPay=True, duration=1)
    req2 = PayTimingChangeReqModel('i-***', cdsList=['all'], duration=1)
    req = [req1, req2]
    # req集合元素最多100个
    resp = client.batch_change_instance_to_prepay(req)
    print(resp)
```

批量转后付费

使用以下代码可以将实例批量转后付费，若需要对实例关联的cds变更，则必须在CdsList中传入所有需要转为后付费的cdsId，若所有cds均变更计费传入all。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc.bcc_model import PayTimingChangeReqModel

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    req1 = PayTimingChangeReqModel('i-***', cdsList=['all'])
    req2 = PayTimingChangeReqModel('i-***', cdsList=['all'])
    req = [req1, req2]
    # req集合元素最多100个
    resp = client.batch_change_instance_to_postpay(req)
    print(resp)
```

获取实例角色列表

使用以下代码可以获取实例角色列表

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 执行获取实例角色列表的操作
    resp = client.list_instance_roles()
    print(resp)
```

绑定角色

使用以下代码可以为BCC实例绑定角色

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 执行绑定角色的操作
    resp = client.bind_instance_role(['i-***', 'i-***'],  # 实例ID列表
                                     role_name='BCC_Test', # 角色名称
                                    )
    print(resp)
```

解绑角色

使用以下代码可以为BCC实例解绑角色

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 执行解绑角色的操作
    resp = client.unbind_instance_role(['i-***', 'i-***'], # 实例ID列表
                                       role_name='BCC_Test' # 角色名称
                                       )
    print(resp)
```

添加Ipv6

使用以下代码可以为BCC实例添加Ipv6

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.add_ipv6(
        instance_id='i-***',  # 实例id
        ipv6_address='240c:4081:8005:ca0b:***:***', # ipv6地址，为空表示自动分配ipv6
        reboot=True) # 是否重启，true表示重启，为空表示不重启
    print(resp)
```

释放Ipv6

使用以下代码可以为BCC实例释放Ipv6

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    resp = client.delete_ipv6(
        instance_id='i-***', # 实例id
        reboot=True) # 是否重启，true表示重启，为空表示不重启
    print(resp)
```

批量添加主网卡辅助IP

如下代码可以批量添加主网卡辅助IP：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    private_ips = ["192.168.***.***", "192.168.***.***"]
    res = client.batch_add_ip(instance_id="i-***", # 实例ID
                              private_ips=private_ips, # 辅助IP，最多100个，和SecondaryPrivateIpAddressCount不可同时使用
                              # secondary_private_ip_address_count=2 # 自动分配IP数量，和PrivateIps不可同时使用
                              allocate_multi_ipv6_addr=True  # 是否分配IPV6，创建IPV6必须是True
                              )
    print(res)
```

批量删除网卡辅助IP

如下代码可以批量删除主网卡辅助IP：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    private_ips = ["192.168.***.***", "192.168.***.***"]
    res = client.batch_delete_ip(instance_id="i-***", # 实例ID
                                 private_ips=private_ips, # 辅助IP，最多100个
                                 )
    print(res)
```

释放实例（POST）

使用以下代码释放单个后付费云服务器实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.release_instance_with_related_resources(instance_id="i-***", # 实例id
                                                         related_release_flag=True, # 是否关联释放当前时刻，实例挂载的EIP和数据盘
                                                         delete_cds_snapshot_flag=True, # 是否释放云磁盘快照
                                                         delete_related_enis_flag=True, # 实例释放时是否删除关联的ENI
                                                         bcc_recycle_flag=True # 实例释放时是否进入回收站
                                                         )
    print(res)
```

释放预付费实例（POST）

释放预付费云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。 释放的时候默认只释放云服务器实例和系统盘，用户可以选择是否关联释放云服务器实例挂载的EIP和数据盘，关联的EIP和数据盘需全部释放或全部不释放，不支持选择部分释放，部分不释放。 当选择关联释放EIP和数据盘，云磁盘快照可选择一同关联释放，若数据盘不关联释放，云磁盘快照也不支持关联释放。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    res = client.release_prepaid_instance_with_related_resources(instance_id="i-***", # 实例id
                                                 related_release_flag=True, # 是否关联释放当前实例挂载的EIP和数据盘（只有该字段为true时 delete_cds_snapshot_flag字段才会有效，若该字段为false,delete_cds_snapshot_flag字段的值无效）
                                                 delete_cds_snapshot_flag=True, # 是否释放云磁盘快照
                                                 delete_related_enis_flag=True, # 实例释放时是否删除关联的ENI
                                                 )
    print(res)
```

磁盘

直接创建磁盘

使用以下代码可以创建磁盘:

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)


    # 标签列表
    tags = [bcc_model.TagModel("example", "createCds")]

    # 自动快照策略
    auto_snapshot_policy = bcc_model.AutoSnapshotPolicyModel(name="aspExample",     # 自动快照策略名称
                                                             timePoints=[1, 2, 3],  # 自动快照创建时间点, 参数为0~23的列表
                                                             repeatWeekdays=[1, 5], # 自动快照重复日期, 参数为0~6的列表
                                                             retentionDays=30)     # 自动快照的保留时间，单位为天。 -1：永久保存
1~65536：指定保存天数

    resp = client.create_volume_with_cds_size(cds_size_in_gb=5,
                                              charge_type='Postpaid',
                                              zone_name='cn-bj-a', # 可用区名称
                                              instance_id='', # 创建完成后待挂载的实例ID
                                              encrypt_key='c6c0c0b3-****-****-****-ca6d5751504f', # 加密密钥ID
                                              name='test-name', # 磁盘名称
                                              description='desc', # 磁盘描述
                                              renew_time_unit='month', # 续费时长单位,支持:month(月) / year(年)
                                              renew_time=2, # 续费时长,
                                              relation_tag=True,  # 关联快照和快照链是否统一加标签
                                              auto_snapshot_policy=auto_snapshot_policy, # 自动快照策略
                                              tags=tags) # 绑定标签
    print (resp)
```

各磁盘类型可配容量范围详见磁盘性能。

从快照创建CDS磁盘

使用以下代码可以从快照创建磁盘：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)


    resp = client.create_volume_with_snapshot_id(snapshot_id="s-er2Q****",  # 快照ID
                                         charge_type='Postpaid',  # 支付方式
                                         cluster_id="DC-jh5q****",  # 专属集群ID
                                         purchase_count=2)  # 购买数量
    print (resp)
```

查询磁盘列表

使用以下代码查询所有的磁盘列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_volumes(instance_id="i-cVh8****",  # 磁盘挂载的实例ID
                               zone_name="cn-bj-c",  # 可用区
                               marker="v-3uIJ****",  # 起始卷ID
                               max_keys=10, # 返回列表的最大数量
                               cluster_id="DC-rWKx****") # CDS集群id
    print (resp)
```

查询磁盘详情

使用以下代码可以根据磁盘ID查询单个磁盘的详细信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.get_volume(volume_id='v-3uIJ****') # 指定磁盘ID
    print (resp)
```

挂载CDS磁盘

使用以下代码将磁盘挂载在指定的BCC实例下，注意CDS磁盘需要挂载在与其处在相同可用区下的BCC实例上。

只有磁盘状态为 Available 且实例状态为 Running 或 Stopped 时才允许挂载：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置要挂载的磁盘ID
    volume_id = 'v-***'
    # 设置要挂载的实例ID
    instance_id = 'i-***'
    resp = client.attach_volume(volume_id, instance_id)
    print(resp)
```

卸载CDS磁盘

使用以下代码将指定的磁盘从实例中卸载。

只有实例状态为 Running 或 Stopped 时，磁盘才可以执行此操作：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置要卸载的磁盘ID
    volume_id = 'v-***'
    # 设置所挂载的实例ID
    instance_id = 'i-***'
    resp = client.detach_volume(volume_id, instance_id)
    print(resp)
```

释放CDS磁盘

使用以下代码释放未挂载的CDS磁盘，系统盘不能释放：

- 磁盘释放后不可恢复。默认情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作。

  def release_volume(self):

```
 #设置要释放的磁盘ID
volume_id = 'your-choose-volume-id'

self.assertEqual(

    type(self.client.release_volume(volume_id)),

    baidubce.bce_response.BceResponse)
```

磁盘扩容

使用以下代码对磁盘进行扩容：

- 磁盘只能进行扩容操作，不支持缩容。
- 只有Available状态的磁盘，才能进行扩容操作。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作。
- 各磁盘类型可配容量范围详见磁盘性能。

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"

    # 请求扩容磁盘，变更磁盘类型和大小
    response = client.resize_volume(volume_id=volume_id,            # 磁盘ID
                                    new_cds_size=50,                # 扩容后磁盘大小, 单位为GB
                                    new_volume_type="cloud_hp1")    # 扩容后磁盘类型, 可取值见：
https://cloud.baidu.com/doc/BCC/s/6jwvyo0q2#storagetype
    print(response)
```

回滚磁盘

使用以下代码用指定磁盘自身的快照回滚磁盘内容：

- 磁盘状态必须为 Available 才可以执行此操作。
- 指定 snapshotId 必须是由 volumeId 创建的快照。
- 若是回滚系统盘，实例状态必须为 Running 或 Stopped 才可以执行此操作。
- 回滚系统盘快照，自本次快照以来的系统盘数据将全部丢失，不可恢复：

  from baidubce.auth.bce_credentials import BceCredentials from baidubce.bce_client_configuration import BceClientConfiguration from baidubce.services.bcc import bcc_client, bcc_model

  if **name** == '**main**': # 设置您的ak、sk和要访问的地域 HOST = b'http://bcc.bj.baidubce.com' AK = b'ak' SK = b'sk' # 设置默认配置 config = BceClientConfiguration(credentials=BceCredentials(AK, SK), endpoint=HOST) # 创建bcc client client = bcc_client.BccClient(config) volume_id = 'v-vG0z**' **# 用于回滚指定磁盘的快照ID，必须为 volumeId 所创建的快照 snapshot_id = 's-HkbJ**'** resp = client.rollback_volume(volume_id, snapshot_id) print (resp)

磁盘续费

使用以下代码可以实现磁盘续费。

磁盘扩缩容期间不能进行续费操作：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 待续费磁盘ID
    volume_id = 'v-PGE6****'
    # 设置续费时长,目前只支持续费单位为月
    billing = bcc_model.Billing('Prepaid', 2)
    resp = client.purchase_reserved_volume(volume_id=volume_id, billing=billing)
    print (resp)
```

修改磁盘属性（新）

使用以下代码可以修改磁盘属性：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 待修改磁盘ID
    volume_id = 'v-vG0z****'
    # 设置磁盘新名称
    cds_name = 'new-cds-name'
    resp = client.modify_volume_Attribute(volume_id=volume_id, cds_name=cds_name)
    print (resp)
```

修改磁盘计费方式（新）

使用以下代码可以将按量付费（后付费）磁盘变更为包年包月（预付费）： # !/usr/bin/env python # coding=utf-8 from baidubce.services.bcc import bcc_client from baidubce.services.bcc import bcc_model from baidubce.auth.bce_credentials import BceCredentials from baidubce.bce_client_configuration import BceClientConfiguration

```
if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"
    post_paid_billing = bcc_model.Billing(paymentTiming='Prepaid',          # 预付费计费方式, 指定paymentTiming为Prepaid,
可不填
                                          reservationLength=3,              # 预付费时长, 单位为月时取值范围为1-9
                                          reservationTimeUnit='Month')      # 预付费时长单位，当前支持取值为Month

    # 请求将后付费磁盘变更计费方式为预付费
    response = client.modify_volume_charge_type(volume_id=volume_id,            # 磁盘ID
                                                billing=post_paid_billing)      # 计费方式
    print(response)
```

使用以下代码可以将包年包月（预付费）磁盘变更为按量付费（后付费）： # !/usr/bin/env python # coding=utf-8 from baidubce.services.bcc import bcc_client from baidubce.services.bcc import bcc_model from baidubce.auth.bce_credentials import BceCredentials from baidubce.bce_client_configuration import BceClientConfiguration

```
if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 预付费磁盘变更计费方式为后付费
    volume_id = "v-***"
    post_paid_billing = bcc_model.Billing(paymentTiming='Postpaid',       # 后付费计费方式, 指定paymentTiming为
Postpaid, 可不填
                                          reservationLength=0)              # 后付费情况下，必须需要设置reservationLength为0

    # 请求将预付费磁盘变更计费方式为后付费
    response = client.modify_volume_charge_type(volume_id=volume_id,        # 磁盘ID
                                                billing=post_paid_billing)  # 计费方式
    print(response)
```

释放cds磁盘(新)

使用以下代码可以实现释放CDS磁盘

- 该接口用于释放未挂载的CDS磁盘，系统盘不能释放
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误
- 如果 volumeId 的磁盘不存在，将提示 404 错误
- 可以控制是否释放关联的自动/手动快照、是否进入回收站

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"

    # 请求释放cds盘
    response = client.release_volume_new(volume_id=volume_id,                    # 磁盘ID
                                         auto_snapshot="on",                     # 是否删除磁盘关联的自动快照
                                         manual_snapshot="on",                   # 是否删除磁盘关联的手动快照
                                         recycle="off")                          # 是否将磁盘回收至回收站，为off时直接释放，不进入回收站
    print(response)
```

磁盘开通自动续费

使用以下代码可以开通磁盘自动续费：

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"
    # 请求开通自动续费
    response = client.auto_renew_cds_volume(volume_id=volume_id,        # 磁盘ID
                                            renew_time=3,               # 续费时长, 单位为月时取值范围为1-9
                                            renew_time_unit="month")    # 续费时长单位，当前支持取值为: month
    print(response)
```

磁盘取消自动续费

使用以下代码可以取消磁盘自动续费：

```
#### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"
    # 请求取消自动续费
    response = client.cancel_auto_renew_cds_volume(volume_id=volume_id)     # 磁盘ID
    print(response)
```

查询可用区的磁盘信息

使用以下代码可以查询可用区的磁盘信息：

```
#### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    azone_name = "cn-bj-c"                  # 可用区名称；当传入azone_name为空串或为非法azone时，会返回全部可用区的
可购买磁盘信息。

    # 请求查询对应可用区下可购买磁盘信息
    response = client.get_available_disk_info(zone_name=azone_name)
    print(response)
```

磁盘绑定标签

使用以下代码可以给指定磁盘绑定标签：

```
#### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"                                  # 磁盘ID
    tag1 = bcc_model.TagModel(tagKey='TagKey01',               # Tag key
                              tagValue='TagValue01')           # Tag value
    tag2 = bcc_model.TagModel(tagKey='TagKey02',               # Tag key
                              tagValue='TagValue02')           # Tag value
    tag_list = list()                                    # 标签列表
    tag_list.append(tag1)
    tag_list.append(tag2)

    # 请求为磁盘绑定标签
    response = client.tag_volume(volume_id=volume_id,       # 磁盘ID
                                 relation_tag=False,        # 是否为相关资源绑定标签
                                 tags=tag_list)             # 标签列表
    print(response)
```

磁盘解绑标签

使用以下代码可以给指定磁盘解绑标签：

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    volume_id = "v-***"                                 # 磁盘ID
    tag1 = bcc_model.TagModel(tagKey='TagKey01',              # Tag key
                              tagValue='TagValue01')          # Tag value
    tag2 = bcc_model.TagModel(tagKey='TagKey02',              # Tag key
                              tagValue='TagValue02')          # Tag value
    tag_list = list()                                   # 标签列表
    tag_list.append(tag1)
    tag_list.append(tag2)

    # 请求为磁盘解绑标签
    response = client.untag_volume(volume_id=volume_id,       # 磁盘ID
                                   relation_tag=False,        # 是否为相关资源绑定标签
                                   tags=tag_list)             # 标签列表
    print(response)
```

镜像

通过实例创建自定义镜像

- 用于创建自定义镜像，默认每个账号配额20个，创建后的镜像可用于创建实例。
- 只有 Running 或 Stopped 状态的实例才可以执行成功。
    - 仅限通过系统盘快照创建自定义镜像。

```
 import uuid

from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    image_name = 'instanceImage1' + client_token                # 创建的镜像名称
    instance_id = 'i-***'                                       # 创建的实例id
    snapshot_id = 'm-***'                                       # 快照id

    resp = client.create_image_from_snapshot_id(image_name,
                                    snapshot_id=snapshot_id,
                                    client_token=client_token)
    print(resp)
```

通过快照创建自定义镜像

通过快照创建时，只有 Available 状态的快照才可以执行成功：

```
 import uuid
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    image_name = 'instanceImage1' + client_token                # 创建的镜像名称
    instance_id = 'i-***'                                       # 创建的实例id
    snapshot_id = 'm-***'                                       # 快照id

    resp = client.create_image_from_snapshot_id(image_name,
                                    snapshot_id=snapshot_id,
                                    client_token=client_token)
    print(resp)
```

查询镜像列表

使用以下接口查询用户所有的镜像信息：

- 查询的镜像信息中包括公共镜像、自定义镜像和服务集成镜像。

- 支持按 imageType 来过滤查询，此参数非必需，未设置时默认为 All,即查询所有类型的镜像。

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    response = client.list_images()

    print(response)
```

查询镜像详情

使用以下接口根据指定镜像ID查询单个镜像的详细信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    imageId = "m-***"
    response = client.get_image(imageId)

    print(response)
```

删除自定义镜像

使用以下接口删除用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能删除

- 镜像删除后无法恢复，不能再用于创建、重置实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    imageId = "m-***"
    response = client.delete_image(imageId)

    print(response)
```

跨地域复制镜像（新）

使用以下接口跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    imageId = "m-***"
    destinationImageName = "new-image"
    destinations = ["hkg"]

    response = client.remote_copy_image(imageId, destinationImageName, destinations)

    print(response)
```

取消跨地域复制镜像（新）

用于取消跨地域复制自定义镜像：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建 client
    client = bcc_client.BccClient(config)
    imageId = "m-***"

    response = client.cancle_remote_copy_image(imageId)

    print(response)
```

共享自定义镜像（新）

使用以下接口可以共享用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能共享：

```
 #### !/usr/bin/env python
#### coding=utf-8

from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    share_image_id = "m-***"
    share_account_id = "***"
    # 请求将镜像共享给指定账号
    response = client.share_image(image_id=share_image_id,              # 镜像ID
                                  account_id=share_account_id)          # 共享账号ID

    print(response)
```

取消共享自定义镜像（新）

使用以下接口可以取消共享用户指定的自定义镜像：

```
 #### !/usr/bin/env python
#### coding=utf-8

from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    unshare_image_id = "m-***"
    unshare_account_id = "***"
    # 请求取消将镜像共享给指定账号
    response = client.unshare_image(image_id=share_image_id,              # 镜像ID
                                    account_id=share_account_id)          # 共享账号ID

    print(response)
```

查询已共享用户列表（新）

使用以下接口查询镜像已共享的用户列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    image_id = 'm-***'              # 镜像id

    print(client.list_shared_user(image_id=image_id).users)
```

根据实例短id批量查询os信息（新）

使用以下接口可以根据实例id批量查询OS信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
 # 设置您的ak、sk和要访问的地域
 HOST = 'http://bcc.bj.baidubce.com'
 AK = 'ak'
 SK = 'sk'
 # 设置默认配置
 config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
 endpoint=HOST)
 # 创建bcc client
 client = bcc_client.BccClient(config)

 # 待查询实例id
 instance_id1 = 'i-***'              # 查询实例Id
 instance_id2 = 'i-***'              # 查询实例Id
 instance_ids = []
 instance_ids.append(instance_id1)
 instance_ids.append(instance_id2)
 resp = client.list_os(instance_ids=instance_ids)
 print(resp.os_info)
```

镜像绑定标签

使用以下代码可以给指定镜像绑定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
 # 设置您的ak、sk和要访问的地域
 HOST = 'http://bcc.bj.baidubce.com'
 AK = 'ak'
 SK = 'sk'
 # 设置默认配置
 config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                 endpoint=HOST)
 # 创建bcc client
 client = bcc_client.BccClient(config)

 tag1 = bcc_model.TagModel(tagKey='key1',  # 标签key value
                           tagValue='value1')
 tag2 = bcc_model.TagModel(tagKey='key2',  # 标签key value
                           tagValue='value2')
 tag_list = []
 tag_list.append(tag1)
 tag_list.append(tag2)
 resp = client.bind_image_to_tags(image_id='m-***', tags=tag_list)
 print(resp)
```

镜像解绑标签

使用以下代码可以给指定镜像绑定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    tag1 = bcc_model.TagModel(tagKey='key1',  # 标签key value
                              tagValue='value1')
    tag2 = bcc_model.TagModel(tagKey='key2',  # 标签key value
                              tagValue='value2')
    tag_list = []
    tag_list.append(tag1)
    tag_list.append(tag2)
    resp = client.unbind_image_to_tags(image_id='m-***', tags=tag_list)
    print(resp)
```

导入镜像

使用以下代码可以导入一个镜像：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.import_custom_image(
        os_name='Centos',  # 操作系统名称
        os_arch='64',  # 操作系统位数
        os_type='linux',  # 操作系统类型
        os_version='7.2',  # 操作系统版本
        name='image-01',  # 镜像名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
        bos_url='https://bcc-bos.bj.bcebos.com/image_test/CentOS-6-x86_64-GenericCloud.qcow2')  # bos镜像地址

    print(resp)
```

根据实例规格查询可用公共镜像

使用以下代码可以根据实例规格查询可用公共镜像：

```
    def test_get_available_images_by_spec(self):
        """
        test case for get_available_images_by_spec
        """
        resp = self.client.get_available_images_by_spec(spec='bcc.ic4.c1m1', os_name='Centos')
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

快照

创建快照

使用以下代码可以从指定的磁盘创建快照，支持创建系统盘快照和CDS磁盘快照。

创建系统盘快照时，实例的状态必须为 Running 或 Stopped 此操作才能正常完成。

创建CDS磁盘快照时，磁盘的状态需要为InUse或 Available 此操作才能正常完成。

可创建快照数量的配额取决于账号下拥有的磁盘总数，当前最多为磁盘数量乘以8。

```
import uuid

from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    client_token = str(uuid.uuid4())
    # 设置快照名称
    snapshot_name = 'snapshot_' + client_token
    # 创建快照的磁盘ID
    volume_id = 'v-***'
    resp = client.create_snapshot(volume_id, snapshot_name, client_token=client_token)
    print(resp)
```

查询快照列表

使用以下代码以查询所有的快照信息，包括系统盘快照和数据盘快照信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 创建快照的磁盘ID
    volume_id = 'v-***'
    # 批量获取列表的查询的起始位置
    marker = "your-marker"
    # 每页包含的最大数量
    max_keys = 100
    resp = client.list_snapshots(marker=marker, max_keys=max_keys, volume_id=volume_id)
    print(resp)
```

查询快照详情

使用以下代码可以通过指定的快照ID查询单个快照的详细信息：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 快照ID
    snapshot_id = 's-EJq3B***'
    resp = client.get_snapshot(snapshot_id)
    print(resp)
```

删除快照

使用以下代码可以通过指定快照ID删除快照。

只有当快照的状态为 CreatedFailed 或 Available时，操作才能正常完成：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 待删除快照ID
    snapshot_id = 's-EJq3B***'
    resp = client.delete_snapshot(snapshot_id=snapshot_id)
    print(resp)
```

查询快照链列表

使用以下代码可以查询快照链列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 磁盘ID
    volume_id = 'v-***'
    resp = client.list_snapshot_chain(volume_id)
    print(resp)
```

快照链绑定标签

使用以下代码可以给指定快照链绑定标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    tag1 = bcc_model.TagModel(tagKey='TestKey02',
                              tagValue='TestValue02')
    tag2 = bcc_model.TagModel(tagKey='TestKey03',
                              tagValue='TestValue03')
    tags = []
    tags.append(tag1)
    tags.append(tag2)
    chain_id = "sl-i7TLI***"
    resp = client.tag_snapshot_chain(chain_id, tags=tags)
    print(resp)
```

快照链解绑标签

使用以下代码可以给指定快照链解绑标签：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    tag1 = bcc_model.TagModel(tagKey='TestKey02',
                              tagValue='TestValue02')
    tag2 = bcc_model.TagModel(tagKey='TestKey03',
                              tagValue='TestValue03')
    tags = []
    tags.append(tag1)
    tags.append(tag2)
    chain_id = "sl-i7TLI***"
    resp = client.untag_snapshot_chain(chain_id, tags=tags)
    print(resp)
```

跨地域复制快照

使用以下代码可以将一份磁盘快照从一个地域复制到另一个地域：

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.services.bcc import bcc_model
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 跨区域复制快照demo
    snapshot_id = "s-GTI*****"          # 快照ID
    dest_region_info_list = list()      # 目标区域信息列表

    dest_region_info1 = bcc_model.DestRegionInfoModel(destRegion="bd",                # 目标区域名称
                                                      name="snapshot_name_for_bd")     # 快照名称
    dest_region_info2 = bcc_model.DestRegionInfoModel(destRegion="gz",                # 目标区域名称
                                                      name="snapshot_name_for_gz")     # 快照名称
    dest_region_info_list.append(dest_region_info1)
    dest_region_info_list.append(dest_region_info2)

    # 请求跨区域复制快照
    response = client.create_remote_copy_snapshot(snapshot_id=snapshot_id,                        # 快照ID
                                                  dest_region_infos=dest_region_info_list)        # 目标区域信息列表

    print(response)
```

自动快照策略

创建自动快照策略

以下代码用于创建自动快照策略：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    #设置快照名称
    asp_name = 'your-asp-name'
    #设置一天中做快照的时间点，取值为0~23，0为午夜12点
    time_points = [0, 22]
    #设置一周中做快照的时间点，取值为0~6，0为周日
    repeat_week_days = [0, 5]
    #设置自动快照保留天数，-1为永久保留
    retention_days = '-1'
    resp = client.create_asp(asp_name=asp_name,
                             time_points=time_points,
                             repeat_week_days=repeat_week_days,
                             retention_days=retention_days)
    print(resp)
```

绑定自动快照策略

以下代码用于绑定自动快照策略：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置需要绑定的快照id
    asp_id = 'asp-hZll9***'
    # 设置需要绑定的磁盘id，可多选
    volume_ids = ['v-***', 'v-***']

    client.attach_asp(asp_id=asp_id, volume_ids=volume_ids)
```

解绑自动快照策略

以下代码用于解绑自动快照策略：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置需要解绑的快照id
    asp_id = 'asp-hZll9***'
    # 设置需要解绑的磁盘id，可多选
    volume_ids = ['v-***', 'v-***']

    client.detach_asp(asp_id=asp_id, volume_ids=volume_ids)
```

删除自动快照策略

以下代码用于删除自动快照策略：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置需要删除的快照id
    asp_id = 'asp-hZll9***'

    client.delete_asp(asp_id=asp_id)
```

查询自动快照策略列表

以下代码用于查询自动快照策略列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_asps()
    print(resp)
```

查询自动快照策略详情

以下代码用于查询自动快照策略详情：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置需要查询的快照id
    asp_id = 'asp-hZll9***'

    resp = client.get_asp(asp_id=asp_id)
    print(resp)
```

自动快照策略变更

以下代码可以更新一个自动快照策略：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.update_asp(name="changeDone1", # 自动快照策略名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
                             asp_id="asp-osTov***", # 自动策略的id
                             time_points=[1, 2], # 一天中做快照时间点，取值为0~23。
                             repeat_week_days=[1, 2], # 一周中做快照的时间，取值为0~6。
                             retention_days=12) # 自动快照保留天数，取-1则永久保留。
    print(resp)
```

安全组

创建安全组

使用如下代码创建安全组以及安全组规则：

```
 def create_security_group(self):

    client_token = generate_client_token()

    #设置安全组名称
    security_group_name = 'your_security_group_name' + client_token

    #设置安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel('rule_' + client_token,# 设置备注
                                            'ingress',# 设置入站/出站，取值ingress或egress，必选参数
                                             portRange='1-65535',# 设置端口范围，默认空时为1-65535，可以指定80等单个端口
                                             protocol='tcp',# 设置协议类型
                                             sourceGroupId='',# 设置源安全组ID
                                             sourceIp='')# 设置源IP地址，与sourceGroupId不能同时设定值

     security_group_rule_list = []

     security_group_rule_list.append(security_group_rule)

     #vpc的Id，空的时候，表示默认VPC的id
     security_group_rule_vpc_id = 'your-vpc-id'

     #设置安全组描述
     security_group_desc = 'your_security_group_desc'

     self.assertEqual(

         type(self.client.create_security_group(name=security_group_name,
                                                rules=security_group_rule_list,
                                                vpc_id=security_group_rule_vpc_id,
                                                desc=security_group_desc,
                                                client_token=client_token)),

         baidubce.bce_response.BceResponse)
```

查询安全组列表

使用如下代码查询用户的所有安全组信息：

```
def list_security_groups(self):

    #批量获取列表的查询的起始位置
    marker = 'your-marker'

    #每页包含的最大数量
    maxKeys = 100

    #实例id，可查询实例关联的安全组列表
    instance_id = 'your-instance-id'

    #vpcid，可查询实例关联的安全组列表
    vpc_id = 'your-vpc-id'

    self.assertEqual(

        type(self.client.list_security_groups(instance_id=instance_id,
                                              vpc_id=vpc_id,
                                              marker=marker,
                                              max_keys=max_keys)),

        baidubce.bce_response.BceResponse)
```

删除安全组

使用如下代码删除某个安全组：

```
def delete_security_group(self):

    #要删除的安全组ID
    security_group_id = 'your-security-group-id'

    self.assertEqual(

        type(self.client.delete_security_group(security_group_id)),

        baidubce.bce_response.BceResponse)
```

授权安全组规则

使用如下代码在安全组中授权新的安全组规则：

```
 def authorize_security_group_rule(self):

    client_token = generate_client_token()

    #设置安全组ID
    security_group_id = 'your-security-group-id'

    #设置安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel(direction='ingress',# 设置入站/出站，取值ingress或
egress，必选参数
                                                           portRange='80-90',# 设置端口范围，默认空时为1-65535，可以指定80等单个端
口
                                                           protocol='tcp')# 设置协议类型

    self.assertEqual(

        type(self.client.authorize_security_group_rule(security_group_id=security_group_id,
                                                       rule=security_group_rule,
                                                       client_token=client_token)),

        baidubce.bce_response.BceResponse)
```

撤销安全组规则

使用如下代码撤销安全组中的规则：

```
 def revoke_security_group_rule(self):

    client_token = generate_client_token()

    #待撤销安全组ID
    security_group_id = 'your-security-group-id'

    #待撤销安全组规则
    security_group_rule = bcc_model.SecurityGroupRuleModel(direction='ingress',# 设置入站/出站，取值ingress或
egress，必选参数
                                                           portRange='80-90',# 设置端口范围，默认空时为1-65535，可以指定80等单个端
口
                                                           protocol='tcp')# 设置协议类型

    self.assertEqual(

        type(self.client.revoke_security_group_rule(security_group_id=security_group_id，
                                                    security_group_rule,
                                                    client_token=client_token)),

        baidubce.bce_response.BceResponse)
```

密钥对

创建密钥对

以下代码用于创建密钥对：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    keypair_name = 'bcc'  # 待创建的密钥对名称
    keypair_desc = '用于bcc'  # 待创建的密钥对的描述
    resp = client.create_keypair(
        keypair_name=keypair_name,
        keypair_desc=keypair_desc)
    print(resp)
```

导入密钥对

以下代码用于导入密钥对：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    keypair_name = 'keyName'
    keypair_desc = 'Desc'
    public_key = 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQA***OGfME75krwo6DbUkpzJPYbBctqD'

    resp = client.import_keypair(keypair_name=keypair_name, # 待创建的密钥对名称
                                 keypair_desc=keypair_desc, # 待创建的密钥对的描述
                                 public_key=public_key), # 公钥内容
    print(resp)
```

查询密钥对列表

以下代码用于查询密钥对列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    resp = client.list_keypairs(
        max_keys=1, # 每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
        marker='k-Su1ZAg78' # 批量获取列表的查询的起始位置，是一个由系统生成的字符串
    )
    print(resp)
```

查询密钥对详情

以下代码用于查询密钥对列表：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    keypair_id = 'k-wgBDZ***'   # 设置被查询密钥对id

    resp = client.get_keypair(keypair_id=keypair_id)
    print(resp)
```

绑定密钥对

以下代码用于绑定密钥对到BCC实例：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    keypair_id = 'k-***'        # 设置被绑定密钥对id
    instance_ids = ['i-***', 'i-***']     # 设置被绑定实例id，可多个
    resp = client.attach_keypair(keypair_id=keypair_id, instance_ids=instance_ids)
    print(resp)
```

解绑密钥对

以下代码用于解绑密钥对：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)
    # 设置要解绑的密钥对id
    keypair_id = 'k-***'
    # 待解绑的虚机的短id列表
    instance_ids = ['i-***', 'i-***']
    # 发起解绑
    resp = client.detach_keypair(keypair_id=keypair_id, instance_ids=instance_ids)
    print(resp)
```

删除密钥对

以下代码用于删除密钥对：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置要删除的密钥对id
    keypair_id = 'k-wgBDZ***'
    # 发起删除请求
    client.delete_keypair(keypair_id=keypair_id)
```

密钥对重命名

以下代码用于密钥对重命名：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置要重命名的密钥对id
    keypair_id = 'k-wgBDZ***'
    # 设置要重命名的密钥对名称
    keypair_name = 'Name1'
    # 更改秘钥对名称
    resp = client.rename_keypair(keypair_id=keypair_id, keypair_name=keypair_name)
    print(resp)
```

更新密钥对描述

以下代码用于更新密钥对描述：

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client

if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = 'http://bcc.bj.baidubce.com'
    AK = 'ak'
    SK = 'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 设置要更新描述的密钥对id
    keypair_id = 'k-wgBDZ***'
    # 设置要更新的密钥对描述
    keypair_desc = 'change desc'
    # 更新密钥对描述
    resp = client.update_keypair_desc(keypair_id=keypair_id, keypair_desc=keypair_desc)
    print(resp)
```

磁盘专属集群

创建磁盘专属集群

使用以下代码可以根据实例ID批量查询实例列表：

```
    def test_create_volume_cluster(self):
        """
        test case for create_volume_cluster
        """
        client_token = generate_client_token()
        cluster_size_in_gb = 97280
        create_response = self.client.create_volume_cluster(cluster_size_in_gb=cluster_size_in_gb, zone_name='cn-bj-a')
        print(create_response)
        self.assertEqual(
            type(create_response),
            baidubce.bce_response.BceResponse)
```

磁盘专属集群列表

使用以下代码可以根据实例ID批量查询实例列表：

```
    def test_list_volume_cluster(self):
        """
        test case for list_volume_cluster
        """
        create_response = self.client.list_volume_cluster()
        print(create_response)
        self.assertEqual(
            type(create_response),
            baidubce.bce_response.BceResponse)
```

磁盘专属集群详情

使用以下代码可以根据实例ID批量查询磁盘专属集群详情：

```
def test_get_volume_cluster(self):
    """
    test case for get_volume_cluster
    """
    create_response = self.client.get_volume_cluster(cluster_id='DC-yWfhpUbN')
    print(create_response)
    self.assertEqual(
        type(create_response),
        baidubce.bce_response.BceResponse)
```

磁盘专属集群扩容

使用以下代码可以给磁盘专属集群扩容：

```
def test_resize_volume_cluster(self):
    """
    test case for resize_volume_cluster
    """
    create_response = self.client.resize_volume_cluster(cluster_id='DC-yWfhpUbN', new_cluster_size=107520)
    print(create_response)
    self.assertEqual(
        type(create_response),
        baidubce.bce_response.BceResponse)
```

磁盘专属集群续费

使用以下代码可以给专属集群续费：

```
def test_renew_volume_cluster(self):
    """
    test case for renew_volume_cluster
    """
    create_response = self.client.renew_volume_cluster(cluster_id='DC-yWfhpUbN')
    print(create_response)
    self.assertEqual(
        type(create_response),
        baidubce.bce_response.BceResponse)
```

磁盘专属集群自动续费

使用以下代码可以给专属集群设置自动续费：

```
def test_autoRenew_volume_cluster(self):
    """
    test case for autoRenew_volume_cluster
    """
    create_response = self.client.autoRenew_volume_cluster(cluster_id='DC-yWfhpUbN')
    print(create_response)
    self.assertEqual(
        type(create_response),
        baidubce.bce_response.BceResponse)
```

磁盘专属集群取消自动续费

使用以下代码可以取消磁盘专属集群自动续费：

```
    def test_cancel_autoRenew_volume_cluster(self):
        """
        test case for cancel_autoRenew_volume_cluster
        """
        create_response = self.client.cancel_autoRenew_volume_cluster(cluster_id='DC-yWfhpUbN')
        print(create_response)
        self.assertEqual(
            type(create_response),
            baidubce.bce_response.BceResponse)
```

部署集

创建部署集

通过以下代码可以依据给定的创建参数创建部署集：

```
 #### !/usr/bin/env python
#### coding=utf-8

from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    response = client.create_deploy_set(name="your deploy set name",          # 部署集名称，非必填项
                                        strategy="HOST_HA",                   # 部署集策略，非必填项，默认值为HOST_HA，可选值为：
HOST_HA、RACK_HA、TOR_HA
                                        concurrency=5,                        # 部署集并发度，非必填项，默认值为1
                                        desc="your deploy set description")   # 部署集描述，非必填项

    print(response)
```

查询部署集列表

通过以下代码可以查询所有部署集实例的列表及详情信息：

```
 #### !/usr/bin/env python
#### coding=utf-8

from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 请求查询部署集列表，返回账号全量部署集信息
    response = client.list_deploy_sets()

    print(response)
```

修改部署集属性

使用以下代码可以修改指定部署集的指定属性值：

```
 #### !/usr/bin/env python
#### coding=utf-8

from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    response = client.modify_deploy_set(deploy_set_id='dset-******',       # 部署集ID, 必填项
                                        name="modify-new-name",            # 部署集名称，非必填项
                                        desc="modify new description")     # 部署集描述，非必填项

    print(response)
```

删除指定的部署集

使用以下代码删除用户自己的指定的部署集：

```
 #### !/usr/bin/env python
#### coding=utf-8
from baidubce.services.bcc import bcc_client
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration


if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    # 设置默认配置
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    # 创建bcc client
    client = bcc_client.BccClient(config)

    # 请求删除部署集
    response = client.delete_deploy_set(deploy_set_id='dset-*******')     # 部署集ID，必填项

    print(response)
```

绑定指定的部署集

使用以下代码绑用户自己指定的部署集：

```
def test_update_instance_deploy(self):
        """
        test case for update_instance_deploy
        """
        resp = self.client.update_instance_deploy(instance_id='iid1', deployset_id_list=['did1', 'did2'], force=True)
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

解绑指定的部署集

使用以下代码解绑用户自己指定的部署集：

```
def test_del_instance_deploy(self):
        """
        test case for del_instance_deploy
        """
        resp = self.client.del_instance_deploy(instance_id_list=['iid1', 'iid2'], deploy_set_id='dsid')
        self.assertEqual(
            type(resp),
            baidubce.bce_response.BceResponse)
        if resp is not None and resp.content is not None:
            print(json.loads(resp.content.decode('utf-8')))
        else:
            print(resp)
```

预留实例券

创建预留实例券

使用以下代码创建预留实例券

```
 from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)
    bcc_client = BccClient(bcc_sample_conf.config)

    response = bcc_client.create_reserved_instances(reserved_instance_name="test_create_reserved_instances",
                                                    scope="AZ",
                                                    zone_name="cn-bj-a",
                                                    spec="bcc.g5.c2m8",
                                                    offering_type="FullyPrepay",
                                                    instance_count=1,
                                                    reserved_instance_count=1,
                                                    reserved_instance_time=1,
                                                    reserved_instance_time_unit="month",
                                                    auto_renew_time_unit="month",
                                                    auto_renew_time=1,
                                                    auto_renew=True,
                                                    tags=[bcc_model.TagModel("k1", "v1")])

    # 打印返回结果
    print(response)
```

调整预留实例券

使用以下代码调整预留实例券

```
 from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)
    bcc_client = BccClient(bcc_sample_conf.config)

    reserved_instance = bcc_model.ModifyReservedInstanceModel(reservedInstanceId='r-UBVQFB5b',
                                                              reservedInstanceName='rename-reservedInstanceName')

    reserved_instances_list = [reserved_instance]
    response = bcc_client.modify_reserved_instances(reserved_instances=reserved_instances_list)

    # 打印返回结果
    print(response)
```

绑定标签

使用以下代码给指定预留实例券绑定标签

```
 from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)
    bcc_client = BccClient(bcc_sample_conf.config)

    # 预留实例券ID，最多支持100个
    reserved_instance_ids = ['r-Qyycx1SX']
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey02',
                                       tagValue='TestValue02')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey03',
                                       tagValue='TestValue03')
    # 待绑定tag列表
    instance_tags = [instance_tag1, instance_tag2]
    bcc_client.bind_reserved_instance_to_tags(reserved_instance_ids=reserved_instance_ids, tags=instance_tags)
```

解绑标签

使用以下代码从指定预留实例券中解绑标签

```
 from baidubce.services.bcc import bcc_model
from baidubce.services.bcc.bcc_client import BccClient
from sample.bcc import bcc_sample_conf

if __name__ == "__main__":
    import logging

    logging.basicConfig(level=logging.DEBUG)
    __logger = logging.getLogger(__name__)

    bcc_client = BccClient(bcc_sample_conf.config)

    # 预留实例券ID，最多支持100个
    reserved_instance_ids = ['r-Qyycx1SX']
    instance_tag1 = bcc_model.TagModel(tagKey='TestKey02',
                                       tagValue='TestValue02')
    instance_tag2 = bcc_model.TagModel(tagKey='TestKey03',
                                       tagValue='TestValue03')
    # 待解绑tag列表
    instance_tags = [instance_tag1, instance_tag2]
    bcc_client.unbind_reserved_instance_from_tags(reserved_instance_ids=reserved_instance_ids, tags=instance_tags)
```

地域

获取域名列表接口

使用以下代码可以获取全部地域或者指定地域的BCC(或CDS、预留实例券)域名列表：

```
from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.bcc import bcc_client, bcc_model
if __name__ == '__main__':
    # 设置您的ak、sk和要访问的地域
    HOST = b'http://bcc.baidubce.com'
    AK = b'ak'
    SK = b'sk'
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    client = bcc_client.BccClient(config)
    response = client.describe_regions(
        region='', # 待查询的地域，为空时代表查询全部地域
    )
    print(response)
```

Go-SDK

概述

本文档主要介绍云服务器BCC Go SDK的安装和使用。在使用本文档前，您需要先了解云服务器BCC的一些基本知识，并已开通了云服务器BCC服务。若您还不了解云服务器BCC，可以参考产品描述和入门指南。

初始化

确认Endpoint

在确认您使用SDK时配置的Endpoint时，可以先阅读开发人员指南中关于BCC访问域名的部分，理解Endpoint相关的概念。百度云目前开放了多区域支持，请参考区域选择说明。

获取密钥

要使用百度云BCC，您需要拥有一个有效的AK(Access Key ID)和SK(Secret Access Key)用来进行签名认证。AK/SK是由系统分配给用户的，均为字符串，用于标识用户，为访问BCC做签名验证。

可以通过如下步骤获得并了解您的AK/SK信息：

注册百度云账号

创建AK/SK

新建BCC Client

BCC Client是BCC服务的客户端，为开发者与BCC服务进行交互提供了一系列的方法。

使用AK/SK新建BCC Client

通过AK/SK方式访问BCC，用户可以参考如下代码新建一个Bcc Client：

```
import (
"github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
// 用户的Access Key ID和Secret Access Key
AK, SK := <your-access-key-id>, <your-secret-access-key>

// 用户指定的Endpoint
ENDPOINT := <domain-name>

// 初始化一个BCCClient
bccClient, err := bcc.NewClient(AK, SK, ENDPOINT)
}
```

在上面代码中，AK对应控制台中的“Access Key ID”，SK对应控制台中的“Access Key Secret”，获取方式请参考《操作指南 管

理ACCESSKEY》。第三个参数ENDPOINT支持用户自己指定域名，如果设置为空字符串，会使用默认域名作为BCC的服务地址。

> **注意：**ENDPOINT参数需要用指定区域的域名来进行定义，如服务所在区域为北京，则为bcc.bj.baidubce.com。

使用STS创建BCC Client

申请STS token

BCC可以通过STS机制实现第三方的临时授权访问。STS（Security Token Service）是百度云提供的临时授权服务。通过STS，您可以为第三方用户颁发一个自定义时效和权限的访问凭证。第三方用户可以使用该访问凭证直接调用百度云的API或SDK访问百度云资源。

通过STS方式访问BCC，用户需要先通过STS的client申请一个认证字符串，申请方式可参见百度云STS使用介绍。

用STS token新建BCC Client

申请好STS后，可将STS Token配置到BCC Client中，从而实现通过STS Token创建BCC Client。

代码示例

GO SDK实现了STS服务的接口，用户可以参考如下完整代码，实现申请STS Token和创建BCC Client对象：

```
import (
"fmt"

"github.com/baidubce/bce-sdk-go/auth"        //导入认证模块
"github.com/baidubce/bce-sdk-go/services/bcc" //导入BCC服务模块
"github.com/baidubce/bce-sdk-go/services/sts" //导入STS服务模块
)

func main() {
// 创建STS服务的Client对象，Endpoint使用默认值
AK, SK := <your-access-key-id>, <your-secret-access-key>
stsClient, err := sts.NewClient(AK, SK)
if err != nil {
fmt.Println("create sts client object :", err)
return
}

// 获取临时认证token，有效期为60秒，ACL为空
stsObj, err := stsClient.GetSessionToken(60, "")
if err != nil {
fmt.Println("get session token failed:", err)
return
  }
fmt.Println("GetSessionToken result:")
fmt.Println("  accessKeyId:", stsObj.AccessKeyId)
fmt.Println("  secretAccessKey:", stsObj.SecretAccessKey)
fmt.Println("  sessionToken:", stsObj.SessionToken)
fmt.Println("  createTime:", stsObj.CreateTime)
fmt.Println("  expiration:", stsObj.Expiration)
fmt.Println("  userId:", stsObj.UserId)

// 使用申请的临时STS创建BCC服务的Client对象，Endpoint使用默认值
bccClient, err := bcc.NewClient(stsObj.AccessKeyId, stsObj.SecretAccessKey, "")
if err != nil {
fmt.Println("create bcc client failed:", err)
return
}
stsCredential, err := auth.NewSessionBceCredentials(
stsObj.AccessKeyId,
stsObj.SecretAccessKey,
stsObj.SessionToken)
if err != nil {
fmt.Println("create sts credential object failed:", err)
return
}
bccClient.Config.Credentials = stsCredential
}
```

> 注意： 目前使用STS配置BCC Client时，无论对应BCC服务的Endpoint在哪里，STS的Endpoint都需配置为http://sts.bj.baidubce.com。上述代码中创建STS对象时使用此默认值。

配置HTTPS协议访问BCC

BCC支持HTTPS传输协议，您可以通过在创建BCC Client对象时指定的Endpoint中指明HTTPS的方式，在BCC GO SDK中使用HTTPS访问BCC服务：

```
// import "github.com/baidubce/bce-sdk-go/services/bcc"

ENDPOINT := "https://bcc.bj.baidubce.com" //指明使用HTTPS协议
AK, SK := <your-access-key-id>, <your-secret-access-key>
bccClient, _ := bcc.NewClient(AK, SK, ENDPOINT)
```

配置BCC Client

如果用户需要配置BCC Client的一些细节的参数，可以在创建BCC Client对象之后，使用该对象的导出字段Config进行自定义配置，可以为客户端配置代理，最大连接数等参数。

使用代理

下面一段代码可以让客户端使用代理访问BCC服务：

```
// import "github.com/baidubce/bce-sdk-go/services/bcc"

//创建BCC Client对象
AK, SK := <your-access-key-id>, <your-secret-access-key>
ENDPOINT := "bcc.bj.baidubce.com"
client, _ := bcc.NewClient(AK, SK, ENDPOINT)

//代理使用本地的8080端口
client.Config.ProxyUrl = "127.0.0.1:8080"
```

设置网络参数

用户可以通过如下的示例代码进行网络参数的设置：

```
// import "github.com/baidubce/bce-sdk-go/services/bcc"

AK, SK := <your-access-key-id>, <your-secret-access-key>
ENDPOINT := "bcc.bj.baidubce.com"
client, _ := bcc.NewClient(AK, SK, ENDPOINT)

// 配置不进行重试，默认为Back Off重试
client.Config.Retry = bce.NewNoRetryPolicy()

// 配置连接超时时间为30秒
client.Config.ConnectionTimeoutInMillis = 30 * 1000
```

配置生成签名字符串选项

```
// import "github.com/baidubce/bce-sdk-go/services/bcc"

AK, SK := <your-access-key-id>, <your-secret-access-key>
ENDPOINT := "bcc.bj.baidubce.com"
client, _ := bcc.NewClient(AK, SK, ENDPOINT)

// 配置签名使用的HTTP请求头为`Host`
headersToSign := map[string]struct{}{"Host": struct{}{}}
client.Config.SignOption.HeadersToSign = HeadersToSign

// 配置签名的有效期为30秒
client.Config.SignOption.ExpireSeconds = 30
```

参数说明

用户使用GO SDK访问BCC时，创建的BCC Client对象的Config字段支持的所有参数如下表所示：

配置项名称	类型	含义
Endpoint	string	请求服务的域名
ProxyUrl	string	客户端请求的代理地址
Region	string	请求资源的区域
UserAgent	string	用户名称，HTTP请求的User-Agent头
Credentials	*auth.BceCredentials	请求的鉴权对象，分为普通AK/SK与STS两种
SignOption	*auth.SignOptions	认证字符串签名选项
Retry	RetryPolicy	连接重试策略
ConnectionTimeoutInMillis	int	连接超时时间，单位毫秒，默认20分钟

说明：

1. `Credentials`字段使用`auth.NewBceCredentials`与`auth.NewSessionBceCredentials`函数创建，默认使用前者，后者为使用STS鉴权时使用，详见“使用STS创建BCC Client”小节。
2. `SignOption`字段为生成签名字符串时的选项，详见下表说明：

名称	类型	含义
HeadersToSign	map[string]struct{}	生成签名字符串时使用的HTTP头
Timestamp	int64	生成的签名字符串中使用的时间戳，默认使用请求发送时的值
ExpireSeconds	int	签名字符串的有效期

其中，HeadersToSign默认为`Host`，`Content-Type`，`Content-Length`，`Content-MD5`；TimeStamp一般为零值，表示使用调用生成认证字符串时的时间戳，用户一般不应该明确指定该字段的值；ExpireSeconds默认为1800秒即30分钟。

3. `Retry`字段指定重试策略，目前支持两种：`NoRetryPolicy`和`BackOffRetryPolicy`。默认使用后者，该重试策略是指定最大重试次数、最长重试时间和重试基数，按照重试基数乘以2的指数级增长的方式进行重试，直到达到最大重试测试或者最长重试时间为止。

可用区

查询可用区列表

如下代码可以查询所有的可用区列表：

```
result, err := client.ListZone()
if err != nil {
    fmt.Println("list zone failed: ", err)
} else {
    fmt.Println("list zone success: ", result)
}
```

查询机型的可用区

使用以下代码可以查询指定机型支持的可用区列表：

```
 args := &api.ListTypeZonesArgs{
   // 选择实例类型，可以选择N1, N2, N3等
   InstanceType: "",
   // 产品类型, 可以选择Prepaid，Postpaid
   ProductType:  "",
   // 实例套餐规格，可以选择bcc.g3.c2m12等
   Spec:         "",
   // 实例套餐规格族,可以选择g3,ic4等
   SpecId:       "",
}
if res, err := BCC_CLIENT.ListTypeZones(args); err != nil {
   fmt.Println("Get the specific zone flavor failed: ", err)
} else {
   fmt.Println("Get the specific zone flavor success, result: ", res)
}
```

实例

创建实例（通过指定实例套餐规格）

使用以下代码可以创建BCC实例，包括普通型BCC、存储优化型BCC、计算优化型BCC、大数据机型BCC、GPU机型BCC、FPGA机型BCC：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 IsOpenHostnameDomain := true
 AutoSeqSuffix := true
 RelationTag := false
 IsOpenIpv6 := true
 CdsAutoRenew := true
 argsV2 := &api.CreateInstanceBySpecArgsV2{
  // 镜像ID
  ImageId: "m-***",
  // 实例规格
  Spec: "bcc.g5.c2m8",
  // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
  RootDiskSizeInGb: 20,
  // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
  RootDiskStorageType: api.StorageTypeEnhancedPl1,
  // 待创建的CDS磁盘列表
  CreateCdsList: []api.CreateCdsModel{
   {
    CdsSizeInGB: 50,
    StorageType: api.StorageTypeEnhancedPl1,
   },
  },
  // 公网带宽，单位为Mbps，必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
```

```
    // 公网带宽，单位为Mbps。范围为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
    NetWorkCapacityInMbps: 10,
    // 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
    EipName: "eipN***",
    // 公网带宽计费方式，可选参数详见internetChargeType，若不指定internetChargeType，默认付费方式同BCC，预付费默认为包年包月按带宽，后付费默认为按使用带宽计费。
    InternetChargeType: "BANDWIDTH_PREPAID",
    // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
    PurchaseCount: 1,
    // 虚拟机名字（可选）。默认都不指定name。如果指定name：批量时name作为名字的前缀。
    //，后缀生成方式：name{ -序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。
    //注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。
    //支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
    Name: "instance-***",
    // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
    Hostname: "hostn***",
    // 是否自动生成hostname domain（可选参数） 是:true 否:false
    IsOpenHostnameDomain: &IsOpenHostnameDomain,
    // 是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
    AutoSeqSuffix: &AutoSeqSuffix,
    // 待创建实例是否开启ipv6，只有当镜像和子网都支持ipv6时才可开启，true表示开启，false表示关闭，不传表示自动适配镜像和子网的ipv6支持情况
    IsOpenIpv6: &IsOpenIpv6,
    // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
    AdminPass: "***",
    // 订单、计费相关参数
    Billing: api.Billing{
     PaymentTiming: api.PaymentTimingPrePaid,
     Reservation: &api.Reservation{
      ReservationLength:   1,
      ReservationTimeUnit: "Month",
     },
    },
    // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
    ZoneName: "cn-bj-a",
    // 指定subnet信息，为空时将使用默认子网
    SubnetId: "sbn-***",
    // 指定securityGroupIds信息，为空时将使用默认安全组
    SecurityGroupIds: []string{"g-***"},
    // 待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
    RelationTag: &RelationTag,
    // cds是否自动续费 是:true 否:false
    CdsAutoRenew: &CdsAutoRenew,
    // 自动快照策略ID
    AspId: "asp-***",
    // 指定实例所在的部署集id列表
    DeployIdList: []string{"dset-***"},
        // 指定弹性高性能计算集群id, 仅支持RDMA网络的套餐
        EhcClusterId: "ehc-***"
  }

  createResult, err := bccClient.CreateInstanceBySpecV2(argsV2)
  fmt.Println(err)
  fmt.Println(createResult)
}
```

提示：

1. 创建BCC请求是一个异步请求，返回200表明订单生成，后续可以通过查询返回的实例id信息了解BCC虚机的创建进度。
2. 本接口用于创建一个或多个同配虚拟机实例。
3. 创建实例需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。
4. 创建计费方式为后付费的实例需要账户现金余额+通用代金券大于100；预付费方式的实例则需要账户现金余额大于等于实例费用。
5. 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
6. 缺省情形下，一个实例最多只能挂载5个云磁盘。
7. 创建CDS磁盘和临时数据盘时，磁盘容量大小限制为5的倍数。
8. 创建实例支持创建和添加临时数据盘，但不支持单独创建或添加临时数据盘。
9. 临时数据盘不支持挂载、卸载、删除。
10. 普通实例的临时数据盘最大不能超过500G。
11. 指定子网和安全组创建，要求子网和安全组必须同时指定或同时不指定，同时指定的子网和安全组必须同属于一个VPC，都不指定会使用默认子网和默认安全组。
12. 指定公网IP带宽创建，计费方式为按照带宽计费。
13. 创建接口为异步创建，可通过查询实例详情接口查询实例状态
14. 每个实例最多只能购买一块临时数据盘。
15. 实例的临时数据盘默认只有hp1类型。
16. 创建存储优化型实例必须购买临时数据盘，通过ephemeralDisks指定临时盘数据盘大小，默认nvme类型数据盘，无需指定。

创建竞价实例

使用以下代码可以创建BCC实例：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.CreateInstanceArgs{
  // 镜像ID
  ImageId: "m-***",
  // 待创建虚拟机实例的类型
  InstanceType:       "N6",
```

```
  CpuCount:          1,
  MemoryCapacityInGB: 4,
  BidModel:          "market",
  // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
  RootDiskSizeInGb: 20,
  // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
  RootDiskStorageType: api.StorageTypeEnhancedPl1,
  // 待创建的CDS磁盘列表
  CreateCdsList: []api.CreateCdsModel{
   {
    CdsSizeInGB: 50,
    StorageType: api.StorageTypeEnhancedPl1,
   },
  },
  // 公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
  NetWorkCapacityInMbps: 10,
  // 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
  EipName: "eipN***",
  // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
  PurchaseCount: 1,
  // 虚拟机名字（可选）。默认都不指定name。如果指定name：批量时name作为名字的前缀。
  //，后缀生成方式：name{ -序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。
  //注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。
  //支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
  Name: "instance-***",
  // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{ -序号}。
  Hostname: "hostn***",
  // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
  AdminPass: "***",
  // 订单、计费相关参数
  Billing: api.Billing{
   PaymentTiming: api.PaymentTimingPrePaid,
   Reservation: &api.Reservation{
    ReservationLength:   1,
    ReservationTimeUnit: "Month",
   },
  },
  // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
  ZoneName: "cn-bj-a",
  // 指定subnet信息，为空时将使用默认子网
  SubnetId: "sbn-***",
  // 指定securityGroupIds信息，为空时将使用默认安全组
  SecurityGroupIds: []string{"g-***"},
  // 自动快照策略ID
  AspId: "asp-***",
  // 指定实例所在的部署集id列表
  DeployIdList: []string{"dset-***"},
 }

 createResult, err := bccClient.CreateBidInstance(args)
 fmt.Println(err)
 fmt.Println(createResult)
}
```

创建实例（接口已废弃，建议使用“创建实例（通过指定实例套餐规格）接口创建）

使用以下代码可以创建BCC实例，包括专属实例、普通型 I 型实例、普通型 II 型实例、存储优化型BCC、计算优化型BCC

```
package main
```

```
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 IsOpenHostnameDomain := true
 AutoSeqSuffix := true
 RelationTag := false
 IsOpenIpv6 := true
 CdsAutoRenew := true
 argsV2 := &api.CreateInstanceArgsV2{
  // 镜像ID
  ImageId: "m-***",
  // 待创建虚拟机实例的类型
  InstanceType: "N6",
  // 实例的CPU核数
  CpuCount: 1,
  // 实例的内存大小，单位为GB
  MemoryCapacityInGB: 4,
  // 待创建虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 2048]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
  RootDiskSizeInGb: 20,
  // 待创建虚拟机实例系统盘介质，默认使用SSD型云磁盘
  RootDiskStorageType: api.StorageTypeEnhancedPl1,
  // 待创建的CDS磁盘列表
  CreateCdsList: []api.CreateCdsModel{
   {
    CdsSizeInGB: 50,
    StorageType: api.StorageTypeEnhancedPl1,
   },
  },
  // 公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps
  NetWorkCapacityInMbps: 10,
  // 长度1～65个字节，字母开头，可包含字母数字-_/.字符。若不传该参数，服务会自动生成name。
  EipName: "eipN***",
  // 公网带宽计费方式，可选参数详见internetChargeType，若不指定internetChargeType，默认付费方式同BCC，预付费默认为包年包月按带宽，后付费默认为按使用带宽计费。
  InternetChargeType: "BANDWIDTH_PREPAID",
  // 批量创建（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
  PurchaseCount: 1,
  // 虚拟机名字（可选）。默认都不指定name。如果指定name：批量时name作为名字的前缀。
  //，后缀生成方式：name{-序号}。如果没有指定name，则自动生成，方式：{instance-八位随机串-序号}。
  //注：随机串从0~9a~z生成；序号按照count的数量级，依次递增，如果count为100，则序号从000~100递增，如果为10，则从00~10递增。
  //支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
  Name: "instance-***",
  // 虚拟机主机名。默认都不指定hostname。如果指定hostname：批量时hostname作为名字的前缀。后端将加上后缀，后缀生成方式：name{-序号}。
  Hostname: "hostn***",
  // 是否自动生成hostname domain（可选参数） 是:true 否:false
  IsOpenHostnameDomain: &IsOpenHostnameDomain,
  // 是否自动生成name和hostname有序后缀（可选参数） 是:true 否:false
```

```
    AutoSeqSuffix: &AutoSeqSuffix,
    // 待创建实例是否开启ipv6，只有当镜像和子网都支持ipv6时才可开启，true表示开启，false表示关闭，不传表示自动适配镜像和子网的ipv6支持情况
    IsOpenIpv6: &IsOpenIpv6,
    // 待创建的实例管理员密码，8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()。
    AdminPass: "***",
    // 订单、计费相关参数
    Billing: api.Billing{
     PaymentTiming: api.PaymentTimingPrePaid,
     Reservation: &api.Reservation{
      ReservationLength:   1,
      ReservationTimeUnit: "Month",
     },
    },
    // 指定zone信息，默认为空，由系统自动选择，可通过调用查询可用区列表接口查询可用区列表。zoneName命名规范是“国家-region-可用区序列"，小写，例如北京可用区A为"cn-bj-a"。
    ZoneName: "cn-bj-a",
    // 指定subnet信息，为空时将使用默认子网
    SubnetId: "sbn-***",
    // 指定securityGroupIds信息，为空时将使用默认安全组
    SecurityGroupIds: []string{"g-***"},
    // 待创建实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在
    RelationTag: &RelationTag,
    // cds是否自动续费 是:true 否:false
    CdsAutoRenew: &CdsAutoRenew,
    // 自动快照策略ID
    AspId: "asp-***",
    // 指定实例所在的部署集id列表
    DeployIdList: []string{"dset-***"},
   }

   createResult, err := bccClient.CreateInstanceV2(argsV2)
   fmt.Println(err)
   fmt.Println(createResult)
}
```

提示：

1. 创建BCC请求是一个异步请求，返回200表明订单生成，后续可以通过查询返回的实例id信息了解BCC虚机的创建进度。
2. 本接口用于创建一个或多个同配虚拟机实例。
3. 创建实例需要实名认证，没有通过实名认证的可以前往百度开放云官网控制台中的安全认证下的实名认证中进行认证。
4. 创建计费方式为后付费的实例需要账户现金余额+通用代金券大于100；预付费方式的实例则需要账户现金余额大于等于实例费用。
5. 支持批量创建，且如果创建过程中有一个实例创建失败，所有实例将全部回滚，均创建失败，如果创建时包含CDS，CDS也会回滚。
6. 缺省情形下，一个实例最多只能挂载5个云磁盘。
7. 创建CDS磁盘和临时数据盘时，磁盘容量大小限制为5的倍数。
8. 创建实例支持创建和添加临时数据盘，但不支持单独创建或添加临时数据盘。
9. 临时数据盘不支持挂载、卸载、删除。
10. 普通实例的临时数据盘最大不能超过500G。
11. 指定子网和安全组创建，要求子网和安全组必须同时指定或同时不指定，同时指定的子网和安全组必须同属于一个

VPC，都不指定会使用默认子网和默认安全组。

12. 指定公网IP带宽创建，计费方式为按照带宽计费。
13. 创建接口为异步创建，可通过查询实例详情接口查询实例状态
14. 可通过该接口指定专属服务器创建实例，专属实例不参与计费。专属实例只能通过ephemeralDisks创建临时盘并指定磁盘类型。
15. 每个实例最多只能购买一块临时数据盘。
16. 实例的临时数据盘默认只有hp1类型。
17. 通过instanceType字段指定需要创建的虚机类型，目前API支持创建的虚机类型参见下述InstanceType。参数(instanceType，cpuCount，memoryCapacityInGB)可以确定需要的机型以及配置。
18. 创建存储优化型实例必须购买临时数据盘，通过ephemeralDisks指定临时盘数据盘大小，默认nvme类型数据盘，无需指定。
19. 创建请求详细使用请参考BCC API 文档创建实例
20. 创建FPGA BCC虚机需要使用指定的(CPU、内存、本地数据盘、FPGA卡类型以及专用镜像), 详细请参考BCC API 文档FPGA型BCC可选规格配置
21. 创建GPU BCC虚机需要使用指定的(CPU、内存、本地数据盘、GPU卡类型), 详细请参考BCC API 文档GPU型BCC可选规格配置

取消竞价实例订单

通过以下代码可以取消竞价实例订单

```
 cancelBidOrderRequest := &CancelBidOrderRequest{
   // 订单ID
   OrderId     string "orderId"
}
if err := bccClient.CancelBidOrder(cancelBidOrderRequest); err != nil {
   fmt.Println("CancelBidOrderRequest failed: ", err)
} else {
   fmt.Println("CancelBidOrderRequest success.")
}
```

查询竞价实例套餐

通过以下代码可以查询竞价实例套餐

```
 if res, err := bccClient.ListBidFlavor(); err != nil {
   fmt.Println("List bidding instance flavors failed: ", err)
} else {
   fmt.Println("List bidding instance flavors success, result: ", res)
}
```

查询套餐列表

使用以下代码可以查询套餐列表

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    //创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.ListFlavorSpecArgs{
  Specs: "bcc.g5.c2m8",
  ZoneName: "cn-bj-b",
 }

    if res, err := bccClient.ListFlavorSpec(args); err != nil {
        fmt.Println("Get flavor list failed: ", err)
    } else {
        fmt.Println("Get flavor list success, result: ", res)
    }
}
```

查询竞价实例市场价

通过以下代码可以查询竞价实例市场价

```
 createCdsList := []api.CreateCdsModel{{
    // 设置CDS磁盘容量，必须为大于0的整数，单位为GB，大小为0~5120G，可选参数
    CdsSizeInGB: 40,
    // 设置CDS磁盘存储类别，默认为高性能型，可选参数
    StorageType: api.StorageTypeHP1,
    // 设置快照ID
    SnapshotId:  "your-snapshot-id",
    // 设置加密密钥，true或false
    EncryptKey:  true,
}}
tagList := []model.TagModel{{
    // 设置要查询的tagKey
    TagKey:   "your-tag-key",
    // 设置要查询的tagValue
    TagValue: "your-tag-value",
}}
args := &api.GetBidInstancePriceArgs{
    // 使用 uuid 生成一个长度不超过64位的ASCII字符串
    ClientToken:          "random-uuid",
     // 设置待查询的虚拟机实例类型，具体可选类型参见InstanceType
    InstanceType:         "your-choose-instance-type",
    // 设置待查询虚拟机实例的CPU核数
    CpuCount:             1,
    // 设置待查询虚拟机实例的内存容量
    MemoryCapacityInGB:   2,
    // 设置待查询虚拟机实例的系统盘大小，单位GB，默认是40GB，范围为[40, 100]GB，超过40GB按照云磁盘价格收费。注意指定的系统盘大小需要满足所使用镜像最小磁盘空间限制。
    RootDiskSizeInGb:     45,
    // 设置待查询虚拟机实例系统盘介质，默认使用SSD型云磁盘，可指定系统盘磁盘类型可参见StorageType
    RootDiskStorageType:  api.StorageTypeCloudHP1,
```

```
    // 设置待查询的CDS磁盘列表
    CreateCdsList:          createCdsList,
    // 设置批量查询（购买）的虚拟机实例个数，必须为大于0的整数，可选参数，缺省为1
    PurchaseCount:          1,
    // 设置虚拟机名字，可选参数
    Name:                   "your-choose-instance-name",
    // 设置实例管理员密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*()),可选参数
    AdminPass:              "your-admin-pass",
    // 设置待查询实例所要绑定的密钥对ID,可选参数
    KeypairId:              "your-keypair-id",
    // 设置自动快照策略ID,可选参数
    AspId:                  "your-asp-id",
    // 设置待查询虚拟机实例的镜像ID,可选参数
    ImageId:                "your-image-id",
    // 设置竞价实例出价模型， 市场价: "market" 自定义："custom",可选参数
    BidModel:               "your-bid-model",
    // 设置竞价实例出价金额，若是自定义出价，且出价金额小于市场价，则不允许创建。当bidModel='custom'时才有效,
可选参数
    BidPrice:               "your-bid-price",
    // 设置公网带宽，单位为Mbps。必须为0~200之间的整数，为0表示不分配公网IP，默认为0Mbps,可选参数
    NetWorkCapacityInMbps: 20,
    // 设置待查询实例指定的标签是否需要和已有标签键进行关联，默认为false。注意值为true时要保证该标签键已存在,
可选参数
    RelationTag:            false,
    // 设置待查询的标签列表,可选参数
    Tags:                   tagList,
    // 设置securityGroup信息，为空时将使用默认安全组,可选参数
    SecurityGroupId:        "your-security-group-id",
    // 设置subnet信息，为空时将使用默认子网,可选参数
    SubnetId:               "your-subnet-id",
    // 设置指定zone信息，默认为空，由系统自动选择,可选参数
    ZoneName:               "your-zone-name",
    // 设置公网带宽计费方式,可选参数
    InternetChargeType:     "your-internet-charge-type",
}
if res, err := bccClient.GetBidInstancePrice(args); err != nil {
    fmt.Println("Get bidding instance price failed: ", err)
} else {
    fmt.Println("Get bidding instance price success, result: ", res)
}
```

查询实例套餐库存

查询实例资源套餐规格对应的库存。

```
 // 实例类型
instanceType := "instanceType"
// CPU核数
cpuCount := cpuCount
// 内存容量（GB）
memoryCapacityInGB := memoryCapacityInGB
// 可用区名称
zoneName := "zoneNamen"
// GPU卡类型，GPU和VGPU可填
gpuCard := "gpuCard"
// GPU卡数量，GPU和VGPU可填
cardCount := "cardCount"
//本地盘信息
ephemeralDisks := []EphemeralDisks{{
    "storageType": "ssd",
    "sizeInGB": sizeInGB,
}}

args := &api.CreateInstanceStockArgs{
    InstanceType:     instanceType,
    CpuCount: cpuCount,
    MemoryCapacityInGB:  memoryCapacityInGB,
    ZoneName:  zoneName,
    GpuCard:  gpuCard,
    CardCount:  cardCount,
    EphemeralDisks:  ephemeralDisks,
}
if res, err := bccClient.GetInstanceCreateStock(args); err != nil {
    fmt.Println("GetInstanceCreateStock failed: ", err)
} else {
    fmt.Println("GetInstanceCreateStock success: ", res)
}
```

实例扩缩容库存查询

实例变配余量查询

```
 // 实例id
instanceId := "instanceId"
// CPU核数
cpuCount := cpuCount
// 内存容量（GB）
memoryCapacityInGB := memoryCapacityInGB

args := &api.CreateInstanceStockArgs{
    InstanceId:     instanceId,
    CpuCount: cpuCount,
    MemoryCapacityInGB:  memoryCapacityInGB,
}
if res, err := bccClient.ResizeInstanceStockArgs(args); err != nil {
    fmt.Println("ResizeInstanceStockArgs failed: ", err)
} else {
    fmt.Println("ResizeInstanceStockArgs success: ", res)
}
```

查询部署集详情

通过以下代码利用部署集ID查询部署集详情

```
 // 设置你要查询的deploySetID
deploySetID := "your-choose-deploy-set-id"
if res, err := bccClient.GetDeploySet(deploySetID); err != nil {
    fmt.Println("Delete deploy set failed: ", err)
} else {
    fmt.Println("Get deploy set success: ", res)
}
```

删除实例与部署集之间的关系

以下代码可以删除实例与部署集之间的关系

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 queryArgs := &api.DelInstanceDeployArgs{
  DeploySetId: "dset-***",  // 部署集ID
  InstanceIds: []string{"i-***", "i-***"}, // 实例ID列表
 }
 res, err := bccClient.DelInstanceDeploySet(queryArgs)
 fmt.Println(err)
 fmt.Println(res)
}
```

查询可关机不计费的实例列表

查询可关机不计费的BCC实例列表：

```
 listInstanceArgs := &ListInstanceArgs{
   // 批量获取列表的查询起始位置，是一个由系统产生的字符串
   Marker          string
 // 设置返回数据大小，缺省为1000
   MaxKeys         int
 // 通过internal Ip过滤
   InternalIp      string
 // 通过DedicatedHostId过滤
   DedicatedHostId string
 // 通过ZoneName过滤
   ZoneName        string
 // 通过KeypairId过滤
   KeypairId       string
}

if res, err := bccClient.GetInstanceNoChargeList(listInstanceArgs); err != nil {
    fmt.Println("GetInstanceNoChargeList failed: ", err)
} else {
    fmt.Println("GetInstanceNoChargeList success, result: ", res)
}
```

查询实例列表

以下代码可以查询BCC虚机实例列表,支持通过内网ip、专属服务器id、可用区名称进行筛选

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 listArgs := &api.ListInstanceArgs{
  // 批量获取列表的查询的起始位置，是一个由系统生成的字符串
  Marker:          "i-***",
  // 每页包含的最大数量，最大数量通常不超过1000，缺省值为1000
  MaxKeys:         1,
  // 内网IP
  InternalIp:      "192.168.***.***",
  // 专属服务器ID
  DedicatedHostId:  "d-***",
  // 可用区信息
  ZoneName:        "cn-bj-d",
  // 多个实例ID，英文逗号分割，最多支持100个
  InstanceIds:     "i-***",
  // 多个实例名称，英文逗号分割，最多支持100个
  InstanceNames:   "instance-***",
  // 多个cds磁盘ID，英文逗号分割，最多支持100个
  CdsIds:          "v-***",
  // 多个部署集ID，英文逗号分割，最多支持100个
  DeploySetIds:    "dset-***",
  // 多个安全组ID，英文逗号分割，最多支持100个
  SecurityGroupIds: "g-***",
  // 支付方式（Prepaid / Postpaid）
  PaymentTiming:   "Postpaid",
  // 实例状态（Recycled / Running / Stopped / Stopping / Starting）
  Status:          "Running",
  // 多个标签，逗号分割，格式：tagKey:tagValue 或 tagKey
  Tags:            "***",
  // vpcId，只能与privateIps查询参数组合使用
  VpcId:           "vpc-***",
  // 多个内网IP，英文逗号分隔，最多支持100个，必须和vpcId组合使用
  PrivateIps:      "192.168.***.***",
 }
 res, err := bccClient.ListInstances(listArgs)
 fmt.Println(err)
 fmt.Println(res)
}
```

查询回收站实例列表

以下代码可以查询回收站中的BCC虚机实例列表,支持通过虚机id，名字进行筛选

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 批量获取列表的查询的起始位置，是一个由系统生成的字符串,可选参数
    marker := "marker***"
    // 每页包含的最大数量，最大数量通常不超过1000。缺省值为1000,可选参数
    maxKeys := 100
    // 设置想要查询的付费类型,可选参数 Prepaid表示预付费，Postpaid表示后付费，不传表示都选
    paymentTiming := "Postpaid"
    // 设置想要查询的虚机id,可选参数
    instanceId := "i-***"
    // 设置想要查询的虚机名称,可选参数
    name := "instance***"
    // 设置想要查询虚机的回收开始时间(北京时间),可选参数 （闭区间）
    recycleBegin := "2023-11-29T09:12:35Z"
    // 设置想要查询虚机的回收结束时间(北京时间),可选参数 （开区间）
    recycleEnd := "2023-11-30T09:12:35Z"
    args := &api.ListRecycleInstanceArgs{
        Marker:        marker,
        MaxKeys:       maxKeys,
        PaymentTiming: paymentTiming,
        InstanceId:    instanceId,
        Name:          name,
        RecycleBegin:  recycleBegin,
        RecycleEnd:    recycleEnd,
    }
    result, err := bccClient.ListRecycleInstances(args)
    if err != nil {
        fmt.Println("list instance failed:", err)
    } else {
        fmt.Println("list instance success: ", result)
    }
}
```

查询指定实例详情

使用以下代码可以查询指定BCC虚机的详细信息

```
// 设置你要操作的instanceId
instanceId := "your-choose-instance-id"

result, err := client.GetInstanceDetail(instanceId)
if err != nil {
    fmt.Println("get instance detail failed:", err)
} else
    fmt.Println("get instance detail success ", result)
}
```

启动实例

如下代码可以启动一个实例

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // 设置你要操作的instanceId
    instanceId := "i-***"
    err := bccClient.StartInstance(instanceId)
    if err != nil {
        fmt.Println("start instance failed:", err)
    } else {
        fmt.Println("start instance success")
    }
}
```

> **提示：**
>
> - 实例状态必须为Stopped，调用此接口才能成功返回，否则返回409错误
> - 该接口调用后，实例会进入Starting状态

停止实例

如下代码可以停止一个实例

```
 // 以下代码可以强制停止一个实例
err := client.StopInstance(instanceId, true)
if err != nil {
    fmt.Println("stop instance failed:", err)
} else {
    fmt.Println("stop instance success")
}
```

停止实例(支持强制停止&关机不计费)

使用以下代码停止实例:

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 err := bccClient.StopInstanceWithNoCharge(
  // 待停止的实例ID
  "i-***",
  // 指定实例是否强制关机，可选值：True、False，缺省为false
  true,
  // 指定实例是否关机不计费，可选值：true、false，缺省为false
  true)
 fmt.Println(err)
}
```

提示：

- 系统后台会在实例实际 Stop 成功后进入“已停止”状态。
- 只有状态为 Running 的实例才可以进行此操作，否则提示 409 错误。
- 实例支持强制停止，强制停止等同于断电处理，可能丢失实例操作系统中未写入磁盘的数据。

重启实例

如下代码可以重启实例

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // 设置你要操作的instanceId
    instanceId := "i-***"
    err := bccClient.RebootInstance(instanceId, true)
    if err != nil {
        fmt.Println("reboot instance failed:", err)
    } else {
        fmt.Println("reboot instance success")
    }
}
```

> **提示：**
>
> - 只有状态为 Running 的实例才可以进行此操作，否则提示 409 错误。
> - 接口调用成功后实例进入 Starting 状态。
> - 支持强制重启，强制重启等同于传统服务器的断电重启，可能丢失实例操作系统中未写入磁盘的数据。

修改实例密码

如下代码可以修改实例密码

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
// 实例ID
    instanceId := "i-***"
    args := &api.ChangeInstancePassArgs{
// 实例管理员密码
        AdminPass: "***",
    }
    err := bccClient.ChangeInstancePass(instanceId, args)
    if err != nil {
        fmt.Println("change instance password failed:", err)
    } else {
        fmt.Println("change instance password success")
    }
}
```

> **提示：** 只有 Running 和 Stopped 状态的实例才可以用调用接口，否则提示 409 错误。

修改实例属性

如下代码可以修改实例属性

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // 设置你要操作的instanceId
    instanceId := "i-***"
    args := &api.ModifyInstanceAttributeArgs{
        // 修改instance的名称
        Name: "Instance***",
    }
    err := bccClient.ModifyInstanceAttribute(instanceId, args)
    if err != nil {
        fmt.Println("modify instance failed:", err)
    } else {
        fmt.Println("modify instance success")
    }
}
```

> **提示：**
>
> - 目前该接口仅支持修改实例名称或网卡队列数

修改实例描述

如下代码可以修改实例描述

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // 设置你要操作的instanceId
    instanceId := "i-***"
    modifyArgs := &api.ModifyInstanceDescArgs{
        // 修改描述
        Description: "test modify",
    }
    err := bccClient.ModifyInstanceDesc(instanceId, modifyArgs)
    if err != nil {
        fmt.Println("modify instance failed:", err)
    } else {
        fmt.Println("modify instance success")
    }
}
```

修改实例主机名

如下代码可以修改实例主机名

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置你要操作的instanceId
    instanceId := "i-***"
    args := &api.ModifyInstanceHostnameArgs{
        // 设置想要修改的新主机名
        Hostname: "new-hostname***",
        // 设置是否开启domain，可选参数 true表示开启 false和null 表示关闭
        IsOpenHostnameDomain: true,
        // 设置是否自动重启，可选参数 true表示重启，false和null表示不重启
        Reboot: true,
    }
    err := client.ModifyInstanceHostname(instanceId, args)
    if err != nil {
        fmt.Println("modify instance hostname failed:", err)
    } else {
        fmt.Println("modify instance hostname success")
    }
}
```

重装实例

如下代码可以重装实例

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 IsOpenHostEye := true
 IsPreserveData := true
 args := &api.RebuildInstanceArgsV2{
  // 指定的镜像ID
  ImageId:        "m-***",
  // 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())，密码需要加密传输
  AdminPass:      "***",
  // 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
  KeypairId:      "k-***",
  // 是否开启主机安全，默认开启
  IsOpenHostEye:  &IsOpenHostEye,
  // 是否保留数据重装，默认保留
  IsPreserveData: &IsPreserveData,
 }
 // i-*** 为指定的实例ID
 err := bccClient.RebuildInstanceV2("i-***", args)
 fmt.Println(err)
}
```

重装实例（批量）

使用以下代码重装实例:

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 InstanceIds := []string{"i-***", "i-***"}
 IsOpenHostEye := true
 IsPreserveData := true
 args := &api.RebuildBatchInstanceArgsV2{
  // 指定的实例ID列表，最多100个
  InstanceIds:    InstanceIds,
  // 指定的镜像ID
  ImageId:        "m-***",
  // 机器密码(8-16位字符，英文，数字和符号必须同时存在，符号仅限!@#$%^*())，密码需要加密传输
  AdminPass:      "***",
  // 待重装实例所要绑定的密钥对ID，必须传递adminPass、keypairId其中一个参数
  KeypairId:      "k-***",
  // 是否开启主机安全，默认开启
  IsOpenHostEye:  &IsOpenHostEye,
  // 是否保留数据重装，默认保留
  IsPreserveData: &IsPreserveData,
 }
 err := bccClient.BatchRebuildInstancesV2(args)
 fmt.Println(err)
}
```

> **提示：**
>
> - 实例重装后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

释放实例

如下代码可以释放实例

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 err := bccClient.DeleteInstance("i-cNpT****") // 待删除虚机ID
 fmt.Println(err)
}
```

释放实例（POST）

使用以下代码释放单个后付费云服务器实例:

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.DeleteInstanceWithRelateResourceArgs{
  // 是否关联释放当前时刻，实例挂载的EIP和数据盘
  RelatedReleaseFlag:    true,
  // 是否释放云磁盘快照
  DeleteCdsSnapshotFlag: true,
  // 实例释放时是否删除关联的ENI
  DeleteRelatedEnisFlag: true,
  // 实例释放时是否进入回收站
  BccRecycleFlag:        true,
 }
 // i-*** 为待释放实例的ID
 err := bccClient.DeleteInstanceWithRelateResource("i-***", args)
 fmt.Println(err)
}
```

批量释放实例（POST）

使用以下代码批量释放实例:

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    deleteInstanceWithRelateResourceArgs := &api.BatchDeleteInstanceWithRelateResourceArgs{
        // 释放的时候是否关联释放当前时刻，实例挂载的eip+数据盘
        // (只有该字段为true时 deleteCdsSnapshotFlag字段才会有效，若该字段为false,deleteCdsSnapshotFlag字段的值无效)
        RelatedReleaseFlag: true,
        // 设置是否释放云磁盘快照
        DeleteCdsSnapshotFlag: true,
        // 设置是否进入回收站
        BccRecycleFlag: true,
        // 设置是否释放弹性网卡
        DeleteRelatedEnisFlag: true,
        // 批量释放的实例id，最多100个
        InstanceIds: []string{"i-***"},
    }
    if err := bccClient.BatchDeleteInstanceWithRelateResource(deleteInstanceWithRelateResourceArgs); err != nil {
        fmt.Println("release instance failed: ", err)
    } else {
        fmt.Println("release instance success.")
    }
}
```

释放预付费实例（POST）

释放预付费云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。

释放的时候默认只释放云服务器实例和系统盘，用户可以选择是否关联释放云服务器实例挂载的EIP和数据盘，关联的EIP和数据盘需全部释放或全部不释放，不支持选择部分释放，部分不释放。

当选择关联释放EIP和数据盘，云磁盘快照可选择一同关联释放，若数据盘不关联释放，云磁盘快照也不支持关联释放。

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    deletePrepaidInstanceWithRelateResourceArgs := &api.DeletePrepaidInstanceWithRelateResourceArgs{
        // 批量释放的实例id
        InstanceId: "i-***",
        // 是否关联释放当前实例挂载的EIP和数据盘(只有该字段为true时 DeleteCdsSnapshotFlag字段才会有效，若该字段为false,DeleteCdsSnapshotFlag字段的值无效)
        RelatedReleaseFlag: true,
        // 设置是否释放云磁盘快照
        DeleteCdsSnapshotFlag: true,
        // 设置是否释放弹性网卡
        DeleteRelatedEnisFlag: true,
    }
    if err := bccClient.DeletePrepaidInstanceWithRelateResource(deletePrepaidInstanceWithRelateResourceArgs); err != nil {
        fmt.Println("release instance failed: ", err)
    } else {
        fmt.Println("release instance success.")
    }
}
```

释放回收站实例

回收站实例7天后自动释放，清理回收站资源，可以使用以下代码将其释放:

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 待操作的实例ID
    instanceId := "i-***"
    // 请求释放回收站中的某个实例
    err := bccClient.DeleteRecycledInstance(instanceId)
    fmt.Println(err)
}
```

> 提示：
>
> - 释放单个云服务器实例，释放后实例所使用的物理资源都被收回，相关数据全部丢失且不可恢复。
> - 只有付费类型为Postpaid或者付费类型为Prepaid且已过期的实例才可以释放。
> - 实例释放后，已挂载的CDS磁盘自动卸载，，基于此CDS磁盘的快照会保留。
> - 实例释放后，基于原系统盘的快照会自动删除，基于原系统盘的自定义镜像会保留。

定时释放 (限定后付费实例)

后付费实例定时释放，到达预设时间后自动释放bcc，自动释放时间可查询实例详情ReleaseTime。设定空字符串可以取消定时释放。请谨慎使用该功能，避免遗忘定时设置

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 err := bccClient.AutoReleaseInstance(
  // 实例ID
  "i-***",
  // 释放时间，格式yyyy-MM-dd'T'HH:mm:ss'Z'
  "2024-05-01T07:58:09Z")
 fmt.Println(err)
}
```

> 提示：
>
> - 只有付费类型为Postpaid的后付费实例允许设定自动释放
> - 本实例系统盘快照及实例快照都会被释放
> - 本实例已挂载CDS云磁盘都会被自动卸载，不会被释放
> - 实例释放后不可恢复
> - 关联的网卡资源会被自动卸载，且被释放

释放保护

使用以下代码可以为BCC实例设置释放保护，实例当前设置可查询实例详情DeletionProtection，默认0不保护，1释放保护中（创建和查询入口限v2版本使用）：

```
 args := &api.DeletionProtectionArgs {
// 释放保护状态 0不启用，1启用
    DeletionProtection : 0,
}
// 设置你要操作的instanceId
instanceId := "your-choose-instance-id"

if err := bccClient.ModifyDeletionProtection(instanceId, args); err != nil {
    fmt.Println("modifyDeletionProtection failed: ", err)
} else {
    fmt.Println("modifyDeletionProtection success.")
}
```

提示：

- 后付费和预付费均可开启释放保护
- 已开启释放保护的实例将无法通过控制台或API释放，只有在关闭的情况下才能被手动释放。定时释放，欠费释放以及实例过期释放不受释放保护属性的影响
- 实例释放保护默认不开启

变配实例

使用以下代码可以选择CPU,MemoryCapacityInGB,EphemeralDisks,EnableJumboFrame变配指定BCC实例:

```
 resizeInstanceArgs := &ResizeInstanceArgs{
    // BCC核数
    CpuCount: CpuCount
    // BCC的内存大小GB
    MemoryCapacityInGB: MemoryCapacityInGB
// 临时盘数据盘大小
    EphemeralDisks       []EphemeralDisk  "ephemeralDisks"
    // 是否开启Jumbo帧，true开启，false不开启。
// 默认为空，为空时：
// 变配到不支持Jumbo帧的规格时，不开启Jumbo帧；
// 从不支持变配到支持的规格时，不开启Jumbo帧；
// 从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
    EnableJumboFrame       *bool           enableJumboFrame,
}
// 设置你要操作的instanceId
instanceId := "your-choose-instance-id"

if err := bccClient.ResizeInstance(instanceId, resizeInstanceArgs); err != nil {
    fmt.Println("resize instance failed: ", err)
} else {
    fmt.Println("resize instance success.")
}
```

变配实例（通过实例套餐规格）

使用以下代码可以选择Spec,EnableJumboFrame变配指定BCC实例:

```
 resizeInstanceArgs := &ResizeInstanceArgs{
   // 实例套餐规格
   Spec              string        "spec"
   // 是否开启Jumbo帧，true开启，false不开启。
 // 默认为空，为空时：
 // 变配到不支持Jumbo帧的规格时，不开启Jumbo帧；
 // 从不支持变配到支持的规格时，不开启Jumbo帧；
 // 从支持到支持得规格是，若实例变配前开启Jumbo帧，变配后也开启Jumbo帧，若变配前不开启，则变配后也不开启。
   EnableJumboFrame   *bool          enableJumboFrame
}
// 设置你要操作的instanceId
instanceId := "your-choose-instance-id"

if err := bccClient.ResizeInstanceBySpec(instanceId, resizeInstanceArgs); err != nil {
   fmt.Println("resize instance failed: ", err)
} else {
   fmt.Println("resize instance success.")
}
```

绑定安全组

使用以下代码绑定安全组:

```
 // 设置你要操作的instanceId
instanceId := "your-choose-instance-id"
// 设置BCC绑定的安全组
SecurityGroupId := "SecurityGroupId"

if err := bccClient.BindSecurityGroup(instanceId, SecurityGroupId); err != nil {
   fmt.Println("Bind Security Group failed: ", err)
} else {
   fmt.Println("Bind Security Group success.")
}
```

解绑安全组

使用以下代码解绑安全组:

```
 // 设置你要操作的instanceId
instanceId := "your-choose-instance-id"
// 设置BCC解绑的安全组
SecurityGroupId := "SecurityGroupId"

if err := bccClient.UnBindSecurityGroup(instanceId, SecurityGroupId); err != nil {
   fmt.Println("UnBind Security Group failed: ", err)
} else {
   fmt.Println("UnBind Security Group success.")
}
```

> **提示：**
>
> - 每个实例至少关联一个安全组，默认关联默认安全组。
> - 如果实例仅属于一个安全组，尝试移出时，请求会报 403 错。

实例扩缩容

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 待变配的实例配置参数
 resizeArgs := &api.ResizeInstanceArgs{
  CpuCount:          4,    // CPU核数
  MemoryCapacityInGB: 16,   // 内存大小，单位GB
  LiveResize:        true, // 是否支持在线扩容，默认为false
 }
 err := bccClient.ResizeInstance("i-SrzD****", resizeArgs)
 fmt.Println(err)
}
```

提示：

- 实例计费方式为预付费时，不能进行缩容操作
- 实例计费方式为后付费时，可弹性扩缩容
- 只有实例Running或Stopped状态时可以进行扩缩容操作
- 实例扩缩容之后会重启一次，可选择热升级进行扩缩容，不影响业务中断，热升级的限制请参考文档热升级的限制
- 异步接口，可通过查询实例详情接口查看扩缩容状态是否完成
- 专属实例可以通过指定的cpu、内存以及临时盘大小，专属实例临时盘大小只支持扩容而不支持缩容，具体请参考API文档 实例扩缩容

查询实例VNC地址

如下代码可以查询实例的VNC地址

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // i-*** 为虚拟机实例ID
 res, err := bccClient.GetInstanceVNC("i-***")
 fmt.Println(err)
 fmt.Println(res)
}
```

> **提示：**
>
> - VNC地址一次使用后即失效
> - URL地址有效期为10分钟

实例续费

对BCC虚机的续费操作，可以延长过期时长，以下代码可以对实例及关联产品进行续费

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 虚拟机实例ID
    instanceId := "i-***"
 args := &api.PurchaseReservedArgs{
  // 设置实例关联续费标识，默认为空字符串
  RelatedRenewFlag: "CDS",
  // 订单详情
  Billing: api.Billing{
   // 续费方式，本接口针对预付费实例进行续费
   PaymentTiming: api.PaymentTimingPrePaid,
   // 续费时长和单位
   Reservation: &api.Reservation{
    // 续费时长
    ReservationLength:   1,
    // 续费单位为月
    ReservationTimeUnit: "month",
   },
  },
  // 幂等性Token，是一个长度不超过64位的ASCII字符串
  ClientToken: "random-uuid",
 }

 res, err := bccClient.InstancePurchaseReserved(instanceId, args)
 fmt.Println(err)
 fmt.Println(res)
}
```

注意：

关联续费产品(relatedRenewFlag)可选：

- CDS 只对BCC实例关联的预付费CDS进行续费
- EIP 只对BCC实例关联的预付费EIP进行续费
- MKT 只对BCC实例关联的预付费MKT进行续费
- CDS_EIP 只对BCC实例关联的预付费CDS、EIP进行续费
- CDS_MKT 只对BCC实例关联的预付费CDS、MKT进行续费
- EIP_MKT 只对BCC实例关联的预付费EIP、MKT进行续费
- CDS_EIP_MKT 只对BCC实例关联的预付费CDS、EIP、MKT进行续费

> 提示：
>
> - BCC虚机实例扩缩容期间不能进行续费操作。
> - 续费时若实例已欠费停机，续费成功后有个BCC虚机实例启动的过程。
> - 该接口是一个异步接口。
> - 专属实例不支持续费。

实例变更子网

如下代码可以变更实例的子网

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.InstanceChangeSubnetArgs{
 // 虚拟机实例ID
 InstanceId: "i-***",
 // 变更后的子网ID
 SubnetId:   "sbn-***",
 // 指定内网IP，不指定则随机生成
 InternalIp: "10.59.***.***",
 // 变更后是否重启
 Reboot:     true,
       // 普通安全组，仅变更vpc时生效，不支持同时指定EnterpriseSecurityGroupIds
       SecurityGroupIds: []string{"g-i24fkh******"},
       // 企业安全组，仅变更vpc时生效，不支持同时指定SecurityGroupIds
       EnterpriseSecurityGroupIds: []string{"esg-i24fkh******"},
 }

 err := bccClient.InstanceChangeSubnet(args)
 fmt.Println(err)
}
```

> 提示：
>
> - 变更子网后默认自动重启，用户选择是否执行该操作。
> - 变更子网的范围目前仅支持在同AZ下变更子网，不支持跨AZ或跨VPC变更子网，如果从普通子网变更至NAT专属子网请先手动解绑EIP。

实例变更VPC

如下代码可以变更实例的VPC

```
 package main

import (
 "fmt"

 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.InstanceChangeVpcArgs{
  InstanceId:       "i-AgPa****",              // 实例ID
  SubnetId:         "sbn-wwqfr3rd****",        // 子网ID
  InternalIp:       "192.168.*.*",            // 内网IP
  SecurityGroupIds: []string{"g-cfbk8xa0****"}, // 安全组ID
 }
 err := bccClient.InstanceChangeVpc(args)
 if err != nil {
  fmt.Println(err)
 }
}
```

提示：

- 变更VPC后默认自动重启，用户选择是否执行该操作。
- 变更VPC后仅保留主网卡主IP（在新子网中自动分配），实例上的辅助IP、弹性网卡和安全组等信息不跟随主体迁移。
- 安全组和企业安全组不能同时指定。

向指定实例批量添加指定ip

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 //创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 privateIps := []string{"192.168.***.***", "192.168.***.***"}
 batchAddIpArgs := &api.BatchAddIpArgs{
  // 实例ID
  InstanceId:                "i-***",
  // 辅助IP，最多100个，和SecondaryPrivateIpAddressCount不可同时使用
  PrivateIps:                privateIps,
  // 是否是多IPV6，添加IPV6辅助地址必须是true
  AllocateMultiIpv6Addr:     false,
  // 自动分配IP数量，和PrivateIps不可同时使用
  //SecondaryPrivateIpAddressCount: 2,
  // 幂等性Token，使用 uuid 生成一个长度不超过64位的ASCII字符串，可选参数
  ClientToken:               "random-uuid",
 }
 res, err := bccClient.BatchAddIP(batchAddIpArgs)
 fmt.Println(err)
 fmt.Println(res)
}
```

批量删除指定实例的ip

```
 privateIps := []string{"192.168.1.25"}
instanceId := "your-choose-instance-id"
// 幂等性Token，使用 uuid 生成一个长度不超过64位的ASCII字符串，可选参数
clientToken := "clientToken"
batchDelIpArgs := &api.BatchDelIpArgs{
 InstanceId:    instanceId,
   // 辅助IP，最多100个
 PrivateIps:    privateIps,
   ClientToken:   clientToken,
}
if err := client.BatchDelIP(batchDelIpArgs); err != nil {
   fmt.Println("delete ips failed: ", err)
} else {
   fmt.Println("delete ips success.")
}
```

开通自动续费（包含关联产品）

自动续费仅限预付费产品

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 bccAutoRenewArgs := &api.BccCreateAutoRenewArgs{
  // 实例ID
  InstanceId:    "i-***",
  // 续费周期单位，可选值：year, month
  RenewTimeUnit: "year",
  // 续费时长
  RenewTime:     1,
  // 是否合并cds自动续费，默认值为true
  RenewCds:      false,
  // 是否合并eip自动续费，默认值为true
  RenewEip:      false,
 }
 err := bccClient.BatchCreateAutoRenewRules(bccAutoRenewArgs)
 fmt.Println(err)
}
```

关闭自动续费（包含关联产品）

自动续费仅限预付费产品

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 bccDeleteAutoRenewArgs := &api.BccDeleteAutoRenewArgs{
  // 实例ID
  InstanceId: "i-***",
  // 是否合并cds自动续费，默认值为true。
  RenewCds:   false,
  // 是否合并eip自动续费，默认值为true。
  RenewEip:   false,
 }
 err := bccClient.BatchDeleteAutoRenewRules(bccDeleteAutoRenewArgs)
 fmt.Println(err)
}
```

后付费资源从回收站恢复计费

仅限后付费产品，预付费资源走续费接口

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.RecoveryInstanceArgs{
        InstanceIds: []api.RecoveryInstanceModel{
            {
                InstanceId: "i-***",
            },
            {
                InstanceId: "i-***",
            },
        },
    }
    if err := bccClient.RecoveryInstance(args); err != nil {
        fmt.Println("recovery instance failed: ", err)
    } else {
        fmt.Println("recovery instance success")
    }
}
```

计费变更-转预付费

使用以下代码对实例计费变更-转预付费:

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.ChangeToPrepaidRequest{
  Duration:    1,    // 转预付费时长，单位为月
 }
 result, err := bccClient.ChangeToPrepaid("i-cVh8****", args) // 虚机ID, 转预付费参数
 fmt.Println(err)
 fmt.Println(result)
}
```

实例绑定标签

使用以下代码对实例绑定标签:

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/model"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置想要绑定的标签
    args := &api.BindTagsRequest{
        ChangeTags: []model.TagModel{
            {
                TagKey:   "Key***",
                TagValue: "Value***",
            },
        },
    }
    // 设置你要操作的instanceId
    instanceId := "i-***"
    if err := bccClient.BindInstanceToTags(instanceId, args); err != nil {
        fmt.Println("BindInstanceToTags failed: ", err)
    } else {
        fmt.Println("BindInstanceToTags success.")
    }
}
```

实例解绑标签

使用以下代码对实例解绑标签:

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/model"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置想要解绑的标签
    args := &api.UnBindTagsRequest{
        ChangeTags: []model.TagModel{
            {
 TagKey:   "Key***",
 TagValue: "Value***",
            },
        },
    }
    // 设置你要操作的instanceId
    instanceId := "i-***"
    if err := bccClient.UnBindInstanceToTags(instanceId, args); err != nil {
        fmt.Println("UnBindInstanceToTags failed: ", err)
    } else {
        fmt.Println("UnBindInstanceToTags success.")
    }
}
```

查询可以变配的实例规格

使用以下代码可以查询可以变配的实例规格

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    listAvailableResizeSpecsArgs := &api.ListAvailableResizeSpecsArgs{
// 实例规格
Spec: "bcc.ic5.c1m1",
// 可用区
        Zone: "cn-bj-d",
    }
    res, _ := bccClient.ListAvailableResizeSpecs(listAvailableResizeSpecsArgs)
    fmt.Println(res)
}
```

批量转预付费

使用以下代码可以将实例批量转预付费，若需要对实例关联的cds变更，则必须在CdsList中传入所有需要转为预付费的cdsId，若所有cds均变更计费传入all。

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // InstanceId最多100个
    batchChangeInstanceToPrepayArgs := &api.BatchChangeInstanceToPrepayArgs{
        Config: []api.PrepayConfig{
            {
                InstanceId: "i-***",
                Duration:   1,
                CdsList: []string{
                    "v-***",
                    "v-***",
                },
            },
            {
                InstanceId: "i-***",
                Duration:   1,
                CdsList: []string{
                    "all",
                },
            },
        },
    }
    result, err := bccClient.BatchChangeInstanceToPrepay(batchChangeInstanceToPrepayArgs)
    if err != nil {
        fmt.Println("batch change instance to prepay failed:", err)
    } else {
        fmt.Println("batch change instance to prepay success: ", result)
    }
}
```

批量转后付费

使用以下代码可以将实例批量转后付费， 若需要对实例关联的cds变更，则必须在CdsList中传入所有需要转为后付费的cdsId，若所有cds均变更计费传入all。

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
    // InstanceId最多100个
    batchChangeInstanceToPostArgs := &api.BatchChangeInstanceToPostpayArgs{
        Config: []api.PostpayConfig{
            {
                InstanceId: "i-***",
                CdsList: []string{
                    "v-***",
                    "v-***",
                },
            },
            {
                InstanceId: "i-***",
                CdsList: []string{
                    "all",
                },
            },
        },
    }
    result, err := bccClient.BatchChangeInstanceToPostpay(batchChangeInstanceToPostArgs)
    if err != nil {
        fmt.Println("batch change instance to postpay failed:", err)
    } else {
        fmt.Println("batch change instance to postpay success: ", result)
    }
}
```

获取实例角色列表

使用以下代码可以获取实例角色列表

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    result, err := bccClient.ListInstanceRoles()
    if err != nil {
        fmt.Println("list instance roles failed:", err)
    } else {
        fmt.Println("list instance roles success: ", result)
    }
}
```

绑定角色

使用以下代码可以为BCC实例绑定角色

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置想要绑定的标签
    bindInstanceRoleArgs := &api.BindInstanceRoleArgs{
        RoleName: "BCC_***",
        Instances: []api.Instances{
            {
                InstanceId: "i-***",
            },
        },
    }

    if err := bccClient.BindInstanceRole(bindInstanceRoleArgs); err != nil {
        fmt.Println("bind instance role failed: ", err)
    } else {
        fmt.Println("bind instance role success.")
    }
```

解绑角色

使用以下代码可以为BCC实例解绑角色

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置想要绑定的标签
    unbindInstanceRoleArgs := &api.UnBindInstanceRoleArgs{
        RoleName: "BCC_***",
        Instances: []api.Instances{
            {
                InstanceId: "i-***",
            },
        },
    }

    if err := bccClient.UnBindInstanceRole(unbindInstanceRoleArgs); err != nil {
        fmt.Println("unbind instance role failed: ", err)
    } else {
        fmt.Println("unbind instance role success.")
    }
}
```

添加Ipv6

使用以下代码可以为BCC实例添加Ipv6

```
 package main

import (
 "fmt"

 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 addIpv6Args := &api.AddIpv6Args{
  // 实例ID
  InstanceId:  "i-***",
  // IPv6地址
  Ipv6Address: "240c:4081:8005:ca0b:***:***",
  // 是否重启实例
  Reboot:      true,
 }

 result, err := bccClient.AddIpv6(addIpv6Args)
 fmt.Println(result)
 fmt.Println(err)
}
```

释放Ipv6

使用以下代码可以为BCC实例释放Ipv6

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 addIpv6Args := &api.DeleteIpv6Args{
  // 实例ID
  InstanceId: "i-***",
  // 是否重启实例
  Reboot:     true,
 }
 err := bccClient.DeleteIpv6(addIpv6Args)
 fmt.Println(err)
}
```

根据实例ID批量查询实例列表

以下代码可以根据实例ID批量查询实例列表

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.ListInstanceByInstanceIdArgs{
        // 待查询的实例id列表，最多100个
        InstanceIds: []string{
            "i-***",
            "i-***",
        },
    }
    result, err := bccClient.ListInstanceByInstanceIds(args)
    if err != nil {
        fmt.Println("list instance failed:", err)
    } else {
        fmt.Println("list instance success: ", result)
    }
}
```

查询实例绑定的弹性网卡列表

使用以下代码可以查询实例绑定的弹性网卡列表

```
// 设置你要操作的instanceId，最多100个
instanceId := "instanceId"
if res, err := BCC_CLIENT.ListInstanceEnis(instanceId); err != nil {
 fmt.Println("Get specific instance eni failed: ", err)
} else {
 fmt.Println("Get specific instance eni success, result: ", res)
}
```

查询实例自定义数据

使用以下代码可以查询实例自定义数据

```
// 设置你要操作的instanceId
  args := &api.DescribeInstanceUserDataArg{
 InstanceId: "i-b34ycow2",
}
result, err := BCC_CLIENT.getInstanceUserData(args)
fmt.Println(result)
fmt.Println(err)
```

磁盘

创建CDS磁盘

支持新建CDS云磁盘或者从CDS数据盘快照创建CDS磁盘，参考以下代码可以创建CDS磁盘：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.CreateCDSVolumeArgs{
  PurchaseCount: 2,                  // 购买数量
  CdsSizeInGB:   5,                  // 磁盘大小
  StorageType:   api.StorageTypeHdd, // 存储类型
        ChargeType:   api.PaymentTimingPostPaid, // 付费方式
  ZoneName:     "cn-bj-c",          // 可用区
  Name:         "test-create-cds", // 磁盘名称
  Description: "test-desc",       // 磁盘描述
        Tags：[]model.TagModel{          // 绑定标签
           {
              // 标签key
              TagKey:   "Key-***",
              // 标签value
              TagValue: "Value***",
           },
        },
 }
 result, err := bccClient.CreateCDSVolume(args)
 if err != nil {
           fmt.Println("create CDS volume failed:", err)
        } else {
           fmt.Println("create CDS volume success: ", result)
        }
}
```

> **提示：**
>
> - 创建CDS磁盘接口为异步接口，可通过查询磁盘详情接口查询磁盘状态，详细接口使用请参考BCC API 文档查询磁盘详情

查询磁盘列表

以下代码可以查询所有的磁盘列表，支持分页查询以及通过云磁盘所挂载的BCC实例ID进行过滤筛选:

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 磁盘列表查询参数
 queryArgs := &api.ListCDSVolumeArgs{
  Marker:     "v-3uIJ****",  // 起始磁盘ID
  MaxKeys:    10,            // 返回结果的最大数量
  InstanceId: "i-cVh8****",  // 磁盘挂载的实例ID
  ZoneName:   "cn-bj-a",     // 可用区
  ClusterId:  "DC-rWKx****", // 专属集群ID
 }

 res, err := bccClient.ListCDSVolume(queryArgs)
 if err != nil {
  fmt.Println(err)
 } else {
  fmt.Println(res)
 }
}
```

查询磁盘详情

通过磁盘ID可以获取对应磁盘的详细信息，以下代码可以查询磁盘详情：

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 获取指定CDS卷的详细信息
 res, err := bccClient.GetCDSVolumeDetail("v-3uIJ****")  // 指定CDS磁盘ID
 if err != nil {
  fmt.Println(err)
 } else {
  fmt.Println(res)
 }
}
```

挂载CDS磁盘

可以将未挂载的磁盘挂载在对应的BCC实例下，以下代码将一个CDS挂载在对应的BCC实例下:

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.AttachVolumeArgs{
  // 设置要挂载的实例ID
  InstanceId: "i-***",
 }
 result, err := bccClient.AttachCDSVolume(
  // 设置要挂载的磁盘ID
  "v-***",
  args)
 fmt.Println(result)
 fmt.Println(err)
}
```

> **提示：**
>
> - CDS磁盘需要挂载在与其处在相同zone下的BCC实例上，否则将返回403错误。
> - 只有磁盘状态为 Available 且实例状态为 Running 或 Stopped 时才允许挂载，否则调用此接口将返回 409 错误。

卸载CDS磁盘

可以将已挂载的磁盘从对应的BCC虚机上卸载下来，以下代码卸载CDS磁盘:

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.DetachVolumeArgs{
  // 设置要挂载的实例ID
  InstanceId: "i-***",
 }
 err := bccClient.DetachCDSVolume(
  // 设置要挂载的磁盘ID
  "v-***",
  args)

 fmt.Println(err)
}
```

提示：

- 只有实例状态为 Running 或 Stopped 时，磁盘才可以执行此操作，否则将提示 409 错误。
- 如果 volumeId 的磁盘不挂载在 instanceId 的实例上，该操作失败，提示 404 错误。

释放CDS磁盘

用于释放未挂载的CDS磁盘，可指定是否删除磁盘关联的快照，缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系:

```
 err = client.DeleteCDSVolume(volumeId)
if err != nil {
    fmt.Println("delete CDS volume failed:", err)
} else {
    fmt.Println("delete CDS volume success")
}
```

提示：

- 已挂载的CDS磁盘不能释放，系统盘不能释放。
- 若磁盘释放后不可恢复。缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误。

释放磁盘（POST）

提示：

- 该接口用于释放未挂载的CDS磁盘，系统盘不能释放。

- 用户可指定是否删除磁盘关联的快照，缺省情况下，该磁盘的所有快照将保留，但会删除与磁盘的关联关系。
- 用户可以指定是否将磁盘放入回收站
- 只有磁盘状态为 Available 或 Expired 或 Error 时才可以执行此操作，否则将提示 409 错误。
- 如果 volumeId 的磁盘不存在，将提示 404 错误 使用以下代码可以释放磁盘：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 磁盘ID
 volumeId := "v-***"

 args := &api.DeleteCDSVolumeArgs{
  // 是否删除磁盘关联的自动快照
  AutoSnapshot:   "on",
  // 是否删除磁盘关联的手动快照
  ManualSnapshot: "on",
  // 是否将磁盘回收至回收站，为off时直接释放，不进入回收站
  Recycle:        "off",
 }

    // 请求释放磁盘
 err := bccClient.DeleteCDSVolumeNew(volumeId, args)

 fmt.Println(err)
}
```

磁盘重命名

以下代码可以给一个CDS磁盘重命名：

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 volumeId := "" // 待重命名的磁盘ID
 args := &api.RenameCSDVolumeArgs{
  Name: "testVolume",  // 新的磁盘名称
 }
 err := bccClient.RenameCDSVolume(volumeId, args)
 if err != nil {
  fmt.Println("rename CDS volume failed", err)
 } else {
  fmt.Println("rename CDS volume success")
 }
}
```

修改磁盘属性

可以使用以下代码修改指定磁盘名称、描述信息：

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.ModifyCSDVolumeArgs{
  CdsName: "name", // 待修改的CDS名称
  Desc:    "desc", // 修改后的CDS描述
 }
 volumeId := "v-vG0z****" // 磁盘ID
 err := bccClient.ModifyCDSVolume(volumeId, args)
 if err != nil {
  fmt.Println("modify CDS volume failed: ", err)
 } else {
  fmt.Println("modify CDS volume success")
 }
}
```

磁盘计费变更

可以使用以下代码变更磁盘计费方式，仅支持按量付费（后付费）转包年包月（预付费）、包年包月（预付费）转按量付费（后付费）两种方式。变更为包年包月（预付费）需要指定购买时长。

- 使用以下代码可以完成将包年包月（预付费）磁盘计费变更为按量付费（后付费）。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 磁盘ID
 volumeId := "v-***"
 args := &api.ModifyChargeTypeCSDVolumeArgs{
  Billing: &api.Billing{
   // 后付费标志, 可不填
   PaymentTiming: api.PaymentTimingPostPaid,
  },
 }

    // 请求将预付费磁盘变更为后付费
 err := bccClient.ModifyChargeTypeCDSVolume(volumeId, args)
 fmt.Println(err)
}
```

- 使用以下代码可以完成将按量付费（后付费）磁盘变更为包年包月（预付费）：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 磁盘ID
 volumeId := "v-***"
 args := &api.ModifyChargeTypeCSDVolumeArgs{
  Billing: &api.Billing{
   // 预付费标志, 可不填
   PaymentTiming: api.PaymentTimingPrePaid,
   Reservation: &api.Reservation{
    // 预留时长，预留时长单位为month，取值范围[1,9]
    ReservationLength:   1,
    // 预留时长单位, 当前仅支持month，可不填
    ReservationTimeUnit: "month",
   },
  },
 }

 // 请求将后付费磁盘变更为预付费
 err := bccClient.ModifyChargeTypeCDSVolume(volumeId, args)
 fmt.Println(err)
}
```

磁盘开通自动续费

通过以下代码可以开通磁盘自动续费：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    volumeId := "v-***"
 renewTimeUnit := "month"
 // 设置自动续费的时间 按月是1-9 按年是 1-3
 renewTime := 4
 args := &api.AutoRenewCDSVolumeArgs{
  // 磁盘ID
  VolumeId:      volumeId,
  // 续费时间单位, 当前支持取值为：month
  RenewTimeUnit: renewTimeUnit,
  // 续费时间, 以month为单位续费时支持取值为：1-9
  RenewTime:     renewTime,
 }
 err := bccClient.AutoRenewCDSVolume(args)
 fmt.Println(err)
}
```

- 注意数据盘才能进行自动续费操作。
- 注意过期状态磁盘不能进行自动续费操作。
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间。

磁盘取消自动续费

通过以下代码可以取消磁盘自动续费：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    volumeId := "v-***"
 args := &api.CancelAutoRenewCDSVolumeArgs{
  // 磁盘ID
  VolumeId: volumeId,
 }

 err := bccClient.CancelAutoRenewCDSVolume(args)
 fmt.Println(err)
}
```

- 注意数据盘才能取消进行自动续费操作。
- 异步接口，可通过查询磁盘详情接口查询磁盘到期时间。

磁盘扩容

使用以下代码可以对磁盘进行扩大容量操作：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 磁盘ID
    volumeId := "v-***"
 // 请求扩容参数
 args := &api.ResizeCSDVolumeArgs{
  // 扩容大小，单位为GB
  NewCdsSizeInGB: 60,
  // 扩容类型
  NewVolumeType:  api.StorageTypeEnhancedPl1,
 }

 // 请求扩容cds盘，变更磁盘类型和大小
 err := bccClient.ResizeCDSVolume(volumeId, args)

 fmt.Println(err)
}
```

提示：

- 磁盘只能进行扩大容量，不支持缩小容量。
- 只有Available状态的磁盘，才能进行扩容操作
- 磁盘扩容是一个异步接口，可通过查询磁盘详情接口查询磁盘扩容状态。

回滚磁盘

可以使用指定磁盘自身的快照回滚磁盘内容，使用以下代码可以对磁盘进行回滚：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 volumeId := "v-zBms****"   // 磁盘ID

 args := &api.RollbackCSDVolumeArgs{
  SnapshotId: "s-HkbJ****",   // 快照ID
 }
 err := bccClient.RollbackCDSVolume(volumeId, args)
 if err != nil {
  fmt.Println(err)
 }
}
```

提示：

- 磁盘状态必须为 Available 才可以执行回滚磁盘操作。
- 指定快照id必须是由该磁盘id创建的快照。
- 若是回滚系统盘，实例状态必须为 Running 或 Stopped 才可以执行此操作。
- 回滚系统盘快照，自本次快照以来的系统盘数据将全部丢失，不可恢复。

磁盘续费

对磁盘的续费操作，可以延长过期时长，以下代码可以对磁盘进行续费：

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 待变配的实例配置参数
 args := &api.PurchaseReservedCSDVolumeArgs{
  Billing: &api.Billing{
   PaymentTiming: api.PaymentTimingPrePaid,
   Reservation: &api.Reservation{
    ReservationLength:   1,      // 续费时长
    ReservationTimeUnit: "month", // 续费时长单位
   },
  },
 }
 volumeId := "v-dj5Q****"    // 磁盘ID
 err := bccClient.PurchaseReservedCDSVolume(volumeId, args)
 if err != nil {
  fmt.Println("purchase reserve CDS volume failed:", err)
 } else {
  fmt.Println("purchase reserve CDS volume success")
 }
}
```

释放CDS磁盘（新）

如下代码可以释放一个CDS磁盘及相关联的快照，并指定是否进入回收站

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 磁盘ID
 volumeId := "v-***"

 args := &api.DeleteCDSVolumeArgs{
  // 是否删除磁盘关联的自动快照
  AutoSnapshot:   "on",
  // 是否删除磁盘关联的手动快照
  ManualSnapshot: "on",
  // 是否将磁盘回收至回收站，为off时直接释放，不进入回收站
  Recycle:        "off",
 }

    // 请求释放磁盘
 err := bccClient.DeleteCDSVolumeNew(volumeId, args)

 fmt.Println(err)
}
```

> **提示：**
>
> - 该接口用于释放未挂载的CDS磁盘，系统盘不能释放。
> - 与老接口功能上的区别在于，可以控制是否删除与磁盘关联的快照。

查询可用区可购买的磁盘信息

使用以下代码可以查询指定可用区可购买的磁盘信息

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 可用区名称；当传入zoneName为空串或为非法zone时，会返回全部可用区的可购买磁盘信息。
    zoneName := "cn-bj-d"

    // 请求查询可用区下可购买磁盘信息
 result, err := bccClient.GetAvailableDiskInfo(zoneName)
 fmt.Println(err)
 fmt.Println(result)
}
```

磁盘绑定标签

使用以下代码可以给指定磁盘绑定标签

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 磁盘ID
    volumeId := "v-***"
 tagArgs := &api.TagVolumeArgs{
  // 是否为关联资源绑定标签
  RelationTag: false,
  ChangeTags: []api.Tag{
   {
    // 标签key
    TagKey:   "Key-***",
    // 标签value
    TagValue: "Value***",
   },
   {
    // 标签key
    TagKey:   "Key-***",
    // 标签value
    TagValue: "Value***",
   },
  },
 }

    // 请求为磁盘绑定标签
 err := bccClient.TagVolume(volumeId, tagArgs)
 fmt.Println(err)
}
```

磁盘解绑标签

使用以下代码可以给指定磁盘绑定标签

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 磁盘ID
    volumeId := "v-***"
    tagArgs := &api.TagVolumeArgs{
  // 是否为关联资源解绑标签
  RelationTag: false,
        ChangeTags: []api.Tag{
            {
    // 标签key
    TagKey:   "Key-***",
    // 标签value
    TagValue: "Value***",
   },
            {
    // 标签key
    TagKey:   "Key-***",
    // 标签value
    TagValue: "Value***",
   },
        },
    }

    // 请求为磁盘解绑标签
    err := bccClient.UntagVolume(volumeId, tagArgs)
    fmt.Println(err)
}
```

镜像

创建自定义镜像

- 该接口用于创建自定义镜像，默认配额每账号每地域20个，创建后的自定义镜像可用于创建实例。
- 支持通过实例创建和通过快照创建两种方式。
- 当通过实例创建时，只有 Running 或 Stopped 状态的实例才可以执行成功，否则会提示 409 错误。
- 仅限通过系统盘快照创建自定义镜像。
- 当通过快照创建时，只有 Available 状态的快照才可以执行成功，否则会提示 409 错误

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 待创建的自定义镜像名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
 imageName := "image02"
 // 当从快照创建镜像时，此参数是指用于创建镜像的快照ID。可选参数，与 snapshotId 不同时存在，同时存在时只取
instanceId
 snapshotId := "m-***"
 // 当从实例创建镜像时，此参数是指用于创建镜像的实例ID。可选参数，与 snapshotId 不同时存在，同时存在时只取
instanceId
 instanceId := "i-***"
 // 是否关联CDS，默认为false
 relateCds := false
 args := &api.CreateImageArgs{
  SnapshotId:  snapshotId,
  IsRelateCds: relateCds,
  ImageName:   imageName,
  InstanceId:  instanceId,
 }
 resp, err := bccClient.CreateImage(args)
 fmt.Println(resp, err)
}
```

> 注意，创建自定义镜像，默认配额每账号每地域20个。

查询镜像列表

- 使用以下代码可以查询有权限的镜像列表。
- 查询的镜像信息中包括公共镜像、自定义镜像和服务集成镜像。
- 支持按 imageType 来过滤查询，此参数非必需，缺省为 All,即查询所有类型的镜像。
- 支持按 imageName 来过滤查询自定义镜像，返回名称中包含该字符串的镜像。此参数非必需，设定ImageName时，必须同时指定ImageType为'Custom'。

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 createInstanceArgs := &api.ListImageArgs{
  // 镜像类型
  ImageType: "ALL",
 }
 images, err := bccClient.ListImage(createInstanceArgs)
 fmt.Println(err)
 fmt.Println(images)
}
```

> 具体的镜像类型可详细参考BCC API文档查询镜像列表

查询镜像详情

以下代码可以查询镜像详细信息：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 imageId := "m-***"
 imageDetail, err := bccClient.GetImageDetail(imageId)
 fmt.Println(err)
 fmt.Println(imageDetail.Image)
}
```

删除自定义镜像

- 该接口用于删除用户指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能删除。
- imageId 所指定的镜像不存在，提示404错误。
- 镜像删除后无法恢复，不能再用于创建、重置实例。

以下代码可以删除一个自定义镜像：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 imageId := "m-***"
 err := bccClient.DeleteImage(imageId)
 fmt.Println(err)
}
```

跨地域复制自定义镜像

- 用于用户跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制。
- regions如北京"bj",广州"gz",苏州"su"，可多选：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 imageId := "m-***"
 destinations := &api.RemoteCopyImageArgs{
  Name:       "new-image",
  DestRegion: []string{"hkg"},
 }
 err := bccClient.RemoteCopyImage(imageId, destinations)
 fmt.Println(err)
}
```

跨地域复制自定义镜像并返回目的region的镜像镜像ID

- 用于用户跨地域复制自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能复制。
- regions如北京"bj",广州"gz",苏州"su"，可多选：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 imageId := "m-***"
 destinations := &api.RemoteCopyImageArgs{
  Name:       "new-image",
  DestRegion: []string{"hkg"},
 }
 imageCopyResult, err := bccClient.RemoteCopyImageReturnImageIds(imageId, destinations)
 fmt.Println(err)
 fmt.Println(imageCopyResult)
}
```

取消跨地域复制自定义镜像

以下代码用于取消跨地域复制自定义镜像：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 imageId := "m-***"
 err := bccClient.DeleteImage(imageId)
 fmt.Println(err)
}
```

共享自定义镜像

- 该接口用于共享用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能共享。
- imageId 所指定的镜像不存在，提示404错误。
- 镜像共享后，被共享的用户可以使用此镜像创建、重置实例。
- 请求参数中的account和accountId均为可选参数，但不能全部为空，当两个参数同时出现时，服务端会自动去重。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    ShareImageId := "m-***"

 // 请求将镜像分享给指定用户id
 args := &api.SharedUser{
  // 账号ID
  AccountId: "***",
 }
 err := bccClient.ShareImage(ShareImageId, args)
 fmt.Println(err)
}
```

取消共享自定义镜像

- 该接口用于取消共享用户自己的指定的自定义镜像，仅限自定义镜像，公共镜像和服务集成镜像不能共享。
- imageId 所指定的镜像不存在，提示404错误。
- 镜像取消共享后，被取消共享的用户不能再使用此镜像创建、重置实例。
- 请求参数中的account和accountId均为可选参数，但不能全部为空，当两个参数同时出现时，服务端会自动去重。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    UnShareImageId := "m-***"

 // 请求取消将镜像分享给指定用户id
 args := &api.SharedUser{
  // 账号ID
  AccountId: "***",
 }
 err := bccClient.UnShareImage(UnShareImageId, args)
 fmt.Println(err)
}
```

查询镜像已共享用户列表

- imageId 所指定的镜像不存在，提示404错误。 如下代码用于查询镜像已共享的用户列表：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 镜像id
    imageId := "m-***"
 result, err := bccClient.GetImageSharedUser(imageId)
 fmt.Println(result)
 fmt.Println(err)
}
```

根据实例ID批量查询OS信息

如下代码可以根据实例的ID来查询相应OS的信息：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.GetImageOsArgs{
  // 实例ID列表
  InstanceIds: []string{"i-***", "i-***"},
 }
 result, err := bccClient.GetImageOS(args)
 fmt.Println(result)
 fmt.Println(err)
}
```

镜像绑定标签

使用以下代码可以给指定镜像绑定标签：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/model"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.BindTagsRequest{
  ChangeTags: []model.TagModel{
   {
    // 标签键
    TagKey:   "Key***",
    // 标签值
    TagValue: "Value***",
   },
   {
    // 标签键
    TagKey:   "Key***",
    // 标签值
    TagValue: "Value***",
   },
  },
 }

 result := bccClient.BindImageToTags("m-***", args)
 fmt.Println(result)
}
```

镜像解绑标签

使用以下代码可以给指定镜像绑定标签：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/model"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.UnBindTagsRequest{
  ChangeTags: []model.TagModel{
   {
    // 标签键
    TagKey:   "Key***",
    // 标签值
    TagValue: "Value***",
   },
  },
 }
 err := bccClient.UnBindImageToTags(
  // 镜像Id
  "m-***",
  args)
 fmt.Println(err)
}
```

导入镜像

使用以下代码可以导入镜像：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)
func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.ImportCustomImageArgs{
  // 操作系统名称
  OsName:    "Centos",
  // 操作系统位数
  OsArch:    "64",
  // 操作系统类型
  OsType:    "linux",
  // 操作系统版本
  OsVersion: "6.5",
  // 镜像名称,支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
        Name:      "image-2",
  // bos镜像地址
  BosURL:    "https://bcc-bos.bj.bcebos.com/image_test/CentOS-6-x86_64-GenericCloud.qcow2",
 }
 result, _ := bccClient.ImportCustomImage(args)
 fmt.Println(result)
}
```

根据实例规格查询可用公共镜像

使用以下代码可以根据实例规格查询可用的公共镜像：

```
 func TestGetAvailableImagesBySpec(t *testing.T) {

 args := &api.GetAvailableImagesBySpecArg{
  OsName: "Centos",
  Spec:    "bcc.ic4.c1m1",
  MaxKeys: 10,
  Marker: "m-21bmeYvH",
 }

 result, _ := BCC_CLIENT.GetAvailableImagesBySpec(args)
 fmt.Println(result)
}
```

快照

创建快照

如下代码可以创建一个快照：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.CreateSnapshotArgs{
        // 磁盘ID
        VolumeId:     "v-***",
        SnapshotName: "snapshotName",
        Description:  "create by sdk",
    }
    result, err := bccClient.CreateSnapshot(args)
    if err != nil {
        fmt.Println("create snapshot failed:", err)
    } else {
        fmt.Println("create snapshot success: ", result)
    }
}
```

查询快照列表

如下代码可以查询当前账户下所有快照的列表：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.ListSnapshotArgs{}
    result, err := bccClient.ListSnapshot(args)
    if err != nil {
        fmt.Println("list all snapshot failed:", err)
    } else {
        fmt.Println("list all snapshot success: ", result)
    }
}
```

查询快照详情

如下代码可以查询特定快照的详细信息：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 快照id
    snapshotId := "s-***"
    result, err := bccClient.GetSnapshotDetail(snapshotId)
    if err != nil {
        fmt.Println("get snapshot detail failed:", err)
    } else {
        fmt.Println("get snapshot detail success: ", result)
    }
}
```

删除快照

如下代码可以删除一个快照：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 快照id
    snapshotId := "s-***"
    err := bccClient.DeleteSnapshot(snapshotId)
    if err != nil {
        fmt.Println("delete snapshot failed:", err)
    } else {
        fmt.Println("delete snapshot success")
    }
}
```

查询快照链列表

通过以下代码可以查询快照链列表：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.ListSnapshotChainArgs{
  // 设置排序属性：chainId(快照链id,默认值),chainSize(快照链大小),volumeSize(磁盘大小)。可选参数
  OrderBy:  "your-order-by",
  // 设置排序方式:asc(正序,默认值), desc(倒序)。可选参数
  Order:    "your-order",
  // 设置每页容量，默认值为1000，可选参数
  PageSize: 100,
  // 设置页数，默认为1，可选参数
  PageNo:   1,
  // 设置磁盘ID，该字段非空则只返回这个磁盘的快照链信息,可选
        VolumeId: "v-***",
    }
    if res, err := bccClient.ListSnapshotChain(args); err != nil {
        fmt.Println("get snapshot chain list failed: ", err)
    } else {
        fmt.Println("get snapshot chain list success, Snapshots: ", res)
    }
}
```

快照链绑定标签

使用以下代码可以给指定快照链绑定标签：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    tagArgs := &api.TagVolumeArgs{
        ChangeTags: []api.Tag{
            {
    TagKey:   "Key***",
    TagValue: "Value***",
            },
        },
    }
    err := bccClient.TagSnapshotChain("your-chain-id", tagArgs)
    if err != nil {
        fmt.Println("tag snapshotChain failed:", err)
    } else {
        fmt.Println("tag snapshotChain success")
    }
}
```

快照链解绑标签

使用以下代码可以给指定快照链绑定标签：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "http://bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    tagArgs := &api.TagVolumeArgs{
        ChangeTags: []api.Tag{
            {
    TagKey:   "Key***",
    TagValue: "Value***",
            },
        },
    }
    err := bccClient.UntagSnapshotChain("your-chain-id", tagArgs)
    if err != nil {
        fmt.Println("untag snapshotChain failed:", err)
    } else {
        fmt.Println("untag snapshotChain success")
    }
}
```

跨地域复制快照

使用以下代码可以将一份磁盘快照从一个地域复制到其他地域：

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 快照ID
 snapshotId := "s-***"
 args := &api.RemoteCopySnapshotArgs{
  DestRegionInfos: []api.DestRegionInfo{
   {
    // 快照名称
    Name:       "your_snap_name_for_hkg",
    // 目标区域
    DestRegion: "hkg",
   },
   {
    // 快照名称
    Name:       "your_snap_name_for_fwh",
    // 目标区域
    DestRegion: "fwh",
   },
  },
 }

 // 请求跨地域复制快照
 result, err := bccClient.CreateRemoteCopySnapshot(snapshotId, args)

 fmt.Println(err)
 fmt.Println(result)
}
```

自动快照策略

创建自动快照策略

如下代码可以创建一个自动快照策略：

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.CreateASPArgs{
  // 设置自动快照策略名称
  Name: "sdkCreate",
  // 设置一天中做快照的时间点，取值为0~23，0为午夜12点
  // 例如设置做快照的时间点为下午两点：
  TimePoints: []string{"14"},
  // 设置一周中做快照的时间，取值为0~6，0代表周日，1~6代表周一到周六
  // 例如设置做快照的时间为礼拜五：
  RepeatWeekdays: []string{"5"},
  // 设置自动快照保留天数，取-1则永久保留
  RetentionDays: "7",
 }
 result, err := bccClient.CreateAutoSnapshotPolicy(args)
 if err != nil {
  fmt.Println("create auto snapshot policy failed:", err)
 } else {
  fmt.Println("create auto snapshot policy success: ", result)
 }
}
```

绑定自动快照策略

如下代码可以将自动快照策略绑定到某个CDS磁盘：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

   // 设置需要绑定的自动快照策略ID
 aspId := "asp-***"
 args := &api.AttachASPArgs{
  // 设置需要绑定的磁盘ID列表
  VolumeIds: []string{"v-***", "v-***"},
 }
 err := bccClient.AttachAutoSnapshotPolicy(aspId, args)
 if err != nil {
  fmt.Println("attach auto snapshot policy with CDS volume failed:", err)
 } else {
  fmt.Println("attach auto snapshot policy with CDS volume success")
 }
}
```

解绑自动快照策略

如下代码可以将自动快照策略与指定CDS磁盘解除绑定：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置需要解绑的自动快照策略ID
 aspId := "asp-***"
 args := &api.DetachASPArgs{
  // 设置需要解绑的磁盘ID列表
  VolumeIds: []string{"v-***", "v-***"},
 }
 err := bccClient.DetachAutoSnapshotPolicy(aspId, args)
 if err != nil {
  fmt.Println("detach auto snapshot policy from CDS volume failed:", err)
 } else {
  fmt.Println("detach auto snapshot policy from CDS volume success")
 }
}
```

删除自动快照策略

如下代码可以删除自动快照策略：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置需要删除的自动快照策略ID
 aspId := "asp-***"
 err := bccClient.DeleteAutoSnapshotPolicy(aspId)
    if err != nil {
        fmt.Println("delete auto snapshot policy failed:", err)
    } else {
        fmt.Println("delete auto snapshot policy success")
    }
}
```

查询自动快照策略列表

如下代码可以查询到用户当前地域所有自动快照策略的列表：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.ListASPArgs{}
    result, err := bccClient.ListAutoSnapshotPolicy(args)
    if err != nil {
        fmt.Println("list all auto snapshot policy failed:", err)
    } else {
        fmt.Println("list all auto snapshot policy success: ", result)
    }
}
```

查询自动快照策略详情

如下代码可以查询到特定自动快照策略的详细信息：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    // 设置需要查询的自动快照策略ID
    aspId := "asp-***"
    result, err := bccClient.GetAutoSnapshotPolicy(aspId)
 if err != nil {
  fmt.Println("get auto snapshot policy detail failed:", err)
 } else {
  fmt.Println("get auto snapshot policy detail success", result)
 }
}
```

自动快照策略变更

如下代码可以更新一个自动快照策略：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 aspId := "asp-***"
 args := &api.UpdateASPArgs{
  // 自动快照策略名称，支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母开头，长度1-65。
  Name:           "changeDone",
  // 一天中做快照时间点，取值为0~23。
  TimePoints:     []string{"10"},
  // 一周中做快照的时间，取值为0~6。
  RepeatWeekdays: []string{"0", "1"},
  // 自动快照保留天数，取-1则永久保留。
  RetentionDays:  "2",
  // 自动策略的id
  AspId:          "asp-***",
 }
 err := bccClient.UpdateAutoSnapshotPolicy(args)
 fmt.Println(err)
}
```

安全组

查询安全组列表

以下代码可以查询安全组列表：

```
 args := &api.ListSecurityGroupArgs{}

// 设置筛选的实例Bcc实例id
args.InstanceId = instanceId

// 设置筛选的安全组绑定的VPC实例ID
args.VpcId = vpcId

result, err := client.ListSecurityGroup(args)
if err != nil {
    fmt.Println("list all security group failed:", err)
} else {
    fmt.Println("list all security group success: ", result)
}
```

创建安全组

以下代码可以创建一个安全组：

```
 args := &api.CreateSecurityGroupArgs{
   // 设置安全组名称
 Name: "sdk-create",
   // 设置安全组规则
 Rules: []api.SecurityGroupRuleModel{
  {
         // 设置安全组规则备注
   Remark:      "备注",
         // 设置协议类型
   Protocol:    "tcp",
         // 设置端口范围，默认空时为1-65535，可以指定80等单个端口
   PortRange:   "1-65535",
         // 设置入站出站，取值ingress/egress
   Direction:   "ingress",
         // 设置源IP地址，与sourceGroupId不能同时设置
   SourceIp:    "",
  },
 },
}
result, err := client.CreateSecurityGroup(args)
if err != nil {
    fmt.Println("create security group failed:", err)
} else {
    fmt.Println("create security group success: ", result)
}
```

同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，重复记录会报409错误。 protocol的取值（tcp|udp|icmp），默认值为空，代表all。 具体的创建安全组规则接口描述BCC API 文档创建安全组。

删除安全组

以下代码可以删除指定的安全组:

```
 err := client.DeleteSecurityGroup(securityGroupId)
if err != nil {
    fmt.Println("delete security group failed:", err)
} else {
    fmt.Println("delete security group success")
}
```

授权安全组规则

使用以下代码可以在指定安全组中添加授权安全组规则:

```
 args := &api.AuthorizeSecurityGroupArgs{
Rule: &api.SecurityGroupRuleModel{
 Remark:    "备注",
 Protocol:  "udp",
 PortRange: "1-65535",
 Direction: "ingress",
},
}
err := client.AuthorizeSecurityGroupRule(securityGroupId, args)
if err != nil {
    fmt.Println("authorize security group new rule failed:", err)
} else {
    fmt.Println("authorize security group new rule success")
}
```

- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，重复记录会报409错误。
- 具体的接口描述BCC API 文档授权安全组规则。

撤销安全组规则

使用以下代码可以在指定安全组中撤销指定安全组规则授权:

```
 args := &api.RevokeSecurityGroupArgs{
   Rule: &api.SecurityGroupRuleModel{
    Remark:      "备注",
    Protocol:    "udp",
    PortRange:   "1-65535",
    Direction:   "ingress",
    SourceIp:    "",
},
}
err := client.RevokeSecurityGroupRule(securityGroupId, args)
if err != nil {
    fmt.Println("revoke security group rule failed:", err)
} else {
    fmt.Println("revoke security group rule success")
}
```

- 同一安全组中的规则以remark、protocol、direction、portRange、sourceIp|destIp、sourceGroupId|destGroupId六元组作为唯一性索引，若安全组中不存在对应的规则将报404错误。
- 具体的接口描述BCC API 文档撤销安全组规则。

更新普通安全组规则

使用以下代码可以在指定普通安全组中更新安全组规则:

```
  args := &api.UpdateSecurityGroupRuleArgs{
    SecurityGroupRuleId: SecurityGroupRuleId,
    Remark:              Remark,
}
err := client.UpdateSecurityGroupRule(args)
if err != nil {
    fmt.Println("update security group new rule failed:", err)
} else {
    fmt.Println("update security group new rule success")
}
```

删除普通安全组规则

使用以下代码可以在指定普通安全组中删除指定安全组规则:

```
  args := &api.DeleteSecurityGroupRuleArgs{
    SecurityGroupRuleId: SecurityGroupRuleId,
}
err := client.DeleteSecurityGroupRule(args)
if err != nil {
    fmt.Println("delete security group rule failed:", err)
} else {
    fmt.Println("delete security group rule success")
}
```

关联安全组

使用以下代码可以在关联指定安全组:

```
  args := &api.BindSgV2Req{
  InstanceIds: []string{
   "InstanceIds",
  },
  SecurityGroupIds: []string{
   "SecurityGroupIds",
  },
  SecurityGroupType: "SecurityGroupType",
 }
res, err := BCC_CLIENT.InstanceBindSecurityGroup(args)
if err != nil {
    fmt.Println("bind security group failed:", err)
} else {
    fmt.Println("bind security group success")
}
```

解绑安全组

使用以下代码可以解绑指定安全组:

```
 args := &api.UnbindSgV2Req{
  InstanceIds: []string{
   "InstanceIds",
  },
  SecurityGroupIds: []string{
   "SecurityGroupIds",
  },
  SecurityGroupType: "SecurityGroupType",
 }
 res, err := BCC_CLIENT.InstanceUnbindSecurityGroup(args)
if err != nil {
    fmt.Println("unbind security group failed:", err)
} else {
    fmt.Println("unbind security group success")
}
```

替换安全组

使用以下代码可以替换为指定安全组:

```
 args := &api.ReplaceSgV2Req{
  InstanceIds: []string{
   "InstanceIds",
  },
  SecurityGroupIds: []string{
   "SecurityGroupIds",
  },
  SecurityGroupType: "enterprise",
 }
 res, err := BCC_CLIENT.InstanceReplaceSecurityGroup(args)
if err != nil {
    fmt.Println("replace security group failed:", err)
} else {
    fmt.Println("replace security group success")
}
```

密钥对

创建密钥对

如下代码用于创建密钥对，创建后的密钥对可植入实例以实现远程登录BCC实例：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)

 // 待创建的密钥对名称
 name := "bbc"
 // 待创建的密钥对的描述,可选参数
 description := "用于创建bbc"
 args := &api.CreateKeypairArgs{
  Name:        name,
  Description: description,
 }
 res, err := bccClient.CreateKeypair(args)
 fmt.Println(res, err)
}
```

导入密钥对

如下代码用于用户自行导入创建密钥对：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 创建密钥对名称
 name := "keyName"
 // 待创建的密钥对的描述,可选参数
 description := "description"
 // 待创建的密钥对的公钥
 publicKey := "ssh-***"
 args := &api.ImportKeypairArgs{
  Name:        name,
  Description: description,
  PublicKey:   publicKey,
 }
 res, err := bccClient.ImportKeypair(args)
 fmt.Println(res, err)
}
```

绑定密钥对

如下代码可以将所选密钥对（限单个）绑定到所选BCC实例（支持多台）。 目前一台BCC实例只能绑定一个密钥对，若操作的BCC实例已经绑定密钥对，则此操作将替换该BCC实例原有的密钥对。此操作仅适用于linux系统的BCC实例，且所选BCC实例必须处于运行中或已关机状态。

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 设置被绑定实例id，可多个
 instanceIds := []string{"i-***", "i-***"}
 // 设置被绑定密钥对id
 keypairId := "k-***"
 args := &api.AttackKeypairArgs{
  InstanceIds: instanceIds,
  KeypairId:   keypairId,
 }
 err := bccClient.AttachKeypair(args)
 fmt.Println(err)
}
```

解绑密钥对

如下代码可以将所选BCC实例与它们各自绑定的密钥对解绑。 目前一台BCC实例只能绑定一个密钥对，此操作将使所选BCC实例与其通过百度云控制台操作所绑定的密钥对解绑，若该虚机不存在这种密钥对，则不进行处理。 此操作仅适用于linux系统的BCC实例，且所选BCC实例必须处于运行中或关机状态。 注： 1）用户所选BCC实例可能并未绑定任何密钥对，则此操作对该BCC实例无任何作用； 2）用户可能对所选BCC实例手动绑定了密钥对，则此操作对其手动绑定的密钥对不产生任何影响； 3）用户如若此前通过百度云控制台操作，为所选BCC实例绑定过密钥对，且该密钥对状态正常，则此操作将从该BCC实例中删除该密钥对。

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 待解绑的虚机的短id列表
 instanceIds := []string{"i-***", "i-***"}
 // 待解绑的密钥对id
 keypairId := "k-***"
 args := &api.DetachKeypairArgs{
  InstanceIds: instanceIds,
  KeypairId:   keypairId,
 }
 // 发起解绑
 err := bccClient.DetachKeypair(args)
 fmt.Println(err)
}
```

删除密钥对

如下代码用于删除密钥对，已经绑定BCC实例的密钥对无法被删除：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 设置要删除的密钥对id
 keypairId := "k-***"
 // 构建请求体
 args := &api.DeleteKeypairArgs{
  KeypairId: keypairId,
 }
 err := bccClient.DeleteKeypair(args)
 fmt.Println(err)
}
```

查询密钥对详情

如下代码用于查询单个密钥对的详细信息：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 设置被查询密钥对id
 keypairId := "k-***"
 res, err := bccClient.GetKeypairDetail(keypairId)
 fmt.Println(res, err)
}
```

查询密钥对列表

如下代码用于查询密钥对列表：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.ListKeypairArgs{
  // 批量获取列表的查询的起始位置，是一个由系统生成的字符串
  Marker:  "k-***",
  // 每页包含的最大数量，最大数量通常不超过1000。缺省值为1000
  MaxKeys: 1,
 }
 res, err := bccClient.ListKeypairs(args)
 fmt.Println(res, err)
}
```

重命名密钥对

如下代码用于重命名密钥对：

```
 package main
import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 新的密钥对名称
 name := "Name3"
 // 待重命名的密钥对id
 keypairId := "k-***"
 args := &api.RenameKeypairArgs{
  Name:      name,
  KeypairId: keypairId,
 }
 err := bccClient.RenameKeypair(args)
 fmt.Println(err)
}
```

更改密钥对描述

如下代码用于更改密钥对描述，若用户提供的新的描述内容为空，则删除所操作虚机的描述：

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
 // 设置您的ak、sk和要访问的endpoint
 ak := "ak"
 sk := "sk"
 endpoint := "http://bcc.bj.baidubce.com"
 // 创建bcc client
 bccClient, _ := bcc.NewClient(ak, sk, endpoint)
 // 设置要更新的密钥对描述
 description := "DESC"
 // 设置要更新描述的密钥对id
 keypairId := "k-***"
 args := &api.KeypairUpdateDescArgs{
  Description: description,
  KeypairId:   keypairId,
 }
 // 更新keypair的描述信息
 err := bccClient.UpdateKeypairDescription(args)
 fmt.Println(err)
}
```

部署集

创建部署集

通过以下代码可以根据指定的部署集策略创建部署集。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    CreateDploySetArgs := &api.CreateDeploySetArgs{
  // 部署集策略，默认为HOST_HA
  Strategy:        "HOST_HA",
  // 部署集名称
  Name:            "testDeploySetName",
  // 部署集描述
  Desc:            "test Deploy Desc",
  // 部署集并发度
  Concurrency:     2,
 }
 result, err := bccClient.CreateDeploySet(CreateDploySetArgs)

    fmt.Println(err)
    fmt.Println(result)
}
```

查询部署集列表

通过以下代码可以查询所有部署集实例的列表及详情信息。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    result, err := bccClient.ListDeploySets()

 fmt.Println(err)
 fmt.Println(result)
}
```

修改部署集属性

使用以下代码可以修改指定部署集的属性。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    DeployId := "dset-***"
 ModifyDeploySetName := "ModifyName"
 ModifyDeployDesc := "Modify Desc"
 queryArgs := &api.ModifyDeploySetArgs{
  // 部署集名称
  Name: ModifyDeploySetName,
  // 部署集描述
  Desc: ModifyDeployDesc,
 }
 result, err := bccClient.ModifyDeploySet(DeployId, queryArgs)

 fmt.Println(err)
 fmt.Println(result)
}
```

删除指定的部署集

使用以下代码删除用户自己的指定的部署集。

```
 package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.bj.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    DeployId := "dset-***"
    result := bccClient.DeleteDeploySet(DeployId)
    fmt.Println(result)
}
```

绑定指定的部署集

使用以下代码绑定用户自己指定的部署集。

```
 queryArgs := &api.UpdateInstanceDeployArgs{
      // 设置你要绑定的InstanceId
 InstanceId:   "InstanceId",
      // 设置你要绑定的DeploySetIds
 DeploySetIds: []string{"DeploySetId"},
}
rep, err := BCC_CLIENT.UpdateInstanceDeploySet(queryArgs)
fmt.Println(rep)
ExpectEqual(t.Errorf, err, nil)
```

解绑指定的部署集

使用以下代码解绑用户自己指定的部署集。

```
 queryArgs := &api.DelInstanceDeployArgs{
      // 设置你要解绑的DeploySetId
 DeploySetId: "DeploySetId",
      // 设置你要解绑的InstanceIds
 InstanceIds: []string{"InstanceId"},
}
rep, err := BCC_CLIENT.DelInstanceDeploySet(queryArgs)
fmt.Println(rep)
ExpectEqual(t.Errorf, err, nil)
```

磁盘专属集群

创建磁盘专属集群

以下代码可以创建磁盘专属集群：

```
 args := &CreateVolumeClusterArgs{
   // 创建一个磁盘磁盘专属集群，若要同时创建多个，可以修改此参数
PurchaseCount:   1,
// 集群大小,支持最小容量:85TB（87040GB）,支持最大容量：1015TB（1039360GB）,购买步长：10TB
ClusterSizeInGB: 97280,
   // 集群名称
ClusterName:    "dbsc",
// 集群磁盘类型：通用型HDD，通用型SSD
StorageType:    StorageTypeHdd,
Billing: &Billing{
      // 只支持预付费
 Reservation: &Reservation{
         // 购买时长
  ReservationLength:   6,
  ReservationTimeUnit: "MONTH",
 },
},
   // 自动续费时长
 RenewTimeUnit: "MONTH",
 RenewTime:    6,
}
result, err := DBSC_CLIENT.CreateVolumeCluster(args)
if err != nil {
 fmt.Println(err)
}
clusterId := result.ClusterIds[0]
fmt.Print(clusterId)
```

磁盘专属集群列表

以下代码可以根据专属集群ID批量查询专属集群列表：

```
 args := &ListVolumeClusterArgs{
}
result, err := DBSC_CLIENT.ListVolumeCluster(args)
if err != nil {
 fmt.Println(err)
}
fmt.Println(result)
```

磁盘专属集群详情

以下代码可以根据专属集群ID批量查询专属集群列表：

```
 clusterId := "clusterId"
result, err := DBSC_CLIENT.GetVolumeClusterDetail(clusterId)
if err != nil {
 fmt.Println(err)
}
fmt.Println(result)
```

磁盘专属集群扩容

以下代码可以给磁盘专属集群扩容：

```
 clusterId := "clusterId"
args := &ResizeVolumeClusterArgs{
 NewClusterSizeInGB int  `json:"newClusterSizeInGB"`
}
err := DBSC_CLIENT.ResizeVolumeCluster(clusterId, args)
if err != nil {
 fmt.Println(err)
}
```

磁盘专属集群续费

以下代码可以给专属集群续费：

```
 args := &PurchaseReservedVolumeClusterArgs{
 Billing: &Billing{
  Reservation: &Reservation{
          // 续费时长
   ReservationLength:   6,
   ReservationTimeUnit: "MONTH",
  },
 },
}
clusterId := "clusterId"
err := DBSC_CLIENT.PurchaseReservedVolumeCluster(clusterId, args)
if err != nil {
 fmt.Println(err)
}
```

磁盘专属集群自动续费

以下代码可以给专属集群设置自动续费：

```
 clusterId := "clusterId"
args := &AutoRenewVolumeClusterArgs{
 ClusterId:    clusterId,
 RenewTime:    6,
 RenewTimeUnit: "month",
}
err := DBSC_CLIENT.AutoRenewVolumeCluster(args)
if err != nil {
 fmt.Println(err)
}
```

磁盘专属集群取消自动续费

以下代码可以取消磁盘专属集群自动续费：

```
 clusterId := "clusterId"
args := &CancelAutoRenewVolumeClusterArgs{
 ClusterId: clusterId,
}
err := DBSC_CLIENT.CancelAutoRenewVolumeCluster(args)
if err != nil {
 fmt.Println(err)
}
```

预留实例券

查询预留实例券列表

使用以下代码查询预留实例券列表

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.ListReservedInstanceArgs{
       ZoneName: "cn-bj-a",
  MaxKeys:  10,
       Marker: "r-*****",
 }

 result, err := client.ListReservedInstances(args)
 if err != nil {
  panic(err)
 }

 fmt.Println(result)
}
```

创建预留实例券

使用以下代码创建预留实例券

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.CreateReservedInstanceArgs{
  ClientToken:              "myClientToken",
  ReservedInstanceName:     "myReservedInstance",
  Scope:                    "AZ",
  ZoneName:                 "cn-bj-a",
  Spec:                     "bcc.g5.c2m8",
  OfferingType:             "FullyPrepay",
  InstanceCount:            1,
  ReservedInstanceCount:    1,
  ReservedInstanceTime:     365,
  ReservedInstanceTimeUnit: "month",
  AutoRenewTimeUnit:        "month",
  AutoRenewTime:            1,
  AutoRenew:                true,
  Tags: []api.Tag{
   {
    TagKey:   "Env",
    TagValue: "Production",
   },
  },
  EhcClusterId: "ehcCluster123",
  TicketId:     "ticket456",
 }

 result, err := client.CreateReservedInstance(args)
 if err != nil {
  panic(err)
 }

 fmt.Println(result)
}
```

修改预留实例券

使用以下代码修改预留实例券属性

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)

 args := &api.ModifyReservedInstancesArgs{
  ReservedInstances: []api.ModifyReservedInstance{
   {
    ReservedInstanceId:   "r-UBVQFB5b",
    ReservedInstanceName: "update-reserved-instance",
   },
  },
 }

 result, err := client.ModifyReservedInstances(args)
 if err != nil {
  panic(err)
 }

 fmt.Println(result)
}
```

绑定标签

使用以下代码给指定预留实例券绑定标签

```
package main

import (
 "github.com/baidubce/bce-sdk-go/model"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 client, _ := bcc.NewClient(ak, sk, endpoint) // 创建BCC Client
 args := &api.ReservedTagsRequest{
        // 标签列表
  ChangeTags: []model.TagModel{
   {
    TagKey:   "TagKey-go",
    TagValue: "TagValue",
   },
  },
        // 预留实例券id列表，最多支持100个
  ReservedInstanceIds: []string{
   "r-oFpMXKhv", "r-HrztSVk0",
  },
 }
 err := client.BindReservedInstanceToTags(args)
 if err != nil {
  panic(err)
 }
}
```

解绑标签

使用以下代码从指定预留实例券中解绑标签

```
package main

import (
 "github.com/baidubce/bce-sdk-go/model"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 client, _ := bcc.NewClient(ak, sk, endpoint) // 创建BCC Client
 args := &api.ReservedTagsRequest{
        // 标签列表
  ChangeTags: []model.TagModel{
   {
    TagKey:   "TagKey-go",
    TagValue: "TagValue",
   },
  },
        // 预留实例券id列表，最多支持100个
  ReservedInstanceIds: []string{
   "r-oFpMXKhv", "r-HrztSVk0",
  },
 }
 err := client.UnbindReservedInstanceFromTags(args)
 if err != nil {
  panic(err)
 }
}
```

续费预留实例券

使用以下代码续费预留实例券

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.RenewReservedInstancesArgs{
        // 客户端Token
        ClientToken: "myClientToken",
        // 预留实例券ids
        ReservedInstanceIds: []string{
            "test-renew",
        },
        // 预留实例券续费时长，支持3，6，9，12，24，36个月
        ReservedInstanceTime:     1,
        // 预留实例券购买时长单位，默认为month，不可变更
        ReservedInstanceTimeUnit: "month",
        // 预留实例券自动续费时长单位，默认为month，不可变更
        AutoRenewTimeUnit:        "month",
        // 预留实例券自动续费时长，支持3，6，9，12，24，36个月，autoRenew为true时，必选且必须与
reservedInstanceTime一致
        AutoRenewTime:            1,
        // 自动续费开关，默认为false
        AutoRenew:                true,
    }

 result, err := client.RenewReservedInstances(args)
 if err != nil {
  panic(err)
 }

 fmt.Println(result)
}
```

地域

获取域名列表接口

使用以下代码可以获取全部地域或者指定地域的BCC(或CDS、预留实例券)域名列表：

```
package main

import (
    "fmt"
    "github.com/baidubce/bce-sdk-go/services/bcc"
    "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {
    // 设置您的ak、sk和要访问的endpoint
    ak := "ak"
    sk := "sk"
    endpoint := "bcc.baidubce.com"
    // 创建bcc client
    bccClient, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.DescribeRegionsArgs{
        // 地域,为空时查询所有地域的BCC(CDS、预留实例券)域名列表
        Region: "",
    }

    result, err := bccClient.DescriceRegions(args)
    fmt.Println(err)
    fmt.Println(result)
}
```

弹性高性能计算集群

创建EHC高性能集群

使用以下代码创建EHC高性能集群

```
package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)
 args := &api.CreateEhcClusterArg{
 Name:        "test-ehcCluster",
 ZoneName:    "cn-bj-a",
 Description: "test description",
 }
 result, _ := BCC_CLIENT.CreateEhcCluster(args)
 fmt.Println(result)
}
```

查询EHC高性能集群列表

使用以下代码查询EHC高性能集群列表

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)

    args := &api.DescribeEhcClusterListArg{
        EhcClusterIdList: []string{
        "ehc-bk4hM1N3",
        }, NameList: []string{
        "test-modify",
        },
        ZoneName: "cn-bj-a",
        SortKey:  "name",
        SortDir:  "asc",
    }
    result, err := BCC_CLIENT.ListEhcCluster(args)
 fmt.Println(result)
 fmt.Println(err)
}
```

修改EHC高性能集群

使用以下代码修改EHC高性能集群

```
 package main

import (
 "fmt"
 "github.com/baidubce/bce-sdk-go/services/bcc"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"
)

func main() {

 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 // 创建BCC Client
 client, _ := bcc.NewClient(ak, sk, endpoint)

    descriptions := ""
    args := &api.ModifyEhcClusterArg{
     EhcClusterId: "ehc-bk4hM1N3",
     Name:         "test-modify",
     Description:  &descriptions,
    }
    err := BCC_CLIENT.ModifyEhcCluster(args)
    fmt.Println(err)
}
```

删除EHC高性能集群

使用以下代码删除EHC高性能集群

```
 package main

import (
 "github.com/baidubce/bce-sdk-go/model"
 "github.com/baidubce/bce-sdk-go/services/bcc/api"

 "github.com/baidubce/bce-sdk-go/services/bcc"
)

func main() {
 // 初始化AK/SK/Endpoint
 ak, sk, endpoint := "ak", "sk", "bcc.bj.baidubce.com"
 client, _ := bcc.NewClient(ak, sk, endpoint) // 创建BCC Client
    args := &api.DeleteEhcClusterArg{
     EhcClusterIdList: []string{
      "ehc-tBmphmZE",
     },
    }
    err := BCC_CLIENT.DeleteEhcCluster(args)
    fmt.Println(err)
}
```

云助手API参考

服务域名

云助手API的服务域名为：

Region	域名	端口	协议
北京	ca.bj.baidubce.com	80	HTTP
广州	ca.gz.baidubce.com	80	HTTP
苏州	ca.su.baidubce.com	80	HTTP
保定	ca.bd.baidubce.com	80	HTTP
武汉	ca.fwh.baidubce.com	80	HTTP

批量获取实例bsm-agent状态

批量获取实例内bsm-agent状态

接口描述

批量获取最多100台虚机内bsm-agent的状态。

请求结构

```
POST /v1/ca/agent/batch
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
hosts	List<Host>	是	RequestBody参数	虚机列表

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
requestId	String	请求id
code	String	响应状态，成功为success
result	List<Agent>	虚机bsm-agent状态

请求示例

```
POST /v1/ca/agent/batch

{
   "hosts": [
      {
         "instanceId" : "i-LxvHe***"
      },
      {
         "instanceId" : "i-gVGDt***"
      }
   ],
}
```

响应示例

```
{
   "requestId": "d8752367-38e8-45e4-b4c7-e53be3******",
   "code": "success",
   "result": [
      {
         "host": {
            "instanceId": "i-LxvHe***"
         },
         "state": "ONLINE",
         "version": "0.0.0.1"
      },
      {
         "host": {
            "instanceId": "i-gVGDt***"
         },
         "state": "OFFLINE",
         "version": "0.0.0.1"
      }
   ]
}
```

附录-数据结构

Agent

参数名称	类型	描述
host	Host	实例信息
state	String	bsm-agent状态。枚举值：ONLINE（在线），OFFLINE（离线）
version	String	bsm-agent版本号，如4.2.0.1

Host

参数名称	类型	描述
instanceId	String	实例id，如bcc实例id: i-gVGDt***

命令接口

创建命令

接口描述

创建命令，支持保存、仅执行、执行并保存

请求结构

```
POST /v1/ca/action
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
execution	String	是	RequestBody参数	执行动作。枚举值：SAVE(仅保存），RUN（仅执行），SAVE_AND_RUN（保存并执行）
action	Action	是	RequestBody参数	执行动作详情
parameters	Map	否	RequestBody参数	执行命令时的参数值，仅在命令有参数且需要执行时需要
targetSelectorType	String	否	RequestBody参数	实例选择器类型，仅在执行动作为执行或保存并执行时需要。默认值为INSTANCES_LIST。可选值：INSTANCES_LIST（实例列表），ALL_INSTANCES（全部实例），TAG_INSTANCES（实例标签选择），INSTANCES_IMPORT（实例列表导入）
targets	List<Target>	否	Query参数	实例ID列表，仅在targetSelectorType为INSTANCES_LIST时需要
targetSelector	TargetSelector	否	Query参数	实例选择器

响应头域

除公共头域外，无其它特殊头域

响应参数

参数名称	类型	描述
actionId	String	命令id
actionType	String	动作类型。枚举值：COMMAND（脚本命令），FILE_UPLOAD（文件上传）
actionName	String	命令名称
commandType	String	脚本类型。枚举值： SHELL、POWERSHELL
version	int	命令版本号。从0开始，每次修改+1
runId	String	执行id，仅在命令执行时返回

请求示例

```
POST /v1/ca/action

{
    "execution": "RUN",
    "action": {
        "type": "COMMAND",
        "description": "this is a test",
        "timeoutSecond": 300,
        "command": {
            "type": "SHELL",
            "content": "ls",
            "scope": "INDIVIDUAL",
            "enableParameter": false,
            "user": "root",
            "workDir": "/home"
        }
    },
    "targets": [
        {
            "instanceType": "BCC",
            "instanceId": "i-p9L8X***"
        }
    ]
}
```

响应示例

```
{
    "requestId": "83e831f9-8f27-4c1c-bde4-fdfb********",
    "code": "success",
    "result": {
        "actionId": "c-cVtZHfVRD8******",
        "actionType": "COMMAND",
        "commandType": "SHELL",
        "version": 0,
        "runId": "r-h5fTUn2sab******"
    }
}
```

删除命令

接口描述

删除已保存的命令

请求结构

```
DELETE /v1/ca/action/{id}
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
id	String	是	Path参数	命令id

响应头域

除公共头域外，无其它特殊头域

响应参数

参数名称	类型	描述
requestId	String	请求id
code	String	响应状态，成功为success

请求示例

```
DELETE /v1/ca/action/c-cVtZHfVRD8******
```

响应示例

```
{
  "requestId": "b7d01b02-40ea-4ed2-9969-925790******",
  "code": "success",
}
```

查询命令详情

接口描述

查询已保存的命令或公共命令

请求结构

```
GET /v1/ca/action
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
id	String	是	Query参数	命令ID

响应头域

除公共头域外，无其它特殊头域

响应参数

参数名称	类型	描述
requestId	String	请求ID
code	String	响应状态，成功为success
result	Action	命令详情

请求示例

```
GET /v1/ca/action?id=c-cVtZHfVRD8******
```

响应示例

```
{
  "requestId" : "8f713916-d41b-472a-a06b-8a80cb******",
  "code": "success",
  "result": {
      "id": "a-CwkIRFAaoUCx",
      "name": "test_command",
      "description": "安装和升级HAS-Agent到最新版本",
      "timeoutSecond": 300,
      "type": "COMMAND",
      "command": {
         "type": "SHELL",
         "content": "",
         "enableParameter": false,
         "parameters": [
            {
               "name": "参数名",
               "desc" : "desc"
            }
         ],
         "user": "work",
         "workDir": "/home/work/"
      },
      "createdTimestamp": 1659534994121,
      "updatedTimestamp": 1659534994121
  }
}
```


修改我的命令

接口描述

修改我的命令

请求结构

```
PUT /v1/ca/action
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
action	UpdateActionRequest	是	RequestBody参数	执行动作详情

UpdateActionRequest | 参数名称| 类型 | 是否必须 | 参数位置 | 描述 | ---- | ---- | ---- | ---- | ---- | ref | String | 是 | RequestBody参数 | 需要修改的命令ID | name | String | 是 | RequestBody参数 | 修改后的命令名称 | description | String | 是 | RequestBody参数 | 修改后的命令描述 | timeoutSecond | int | 是 | RequestBody参数 | 修改后命令的超时时间（秒） | command | Command | 是 | RequestBody参数 | 修改后的命令

响应头域

除公共头域外，无其它特殊头域

响应参数

参数名称	类型	描述
requestId	String	请求id
code	String	响应状态，成功为success

请求示例

```
PUT /v1/ca/action

{
    "action": {
        "ref": "c-WgEtWtD3sV******",
        "name": "cmd_2024_03_21_16_41_32",
        "description": "this is a test",
        "timeoutSecond": 60,
        "command": {
            "type": "SHELL",
            "scope": "INDIVIDUAL",
            "enableParameter": true,
            "parameters": [
                {
                    "name": "param1",
                    "desc": "this is a text"
                }
            ],
            "user": "root",
            "workDir": "/home",
            "content": "{{param1}}"
        }
    }
}
```

响应示例

```
{
    "requestId": "be82abc6-eb4c-4a51-9f52-f59b97******",
    "code": "success"
}
```

命令列表查询

接口描述

查询我的命令列表或公共命令列表

请求结构

```
POST /v1/ca/action/list
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
action	ActionFilter	是	RequestBody参数	命令过滤器
pageNo	int	是	RequestBody参数	页码
pageSize	int	是	RequestBody参数	页大小
sort	String	否	RequestBody参数	排序字段，可选值createTime（命令创建时间）
ascending	bool	否	RequestBody参数	是否升序，默认false

ActionFilter

参数名称	类型	是否必须	参数位置	描述
command	CommandFilter	是	RequestBody参数	执行动作详情

CommandFilter

参数名称	类型	是否必须	参数位置	描述
scope	String	是	RequestBody参数	按命令可见范围筛选。枚举值：INDIVIDUAL（个人命令），GLOBAL（公共命令）
type	String	否	RequestBody参数	按命令类型筛选。枚举值：SHELL，POWERSHELL
name	String	否	RequestBody参数	按命令名称筛选

响应头域

除公共头域外，无其它特殊头域

响应参数

公共响应体中的result如下：

参数名称	类型	描述
pageNo	int	页码
pageSize	int	页大小
totalCount	int	总数
data	List<Action>	命令列表

请求示例

```
POST /v1/ca/action/list

{
    "action": {
        "type": "COMMAND",
        "command": {
            "scope": "INDIVIDUAL",
            "name": "test"
        }
    },
    "sort": "createTime",
    "ascending": false,
    "pageNo": 1,
    "pageSize": 10
}
```

响应示例

```
{
  "requestId": "ca1316b5-3fab-4c91-aeac-ad51fd******",
  "code": "success",
  "result": {
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 1,
    "data": [
      {
        "id": "c-4CtIumYWlo******",
        "type": "COMMAND",
        "name": "cmd_test_test",
        "timeoutSecond": 30,
        "command": {
          "type": "SHELL",
          "content": "ls",
          "scope": "INDIVIDUAL",
          "enableParameter": false,
          "workDir": "/home",
          "user": "root"
        },
        "createdTimestamp": 1694672762000,
        "updatedTimestamp": 1694672762000
      }
    ]
  }
}
```


执行命令

接口描述

执行已创建的我的命令或公共命令

请求结构

```
POST /v1/ca/actionRun
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必须	参数位置	描述
action	Action	是	RequestBody参数	所要执行的命令，仅需填写id
parameters	Map	否	RequestBody参数	执行命令时的参数值，仅在命令有参数时需要
targetSelectorType	String	否	RequestBody参数	实例选择器类型。默认值为INSTANCES_LIST。INSTANCES_LIST（实例列表），ALL_INSTANCES（全部实例），TAG_INSTANCES（实例标签选择），INSTANCES_IMPORT（实例列表导入）
targets	List<Target>	否	Query参数	实例ID列表，仅在targetSelectorType为INSTANCES_LIST时需要
targetSelector	TargetSelector	否	Query参数	实例选择器

响应头域

除公共头域外，无其它特殊头域

响应参数

公共响应体中的result如下：

参数名称	类型	描述
actionId	String	命令ID
runId	String	执行ID

请求示例

```
POST /v1/ca/actionRun

{
    "action": {
        "ref": "c-WgEtWtD3sV******"
    },
    "parameters": {
        "param1": "pwd"
    },
    "targetSelectorType": "INSTANCES_LIST",
    "targets": [
        {
            "instanceType": "BCC",
            "instanceId": "i-EVLpS***"
        }
    ]
}
```

响应示例

```
{
  "requestId": "87c25fbd-21c9-4082-904a-0d8c85******",
  "code": "success",
  "result": {
    "runId": "r-2lnH1HsWXU******",
    "actionId": "c-WgEtWtD3sV******"
  }
}
```


附录

Action | 参数名称| 类型 | 描述 | ---- | ---- | ---- | id | String |命令ID，仅被保存的命令拥有 | type | String | Action类型。枚举值：COMMAND（命令），FILE_UPLOAD（上传文件） | name | String | 命令名称，仅被保存的命令拥有 | description | String | 动作描述 | timeoutSecond | int | 动作的超时时间（秒） | command | Command | 命令详情 | fileUpload | FileUpload | 文件上传详情 | createdTimestamp| long | 命令创建时间。unix时间戳（毫秒） | updateTimestamp| long | 命令最后一次被修改时间。unix时间戳（毫秒）

Command | 参数名称| 类型 | 描述 | ---- | ---- | ---- | type | String | 脚本类型。枚举值：SHELL，POWERSHELL | scope | String | 命令可见范围。GLOBAL表示公共命令，INDIVIDUAL表示个人命令 | content | long | 命令脚本内容 | enableParameter | bool | 命令是否包含参数 | parameters | List<Parameter> |命令参数列表 | user | String | 命令在虚机的执行用户 | workDir| int | 命令在虚机的执行路径 | execParams | Map<String, String> | 命令执行时的实际参数值

FileUpload | 参数名称| 类型 | 描述 | ---- | ---- | ---- | os | String | 操作系统。枚举值：LINUX，WINDOWS | filename| String | 文件名称 | filepath | String | 目标路径 | bosBucketName | String | bos桶名称 | bosFilePath | String | bos文件路径 | bosEtag | String | 文件唯一标识符 | overwrite| bool | 是否覆盖目标路径下的同名文件 | user| String | 用户，仅Linux文件上传需要 | group | String | 用户组，仅Linux文件上传需要 | mode| String | 文件权限，仅Linux文件上传需要

Parameter | 参数名称| 类型 | 描述 | ---- | ---- | ---- | name| String | 参数名称 | desc | String | 参数描述

TargetSelector

参数名称	类型	描述
instanceType	String	实例类型。枚举值：BCC，BBC
tags	List<Tag>	实例标签列表
importInstances	TargetImport	实例标签列表

Tag

参数名称	类型	描述
tagKey	String	标签Key
tagValue	String	标签Value

TargetImport

参数名称	类型	描述
keywordType	String	实例列表导入类型。枚举值：instanceId（通过实例ID导入），internalIp表示（通过实例内网导入）
instances	List<String>	实例清单列表

Target

参数名称	类型	描述
instanceType	String	实例类型。枚举值：BCC，BBC
instanceId	List<String>	实例ID列表

历史记录

查询执行历史记录详情

接口描述

查询某一命令执行记录或文件上传记录详情.

请求结构

```
GET /v1/ca/actionRun?id=r-KJoCY9kKa9******
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
id	String	是	Query参数	执行ID
pageNo	int	否	Query参数	返回子执行记录页码，默认值1
pageSize	int	否	Query参数	返回子执行记录页大小，默认值10，最大值100
childRunState	String	否	Query参数	根据子执行状态过滤子执行记录。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
requestId	String	执行详情
code	String	响应状态，成功为success
result	ActionRun	执行详情

请求示例

```
GET /v1/ca/actionRun?id=r-N0UrpENajU******
```

响应示例

```
{
  "requestId": "8b051243-0760-4776-b8b5-024309******",
  "code": "success",
  "result": {
    "id": "r-N0UrpENajU******",
    "state": "SUCCESS",
    "action": {
      "type": "COMMAND",
      "name": "auto_run_1711100283",
      "description": "",
      "timeoutSecond": 300,
      "command": {
        "type": "SHELL",
        "content": "echo 'hello world'",
        "scope": "INDIVIDUAL",
        "enableParameter": false,
        "workDir": "/",
        "execParams": {
          "name": "value"
        },
        "user": "root"
      },
      "createdTimestamp": 1711100284000,
      "updatedTimestamp": 1711100316000
    },
    "createdTimestamp": 1711100286000,
    "finishedTimestamp": 1711100286000,
    "children": [
      {
        "id": "d-vVnBqI******",
        "state": "SUCCESS",
        "target": {
          "instanceId": "i-mflg****",
          "instanceType": "BCC",
        },
        "createdTimestamp": 1711100286503,
        "finishedTimestamp": 1711100286576,
        "output": {
          "exitCode": 0,
          "stderr": "",
          "stdout": "hello world\n",
          "isTruncated": false
        }
      }
    ],
    "totalCount": 1
  }
}
```


查询执行记录列表

接口描述

查询命令执行记录或文件上传记录列表

请求结构

```
POST /v1/ca/actionRun/list
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
action	RunFilter	是	RequestBody参数	执行记录过滤器
pageNo	int	是	RequestBody参数	页码
pageSize	int	是	RequestBody参数	页大小
ascending	bool	否	RequestBody参数	是否根据执行开始时间升序排列，默认为false
state	string	否	RequestBody参数	根据执行状态过滤。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）
runId	string	否	RequestBody参数	根据执行ID过滤
startTime	long	否	RequestBody参数	时间筛选，Unix时间戳（毫秒）。执行开始时间 >= endTime,Unix时间戳（毫秒）
endTime	long	否	RequestBody参数	时间筛选，Unix时间戳（毫秒）。执行开始时间 <= endTime，Unix时间戳（毫秒）

RunFilter | 参数名称 | 类型 | 是否必选 | 参数位置 | 描述 |

| --- | ------ | --- | ------- | ------ |

| type | string | 是 | RequestBody参数 | 执行记录类型。枚举值：COMMAND（命令执行记录），FILE_UPLOAD（文件上传记录）| | id | string | 否 | RequestBody参数 |通过命令ID筛选，仅对命令执行记录有效| | name | string | 否 | RequestBody参数 |通过命令名称筛选，仅对命令执行记录有效| | fileUpload| FileUploadFilter | 否 | RequestBody参数 | 筛选命令上传记录 |

FileUploadFilter | 参数名称 | 类型 | 是否必选 | 参数位置 | 描述 |

| --- | ------ | --- | ------- | ------ |

| filename | string | 是 | RequestBody参数 | 通过上传文件名称筛选

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
requestId	String	执行详情
code	String	响应状态，成功为success
result	List<ActionRun>	执行详情

请求示例

```
 POST /v1/ca/actionRun/list
Host:

{
    "ascending": false,
    "pageNo": 1,
    "pageSize": 10,
    "action": {
        "name": "test",
        "type": "COMMAND"
    },
    "startTime": "1710777600000",
    "endTime": "1711382399999"
}
```

响应示例

```
{
  "requestId": "f5c8cbf2-b66b-4894-b98d-dae6cb******",
  "code": "success",
  "result": {
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 2,
    "data": [
      {
        "id": "r-QWR6CCMVj0******",
        "state": "SUCCESS",
        "action": {
          "type": "COMMAND",
          "timeoutSecond": 60,
          "command": {
            "type": "SHELL",
            "content": "ls",
            "scope": "INDIVIDUAL",
            "enableParameter": false,
            "workDir": "/home",
            "user": "root"
          }
        },
        "createdTimestamp": 1710837417000,
        "finishedTimestamp": 1710837419000,
        "statistics": {
          "totalTargetCount": 1,
          "failedTargetCount": 0
        }
      },
      {
        "id": "r-vufYpbTTyd******",
        "state": "SUCCESS",
        "action": {
          "id": "c-5iEoMgGqXL******",
          "type": "COMMAND",
          "name": "auto_test_1709896691",
          "description": "",
          "timeoutSecond": 300,
          "command": {
            "type": "SHELL",
            "content": "echo 'hello world'",
            "scope": "INDIVIDUAL",
            "enableParameter": false,
            "workDir": "/",
            "user": "root"
          }
        },
        "createdTimestamp": 1709896693000,
        "finishedTimestamp": 1709896697000,
        "statistics": {
          "totalTargetCount": 1,
          "failedTargetCount": 0
        }
      }
    ]
  }
}
```


附录

ActionRun

参数名称	类型	描述
id	String	执行ID
action	Action	所执行动作详情
state	String	执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成），PARTIAL_FAILED（部分失败）
createdTimestamp	long	执行开始时间。unix时间戳，单位：毫秒
finishedTimestamp	long	执行结束时间，仅执行结束时返回。unix时间戳，单位：毫秒。
statistics	Statistics	子执行执行情况，仅查询执行记录列表时返回

ChildRun

参数名称	类型	描述
id	String	子执行ID
state	String	执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）
createdTimestamp	long	子执行开始时间。unix时间戳，单位：毫秒
finishedTimestamp	long	子执行结束时间，仅执行结束时返回。unix时间戳，单位：毫秒。
failReason	String	子执行失败原因
output	Output	命令输出结果
Target	Target	子执行对应的实例信息

Output | 参数名称 | 类型 | 描述 |

| -------- | -------- | ---------- | | exitCode | int | 退出码 | | stdout| String| 标准输出 | | stderr | String | 标准错误 | | isTruncated | bool| 标准输出或标准错误是否由于过长（超过4KB）而被截断 |

Statistics | 参数名称 | 类型 | 描述 |

| -------- | -------- | ---------- | | totalTargetCount | int | 执行的实例总数 | | failedTargetCount| int| 已失败的执行实例总数 |

日志接口

日志接口

接口描述

获取命令执行的日志记录。

请求结构

```
POST /v1/ca/log
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

参数名称	类型	是否必选	参数位置	描述
runId	String	是	RequestBody参数	执行ID
childId	String	是	RequestBody参数	执行ID下某一子执行ID
cursor	int	是	RequestBody参数	首次请求为0，后续翻页，填写上一次请求返回的nextCursor

响应头域

除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
requestId	String	请求id
code	String	响应状态，成功为success
result	LogResult	返回日志结果

请求示例

```
POST /v1/ca/log

{
    "runId": "r-KJoCY9kKa9******",
    "childId": "d-ks3GQL******",
    "cursor": 0
}
```

响应示例

```
{
    "requestId": "aaf4bd8f-ed6c-481b-ad1a-a4c974******",
    "code": "success",
    "result": {
        "logs": [
            {
                "msg": "pwd \n",
            }, {
                "msg": "second line",
            }
        ],
        "nextCursor": 2,
        "state" : "RUNNING"
    }
}
```

附录

LogResult | 参数名称 | 类型 | 描述 | | ---------- | ------ | ---------- | | childId |String | 子执行id | | logs | List<Log> | 日志内容，顺序排列 | | state | String | 子执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成） | | nextCursor | int |下一次请求使用的cursor。若子执行已结束，且nextCursor=cursor，表示已无更多日志。否则，表示可能存在更多日志 |

Log

参数名称	类型	描述
msg	String	命令执行日志

错误返回

当用户访问API出现错误时，会返回给用户相应的错误码和错误信息，便于定位问题，并做出适当处理。请求发生错误时通过Response Body返回详细错误信息，遵循如下格式：

参数名	类型	说明
code	String	表示具体错误类型。
message	String	有关该错误的详细说明。
requestId	String	导致该错误的requestId

示例：

```
{
  "requestId": "3ffd9f45-6908-452d-9f4f-5fb1f5******",
  "code": "CommandMissing",
  "message": "Command is empty"
}
```

云助手业务错误码

错误码	错误消息	HTTP状态码	描述
CommandMissing	Command is empty	400	命令为空
CommandInvalidParamName	A command parameter name is invalid	400	命令参数名称不符合规范
CommandInvalidExecParams	The execution parameter does not match the command parameter	400	执行时参数与命令参数不匹配
CommandMismatchedParams	Mismatched parameter list and content	400	命令内容和参数不匹配
CommandNameAlreadyExisted	Command named {{name}} already existed	400	命令名称已存在
CommandTooManyParams	The number of parameters must be less than 20	400	参数数量过多
CommandInvalidContentLength	The command content length is invalid	400	命令内容长度不合法
CommandNameMissing	The command name missing	400	缺少命令名称
CommandNotFound	The command not found	404	命令不存在
EmptyFile	The file is empty	400	所上传文件为空
FileNameExceedLimit	The file name exceed the limit	400	文件名称过长
FileSizeExceedLimit	The file exceeds its maximum permitted size	400	文件大小超过上限
FileNotFound	The file does not exist	404	所下载的文件不存在
UnknownActionRunType	Unknown type of action run	400	未知的历史记录类型
UnknownInstanceSelector	TargetSelector is empty	400	未知的实例筛选方式
EmptyInstanceList	The target instance list is empty	400	目标实例列表为空
InstanceCountExceedLimit	The number of target instance exceeds the limit	400	目标实例数量超过上限
ActionRunNotFound	The action run not found	404	执行记录不存在
AgentCountExceedLimit	The number of agent exceed the limit	400	查询Agent数量超过上限

公共请求与响应头

公共请求头

公共头部	说明	是否必须
Authorization	请求认证签名信息，详细内容参见API认证机制	是

公共响应头

公共头部	说明
x-bce-request-id	请求唯一标识

通用说明

API调用遵循HTTP协议，各Region采用不同的域名，具体域名为ca.{region}.baidubce.com。

数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

API认证机制

所有API的安全认证一律采用Access Key与请求签名机制。 Access Key由Access Key ID和Secret Access Key组成，均为字符串（如何获取AK/SK，请查看获取AK/SK。 对于每个HTTP请求，使用下面所描述的算法生成一个认证字符串。提交认证字符串放在Authorization头域里。服务端根据生成算法验证认证字符串的正确性。 认证字符串的格式为 `bce-auth-v{version}/{accessKeyId}/{timestamp}/{expirationPeriodInSeconds}/{signedHeaders}/{signature}`。

• version是正整数。 • timestamp是生成签名时的UTC时间。 • expirationPeriodInSeconds表示签名有效期限。 • signedHeaders是签名算法中涉及到的头域列表。头域名之间用分号（;）分隔，如host;x-bce-date。列表按照字典序排列。（本API签名仅使用host和x-bce-date两个header） • signature是256位签名的十六进制表示，由64个小写字母组成。

当百度云接收到用户的请求后，系统将使用相同的SK和同样的认证机制生成认证字符串，并与用户请求中包含的认证字符串进行比对。如果认证字符串相同，系统认为用户拥有指定的操作权限，并执行相关操作；如果认证字符串不同，系统将忽略该操作并返回错误码。

鉴权认证机制的详细内容请参见鉴权认证。

通信协议

云助手 API所有region支持HTTP调用方式。

请求结构说明

数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

请求参数包括如下4种：

参数类型	说明
URL	通常用于指明操作实体,如:POST /v{version}/instance/{instanceId}
Query参数	URL中携带的请求参数
HEADER	通过HTTP头域传入,如:x-bce-date
RequestBody	通过JSON格式组织的请求数据体

响应结构说明

响应值分为两种形式：

返回内容	说明
HTTP STATUS CODE	如200,400,403,404等
ResponseBody	JSON格式组织的响应数据体

规范化字符串

通常一个字符串中可以包含任何Unicode字符。在编程中这种灵活性会带来不少困扰。因此引入“规范字符串”的概念。一个规范字符串只包含百分号编码字符以及URI（Uniform Resource Identifier）非保留字符（Unreserved Characters）。 RFC 3986规定URI非保留字符包括以下字符：字母（A-Z，a-z）、数字（0-9）、连字号（-）、点号（.）、下划线（_）、波浪线（~）。 将任意一个字符串转换为规范字符串的方式是:

- 将字符串转换成UTF-8编码的字节流。
- 保留所有URI非保留字符原样不变。
- 对其余字节做一次RFC 3986中规定的百分号编码（Percent-Encoding），即一个%后面跟着两个表示该字节值的十六进制字母。字母一律采用大写形式。 示例： 原字符串：`this is an example for` 测试, 对应的规范字符串：`this%20is%20an%20example%20for%20%E6%B5%8B%E8%AF%95`。

云助手SDK参考

Python SDK

安装SDK

安装SDK工具包

环境准备

1. 运行环境
   Python SDK工具包支持在Python 2.7 和Python 3 的环境运行。
2. 安装pycrypto依赖
   安装SDK之前，需要先执行命令`pip install pycrypto`安装pycrypto依赖。

下载和安装

方式一：通过pip安装

您可以通过pip安装的方式将百度智能云Python SDK安装到您的环境中。 联网状态下，在命令行中执行如下命令：

```
pip install bce-python-sdk
```

即可将Python SDK安装到本地。

方式二：将源码包下载到本地后进行安装

1. 在开发者资源中心下载Python SDK压缩工具包。
2. 命令行移动到压缩包所在路径，执行如下命令（version替换为包名称中的版本号）：
   pip install bce-python-sdk-version.zip
   即可将Python SDK安装到本地。

方式三：使用setup.py install文件进行安装

您也可以解压压缩包后执行如下命令（version替换为包名称中的版本号）

```
cd bce-python-sdk-version

python setup.py install
```

SDK目录结构

```
baidubce
    ├──── auth                      //公共权限目录
   ├──── http                     //Http请求模块
    ├──── services                   //服务公共目录
    │     └──── ca                   //CA目录
    │           ├────ca_client.py       //CA客户端入口类
    │           ├────ca_model.py        //CA内部model
    │           ├────ca_handler.py      //CA异常处理
```

卸载SDK

卸载 SDK 时，执行命令`pip uninstall bce-python-sdk`即可完成卸载。

初始化

初始化

确认Endpoint

用户若要自定义域名，可以通过传入ENDPOINT参数来指定。区域的概念请参考区域选择说明。

在开始SDK使用之前，需要您先确定好要在哪个区域进行操作，从而在配置CaClient时将区域对应的Endpoint做为参数填入。

CA目前支持的域名请参考服务域名。

获取密钥

要使用百度智能云产品，您需要拥有一个百度智能云账号和一个有效的 AK(Access Key ID)、SK(Secret Access Key)用来进行签

名认证。

可以通过如下步骤获得并了解您的AK/SK信息：

1.注册百度智能云账号

2.创建AK/SK

获取到密钥后，需要在配置Client时做为参数填入。SDK集成了鉴权认证机制，您不需要关心鉴权背后的运算方法，只需要将AK/SK按要求填入对应的位置，SDK将自动为您完成鉴权相关的工作。

CaClient

新建CaClient

CaClient是CA服务的客户端，为开发者与CA服务进行交互提供了一系列的方法。

通过AK/SK方式访问CA

1.在新建CaClient之前，需要先创建配置文件对CaClient进行配置，以下将此配置文件命名为`ca_sample_conf.py`，具体配置信息如下所示：

```
 #!/usr/bin/env python
#coding=utf-8

#导入Python标准日志模块
import logging

#从Python SDK导入BCE配置管理模块以及安全认证模块
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.auth.bce_credentials import BceCredentials

#设置CaClient的Host，Access Key ID和Secret Access Key
ca_host = "ca.bj.baidubce.com"
access_key_id = "AK"
secret_access_key = "SK"

#设置日志文件的句柄和日志级别
logger = logging.getLogger('baidubce.http.bce_http_client')
fh = logging.FileHandler("sample.log")
fh.setLevel(logging.DEBUG)

#设置日志文件输出的顺序、结构和内容
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
fh.setFormatter(formatter)
logger.setLevel(logging.DEBUG)
logger.addHandler(fh)

#创建BceClientConfiguration
config = BceClientConfiguration(credentials=BceCredentials(access_key_id, secret_access_key), endpoint = ca_host)
```

> 注意： 针对日志文件，Logging有如下级别：DEBUG，INFO，WARNING，ERROR，CRITICAL。

在上面代码中，access_key_id对应控制台中的“Access Key ID”，secret_access_key对应控制台中的“Access Key Secret”，获取方式请参考管理ACCESSKEY。

上面的方式用户需要自己指定CA务的域名，可以通过赋值给`ca_host`变量来指定。 2.在完成上述配置之后，参考如下代码新建一个CaClient。

```
 #导入CaClient配置文件
import ca_sample_conf

#导入CA相关模块
from baidubce import exception
from baidubce.services import ca
from baidubce.services.ca.ca_client import CaClient

#新建CaClient
ca_client = CaClient(ca_sample_conf.config)
```

批量获取实例内bsm-agent状态

批量获取实例内bsm-agent状态

接口描述

批量获取最多100台虚机内bsm-agent的状态。

请求参数

参数名称	类型	是否必选	描述
instance_list	List<Instance>	是	虚机列表

响应参数 | 参数名称 | 类型 | 描述 | | -------- | ---------- | ---------- | | requestId | String | 请求id | | code | String | 响应状态，成功为success | | result | List<Agent> | 虚机bsm-agent状态 |

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)


    instance_list = [
        ca_model.Instance("i-9y7wPdlG"),
        ca_model.Instance("i-ZCHupg0z")
    ]
    response = client.batch_get_agent(instance_list)
    print(response)
```

响应示例

```
{
  "requestId": "d8752367-38e8-45e4-b4c7-e53be3******",
  "code": "success",
  "result": [
    {
      "host": {
        "instanceId": "i-LxvHe***"
      },
      "state": "ONLINE",
      "version": "0.0.0.1"
    },
    {
      "host": {
        "instanceId": "i-gVGDt***"
      },
      "state": "OFFLINE",
      "version": "0.0.0.1"
    }
  ]
}
```


附录-数据结构

Agent

参数名称	类型	描述
host	Host	实例信息
state	String	bsm-agent状态。枚举值：ONLINE（在线），OFFLINE（离线）
version	String	bsm-agent版本号，如4.2.0.1

Instance

参数名称	类型	描述
instanceId	String	实例id，如bcc实例id: i-gVGDt***

命令接口

创建命令

创建命令，支持保存、仅执行、执行并保存

请求参数

参数名称	类型	是否必须	描述
execution	String	是	执行动作。枚举值：SAVE(仅保存），RUN（仅执行），SAVE_AND_RUN（保存并执行）
action	Action	是	执行动作详情
parameters	Map	否	执行命令时的参数值，仅在命令有参数且需要执行时需要
targetSelectorType	String	否	实例选择器类型，仅在执行动作为执行或保存并执行时需要。默认值为INSTANCES_LIST。可选值：INSTANCES_LIST（实例列表），ALL_INSTANCES（全部实例），TAG_INSTANCES（实例标签选择），INSTANCES_IMPORT（实例列表导入）
targets	List<Target>	否	实例ID列表，仅在targetSelectorType为INSTANCES_LIST时需要
targetSelector	TargetSelector	否	实例选择器

请求

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)

    user_id = ''
    execution = 'SAVE_AND_RUN'
    action = ca_model.Action(
        "", "", "COMMAND", "yyy-python-test", 60,
        ca_model.Command(
            "SHELL", "ls", "INDIVIDUAL", False, [], "root", "/home"
        )
    )
    parameter = {}
    target_selector = ca_model.TargetSelector("BCC", [])
    target_selector_type = "INSTANCES_LIST"
    targets = [ca_model.Target("BCC", "i-kBdE8Tav")]
    response = client.create_action(execution, user_id, action, targets, parameter, target_selector,
                                    target_selector_type)
```

删除命令

删除已保存的命令

接口描述

参数名称	类型	是否必须	描述
id	String	是	命令id

响应参数 | 参数名称 | 类型 | 描述 | | -------- | ---------- | ---------- | | requestId | String | 请求id | | code | String | 响应状态，成功为success |

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = client.CaClient(config)

    id = "c-xadYvQ1FSA5viD4E"
    response = client.delete_action(id)
```

响应示例

```
 {
   "requestId": "b7d01b02-40ea-4ed2-9969-925790******",
   "code": "success",
}
```

查询命令详情

查询已保存的命令或公共命令

请求参数

参数名称	类型	是否必须	描述
id	String	是	命令ID
user_id	String	是	用户ID

响应参数 | 参数名称 | 类型 | 描述 | | --- | --- | --- | | requestId | String | 请求ID | | code | String | 响应状态，成功为success | | result | Action | 命令详情 |

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)

    id = "c-k7zXV794Jjz1WZC8"
    user_id = ""
    response = client.get_action(id, user_id)
    print(response)
```

响应示例

```
{
  "requestId" : "8f713916-d41b-472a-a06b-8a80cb******",
  "code": "success",
  "result": {
      "id": "a-CwkIRFAaoUCx",
      "name": "test_command",
      "description": "安装和升级HAS-Agent到最新版本",
      "timeoutSecond": 300,
      "type": "COMMAND",
      "command": {
         "type": "SHELL",
         "content": "",
         "enableParameter": false,
         "parameters": [
            {
               "name": "参数名",
               "desc" : "desc"
            }
         ],
         "user": "work",
         "workDir": "/home/work/"
      },
      "createdTimestamp": 1659534994121,
      "updatedTimestamp": 1659534994121
  }
}
```


修改我的命令

修改我的命令

请求参数

参数名称	类型	是否必须	描述
action	UpdateActionRequest	是	执行动作详情

UpdateActionRequest | 参数名称| 类型 | 是否必须 | 描述 | ---- | ---- | ---- | ---- | ref | String | 是 | 需要修改的命令ID | name | String | 是 | 修改后的命令名称 | description | String | 是 | 修改后的命令描述 | command | Command | 是 | 修改后的命令

响应参数 | 参数名称 | 类型 | 描述 | | --------- | ---------- | ---------- | | requestId | String | 请求id | | code | String | 响应状态，成功为success |

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)

    action = ca_model.Action(
        "c-k7zXV794Jjz1WZC8", "", "COMMAND", "yyy-python-test-update", 60,
        ca_model.Command(
            "SHELL", "ls", "INDIVIDUAL", False, [], "root", "/home"
        )
    )
    execution = "SAVE"
    response = client.update_action(action, execution)
    print(response)
```

响应示例

```
 {
   "requestId": "be82abc6-eb4c-4a51-9f52-f59b97******",
   "code": "success"
}
```

命令列表查询

查询我的命令列表或公共命令列表

请求参数

参数名称	类型	是否必须	描述
action	ActionFilter	是	命令过滤器
page_no	int	是	页码
page_size	int	是	页大小
sort	String	否	排序字段，可选值createTime（命令创建时间）
ascending	bool	否	是否升序，默认false

ActionFilter

参数名称	类型	是否必须	描述
command	CommandFilter	是	执行动作详情

CommandFilter

参数名称	类型	是否必须	参数位置	描述
scope	String	是	RequestBody参数	按命令可见范围筛选。枚举值：INDIVIDUAL（个人命令），GLOBAL（公共命令）
type	String	否	RequestBody参数	按命令类型筛选。枚举值：SHELL，POWERSHELL
name	String	否	RequestBody参数	按命令名称筛选

响应参数 公共响应体中的result如下：

参数名称	类型	描述
pageNo	int	页码
pageSize	int	页大小
totalCount	int	总数
data	List<Action>	命令列表

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)

    action = ca_model.Action(
        "", "", "COMMAND", "", 0,
        ca_model.Command(
            None, "", "INDIVIDUAL", False, {}, "", ""
        )
    )
    ascending = False
    page_no = 1
    page_size = 10
    sort = "createTime"
    user_id = ""

    response = client.action_list(action, page_no, page_size, sort, ascending, user_id)
    print(response)
```

响应示例

```
{
  "requestId": "ca1316b5-3fab-4c91-aeac-ad51fd******",
  "code": "success",
  "result": {
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 1,
    "data": [
      {
        "id": "c-4CtlumYWlo******",
        "type": "COMMAND",
        "name": "cmd_test_test",
        "timeoutSecond": 30,
        "command": {
          "type": "SHELL",
          "content": "ls",
          "scope": "INDIVIDUAL",
          "enableParameter": false,
          "workDir": "/home",
          "user": "root"
        },
        "createdTimestamp": 1694672762000,
        "updatedTimestamp": 1694672762000
      }
    ]
  }
}
```


执行命令

执行已创建的我的命令或公共命令

请求参数

参数名称	类型	是否必须	描述
action	Action	是	所要执行的命令，仅需填写id
parameters	Map	否	执行命令时的参数值，仅在命令有参数时需要
targetSelectorType	String	否	实例选择器类型。默认值为INSTANCES_LIST。INSTANCES_LIST（实例列表），ALL_INSTANCES（全部实例），TAG_INSTANCES（实例标签选择），INSTANCES_IMPORT（实例列表导入）
targets	List<Target>	否	实例ID列表，仅在targetSelectorType为INSTANCES_LIST时需要
targetSelector	TargetSelector	否	实例选择器

响应参数 公共响应体中的result如下：

参数名称	类型	描述
actionId	String	命令ID
runId	String	执行ID

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':
    HOST = b'http://ca.bj.baidubce.com'
    AK = b'ak'
    SK = b'sk'

    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),endpoint=HOST)
    client = ca_client.CaClient(config)

    user_id = ''
    action = ca_model.Action(
        "c-4cMSvlmdfWp2GdA9"
    )
    parameter = {}
    target_selector = ca_model.TargetSelector("BCC", [])
    target_selector_type = "INSTANCES_LIST"
    targets = [ca_model.Target("BCC", "i-ZCHupg0z")]
    response = client.action_run(action, parameter, user_id, target_selector_type, targets, target_selector)
    print(response)
```

响应示例

```
 {
   "requestId": "87c25fbd-21c9-4082-904a-0d8c85******",
   "code": "success",
   "result": {
      "runId": "r-2lnH1HsWXU******",
      "actionId": "c-WgEtWtD3sV******"
   }
}
```

附录

Action | 参数名称| 类型 | 描述 | ---- | ---- | ---- | id | String |命令ID，仅被保存的命令拥有 | type | String | Action类型。枚举值：COMMAND（命令），FILE_UPLOAD（上传文件） | name | String | 命令名称，仅被保存的命令拥有 | description | String | 动作描述 | timeoutSecond | int | 动作的超时时间（秒） | command | Command | 命令详情 | fileUpload | FileUpload | 文件上传详情 | createdTimestamp| long | 命令创建时间。unix时间戳（毫秒） | updateTimestamp| long | 命令最后一次被修改时间。unix时间戳（毫秒）

Command | 参数名称| 类型 | 描述 | ---- | ---- | ---- | type | String | 脚本类型。枚举值：SHELL，POWERSHELL | scope | String | 命令可见范围。GLOBAL表示公共命令，INDIVIDUAL表示个人命令 | content | long | 命令脚本内容 | enableParameter | bool | 命令是否包含参数 | parameters | List<Parameter> |命令参数列表 | user | String | 命令在虚机的执行用户 | workDir| int | 命令在虚机的执行路径 | execParams | Map<String, String> | 命令执行时的实际参数值

FileUpload | 参数名称| 类型 | 描述 | ---- | ---- | ---- | os | String | 操作系统。枚举值：LINUX，WINDOWS | filename| String | 文件名称 | filepath | String | 目标路径 | bosBucketName | String | bos桶名称 | bosFilePath | String | bos文件路径 | bosEtag | String | 文件唯一标识符 | overwrite| bool | 是否覆盖目标路径下的同名文件 | user| String | 用户，仅Linux文件上传需要 | group | String | 用户组，仅Linux文件上传需要 | mode| String | 文件权限, 仅Linux文件上传需要

Parameter | 参数名称| 类型 | 描述 | ---- | ---- | ---- | name| String | 参数名称 | desc | String | 参数描述

TargetSelector

参数名称	类型	描述
instanceType	String	实例类型。枚举值：BCC，BBC
tags	List<Tag>	实例标签列表
importInstances	TargetImport	实例标签列表

Tag

参数名称	类型	描述
tagKey	String	标签Key
tagValue	String	标签Value

TargetImport

参数名称	类型	描述
keywordType	String	实例列表导入类型。枚举值：instanceId（通过实例ID导入），internalIp表示（通过实例内网导入）
instances	List<String>	实例清单列表

Target

参数名称	类型	描述
instanceType	String	实例类型。枚举值：BCC，BBC
instanceId	List<String>	实例ID列表

历史记录

查询执行历史记录详情

查询某一命令执行记录或文件上传记录详情

请求参数

参数名称	类型	是否必选	描述
id	String	是	执行ID
pageNo	int	否	返回子执行记录页码，默认值1
pageSize	int	否	返回子执行记录页大小，默认值10，最大值100
childRunState	String	否	根据子执行状态过滤子执行记录。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）

响应参数

参数名称	类型	描述
requestId	String	执行详情
code	String	响应状态，成功为success
result	ActionRun	执行详情

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model
if __name__ == '__main__':

    HOST = b'http://ca.bj.baidubce.com'
    AK = b''
    SK = b''
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    ca_client = CaClient(config)
    user_id = ''
    id = 'r-raUPWwr933Liqe54'
    response = ca_client.get_action_run(id, user_id)
    print(response)
```

响应示例

```
{
  "requestId": "8b051243-0760-4776-b8b5-024309******",
  "code": "success",
  "result": {
    "id": "r-N0UrpENajU******",
    "state": "SUCCESS",
    "action": {
      "type": "COMMAND",
      "name": "auto_run_1711100283",
      "description": "",
      "timeoutSecond": 300,
      "command": {
        "type": "SHELL",
        "content": "echo 'hello world'",
        "scope": "INDIVIDUAL",
        "enableParameter": false,
        "workDir": "/",
        "execParams": {
          "name": "value"
        },
        "user": "root"
      },
      "createdTimestamp": 1711100284000,
      "updatedTimestamp": 1711100316000
    },
    "createdTimestamp": 1711100286000,
    "finishedTimestamp": 1711100286000,
    "children": [
      {
        "id": "d-vVnBqI******",
        "state": "SUCCESS",
        "target": {
          "instanceId": "i-mflg****",
          "instanceType": "BCC",
        },
        "createdTimestamp": 1711100286503,
        "finishedTimestamp": 1711100286576,
        "output": {
          "exitCode": 0,
          "stderr": "",
          "stdout": "hello world\n",
          "isTruncated": false
        }
      }
    ],
    "totalCount": 1
  }
}
```


查询执行记录列表

查询命令执行记录或文件上传记录列表

请求参数

参数名称	类型	是否必选	描述
action	RunFilter	是	执行记录过滤器
pageNo	int	是	页码
pageSize	int	是	页大小
ascending	bool	否	是否根据执行开始时间升序排列，默认为false
state	string	否	根据执行状态过滤。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）
runId	string	否	根据执行ID过滤
startTime	long	否	时间筛选，Unix时间戳（毫秒）。执行开始时间 >= endTime,Unix时间戳（毫秒）
endTime	long	否	时间筛选，Unix时间戳（毫秒）。执行开始时间 <= endTime，Unix时间戳（毫秒）

RunFilter | 参数名称 | 类型 | 是否必选 | 参数位置 | 描述 |
| --- | ------ | ---- | ------- | ------ |
| type | string | 是 | RequestBody参数 | 执行记录类型。枚举值：COMMAND（命令执行记录），FILE_UPLOAD（文件上传记录）| | id | string | 否 | RequestBody参数 |通过命令ID筛选，仅对命令执行记录有效| | name | string | 否 | RequestBody参数 |通过命令名称筛选，仅对命令执行记录有效| | fileUpload| FileUploadFilter | 否 | RequestBody参数 | 筛选命令上传记录 |

FileUploadFilter | 参数名称 | 类型 | 是否必选 | 参数位置 | 描述 |
| --- | ------ | ---- | ------- | ------ |
| filename | string | 是 | RequestBody参数 | 通过上传文件名称筛选

响应头域 除公共头域外，无其它特殊头域。

响应参数

参数名称	类型	描述
requestId	String	执行详情
code	String	响应状态，成功为success
result	List<ActionRun>	执行详情

请求示例

```
 from baidubce.auth.bce_credentials import BceCredentials
from baidubce.bce_client_configuration import BceClientConfiguration
from baidubce.services.ca import ca_client, ca_model

if __name__ == '__main__':

    HOST = b'http://ca.bj.baidubce.com'
    AK = b''
    SK = b''
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    ca_client = CaClient(config)
    action = ca_model.Action(
        "", "", "COMMAND"
    )
    ascending = False
    page_no = 1
    page_size = 10
    sort = "createTime"
    user_id = ""
    response = ca_client.action_run_list(action, page_no, page_size, sort, ascending, user_id)
    self.assertEqual(type(response), baidubce.bce_response.BceResponse)
    print(response)
```

响应示例

```
{
  "requestId": "f5c8cbf2-b66b-4894-b98d-dae6cb******",
  "code": "success",
  "result": {
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 2,
    "data": [
      {
        "id": "r-QWR6CCMVj0******",
        "state": "SUCCESS",
        "action": {
          "type": "COMMAND",
          "timeoutSecond": 60,
          "command": {
            "type": "SHELL",
            "content": "ls",
            "scope": "INDIVIDUAL",
            "enableParameter": false,
            "workDir": "/home",
            "user": "root"
          }
        },
        "createdTimestamp": 1710837417000,
        "finishedTimestamp": 1710837419000,
        "statistics": {
          "totalTargetCount": 1,
          "failedTargetCount": 0
        }
      },
      {
        "id": "r-vufYpbTTyd******",
        "state": "SUCCESS",
        "action": {
          "id": "c-5iEoMgGqXL******",
          "type": "COMMAND",
          "name": "auto_test_1709896691",
          "description": "",
          "timeoutSecond": 300,
          "command": {
            "type": "SHELL",
            "content": "echo 'hello world'",
            "scope": "INDIVIDUAL",
            "enableParameter": false,
            "workDir": "/",
            "user": "root"
          }
        },
        "createdTimestamp": 1709896693000,
        "finishedTimestamp": 1709896697000,
        "statistics": {
          "totalTargetCount": 1,
          "failedTargetCount": 0
        }
      }
    ]
  }
}
```


附录

ActionRun

参数名称	类型	描述
id	String	执行ID
action	Action	所执行动作详情
state	String	执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成），PARTIAL_FAILED（部分失败）
createdTimestamp	long	执行开始时间。unix时间戳，单位：毫秒
finishedTimestamp	long	执行结束时间，仅执行结束时返回。unix时间戳，单位：毫秒。
statistics	Statistics	子执行执行情况，仅查询执行记录列表时返回

ChildRun

参数名称	类型	描述
id	String	子执行ID
state	String	执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成）
createdTimestamp	long	子执行开始时间。unix时间戳，单位：毫秒
finishedTimestamp	long	子执行结束时间，仅执行结束时返回。unix时间戳，单位：毫秒。
failReason	String	子执行失败原因
output	Output	命令输出结果
Target	Target	子执行对应的实例信息

Output | 参数名称 | 类型 | 描述 |

| -------- | -------- | ---------- | | exitCode | int | 退出码 | | stdout| String| 标准输出 | | stderr | String | 标准错误 | | isTruncated | bool| 标准输出或标准错误是否由于过长（超过4KB）而被截断 |

Statistics | 参数名称 | 类型 | 描述 |

| -------- | -------- | ---------- | | totalTargetCount | int | 执行的实例总数 | | failedTargetCount| int| 已失败的执行实例总数 |

日志接口

日志接口

获取命令执行的日志记录。

请求参数

参数名称	类型	是否必选	参数位置	描述
runId	String	是	RequestBody参数	执行ID
childId	String	是	RequestBody参数	执行ID下某一子执行ID
cursor	int	是	RequestBody参数	首次请求为0，后续翻页，填写上一次请求返回的nextCursor

响应参数 | 参数名称 | 类型 | 描述 | | --------- | ---------- | ---------- | | requestId | String | 请求id | | code | String | 响应状态，成功为success | | result | LogResult | 返回日志结果 |

请求示例

```
if __name__ == '__main__':

    HOST = b'http://ca.bj.baidubce.com'
    AK = b''
    SK = b''
    config = BceClientConfiguration(credentials=BceCredentials(AK, SK),
                                    endpoint=HOST)
    ca_client = CaClient(config)

    run_id = "r-raUPWwr933Liqe54"
    child_id = "d-7abfIJHODABF"
    response = ca_client.action_log(run_id, child_id, 1)
    print(response)
```

响应示例

```
{
  "requestId": "aaf4bd8f-ed6c-481b-ad1a-a4c974******",
  "code": "success",
  "result": {
     "logs": [
        {
           "msg": "pwd \n",
        }, {
           "msg": "second line",
        }
     ],
     "nextCursor": 2,
     "state" : "RUNNING"
  }
}
```

附录

LogResult | 参数名称 | 类型 | 描述 | | ---------- | ------ | ---------- | | childId |String | 子执行id | | logs | List<Log> | 日志内容，顺序排列 | | state | String | 子执行状态。枚举值：FAILED（执行失败），RUNNING（执行中），SUCCESS（执行完成） | | nextCursor | int |下一次请求使用的cursor。若子执行已结束，且nextCursor=cursor，表示已无更多日志。否则，表示可能存在更多日志 |

Log

参数名称	类型	描述
msg	String	命令执行日志

运维参考

Windows运维参考

系统激活

Windows激活相关配置

描述

百度智能云的所有镜像均采用登录即自动激活的方法，不需要任何的手动操作。如果登录发现系统是未激活状态，即激活失败，可以参照Windows激活失败的处理 ，如果您需要进行手动的重新激活，可以参照文档Windows手动激活系统教程。

百度智能云的所有云服务器操作系统源文件(ISO)均来自[微软官方网站(Microsoft Volume Licensing Service Center)] https://www.microsoft.com/licensing/servicecenter/。

配置

KMS客户端密钥配置

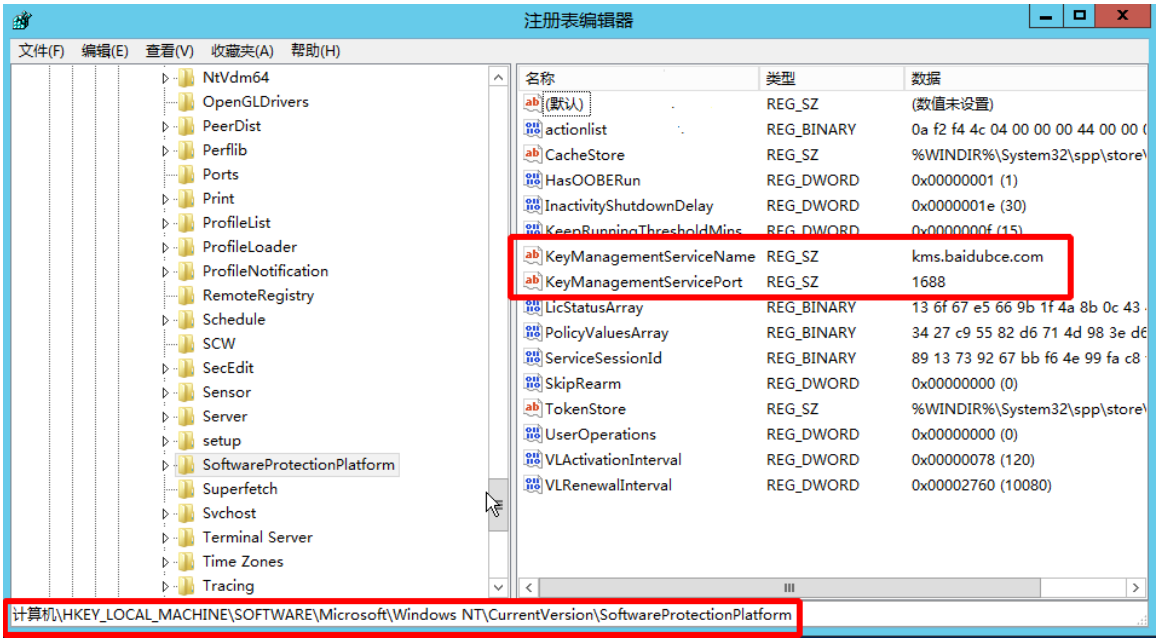
由操作系统源文件制作出来的所有镜像均已经安装了KMS客户端密钥，百度智能云的激活服务器仅能支持激活使用KMS客户端密钥的操作系统镜像，如果密钥被手动更改掉，需要更使用如下方式更改回来：

- 使用管理员权限打开命令行， 输入 slmgr.vbs /ipk [KMS Client Key]
- KMS Client Key 是公开的，可以从微软的官方网站获取 ：[微软官网] https://technet.microsoft.com/en-us/library/jj612867(v=ws.11).aspx

> **注意：**
>
> 打开网站时使用英文文档格式，中文文档偏旧，缺失了Windows Server 2016 的描述。

客户端的激活服务器配置

客户端上的激活服务器信息会存放在注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SoftwareProtectionPlatform 底下， 默认服务器指向kms.baidubce.com，使用默认的端口1688。



其他信息

- 镜像如果激活失败，间隔120分钟会自动重新激活一次。

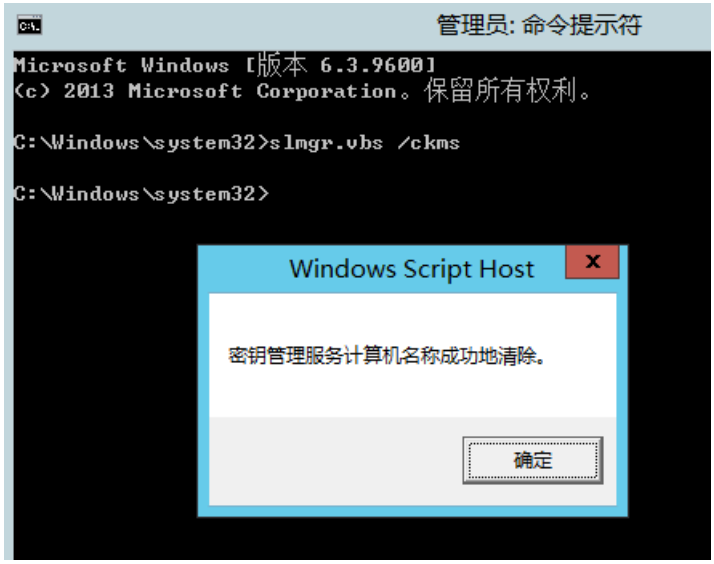
Windows手动激活系统教程

百度智能云的镜像采用自动激活的方式完成系统的激活，在某些特定情况下，可能会需要手动激活Windows系统，比如创建系统自动激活失败；自定义镜像中，客户对激活服务器进行指定性修改。

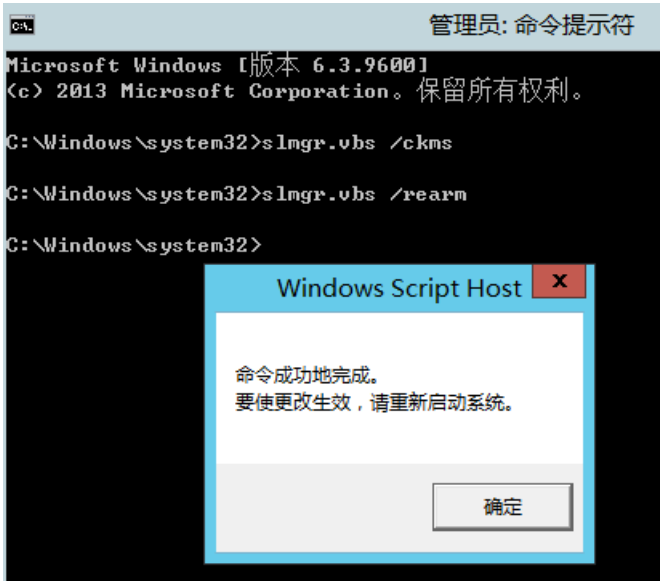
以下是手动激活Windows系统的过程：

1.打开有管理员权限的命令行。

2.在命令行中输入 slmgr.vbs /ckms 清除kms服务器设置。

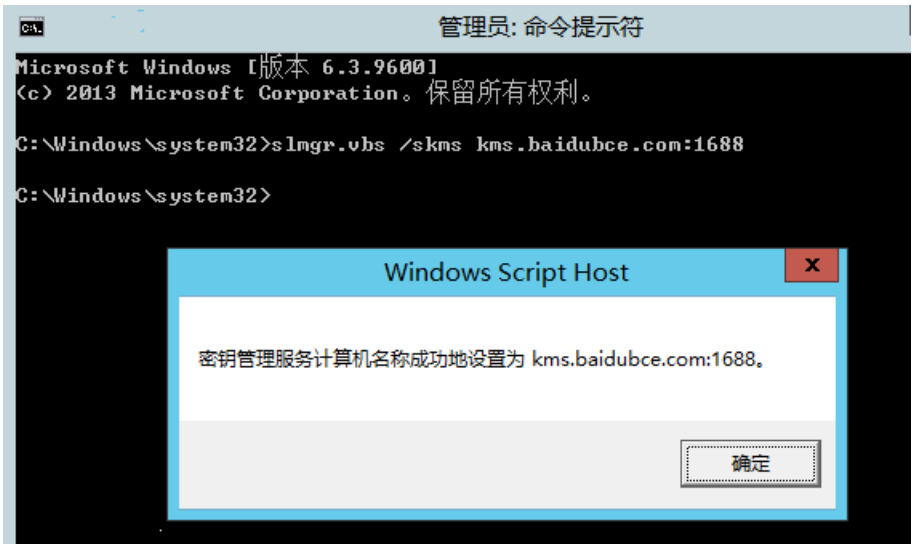


3.在命令行中输入 slmgr.vbs /rearm 重置计算机密钥状态。

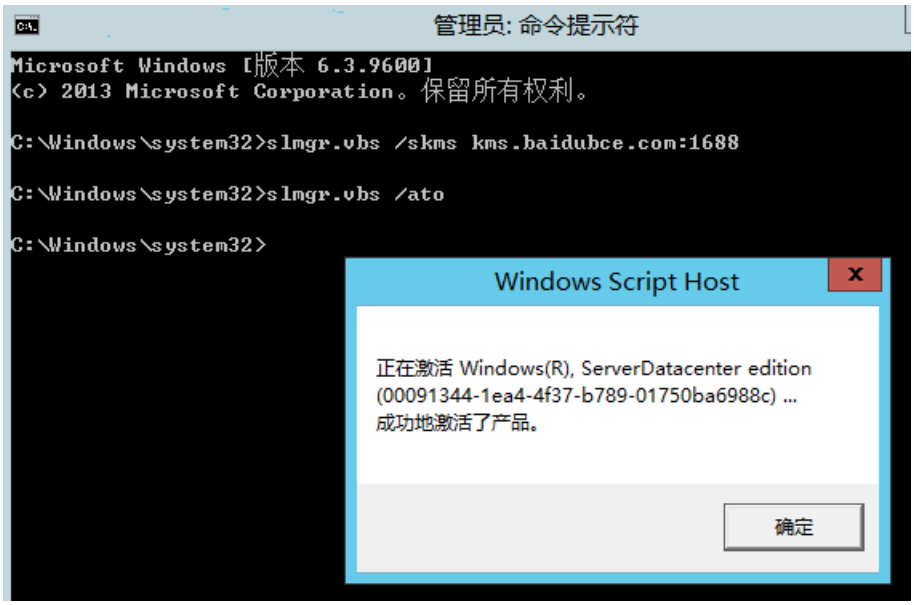


4.重启BCC实例之后，激活状态应该是如下显示。

5.重新使用管理员权限打开命令行， 输入 slmgr.vbs /skms kms.baidubce.com:1688 命令。



6.在命令行输入 slmgr.vbs /ato 命令， 进行激活。



> **注意：**
>
> 如果激活仍然未成功， 可以参照Windows 激活失败的处理，同时也可以[提交工单]（https://console.bce.baidu.com/support/#/ticket/#/ticket/create） 联系百度智能云客服。

Windows激活失败的处理

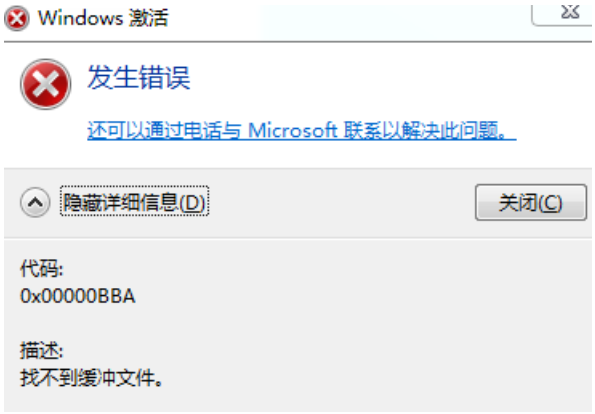
> **注意：**
>
> 此文档中的一部分操作需要您对Windows操作系统有一定的了解， 若您有任何疑惑，请先联系客服人员。 在您进行操作之前，建议对系统盘和数据盘创建快照以防止数据丢失。

激活失败的原因是多种多样的， 下列总结了一些碰到过的一些激活失败情况：

- 通信失败
- 时间和时区不同步
- 客户端密钥错误
- 第三方软件影响
- 服务异常

在我们遇到的激活失败中，经常会出现类似这样的提示窗口 ”错误：[错误码] ， 在运行Microsoft Windows非核心的计算机上，运行 ”slui.exe 0x2a [错误码]“ 以显示文本“, 这句话的意思是运行的程序发生了激活客户端的错误， 但是没有办法显示字符串，需要运行命令才能得到可读的错误信息。slui.exe 是Windows 激活客户端，主要是用于许可证激活的，有这个提示的错误可能主要和许可证相关的问题引起的，这种情况下，先打开命令行工具CMD， 输入提示提及的字符串，看看具体错误是什么。例如：

在命令行中输入“slui.exe 0x2a 0xBBA”后， 弹出窗口,显示可读的详细描述。



接下来我们详细记录一下碰到的一些激活失败的案例。

通信失败

现象

当百度智能云的Windows系统处于未激活状态，并试图激活系统时出错，错误码为0xC004F074。

原因

百度智能云Windows的激活是通过KMS服务器来进行的，出现上述错误的原因是Windows和KMS服务器之间的链路不通。KMS服务器在云平台内网，正常情况下，无论Windows是否添加EIP，都是可以联通的。

处理方法

1. 确认网卡是否被禁用。
2. 确认ping kms.baidubce.com 是否能通、域名 kms.baidubce.com的解析是否正确。
3. 确认是否有软件或设置对端口存在限制。

   3.1检查防火墙是否对kms.baidubce.com TCP 1688端口存在限制。

   3.2检查IP安全组策略是否对kms.baidubce.com TCP 1688端口存在限制。

   3.3检查杀毒软件、安全软件，是否对kms.baidubce.com TCP 1688端口存在限制。
4. 检查路由配置表，打开有管理员权限的CMD，输入 route print 确认是否为内网路由。
5. 确认telnet kms.baidubce.com 是否能通。
6. 若上述步骤均无问题，可以尝试重新激活，在有管理员权限的CMD中，执行如下命令。 6.1 slmgr /skms kms.baidubce.com:1688

   6.2 slmgr /ato

   6.3 更详细的手动激活信息可以查看 Windows 手动激活系统教程。

时间和时区不同步

现象

当百度智能云的Windows系统处于未激活状态，并试图激活系统时出错，错误码为0x80072F8F。

原因

Windows系统的时间和时区需要和KMS服务器保持一致，时钟的偏差有可能会导致激活失败。

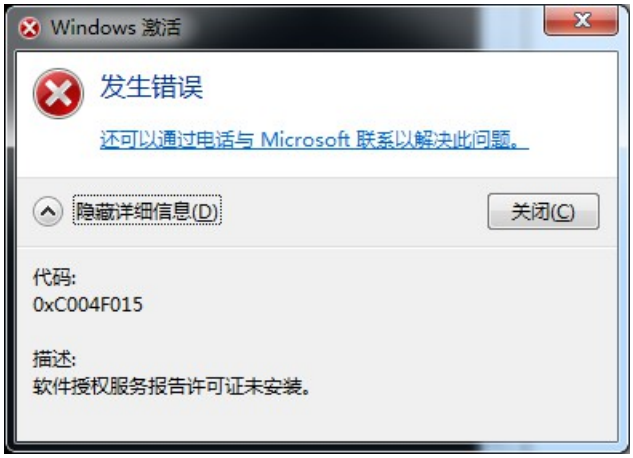
处理方法

1. 在桌面右下方的系统确认是否偏差较大，手动进行时钟及时区的同步。
2. 使用有管理员权限的命令行工具CMD.exe, 输入 "net start W32Timew32tm /resyn" 进行同步之后，重新激活。

客户端密钥错误

现象

Windows显示系统未激活，且错误代码为0xC004F015；或在系统的属性中，激活部分显示"产品ID不可用"。



原因

系统显示授权服务许可证未安装，可能是当前系统不被VLK的Server所认可，检测发现系统使用的激活码与VLSC发布的系统激活码不一致。

处理方法

将KMS 客户端密钥重新安装。

1. 从微软的[官方网站] https://technet.microsoft.com/en-us/library/jj612867(v=ws.11).aspx 下载KMS客户端密钥。 a. 注意：打开网站时使用英文文档格式， 中文文档偏旧，缺失了Windows Server 2016 的描述。 b. Windows 2008 R2 Data Center：74YFP-3QFB3-KQT8W-PMXWJ-7M648。 c. Windows 2012 R2 Data Center：48HP8-DN98B-MYWDG-T2DCC-8W83P。 d. Windows 2016 Data Center： CB7KF-BWN84-R7R2Y-793K2-8XDDG。
2. 使用如下两个命令进行密钥安装和重新激活。 a. slmgr.vbs /ipk [KMS Client Key]。 b. slmgr.vbs /ato。

第三方软件影响

现象

第三方的杀毒软件或者木马病毒也会影响系统激活。 例如，一个需要使用到的服务被“安全狗软件”停止，从而导致激活失败。由第三方软件引起的错误码是多种多样的，比较常见的是“拒绝访问”， 错误码是0xC0000022。

处理方法

第三方软件通常在安全模式下是不会被主动运行的，可以进入安全模式，确认是否能够激活成功。进入安全模式的方法如下：

1. 使用命令行，输入"msconfig" 命令，打开"系统配置"界面。
2. 在系统配置窗口 -> 一般 ->启动选择，选择"诊断启动"，点击确认按钮，并重启计算机。
3. 在系统启动时，按 F8，选择"安全模式"，继续启动系统。
4. 在进入操作系统后，打开管理员权限的命令行，输入"slmgr.vbs /ato" 命令进行重新激活。

服务异常

现象

Windows激活的过程中，会使用到一些服务程序， 比如Software Protection 服务是使用于进行数字许可证激活和检测的，如果这些服务例程异常可能会导致激活失败，错误码是多种多样的，例如错误码是 0x80070002 - 系统无法找到指定文件。

原因

Software Protection服务异常，服务使用的程序sppsvc.exe未启动。

处理方法

1. 重新启动Software Protection 服务： 右键点击"计算机" -> "管理" -> "服务" -> "Software Protection" ,点击启动。
2. 若Software Protection 服务启动失败， 有可能有以下原因：
   - 杀毒软件阻止了服务的运行。 关掉杀毒软件再试试。
   - 服务配置异常，需要重新设置注册表项 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sppsvc。
     1. 导出注册表项作为备份backup.reg。
     2. 从另一台同样版本的系统里面导出正常的注册表项文件normal.reg。 然后导入当前的操作系统。
     3. 如果导入失败，则将备份backup.reg 重新导入回原来的计算机， 再次查找其他原因。

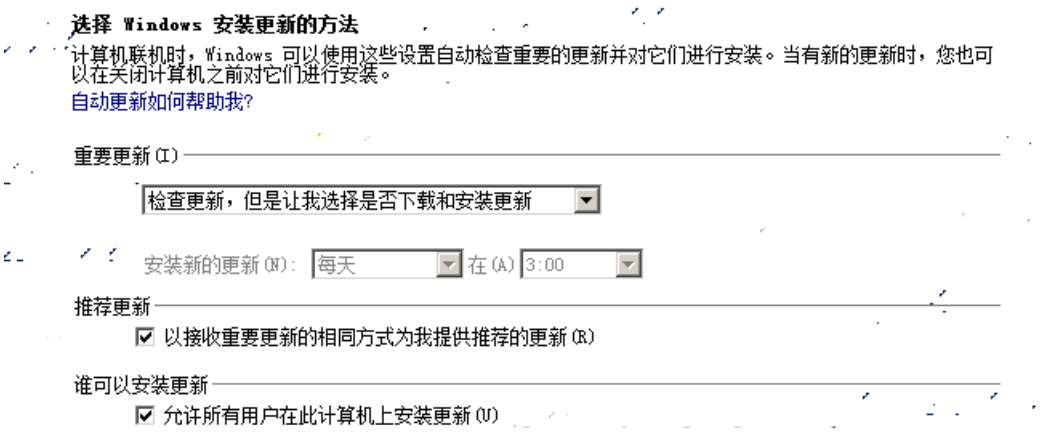
Windows自动更新相关配置

Windows自动更新相关配置

在百度智能云的所有Windows云服务器中，默认使用的是百度智能云内部的更新服务器，而不是从微软的官方源进行更新。采用这种方式的原因有以下几个方面。

1. 即使用户的虚拟机没有申请外网IP，仍然可以获得最新的系统更新包。 2. 减少外网的通道流量。 3. 对微软的更新包进行一定的筛选和审核，避免安装过多的更新包使系统运行缓慢。

目前百度智能云的更新设置如下



百度智能云内部的更新服务器配置如下

更新服务器：http://windowsupdate.baidubce.com:8530

统计服务器：http://windowsupdate.baidubce.com:8530

测试服务器的连通性

当我们需要知道，当前云服务器是否和百度智能云的更新服务器连通的时候，可以使用以下的方式来进行测试。

1. 在命令行中输入： ping windowsupdate.baidubce.com 看是否能够ping通。

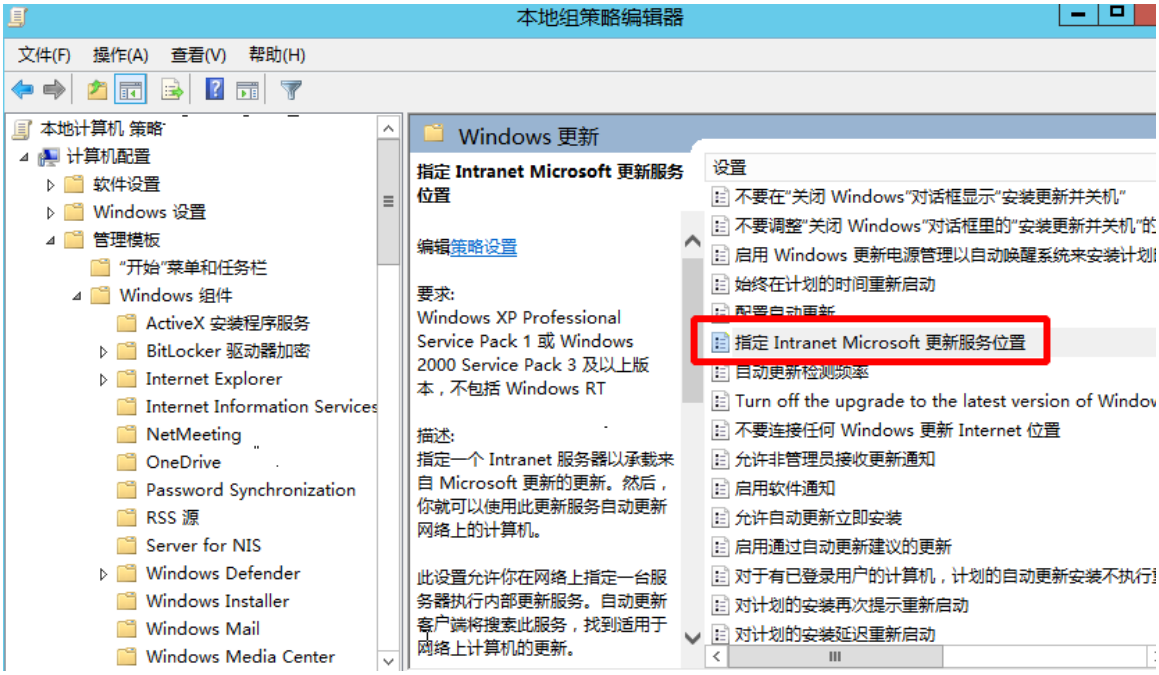
2. 在命令行中输入： telnet windowsupdate.baidubce.com 8530 看端口是否打开。

> **注意：**
>
> telnet 工具的安装可以参考[微软文档] https://technet.microsoft.com/en-us/library/6da83bed-c357-4dd7-af1e-26cc2f6c0364。

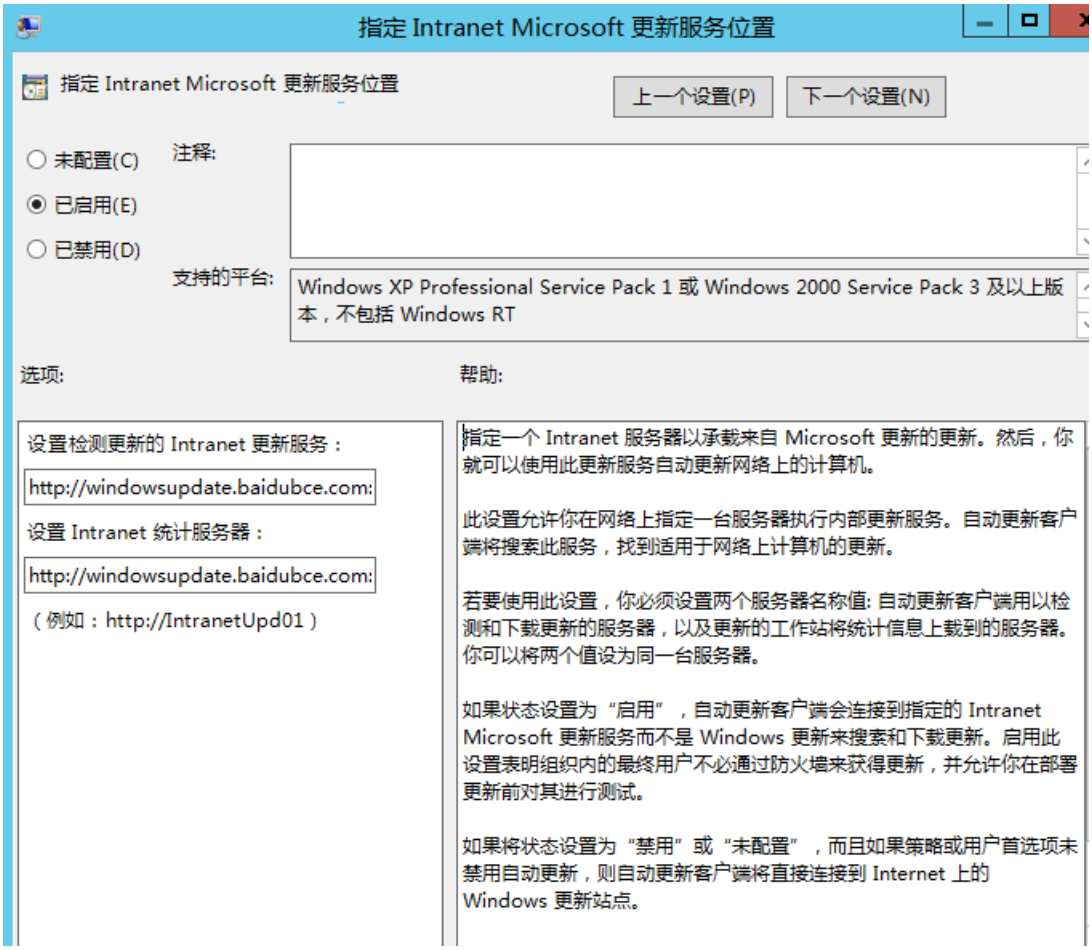
更改更新源

百度智能云的更新服务器从微软的更新源上拉取相应的更新源包， 然后通过验证后发布到服务器上，所以内部的更新服务器上的更新包是被过滤过的，并相对微软更新服务器有一定的延迟。当您希望直接从微软的更新源上下载更新包的时候， 您可以通过修改相应的组策略来进行配置，方法如下：

运行 -> 输入"gpedit.msc" -> 选择"计算机配置" -> 选择"管理模板" -> 选择"Windows组件" -> 选择"Windows更新" -> 选择"指定Intranet Microsoft 更新服务位置"



配置



> **注意：**
>
> 修改组策略之后，如果想要不重启计算机的情况下马上生效，可以使用管理员权限打开CMD，然后运行"gpupdate /force"

> 命令生效。

Windows 更新失败处理

Windows 检查系统更新报错，错误码 0x80072F8F 的处理

现象

在Windows 操作系统中，进行"检查更新"的操作时候报错， 收到的错误码是0x80072F8F，如下图所示：

原因分析

当操作系统和Windows更新服务器的系统时间和时区不一致的时候， 时钟的偏差有可能导致更新失败。

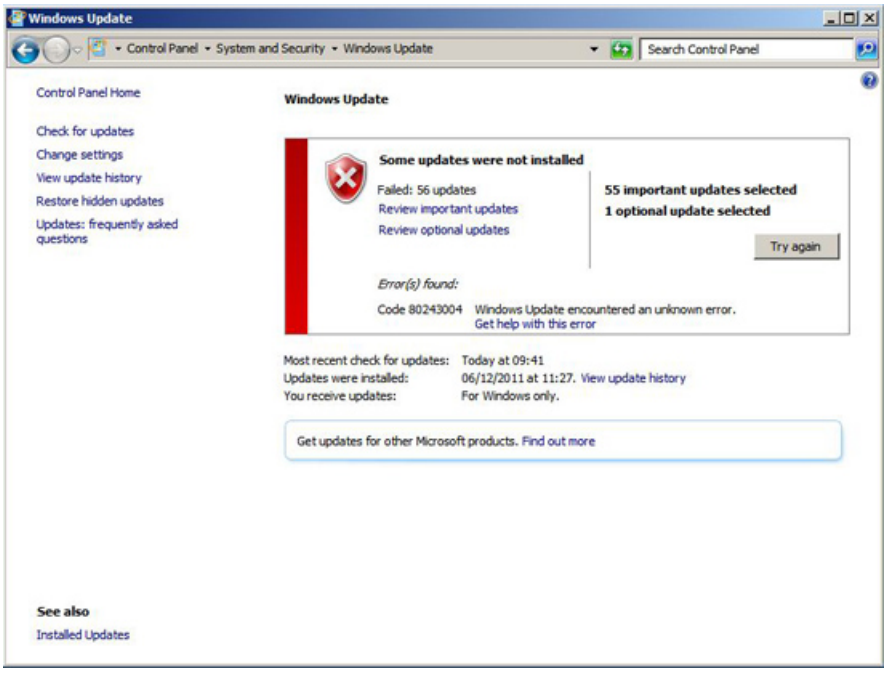
处理方式

1. 在桌面右下方的系统看看是不是偏差较大，可以手动进行时钟和时区同步。
2. 使用既有管理员权限的命令行工具CMD.exe, 输入 "net start W32Timew32tm /resyn" 进行同步之后，进行重新检查更新。

Windows 检查系统更新报错，错误码 0x80243004 的处理

问题现象描述

在Windows 2008操作系统中，进行"检查新的更新"操作时， 收到Windows Update的错误 0x80243004， 如下图所示：



问题原因

Windows Update在创建角标通知的时候发生错误。 原文如下：

0x80243004 WU_E_TRAYICON_FAILURE A failure occurred when trying to create an icon in the taskbar notification area.

解决方案

直接方法就是显示出所有的角标通知。

1. 点击状态栏通知区域的三角形，显示任务栏图标，点击**"自定义"**。

2. 在"**通知区域图标**"中的"**选择在任务栏上出现的通知和图标**"列表里面， 找到"**Windows Update**"，点击右边的下拉列表，选择"**显示图标和通知**"， 点击"**确定**".或者直接勾选"始终在任务栏上显示所有图标和通知"，点击"确认"。

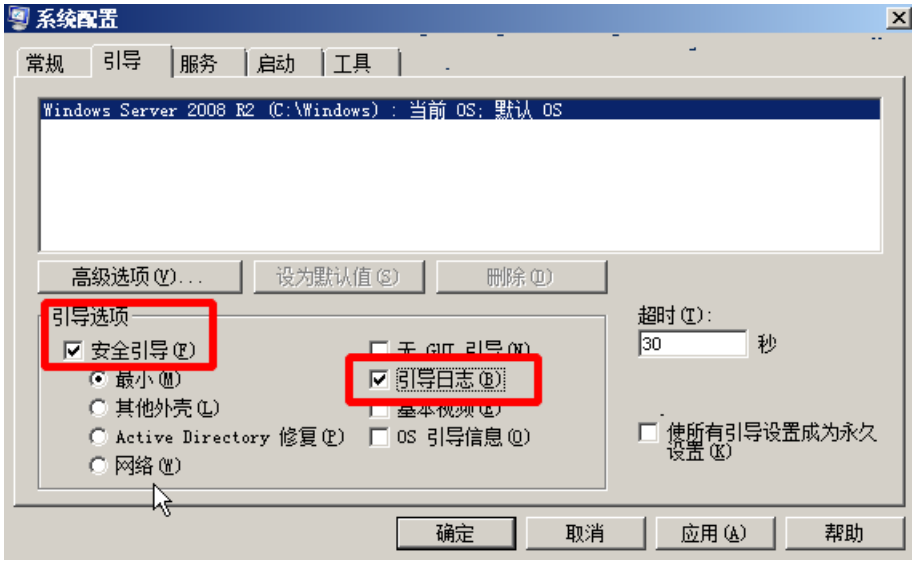
3. 重新检查系统更新。

蓝屏宕机常规处理方法

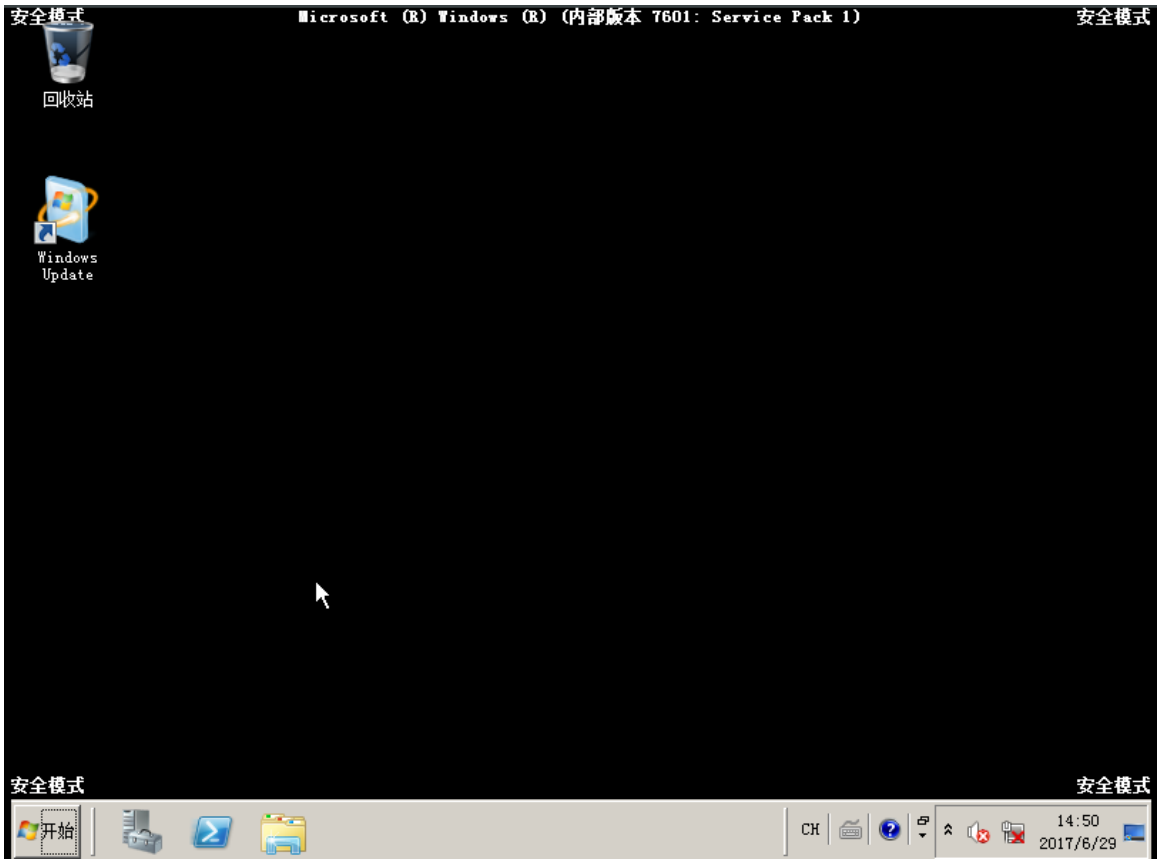
Windows 主动进入安全模式的方法

当系统感染病毒或者有其他需要进入到安全模式时，用户可以主动设置进入Windows安全模式启动系统。 设置如下：

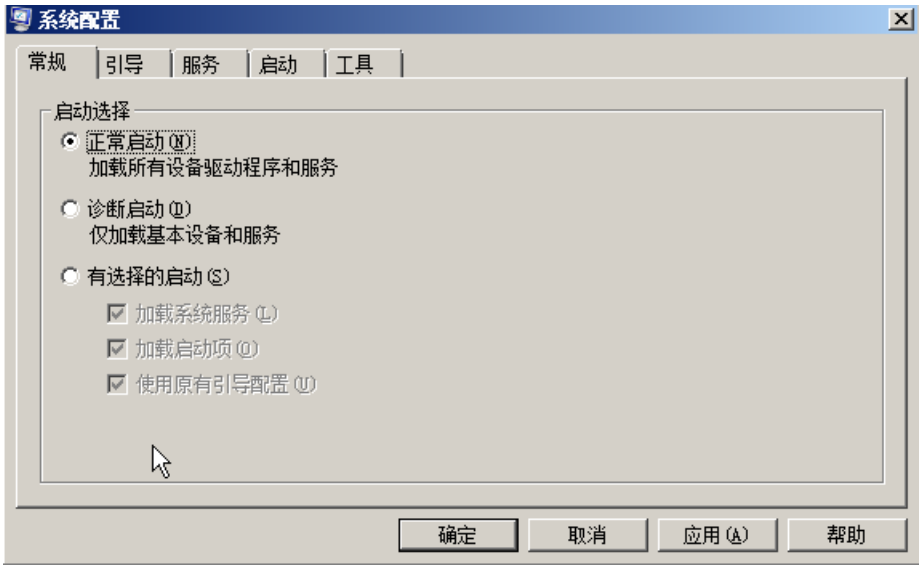
1.打开有管理员权限的命令行工具， 输入"**msconfig**"，进入到"**系统配置**"界面，勾选如下图所示。



2.点击"应用"，点击"确定"后,重启操作系统，系统自动进入安全模式，安全模式下，桌面四角有"安全模式"的水印，如下图所示。



3.当操作完毕之后， 需要恢复到正常启动的模式。 重新打开"系统配置"界面，将启动选择修改成"正常启动"即可。



Windows 内核转储设置

百度智能云的Windows虚拟机默认开启小内存转储(Core Dump)。在某些特定的情况下，比如蓝屏、黑屏、系统hang住，研发人员需要借助转储文件才能快速定位问题点，从而快速处理问题。

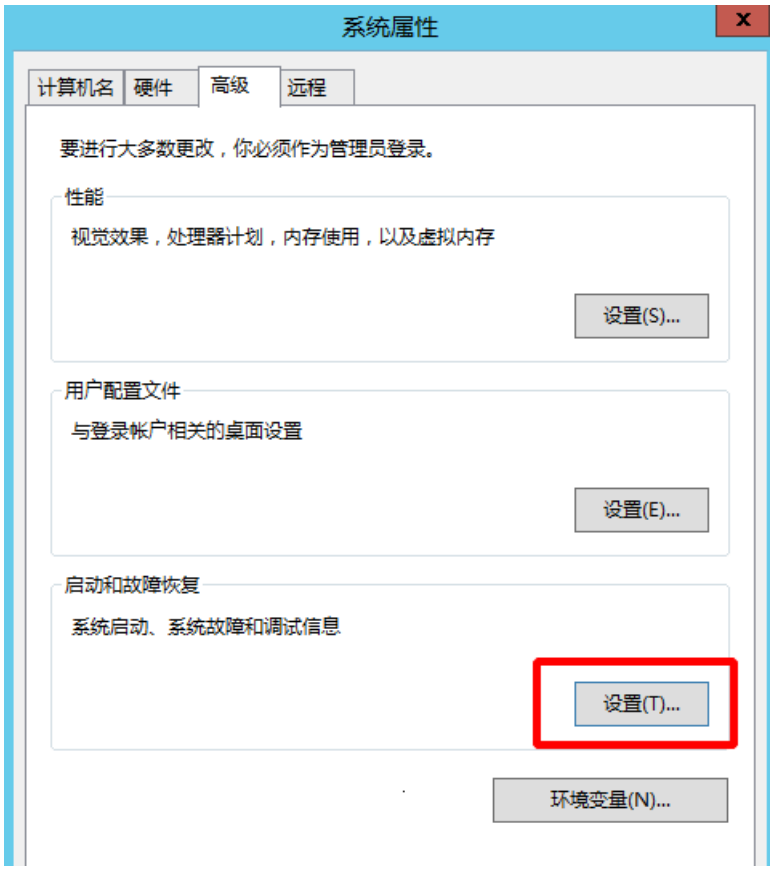
在小内存转储的设置下，Dump文件会存放在%SystemRoot%\Minidump(e.g. C:\Windows\Minidump)目录底下，这些文件的历史记录按照日期命名，查找带有当前系统日期的文件名，即可找到当前的Dump文件。

此外，内存转储还有其他几种方式：核心内存转储、完全内存转储、自动内存转储，其dump文件均是%SystemRoot%\MEMORY.DMP(e.g. C:\Windows\MEMORY.DMP),此文件会自动覆盖前一次产生的dump。

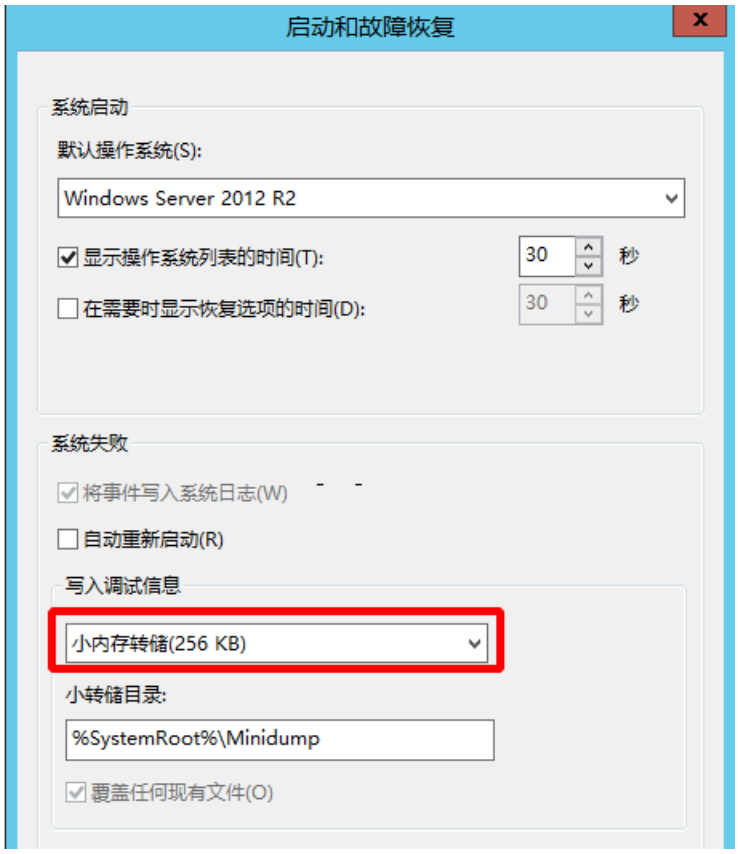
可以通过下列的方式手工修改产生转储文件的方式。

1.打开"**控制面板**"，点击"**系统**"，在系统页面双击"**高级系统设置**"。

2.在"**高级系统设置**"页面下，点击"**高级**"页面，设置 "**启动和故障恢复**"。



3、在"**启动和故障恢复**"页面，选择"**写入调试信息**"的方式。



> **注意：**
>
> 如果C盘空间不足，建议可以将转储文件放到其它磁盘中。

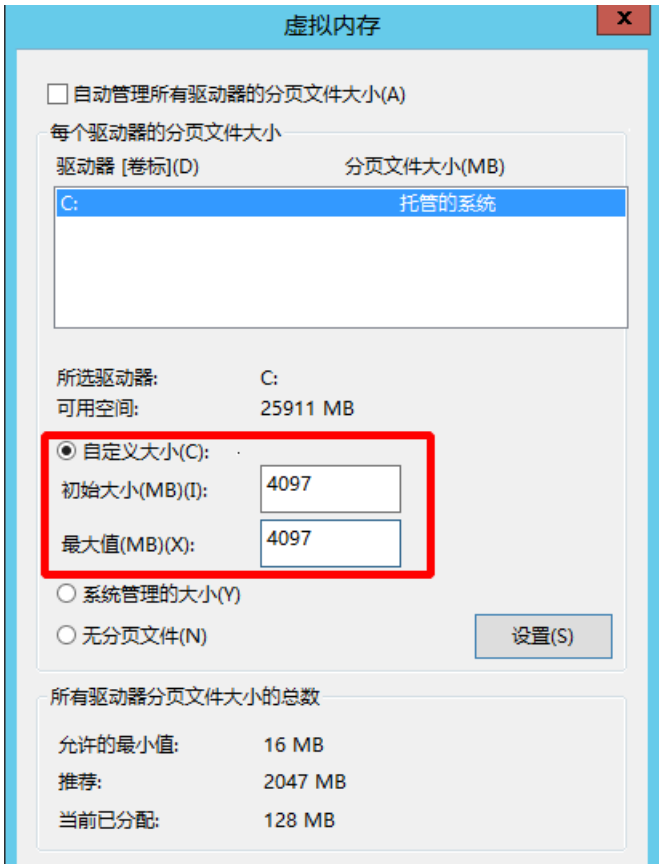
配置页面文件：

配置页面文件需要您对操作系统有一定的了解。当系统运行到一定条件下的时候，会使用到更多的虚拟内存。当内存不足时，Windows 可以将硬盘上的的一块区域(以分页文件的方式)当作RAM来使用。

设置方法如下：

1.在"**高级系统设置**"窗口的"**高级**"页面下，点击"**性能**"设置，进入"**高级**"页面，双击"**更改**"修改虚拟内存。

2.修改分页文件的大小，文件大小通常会比当前使用的内存大一点点。



微软的官方参考文档

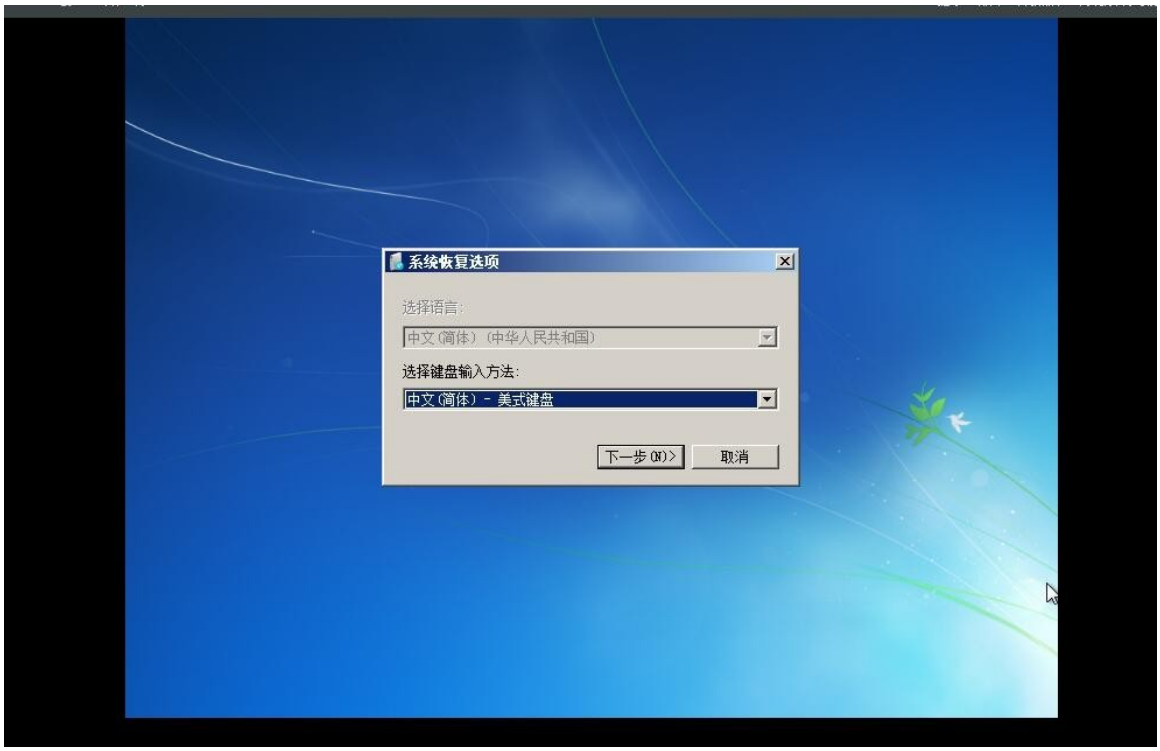
- [如何在 Windows Server 2008 和 Windows Server 2008 R2 中生成内核或完全存储器转储文件]https://support.microsoft.com/zh-cn/kb/969028。
- [Microsoft"蓝屏"或停止错误问题疑难解答] https://support.microsoft.com/kb/3106831

Windows 系统重启后显示系统恢复选项

Windows 系统有可能会遇到重新启动后，无法正常载入系统的情况.这个时候，用户首先会看到"系统恢复"界面。

当进入到"系统恢复"界面的时候，说明此时的系统，已经无法维持正常的运转，原因有很多，如注册表损坏、系统文件损坏、BCD出现问题等等。这些情况很多时候是由于安装的第三方杀毒软件、误操作、异常重启、第三方应用软件引起的。

"**系统恢复**"界面会引导客户进入Windows的修复模式(Windows Recovery Environment) ，修复模式资料参考 Understanding Windows Recovery



处理方式

百度智能云采用微软官方的系统恢复方案。

How Windows RE Works

同时，百度智能云也会通过一些Windows系统命令 sfc工具，尽最大努力的去恢复系统。但是也有可能会因为系统损坏过于严重而无法恢复。

建议

为了更好的使用操作系统， 避免发生系统崩溃，和提高从系统崩溃中恢复的可能性，百度智能云建议：

- 定时创建系统盘和数据盘快照
- 定期运行系统更新，确保当前系统安装了最新的更新包
- 使用商业版本的杀毒软件，并定期更新杀毒软件
- 尽量使用数据盘保护数据系统，并定期备份
- 避免修改您不了解的系统文件，注册表，若确实需要修改，请先备份

Windows 蓝屏宕机的一般处理

蓝屏又称为蓝屏死机(Blue screen of Death – BSoD), 英文文档提到的Stop error,也是这个意思。是指微软的Windows操作系统没有办法从错误中恢复过来，或者已经达到不能安全操作的状态，为保护数据不被破坏而强制显示的界面。百度智能云采用小内核转储方式，蓝屏发生经过搜集信息后会存在%SystemRoot%\Minidump(e.g. C:\Windows\Minidump)目录底下，具体参照[Windows 内核转储设置](#Windows 内核转储设置)

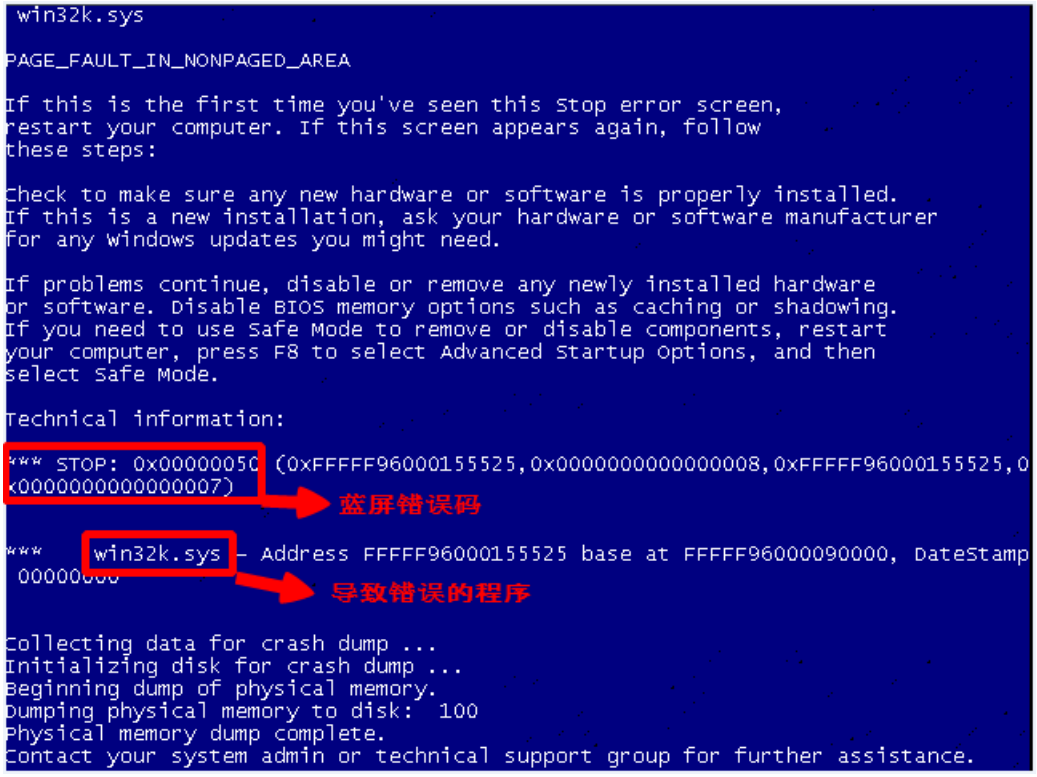
产生蓝屏宕机的原因

- Windows内核态程序出错
- 操作系统文件，注册表的损坏
- 杀毒软件导致的异常
- 病毒感染
- 第三方软件和系统不兼容
- 硬件损坏

- 资源不足

蓝屏信息

在蓝屏界面，通常可以得到蓝屏错误码 如下图所示 Win32k.sys是当时内存出错时，指向的程序；0x00000050代表的是PAGE_FAULT_IN_NONPAGED_AREA错误，以及相应的参数。 具体错误码对应的意义在微软的官方网站上有相应的说明 Bug Code Rerference



当Crash Dump 收集完毕之后， 就可以重启计算机。更多的蓝屏信息，可以在启动计算机之后，查看系统事件簿。如果无法重新启动，请尝试从快照或者备份中还原系统(这可能会丢失一部分数据)， 若有其他问题，请联系百度智能云官方客服。

查看系统事件簿， 可以看到事件ID 为1001的BugCheck类型的事件记录了蓝屏宕机的相关信息， 此外在之前应该还有一个ID为41的来源kernel-power的电源事件，说明系统曾经发生过重启。 如果客户曾经修改过内核转储设置为不生成Dump文件， 则会有一个ID为46的内核转储失败的来源为volmgr的事件。

处理方式

由于蓝屏的分析需要比较长的时间， 建议发生蓝屏后，重启机器，以便快速恢复业务。百度智能云建议用户：

- 定时创建系统盘和数据盘快照
- 定期运行系统更新，确保当前系统安装了最新的更新包
- 使用商业版本的杀毒软件，并定期更新杀毒软件
- 尽量使用数据盘保护数据系统，并定期备份
- 避免修改您不了解的系统文件，注册表。若确实需要修改，请先备份

在百度智能云处理蓝屏的经验中， 一般发生蓝屏比较多的原因是：

1.Windows内核态程序出错（例如：系统驱动程序出错）

2.病毒感染

3.操作系统文件损坏

针对病毒感染的问题，我们建议：

- 安装Windows系统更新到最新补丁包
- 在安全模式下，运行杀毒软件进行杀毒 [Windows 主动进入安全模式的方法](#Windows 主动进入安全模式的方法)

针对文件系统损坏的问题，我们建议：

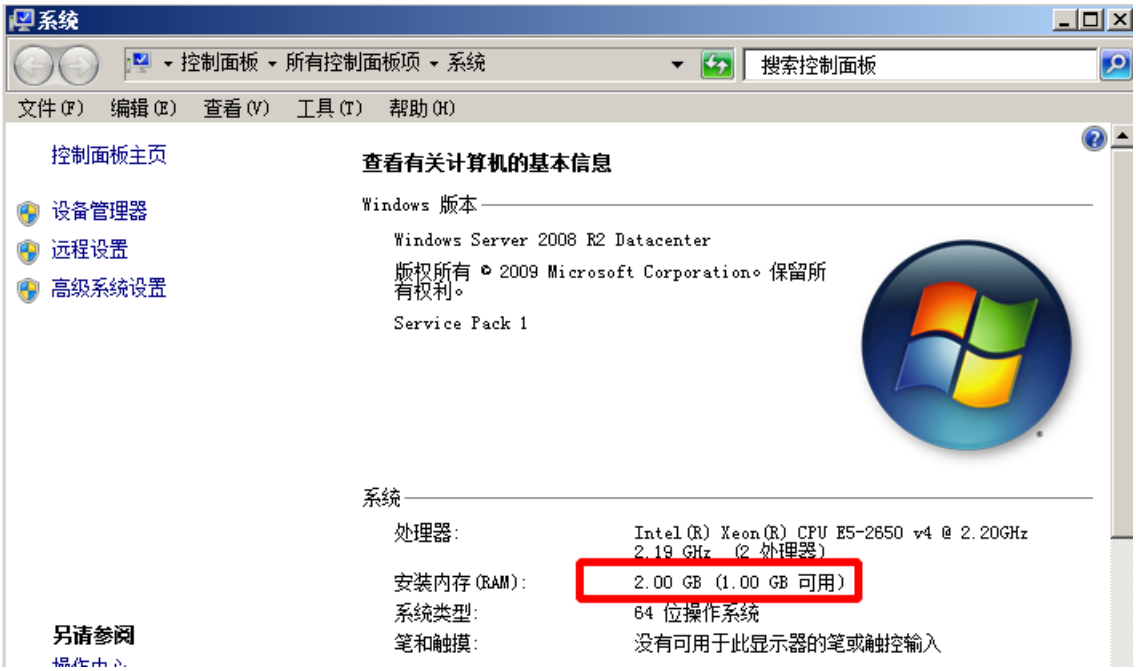
- 若系统能重启成功，进入系统后，运行有管理员权限的CMD， 输入 sfc /scannow 进行系统完整性的检查和修复
- 若无法进入系统，一般会进入系统恢复选项界面，详情查看 [Windows 系统重启后显示系统恢复选项](#Windows 系统重启后显示系统恢复选项)

针对Windows系统驱动程序出错的问题，一般情况下， 重启计算机程序即可恢复，同时可以提交相应的Dump文件给百度智能云客服，百度智能云研发人员会进行分析。

性能资源

Windows 可用内存和申请购买的内存不一致的处理

用户购买了一定量的内存套餐（以2G为例）,通过任务管理器发现可用的内存却达不到购买的内存数量，通过系统属性可以看到如下显示。



问题分析

这种情况是可用内存被限制了，Windows系统本身有最大的支持内存，同时Windows支持内存的限制使用，以帮助系统或者内核调试的场景- 即有足够的调试内存，同时可以在不改变硬件的情况下，修改内存的使用大小。百度智能云的虚拟机不会对内存的使用进行限制，一般情况下是客户对这部分内容主动进行过修改。

Windows 系统本身能支持的最大内存

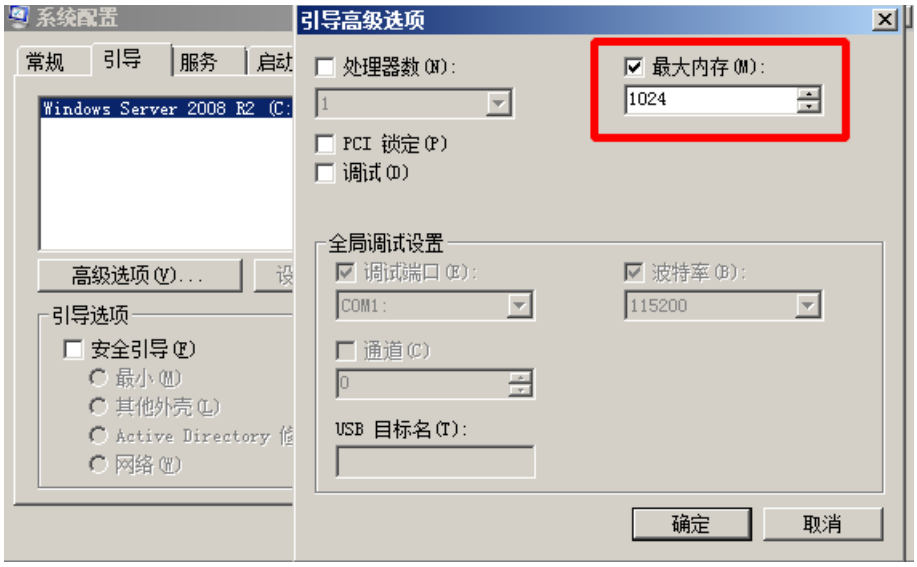
https://msdn.microsoft.com/zh-cn/library/windows/desktop/aa366778(v=vs.85).aspx

解决方法

只要将内存的限制去掉即可，步骤如下：

1.【开始】->【运行】 ， 输入"msconfig"打开微软系统配置。

2.在"**引导**"页面，选择当前系统， 点击"**高级选项**"， 去掉"**最大内存**"的勾选项。



3.重启计算机，可以看到可用内存不再被限制。

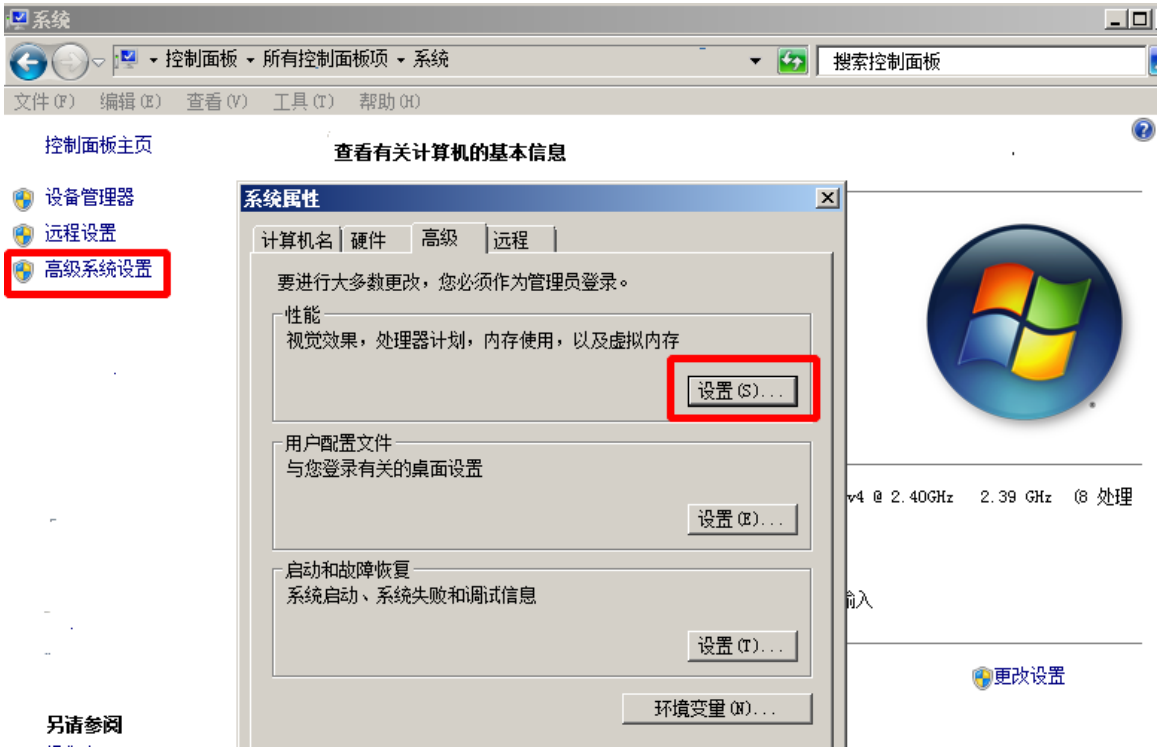
Windows 虚拟内存的设置

由于内存本身的价格要远超硬盘的价格， Windows 提供 了一种增加内存的方式，Windows 可以将硬盘上的的一块区域(以分页文件的方式)当作RAM来使用。这部分RAM的大小就是Windows虚拟内存的大小。

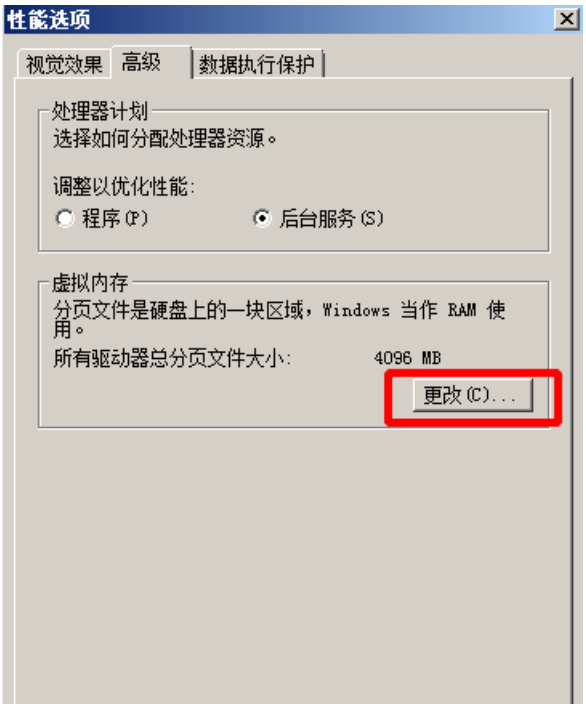
由于开启虚拟内存会导致IO性能的下降，用户可以自由选择是否打开虚拟内存。

可以通过以下方式来设置改变这个虚拟内存默认的大小值。

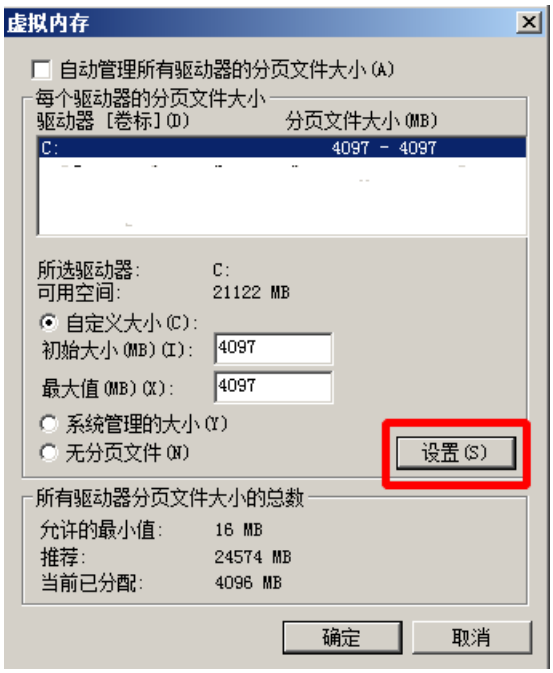
1.打开"**控制面板**"，点击"**系统**"，在左侧窗口，点击"**高级系统设置**"。



2.在"**高级系统设置**"窗口的"高级"页面下，点击"**性能**"设置，进入"**高级**"页面，双击"**更改**"修改虚拟内存。



3.修改分页文件的大小，并且点击"**设置**"来确认修改文件的大小，再点击"**确定**"按钮，重启计算机。

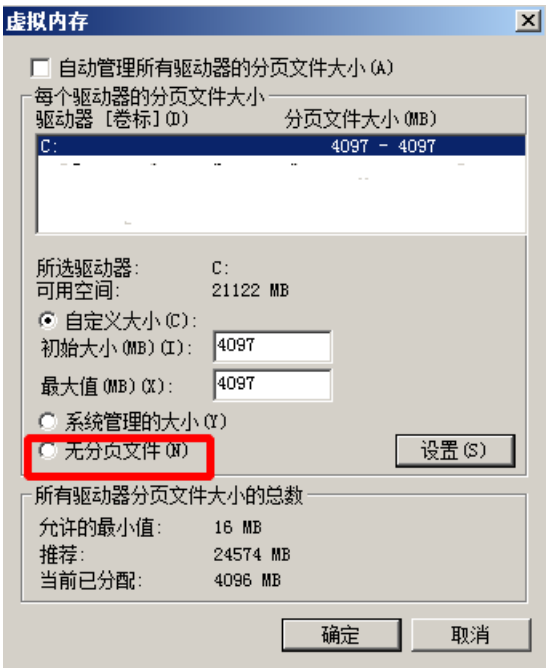


根据微软的推荐策略

- 分页内存最好是比物理内存大一些，但是不超过两倍内存。
- 尽量将页面交换文件设置在非系统磁盘，IO性能会好一些。
- 根据系统宕机时候Dump时候的需求。
- 目前百度智能云生成小转储dump，无需考虑这点，详情可见 Windows 内核转储设置。

> **注意：**
>
> 增加虚拟内存会导致磁盘IO性能下降，建议尽量通过升级物理内存来解决。百度智能云上有一些存量计算机会根据 系统内存自动设置虚拟内存，在申请较大的物理内存时，建议关闭虚拟内存的使用。

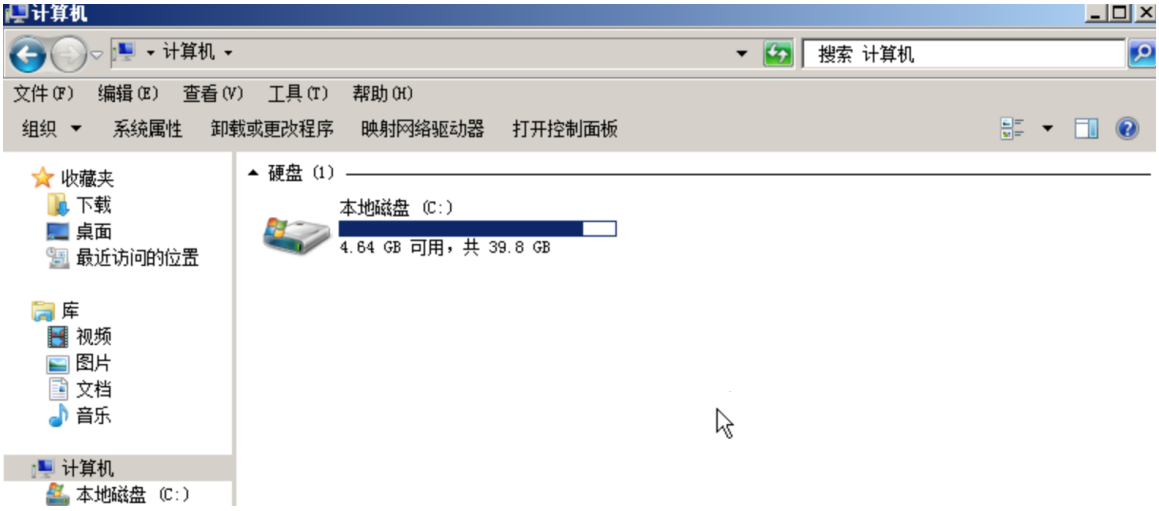
关闭虚拟内存的方法是在虚拟内存设置页选择"**无分页文件**"。



Windows 虚机创建后，系统盘剩余空间过小的处理

问题现象

百度智能云用户创建了一个Windows 虚拟机， 开启启动之后，发现系统盘剩余的空间很小， 如下图所示。



问题分析

上图用户创建的是一个8核16G的虚拟机， 通过打开提供文件管理器， 发现占用磁盘的一个隐藏文件pagefile.sys达到16G，pagefile.sys文件是虚拟内存的分页文件。所以，这个问题的原因是系统创建了和真实内存一样大小的虚拟内存占用了系统盘空间。

解决方法

因为是虚拟内存文件占用了系统盘的空间，所以用户可以将虚拟内存关闭或者调低，或者将虚拟内存文件存放到系统盘中， 设置方法参照 [Windows 虚拟内存的设置](#Windows 虚拟内存的设置)。

Windows 虚机创建后，进程svchost 使用CPU100%的处理

问题现象

创建了1核1G配置的Windows Server 2008 R2的虚拟机，登录虚拟机之后发现CPU使用率达到100%， 打开任务管理器发现占用CPU的是svchost.exe应用程序。

问题分析

Svchost 是从DLL中运行服务的一类进程的统称， 所以svchost的问题是要知道使用他的是具体哪个进程或者服务。打开任务管理器，右键点击进程中的svchost， 然后转到服务，可以看到好几个服务都在使用svchost， 其中Windows Update 占用了大部分CPU，查看Windows更新， 发现正在更新系统，Windows Update 在网络不稳定或者更新内容过多的情况下， 会分别导致CPU和内存过高的时间较长。

解决方法

百度智能云推荐您使用较高CPU和内存套餐，以便获得更好的使用体验，不推荐使用1核1G的配置。在1核1G的配置下，用户可以先暂时关闭自动更新服务，由用户自己选择更新时间，利用空闲时间段去更新系统。同时百度智能云制作的最新镜像均已安装最新的系统更新，将更新量减少到最小，缓解这种情况的发生。

系统黑屏处理方法

Windows 桌面一直显示黑色背景屏幕的处理

问题现象

百度的Windows计算机打开之后，一直显示黑屏，没有任何左面图标，鼠标可以正常移动。

与此同时，点击右上角的Alt + Ctrl + Del 按钮可以显示出面板。

问题分析

现象表明Windows无法显示相关的文件和界面。Windows explorer.exe是Windows文件资源管理器，管理着Windows图形界面，文件系统，这个程序的异常会导致Windows GUI异常，用户可以通过重新启动explorer.exe来修复问题。

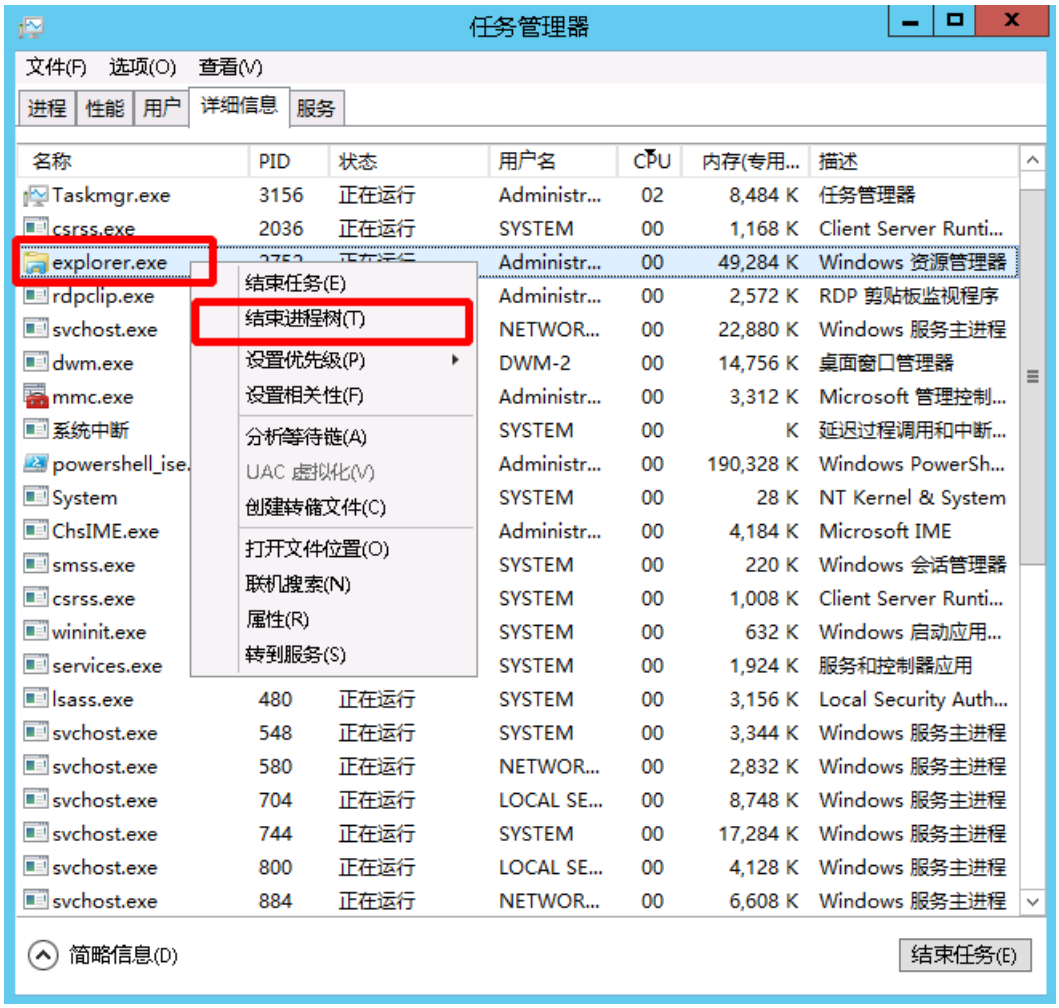
explorer.exe位于C:\Windows目录底下，如果这个目录底下没有这个文件， 说明Windows的系统文件已经被破坏，需要进行修复。

解决方法

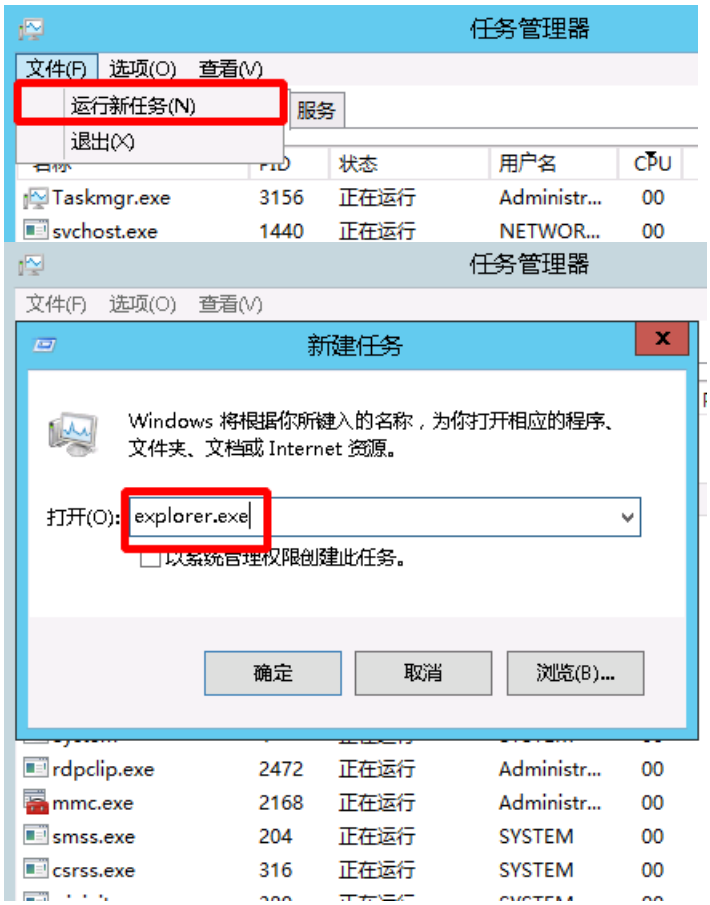
1.点击右上角的Alt + Ctrl + Del 按钮可以显示出面板， 选择"任务管理器"。

2.如果在"任务管理器"里面可以看到explorer.exe程序，则重启explorer.exe即可， 如果没有，则直接启动explorer.exe。

3.重启explorer.exe -先关闭explorer，如果有的话。



4.启动explorer.exe



5.这个时候， Windows桌面就应该启动起来了。 如果在第四步里面输入explorer.exe执行失败， 则很有可能是系统损坏，需要进行修复。

6.修复过程

a.和第四步一样，运行新任务， 输入cmd.exe， 运行命令行工具

b.在命令行中，输入 dir explorer.exe /s 看看在C:\Windows 目录有没有 explorer.exe，已经WinSxS里面有多少个存档

c.运行sfc /scannow 目录修复系统文件

d.如果修复完毕， 在C:\Windows 目录仍然没有explorer.exe文件。 请联系百度智能云客服

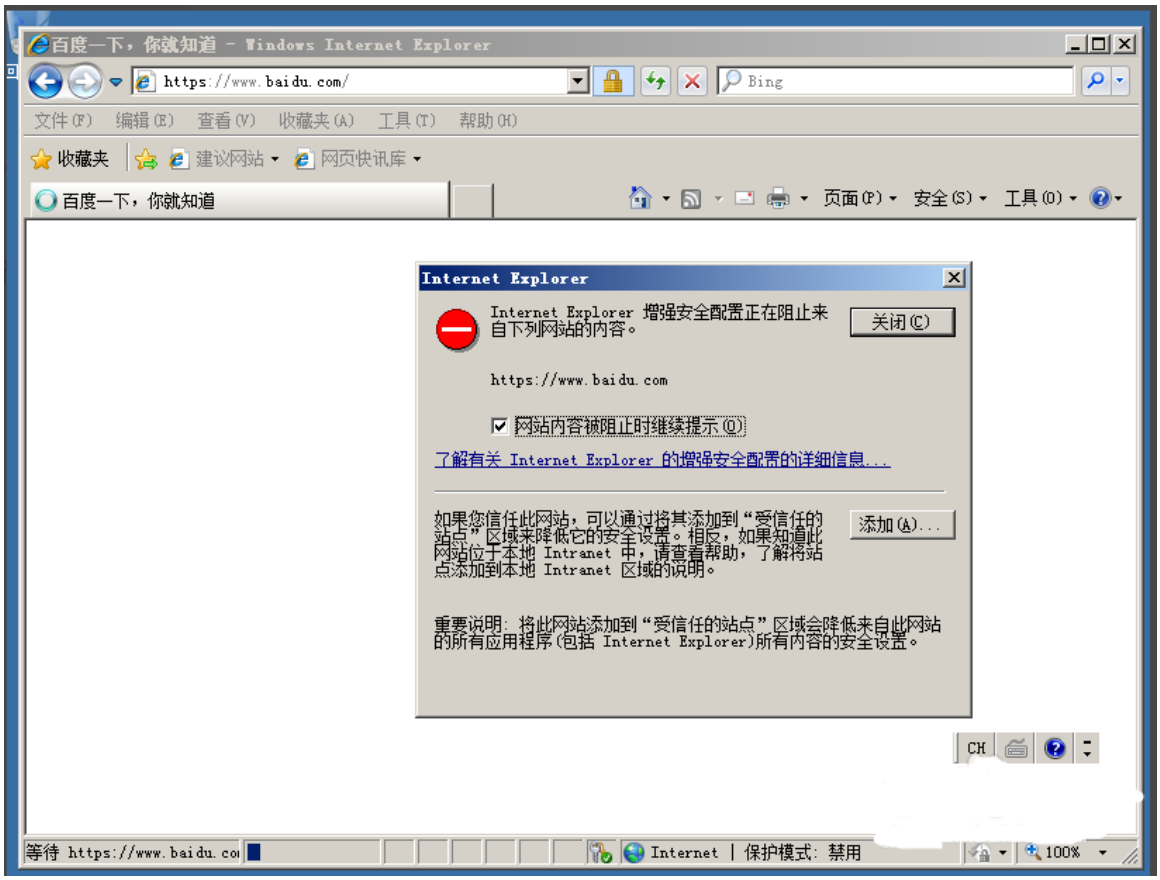
windowsServer关闭IE增强机制教程

本文介绍百度智能云“Windows Server关闭IE增强机制”的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频 《Windows Server 2008关闭IE增强》 《Windows Server 2012关闭IE增强》。

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

安装Windows Server 系统以后，用浏览器打开网址时，系统总是提示“Internet Explorer 增强安全配置正在阻止下列网站的内容。如果把所有网址添加到信任列表是可以打开网址的，但是用起来很吃力。那么如何解决呢？

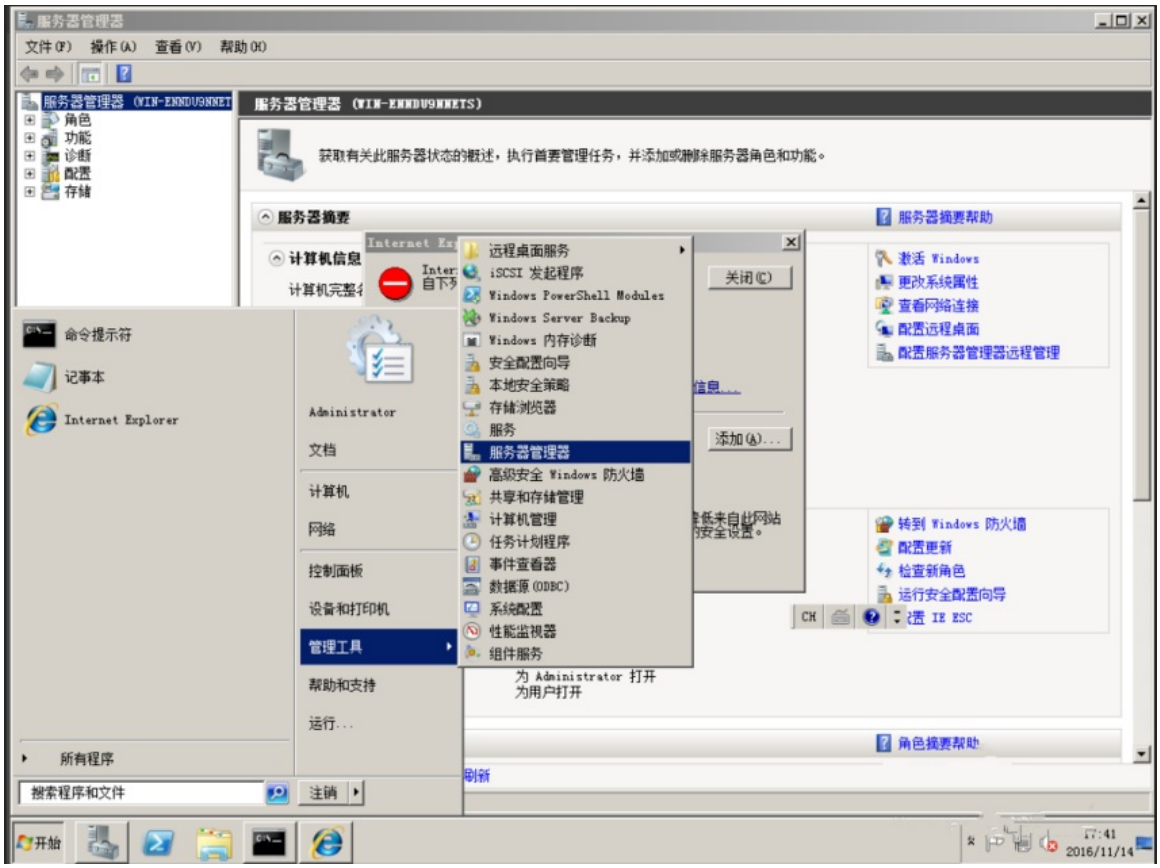
首先，我们打开IE，看到IE显示的警告信息，这就是增强IE所引起的。



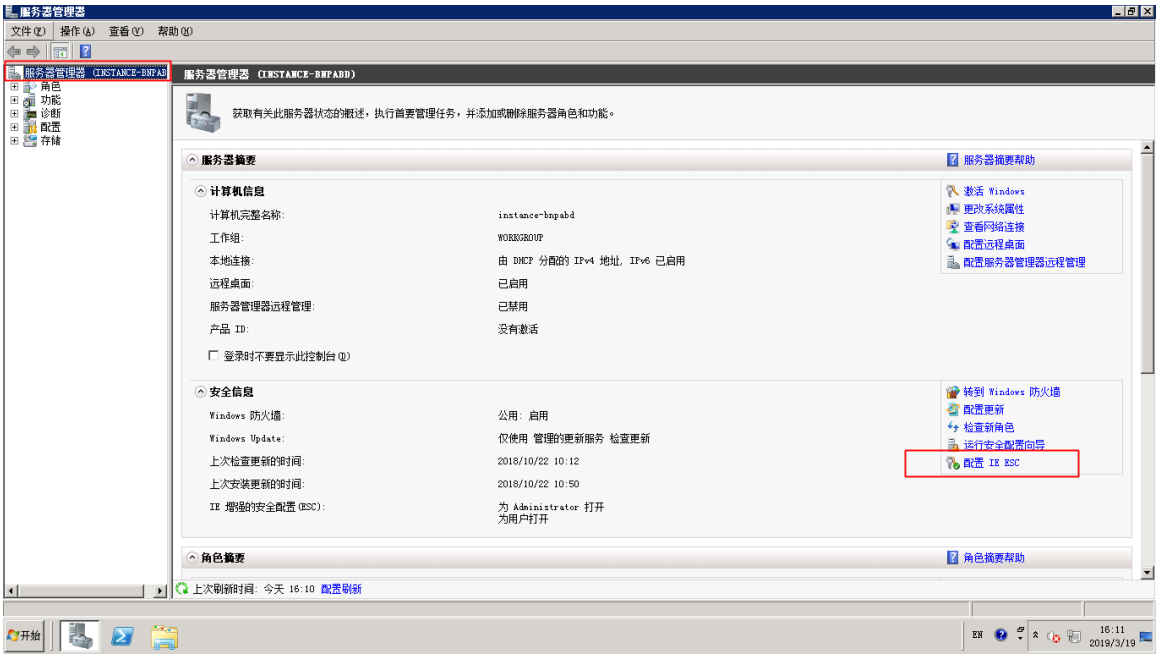
解决方法

Windows Server 2008

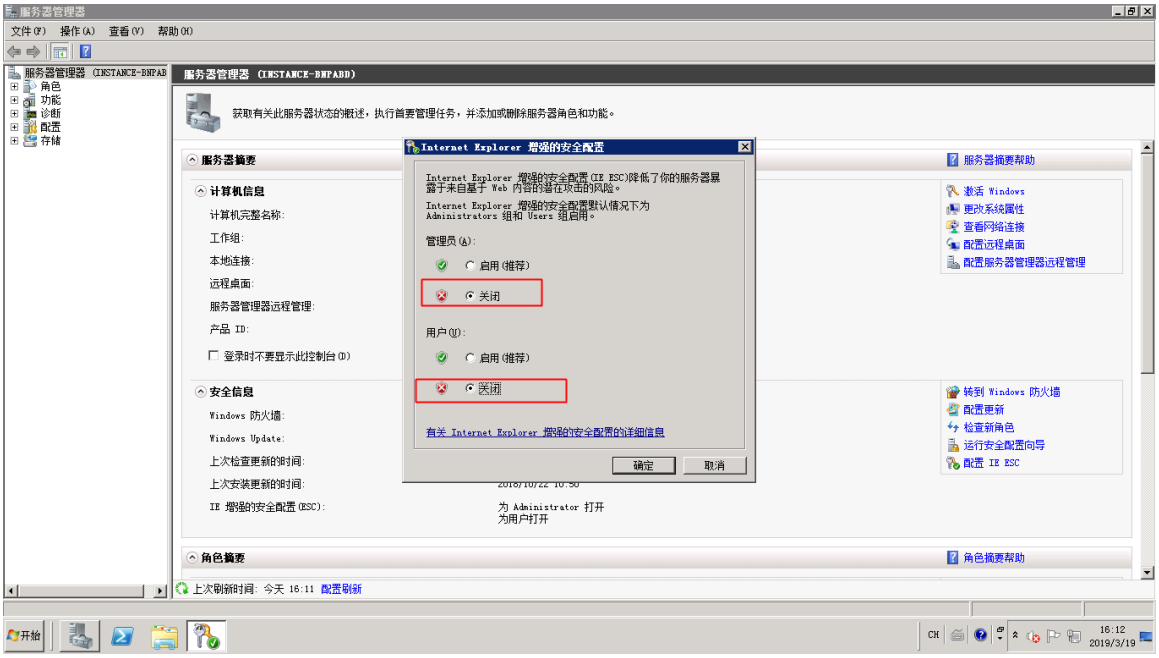
1.点击”WIN“打开"管理工具"，找到”服务器管理器“。



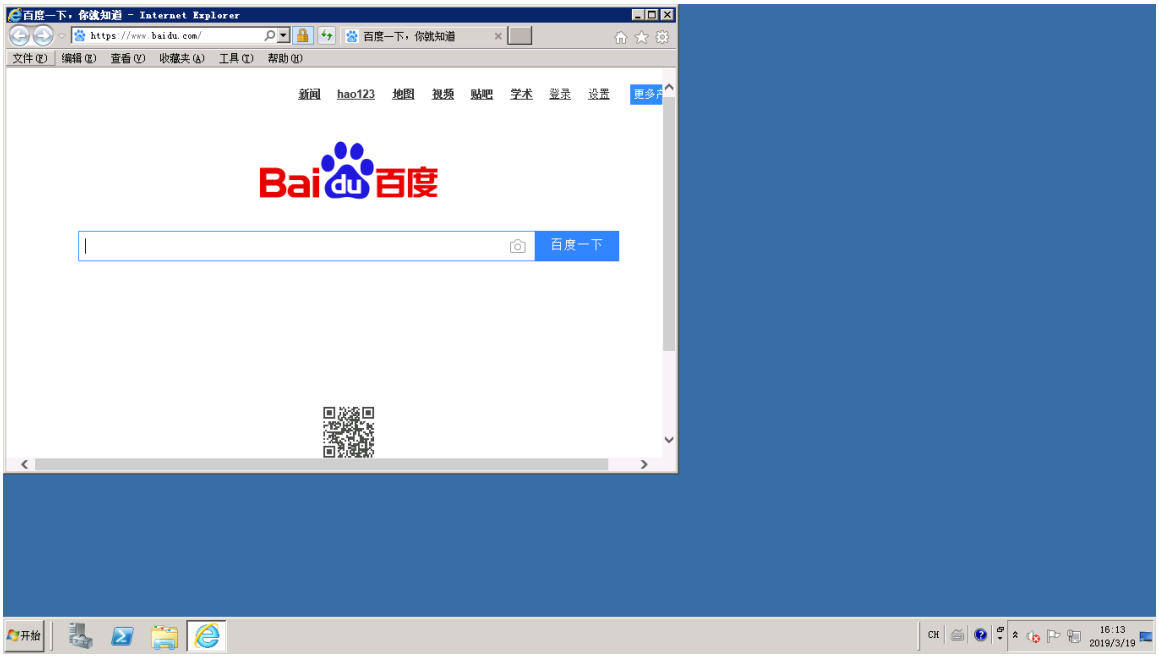
2.在”服务器管理器“中找到”配置IEESC”。



3.点击关闭即可。

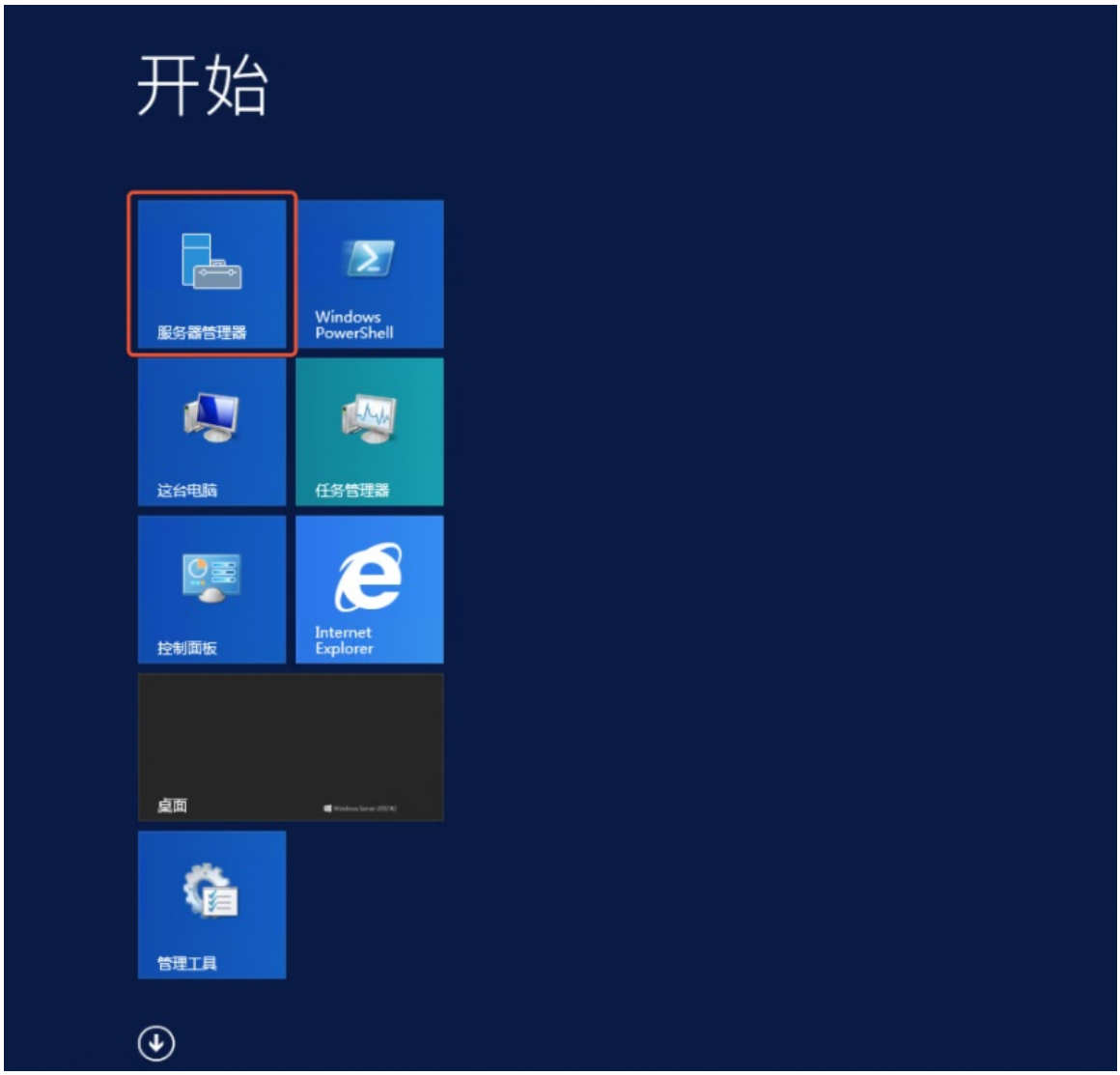


4.可以正常使用IE浏览器访问网站了。

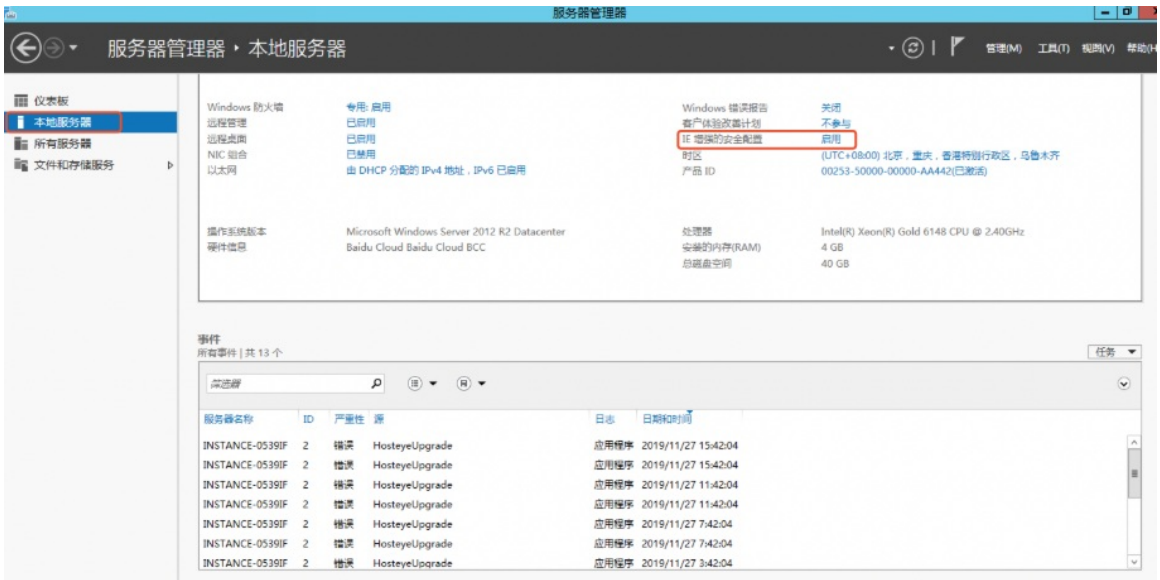


Windows Server 2012

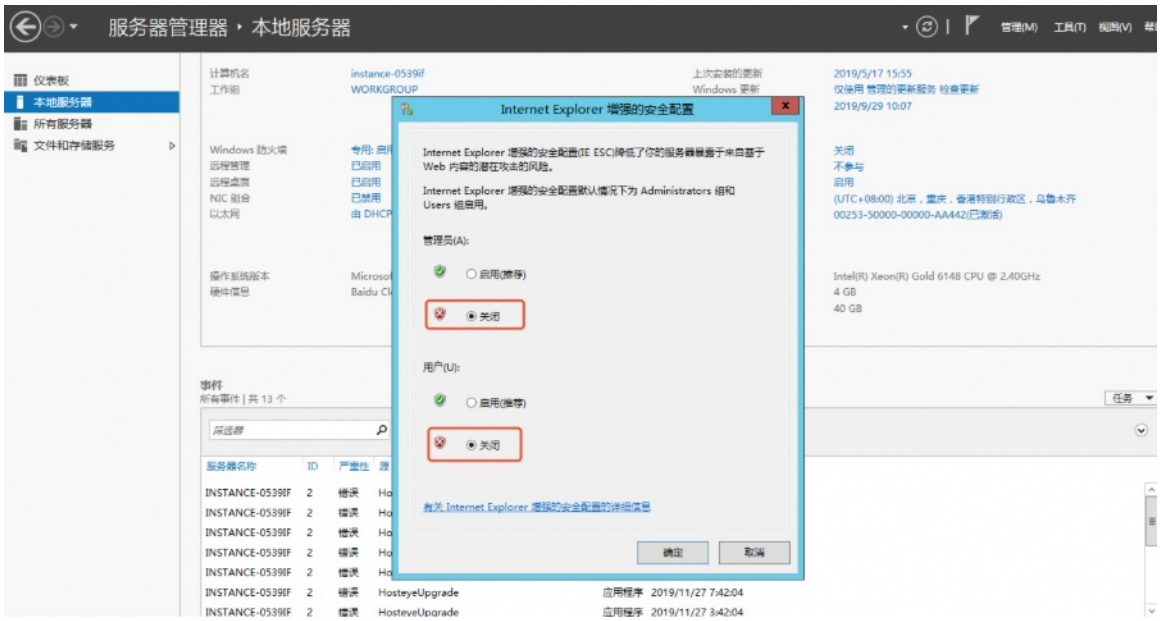
1.点击左下角的"WIN"，找到"服务器管理器"。



2.在”服务器管理器“中点击"本地服务器"，找到"IE增强的安全配置"。



3.点击"IE增强的安全配置"后面的"启用"，点击关闭即可。



4.完成上述步骤后，即可正常使用IE浏览器访问网站了。

使用xshell连接windows服务器

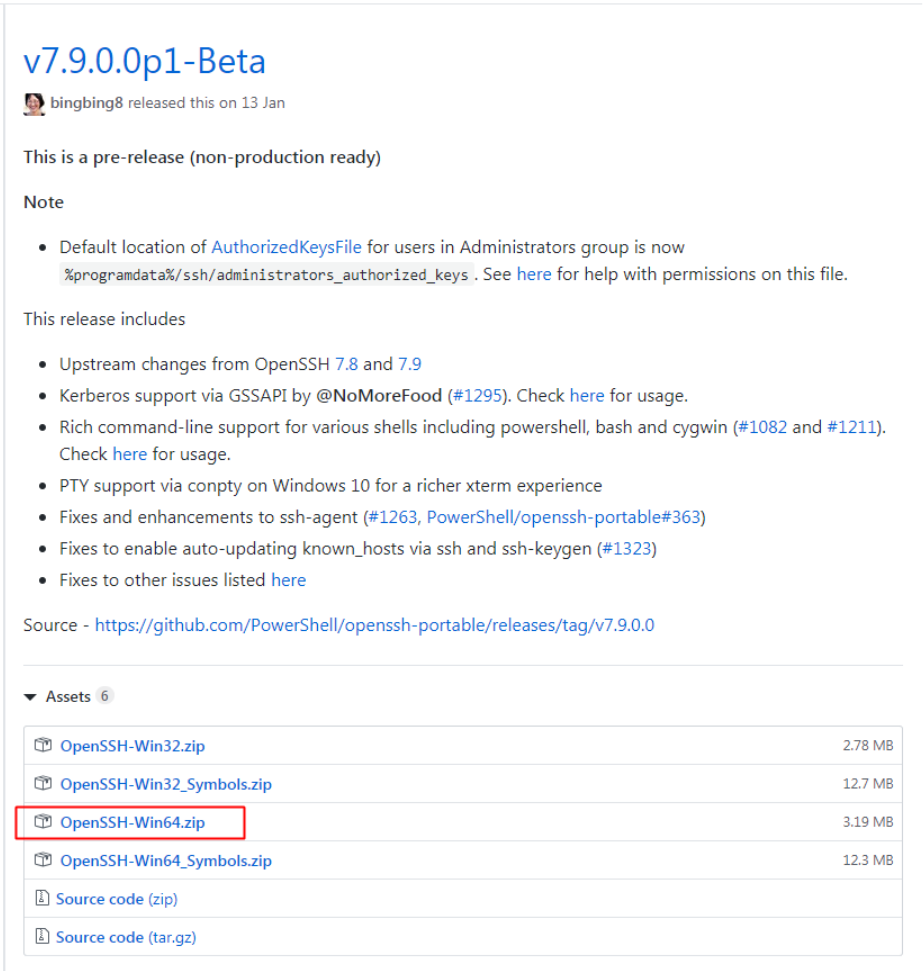
本教程主要讲述如何使用xshell连接Windows服务器，以Windows Server 2008为例。

第一步：远程登录服务器

远程登录服务器

第二步：下载对应服务端软件在BCC内进行解压并安装

1.登录该网页下载对应版本的OpenSSH。下载地址：https://github.com/PowerShell/Win32-OpenSSH/releases。



v7.9.0.0p1-Beta

bingbing8 released this on 13 Jan

This is a pre-release (non-production ready)

Note

- Default location of AuthorizedKeysFile for users in Administrators group is now `%programdata%/ssh/administrators_authorized_keys`. See here for help with permissions on this file.

This release includes

- Upstream changes from OpenSSH 7.8 and 7.9
- Kerberos support via GSSAPI by @NoMoreFood (#1295). Check here for usage.
- Rich command-line support for various shells including powershell, bash and cygwin (#1082 and #1211). Check here for usage.
- PTY support via conpty on Windows 10 for a richer xterm experience
- Fixes and enhancements to ssh-agent (#1263, PowerShell/openssh-portable#363)
- Fixes to enable auto-updating known_hosts via ssh and ssh-keygen (#1323)
- Fixes to other issues listed here

Source - https://github.com/PowerShell/openssh-portable/releases/tag/v7.9.0.0

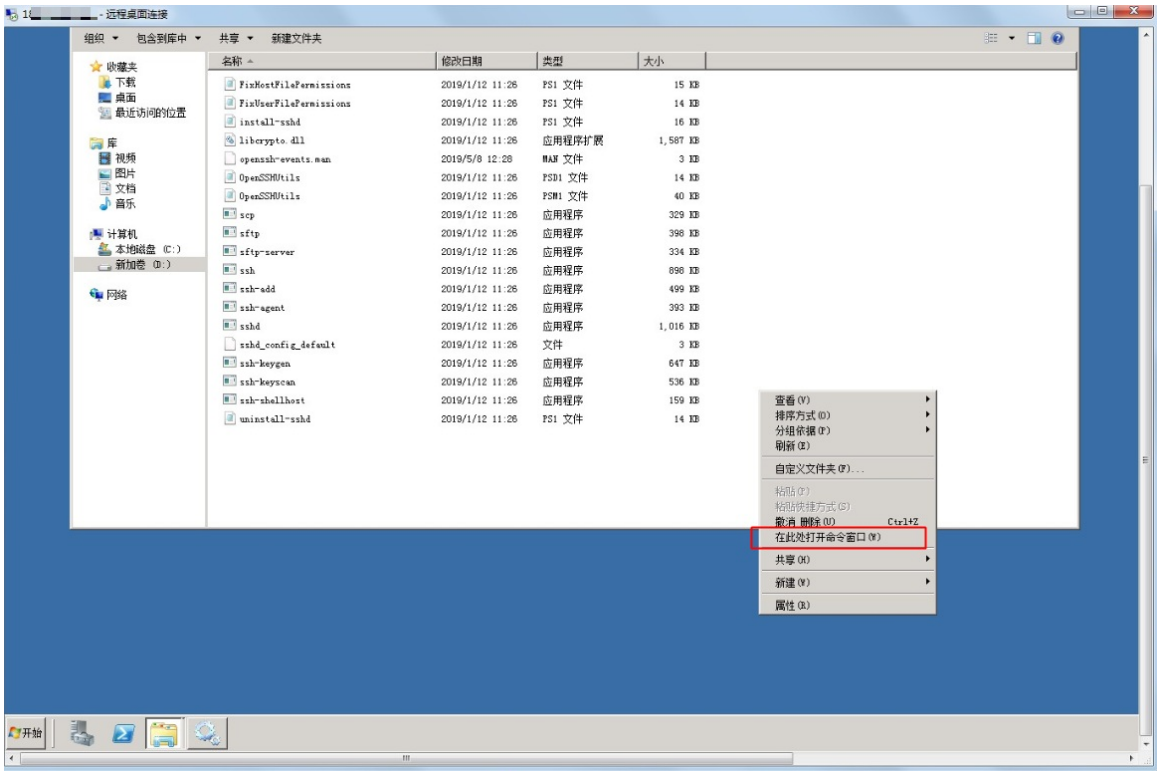
Assets 6

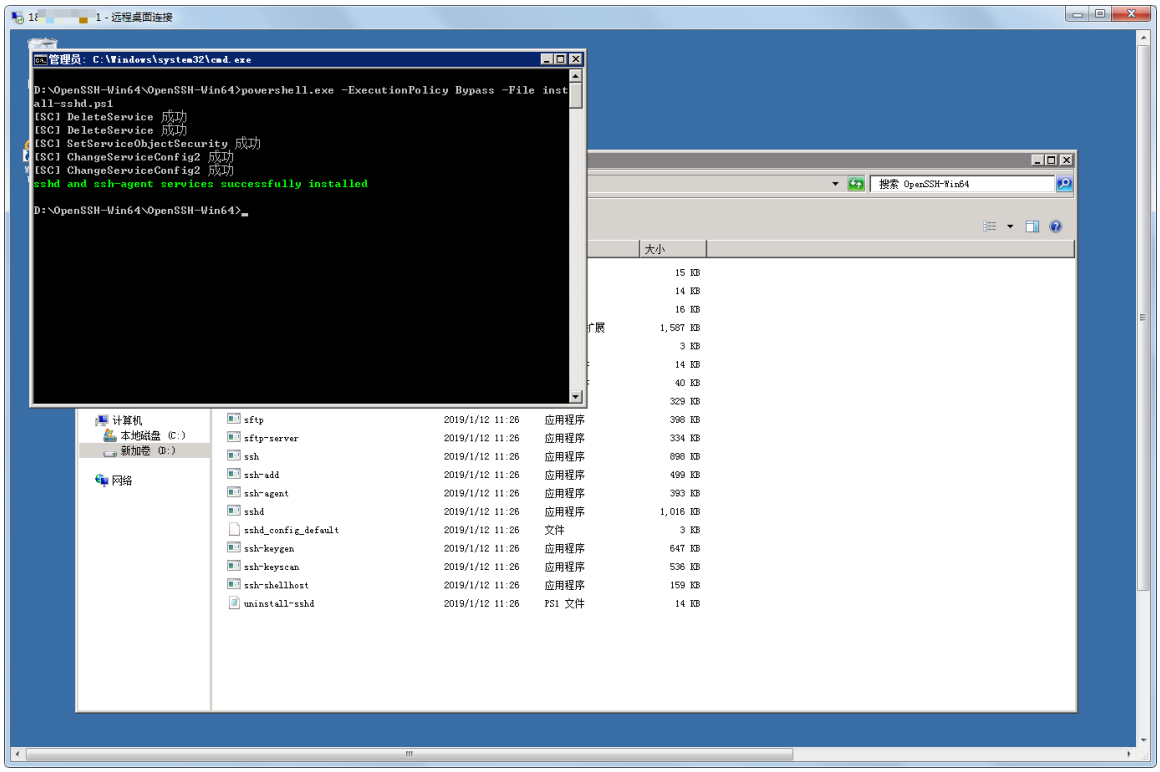
OpenSSH-Win32.zip	2.78 MB
OpenSSH-Win32_Symbols.zip	12.7 MB
OpenSSH-Win64.zip	3.19 MB
OpenSSH-Win64_Symbols.zip	12.3 MB
Source code (zip)	
Source code (tar.gz)	

2.上传该文件到windwos服务器并解压。

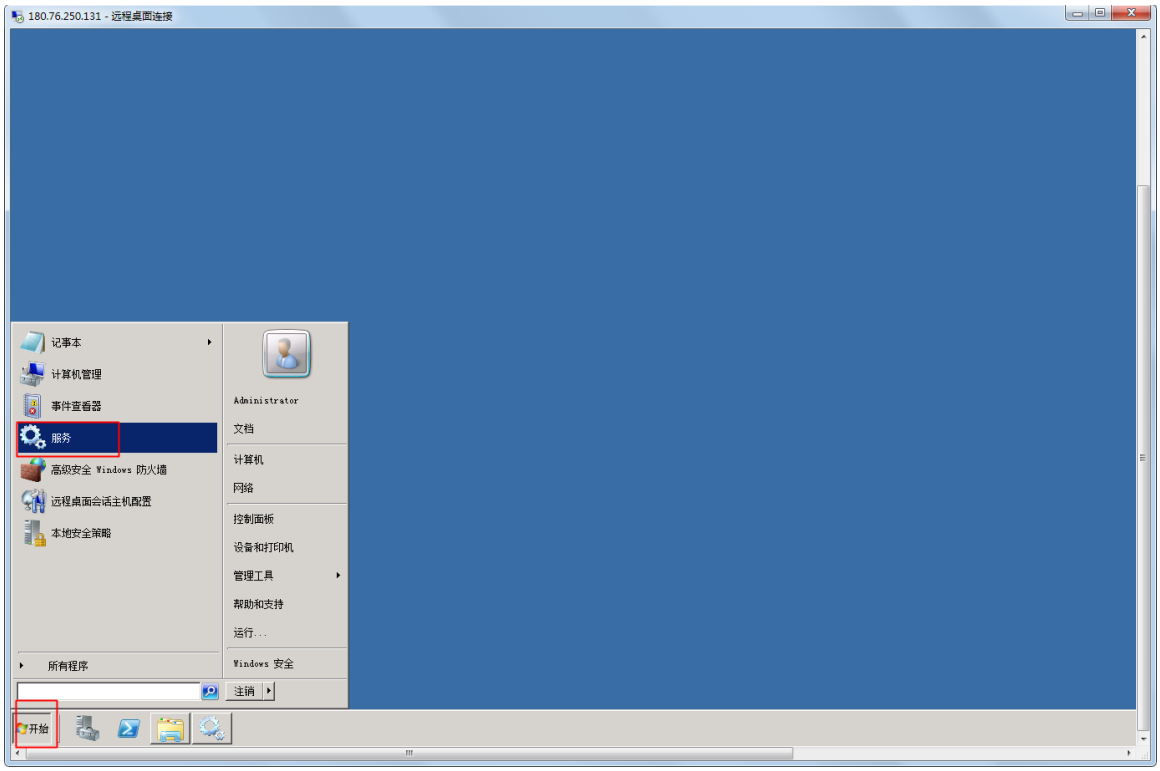
3.在解压后的文件夹内按住shift，同时鼠标右键，打开cmd，执行：

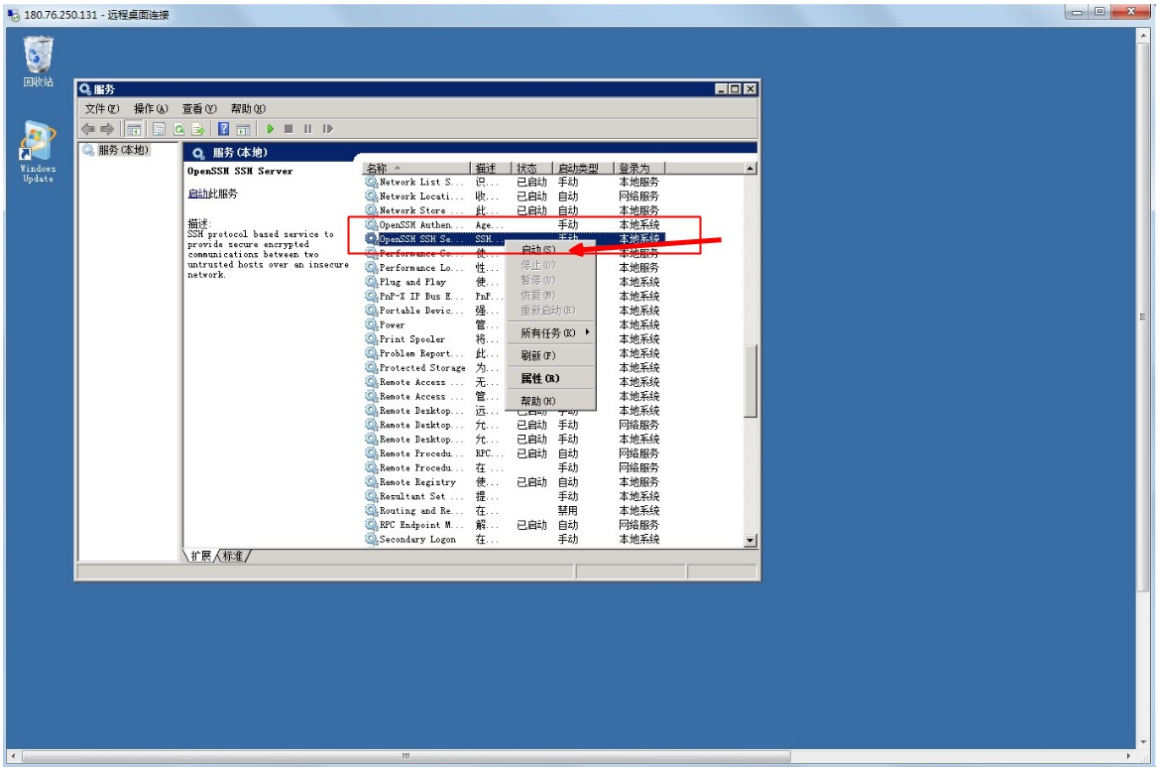
```
powershell.exe -ExecutionPolicy Bypass -File install-sshd.ps1
```

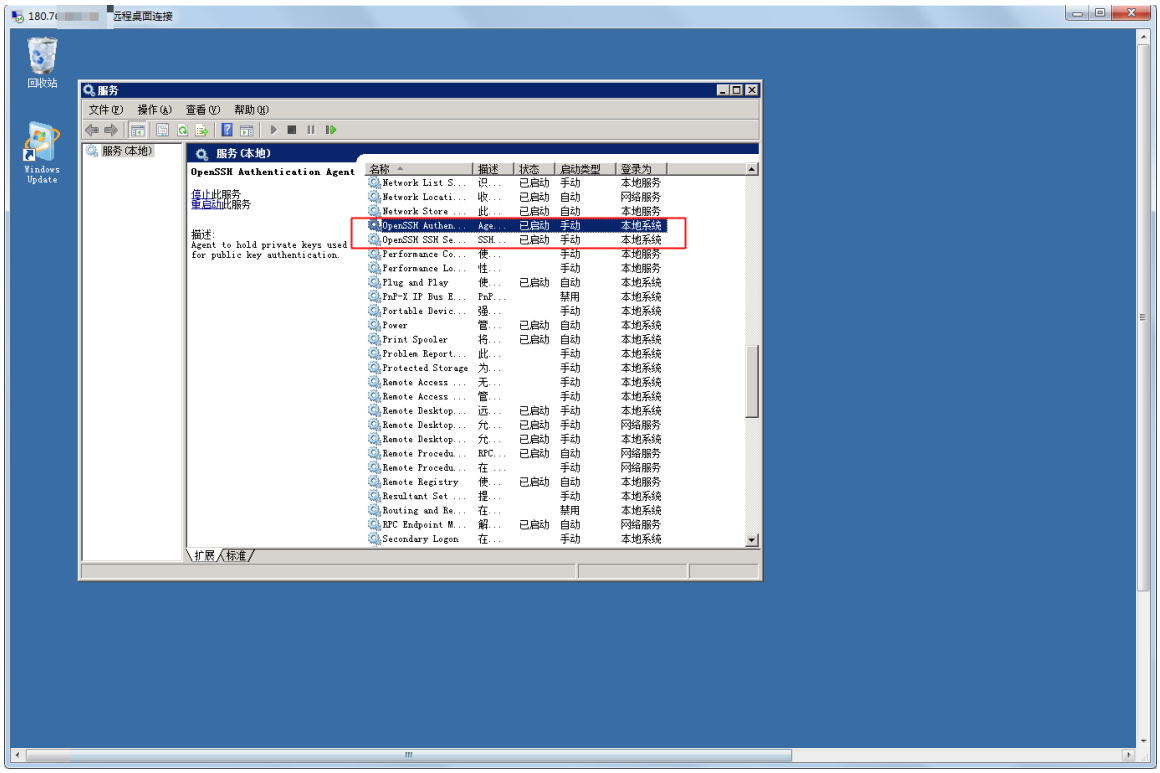




4.点击开始-服务-找到OpenSSH Authen...和OpenSSH SSH Se...这两个服务。右键点击启动。

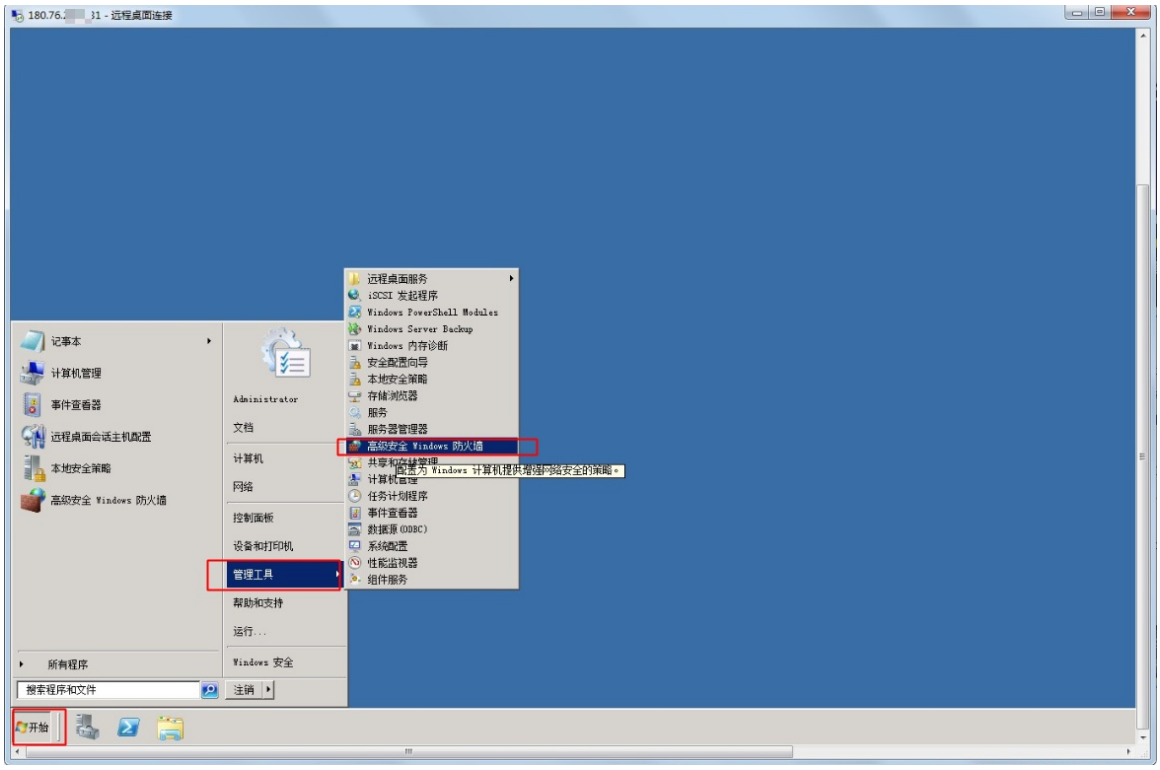

180.76.250.131 - 远程桌面连接
记事本
计算机管理
事件查看器
服务
高级安全 Windows 防火墙
远程桌面会话主机配置
本地安全策略
Administrator
文档
计算机
网络
控制面板
设备和打印机
管理工具
帮助和支持
运行...
Windows 安全
所有程序
注销
开始


180.76.250.131 - 远程桌面连接
回收站
Windows Update
服务
文件(F) 操作(A) 查看(V) 帮助(H)
服务(本地)
OpenSSH SSH Server
启动此服务
描述:
SSH protocol based service to provide secure encrypted communications between two untrusted hosts over an insecure network.
名称 描述 状态 启动类型 登录为
OpenSSH SSH Se... SSH... 本地系统
启动(S)
停止(O)
暂停(U)
恢复(M)
重新启动(E)
所有任务(K)
刷新(F)
属性(R)
帮助(H)
扩展 标准

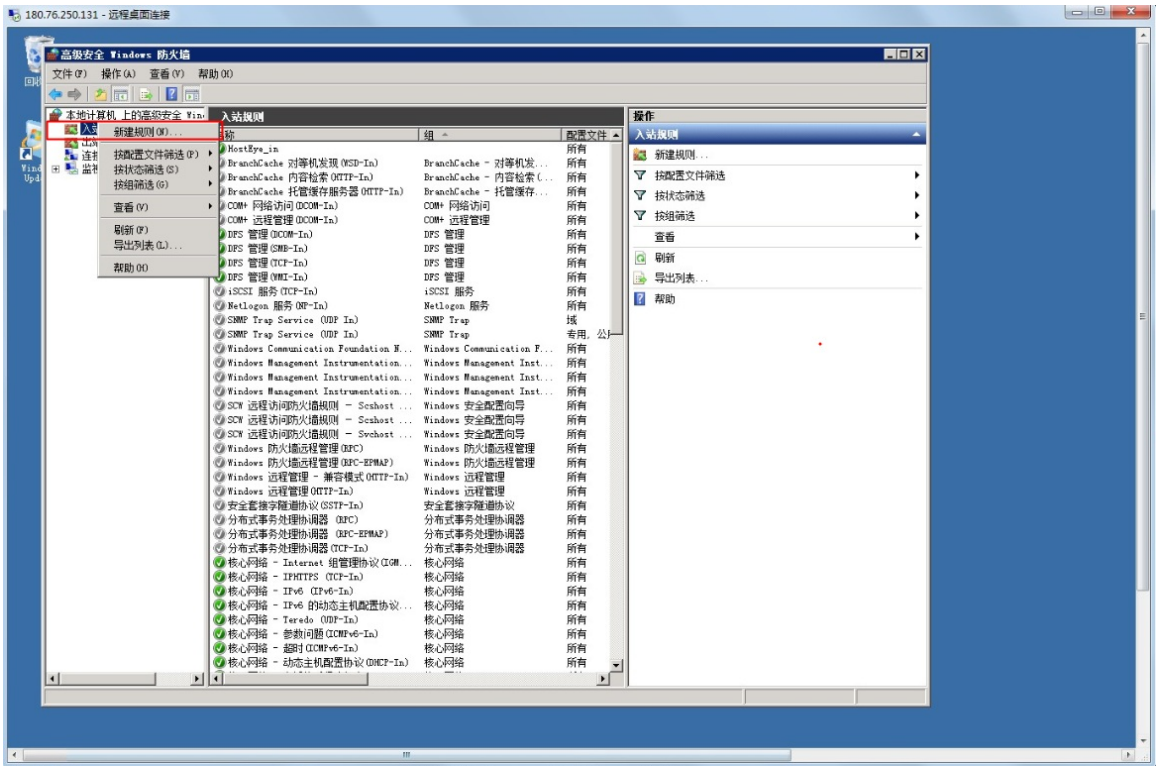


第三步：防火墙放行22端口

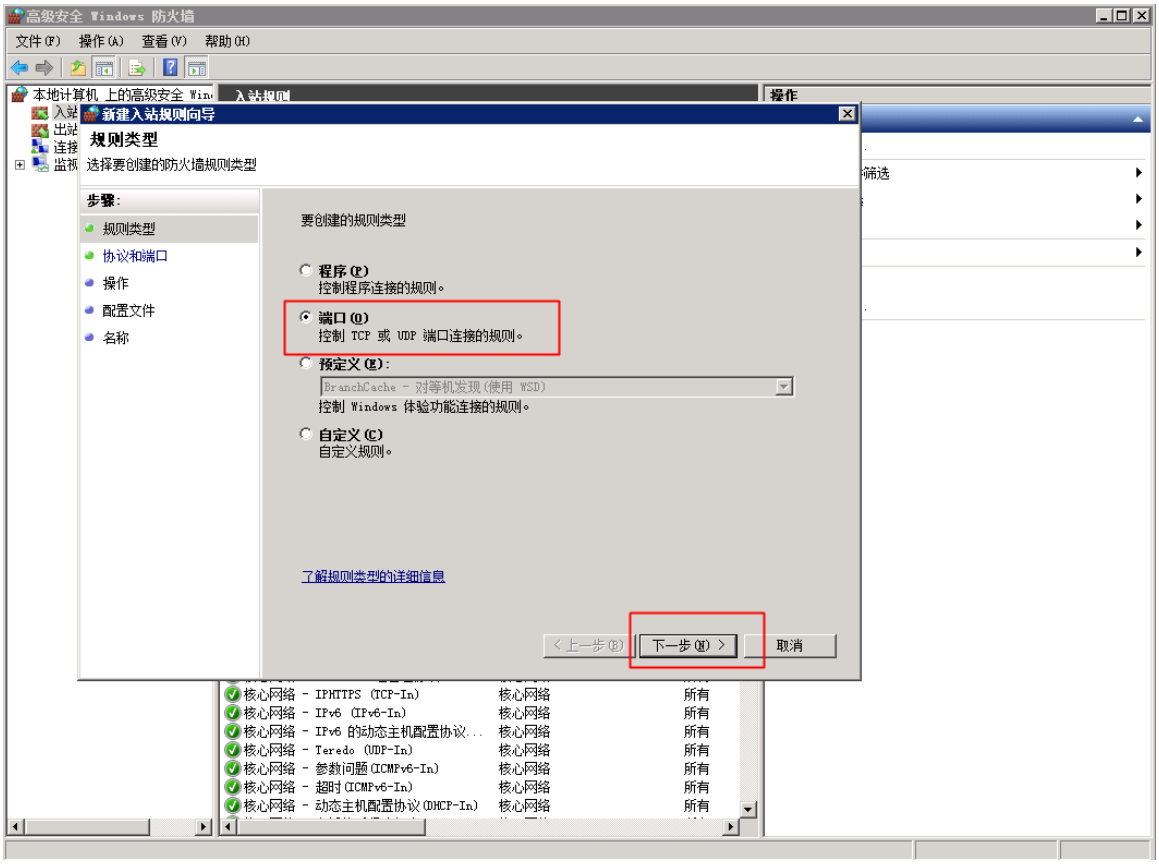
1.在服务器内点击-开始-管理工具-高级安全windows防火墙。



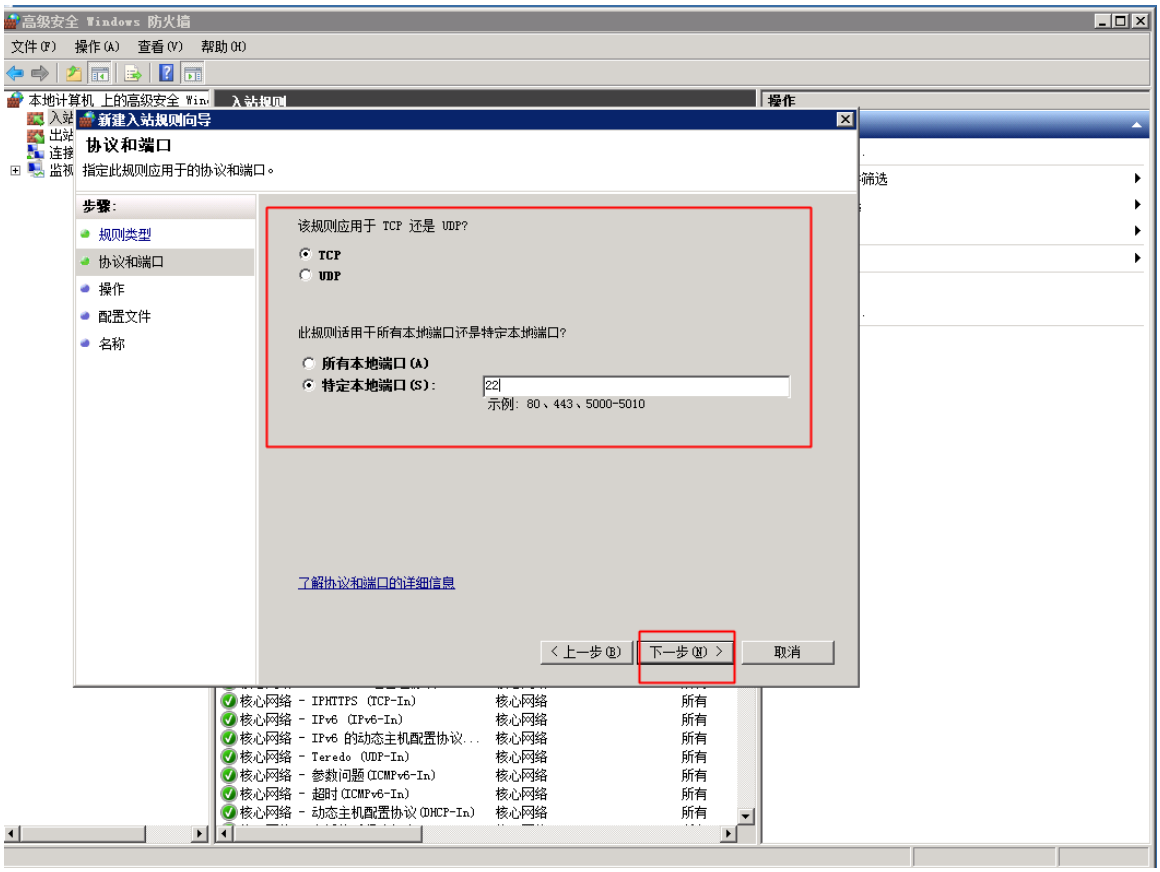
2.右击入站规则-新建规则。



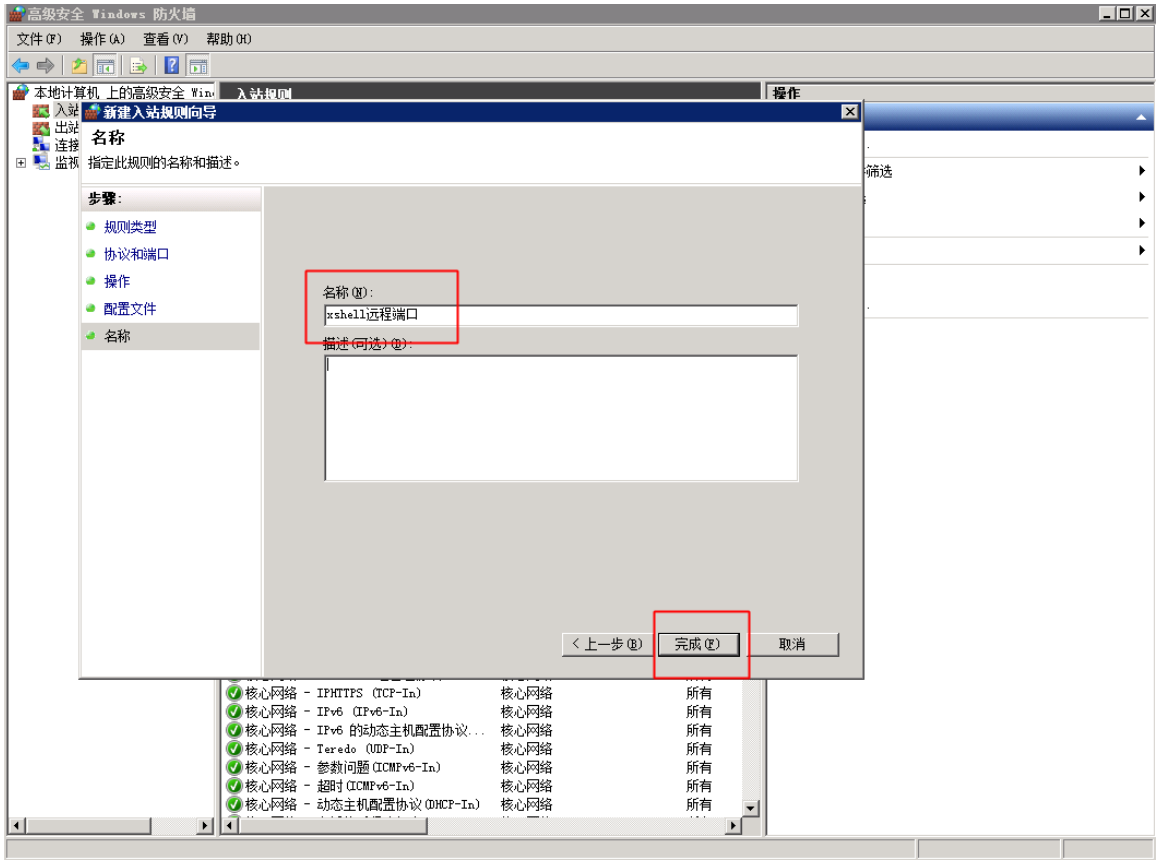
3.选择端口-点击下一步。



4.选择TCP-特定本地端口-端口内填写22-点击下一步。

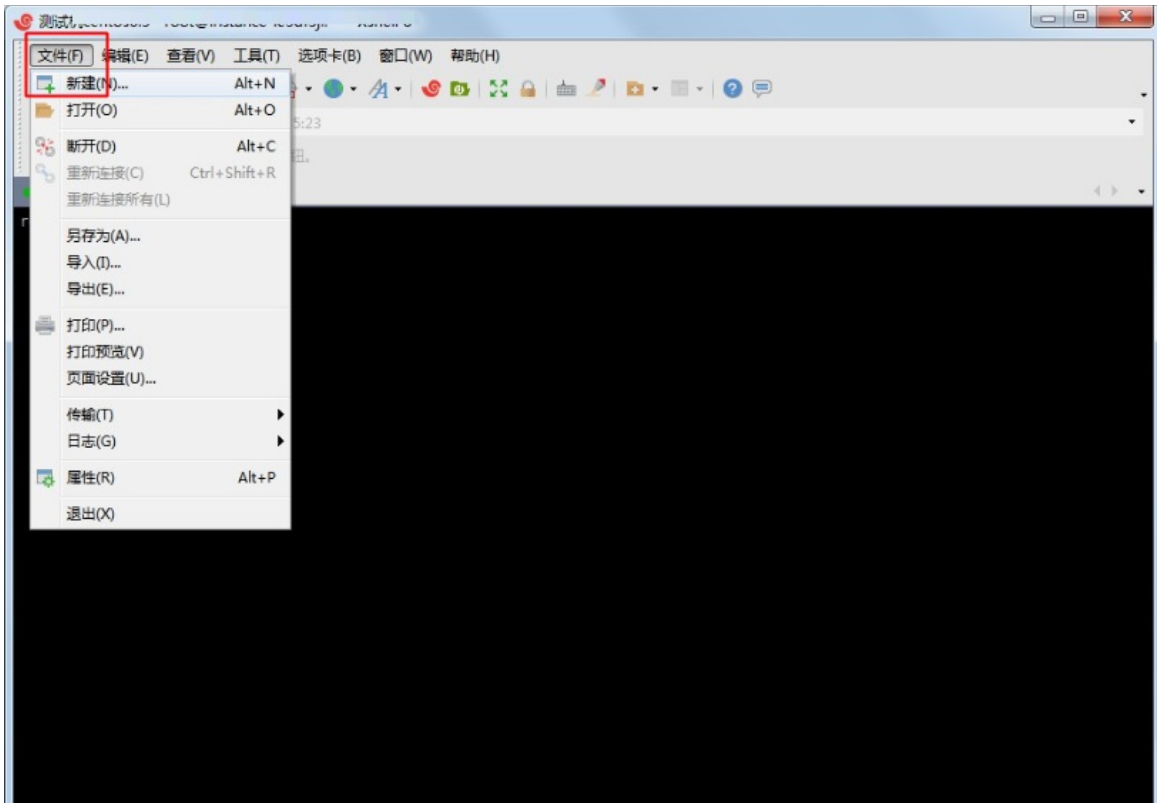


5.一直点击下一步直到”名称“环节，名称中填写xshell远程端口，之后点击完成即可。

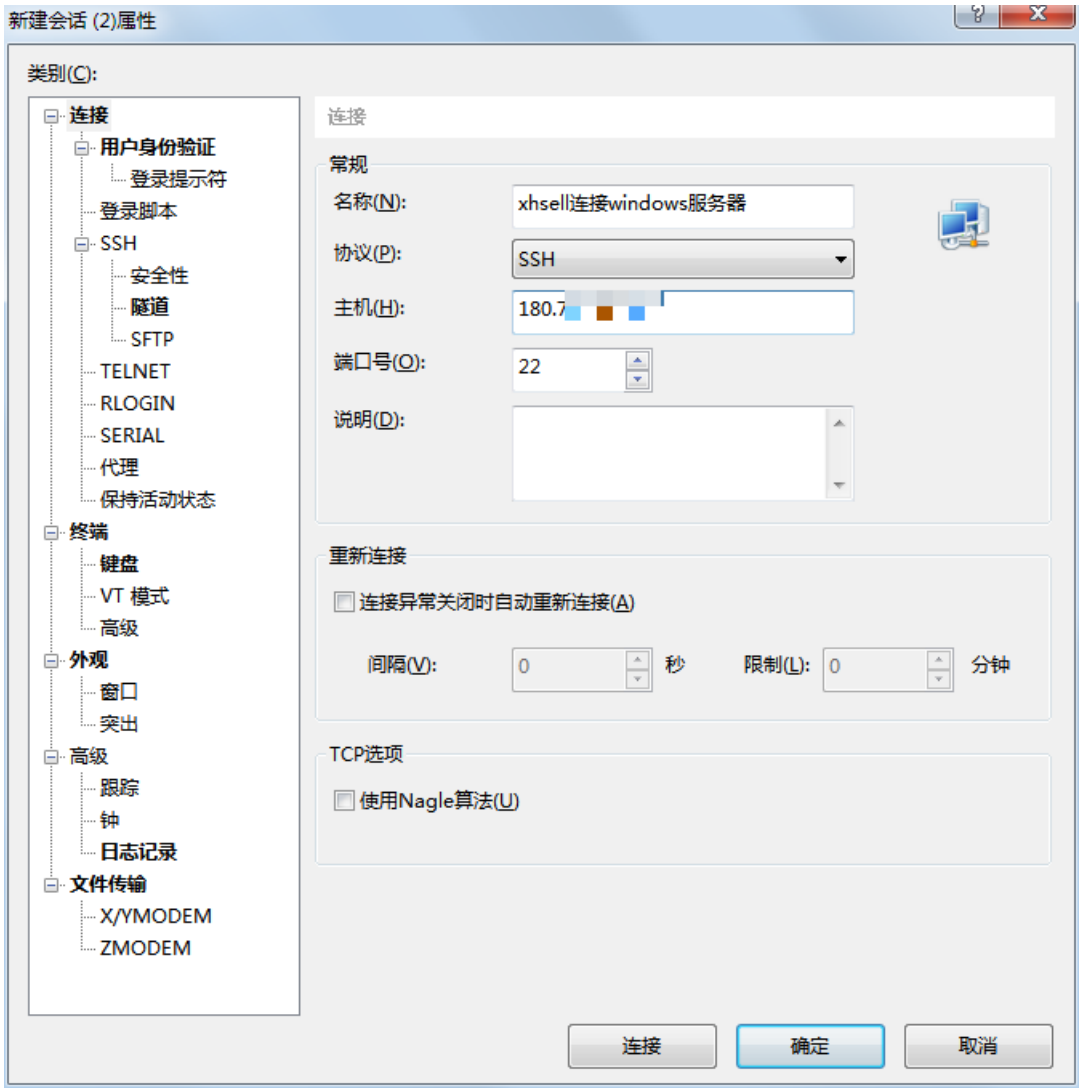


第四步：使用xshell连接windows服务器

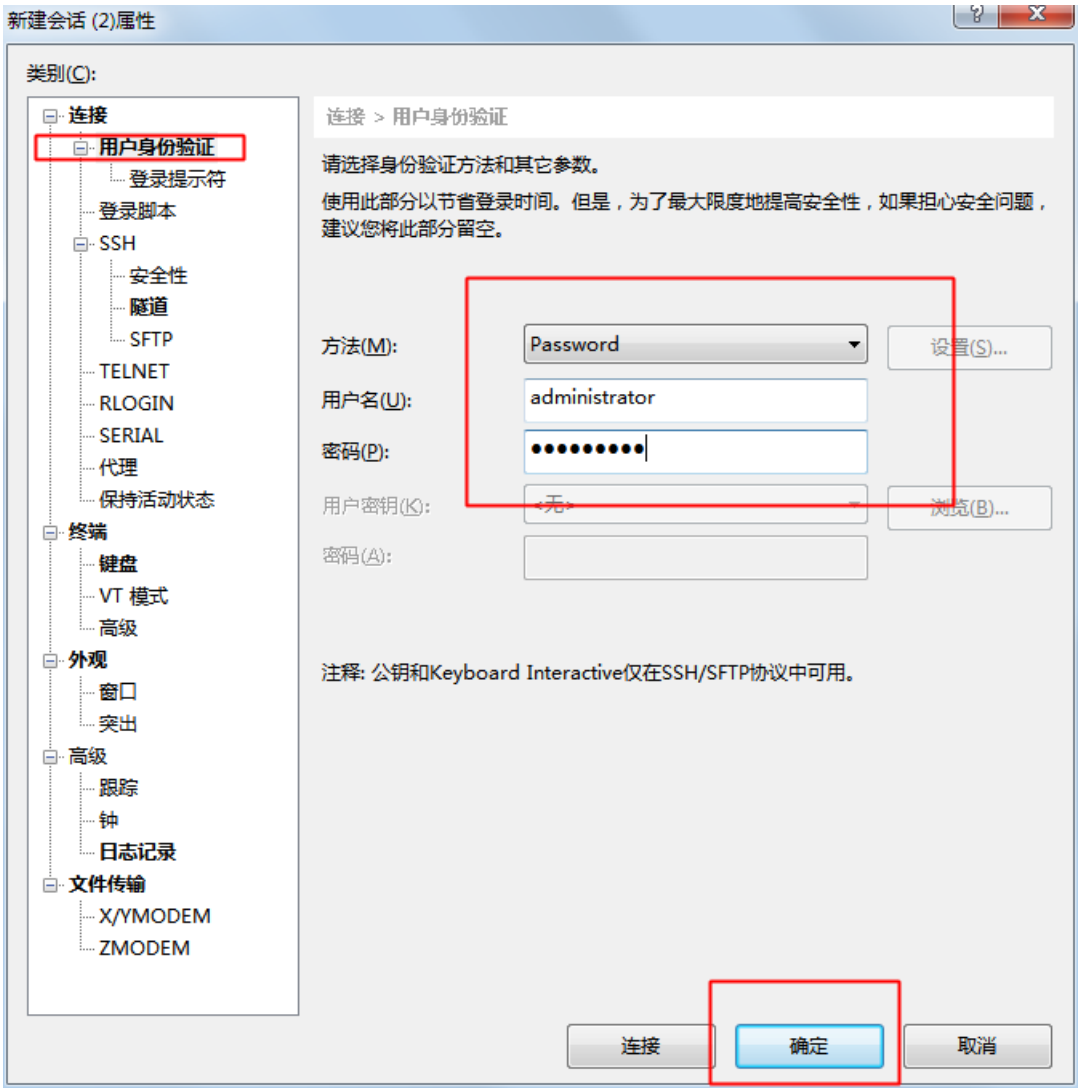
1.在xshell左上角-文件-新建。



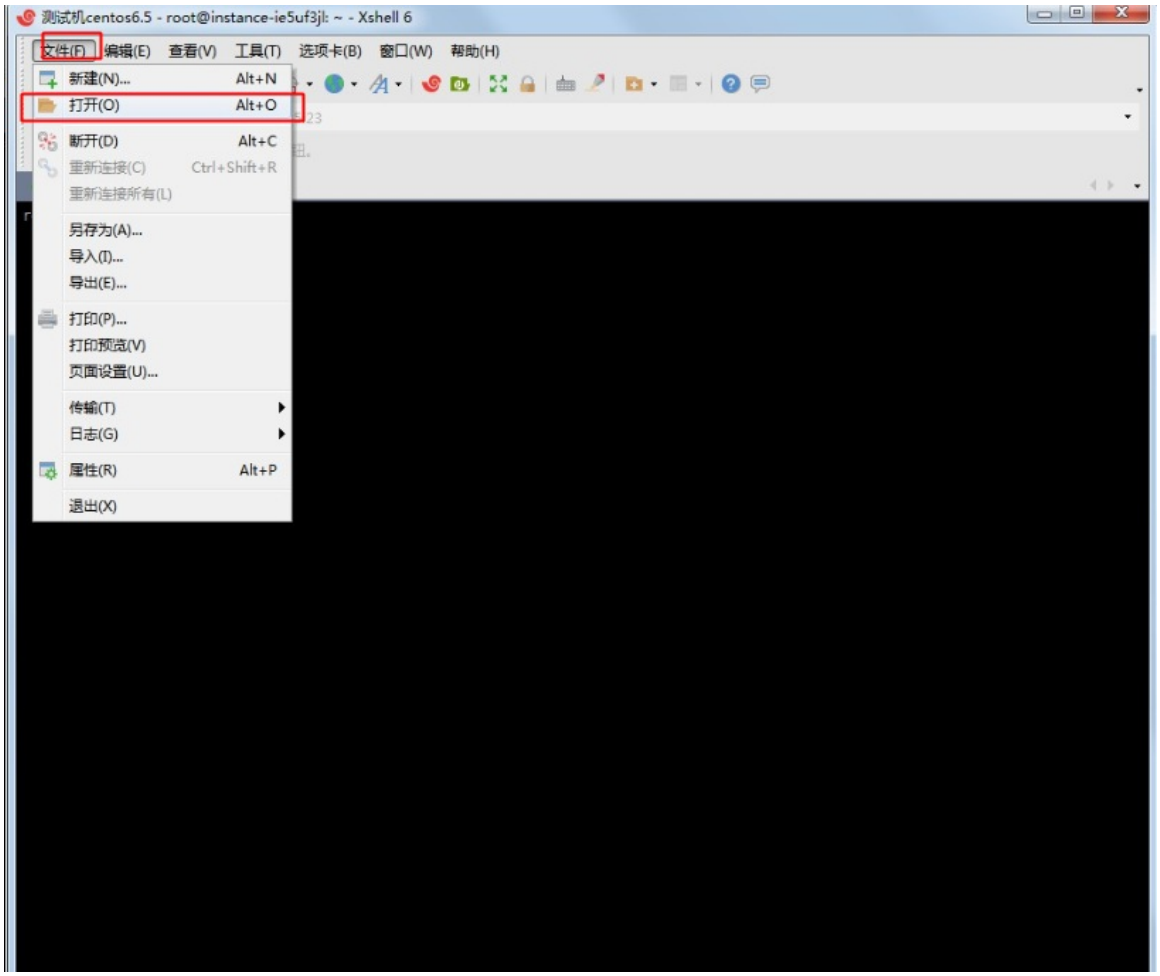
2.在会话中设置对应名称和对应的主机IP（服务器的IP地址）。



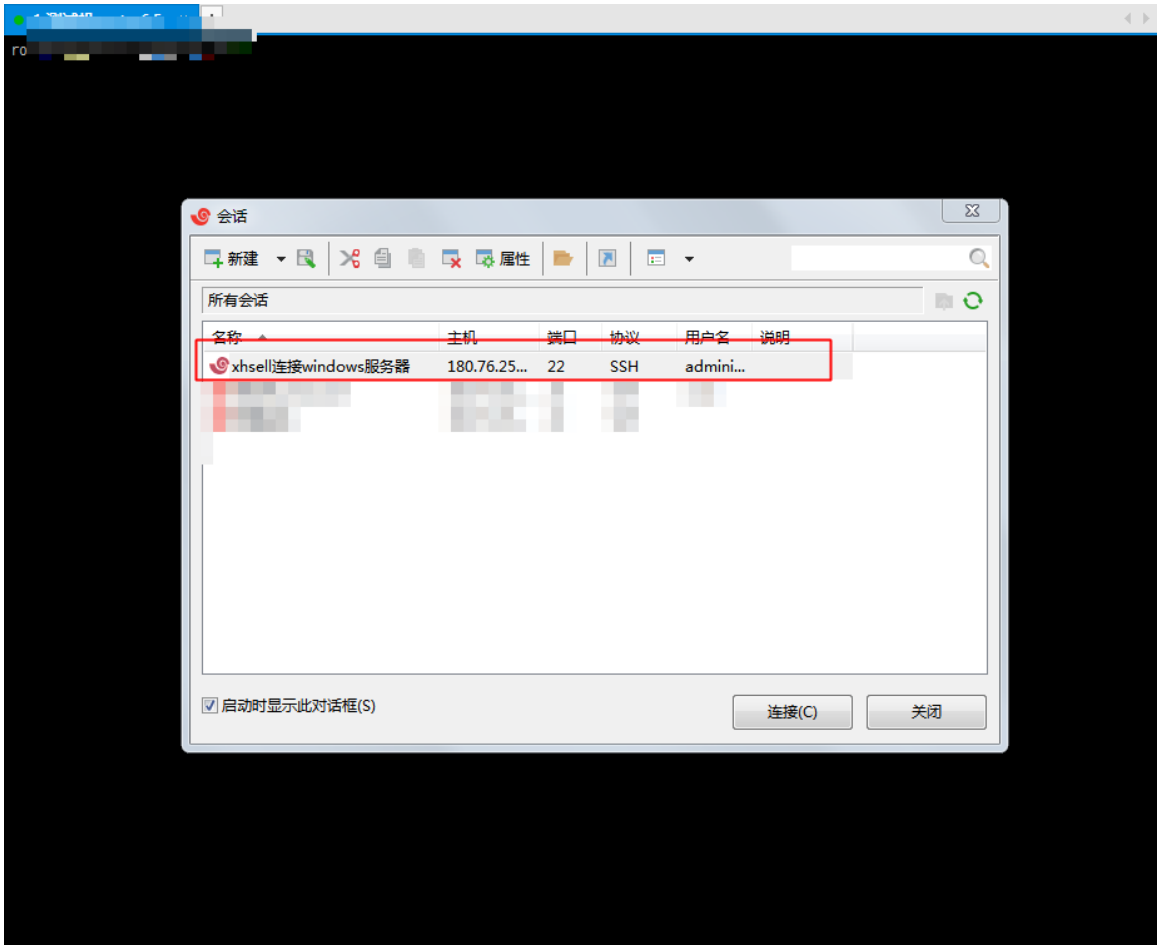
3.点击用户身份验证，输入对应的远程连接账号密码之后点击确定。



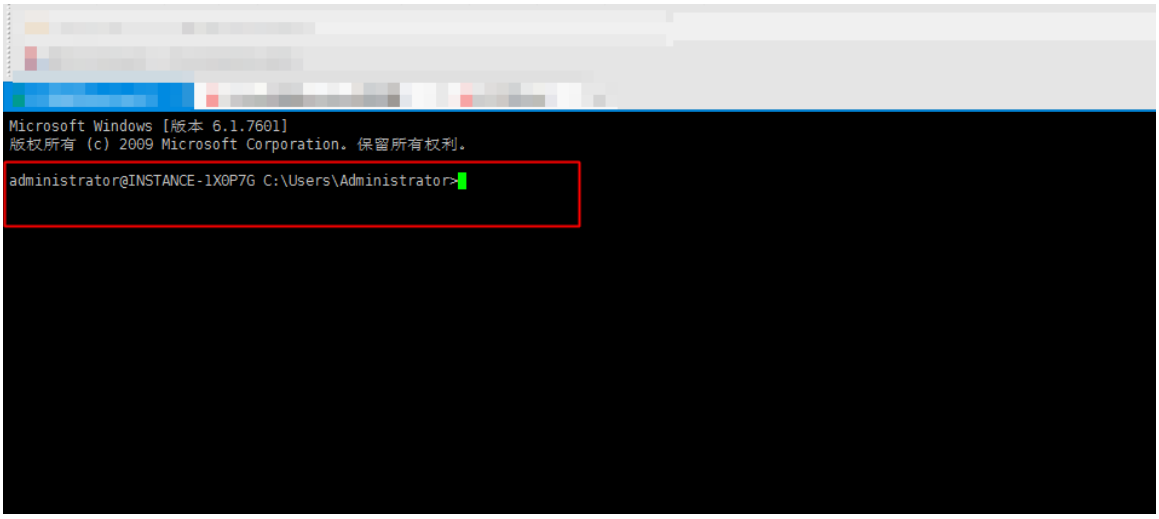
4.点击文件-打开进入会话页面。



5.双击之前创建的会话连接即可。



6.测试成功连接。



关闭windows防火墙教程

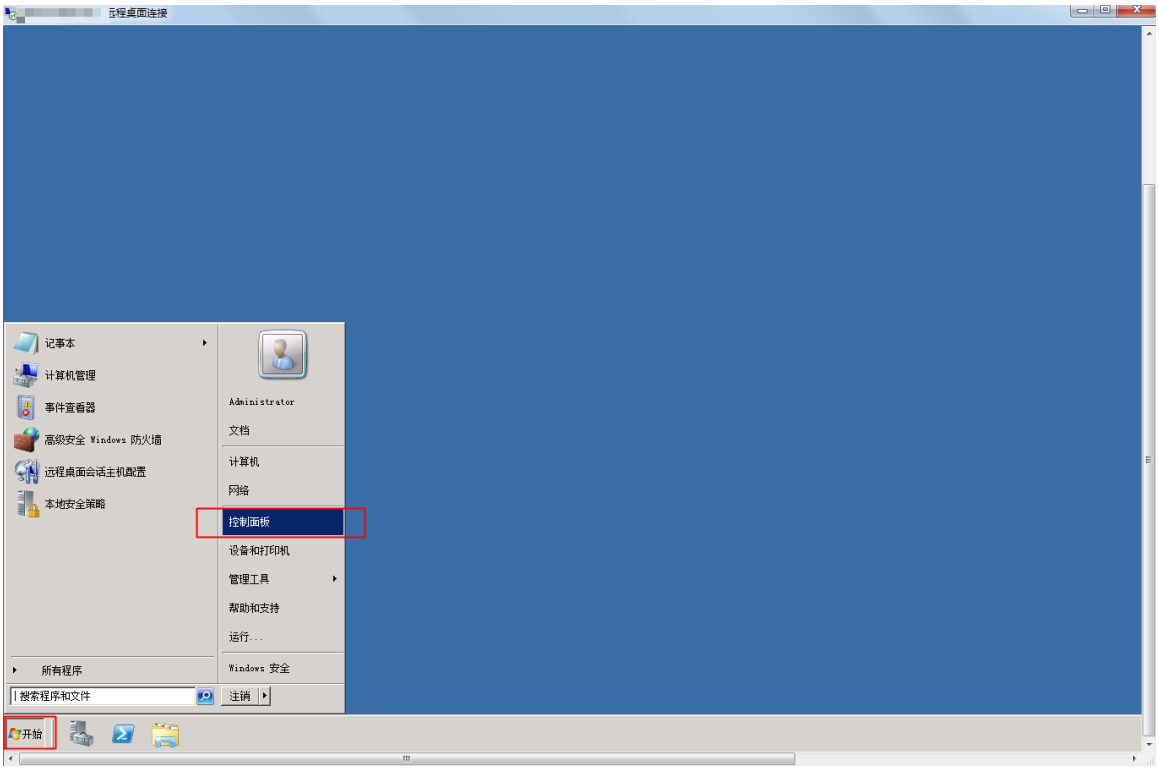
本文以windows2008为例，介绍了如何关闭windows自带防火墙。

第一步：远程登录服务器

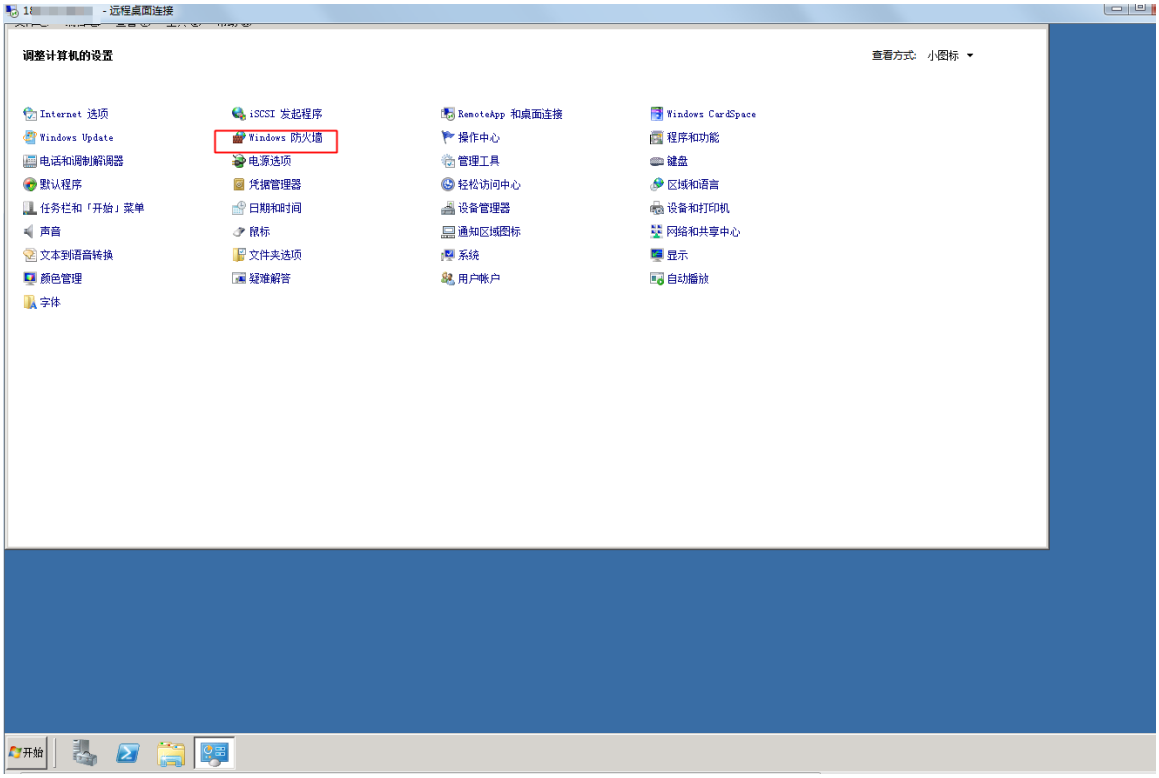
远程登录服务器

第二步:关闭windows防火墙

1.点击开始-控制面板。

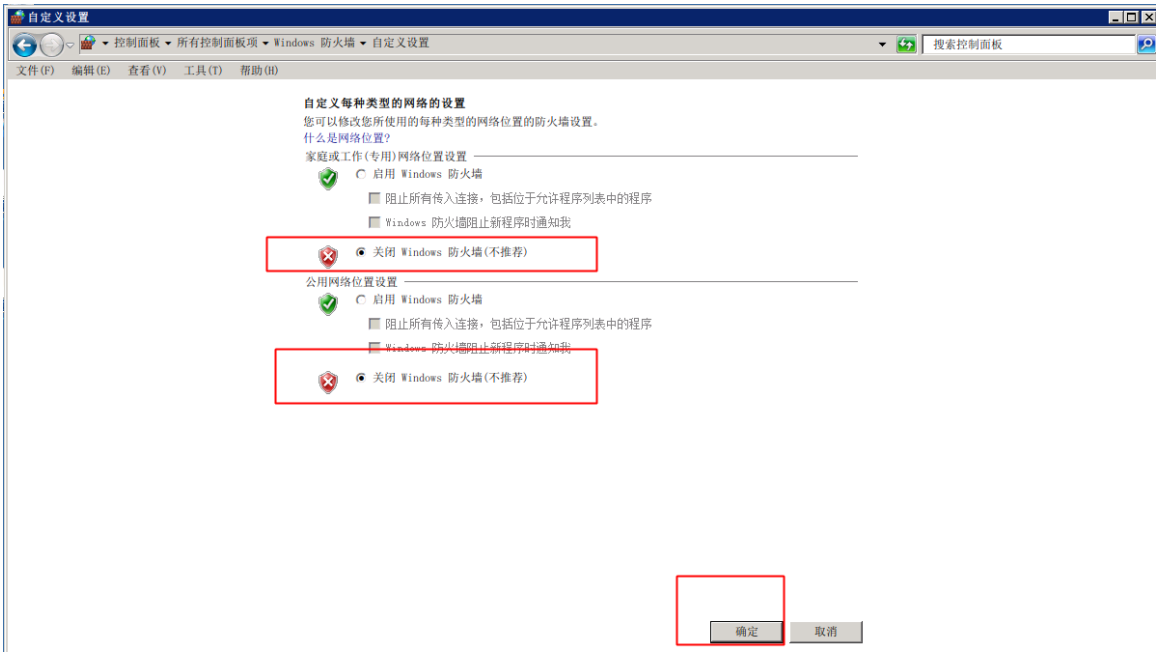


2.点击windows防火墙。



3.点击打开或关闭防火墙。

4.选择关闭windows防火墙点击**确定**。



windows设置只允许本地&固定IP访问远程端口

本篇教程针对需要设置只允许客户本地固定一个或多个IP访问远程端口。

windows设置本地&固定IP访问远程端口，本教程一共分为四步：

第一步：远程登录服务器

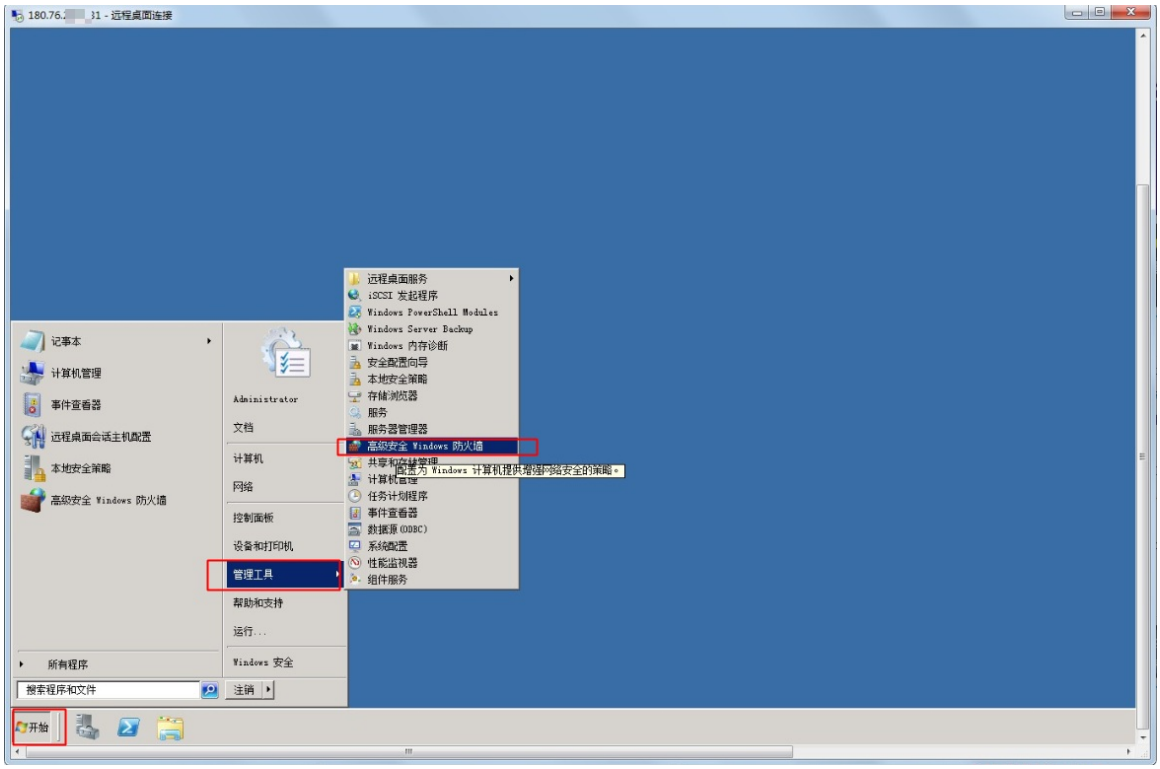
第二步：获取自身本地IP地址

1.在本地电脑打开浏览器，访问百度首页并搜索”IP“关键字。

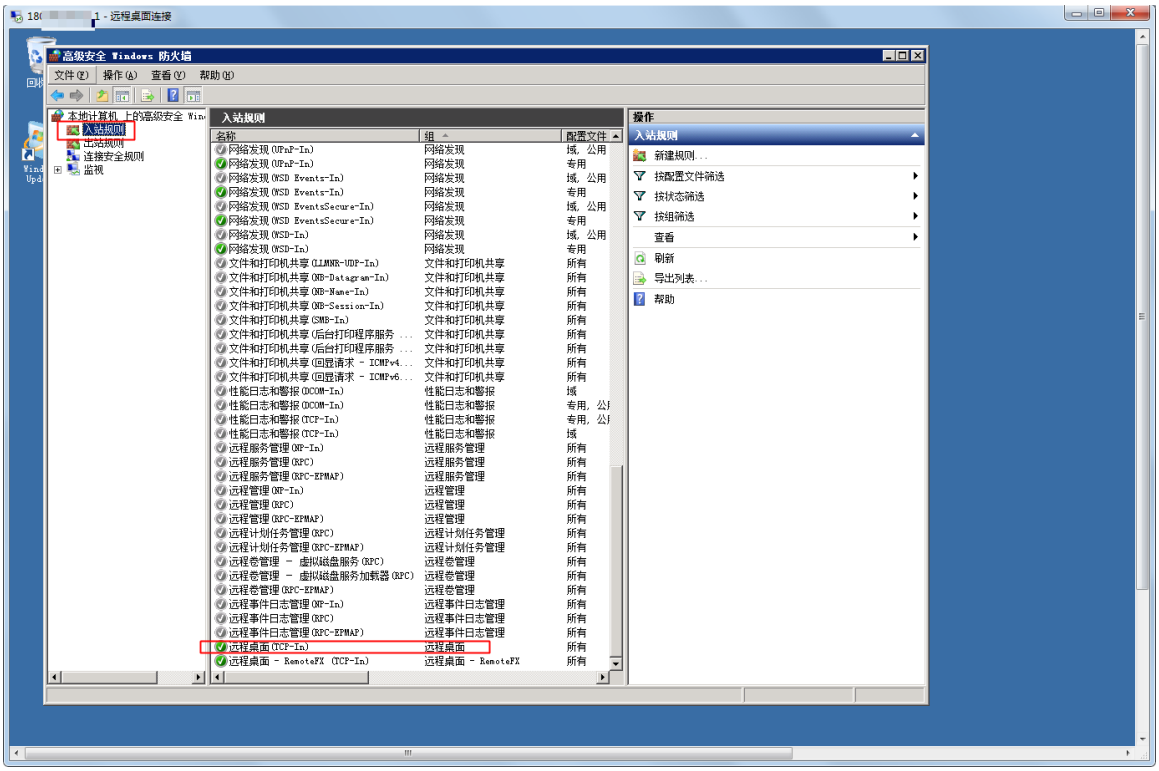
2.从搜索结果第一行中即可看到对应本地IP。（图片中的IP仅供参考，每个人的IP地址不相同，请以您自行搜索到的IP为准）

第三步：使用防火墙设置只允许固定IP访问服务器远程端口

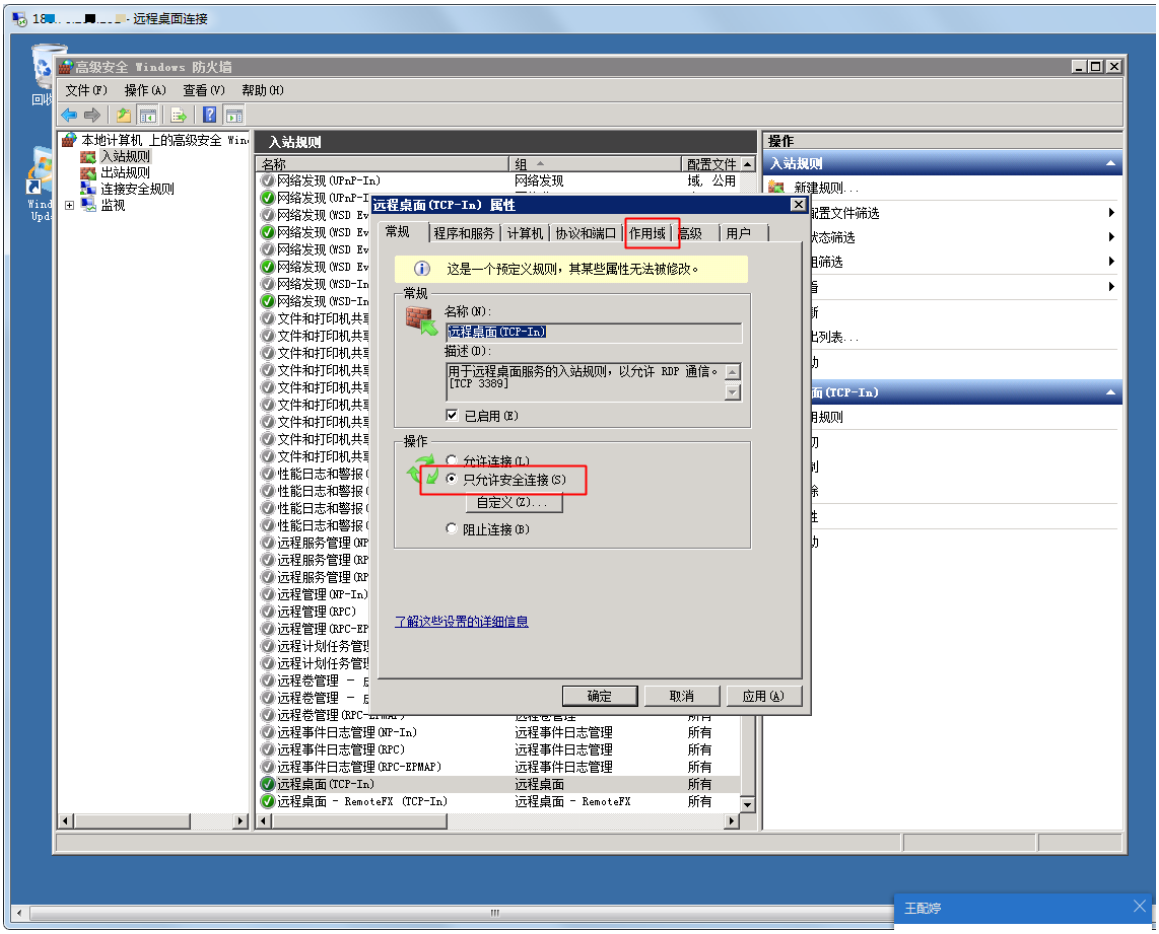
1.在服务器内点击-开始-管理工具-高级安全Windows防火墙。



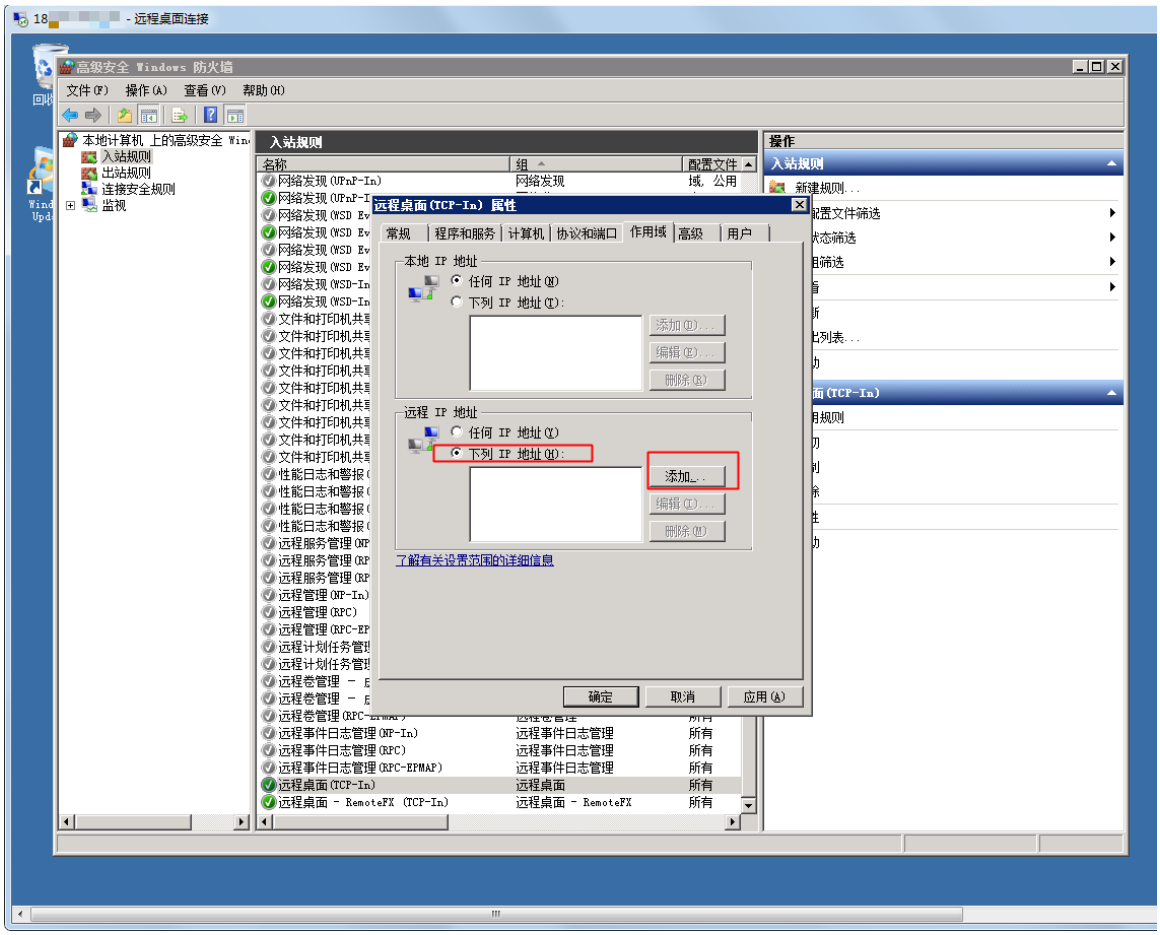
2.点击入站规则-找到”远程桌面（TCP-In）“双击打开设置规则



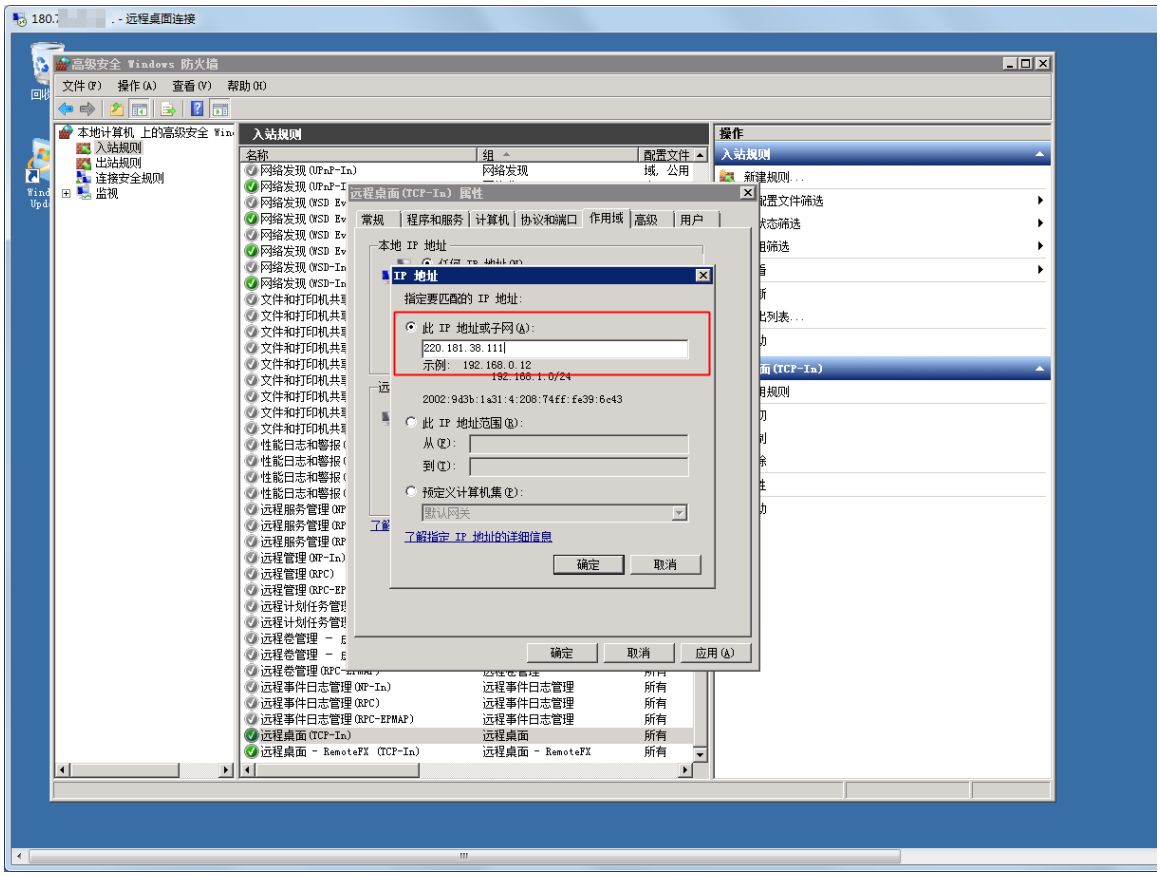
3.设置为只允许安全连接-之后点击作用域



4.选择远程IP地址-下列IP地址-添加

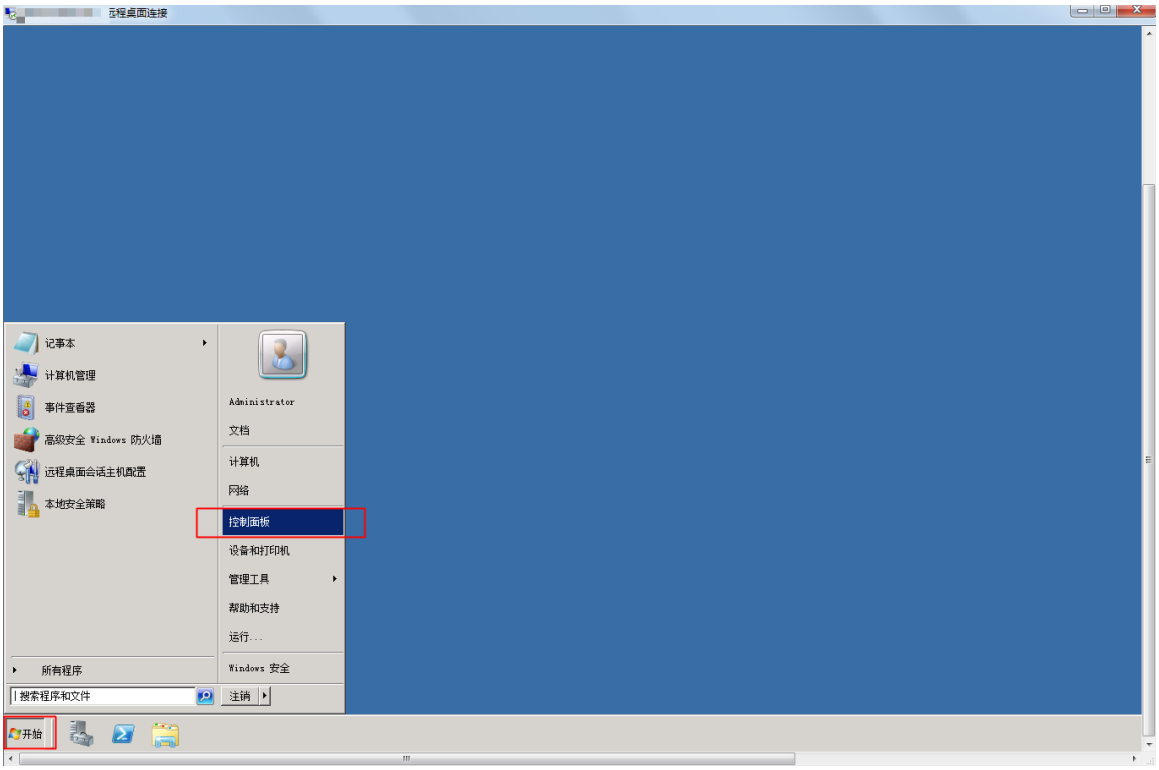


5.将第二步中搜索到的本地IP添加进去点击确定。（如需要其他IP也可以访问您再重复此步骤添加即可）

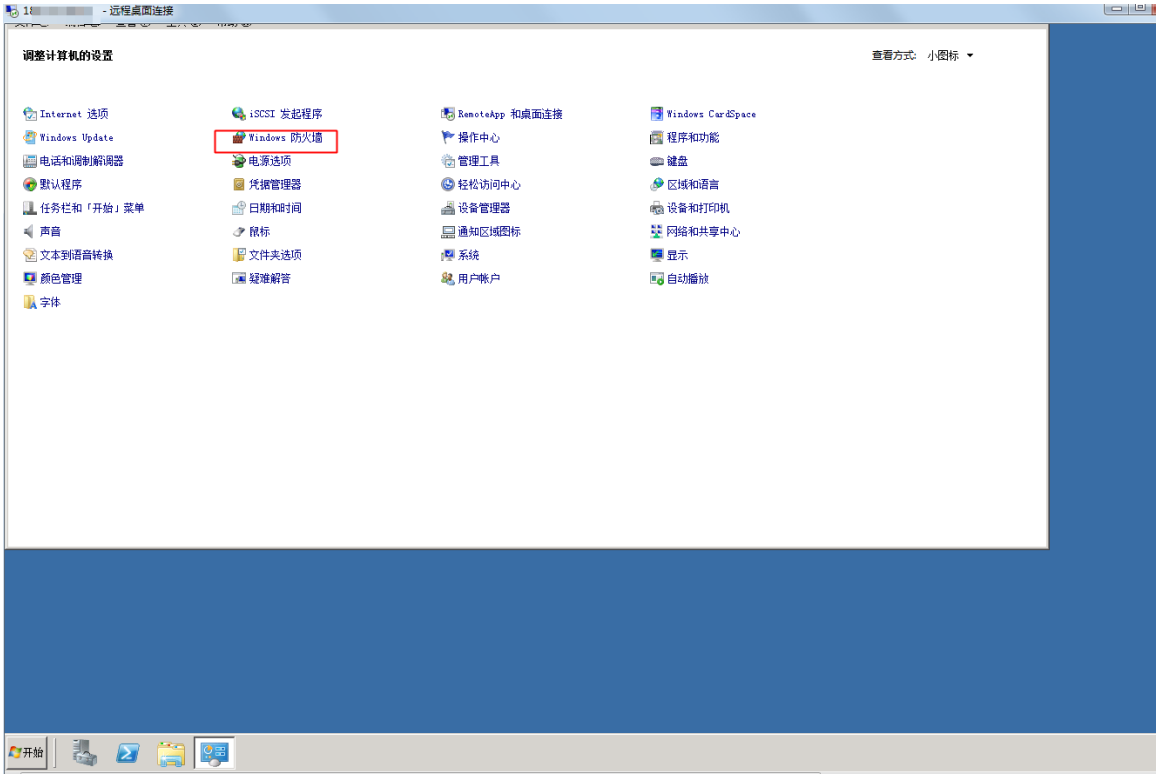


第四步:开启windows防火墙使规则生效。

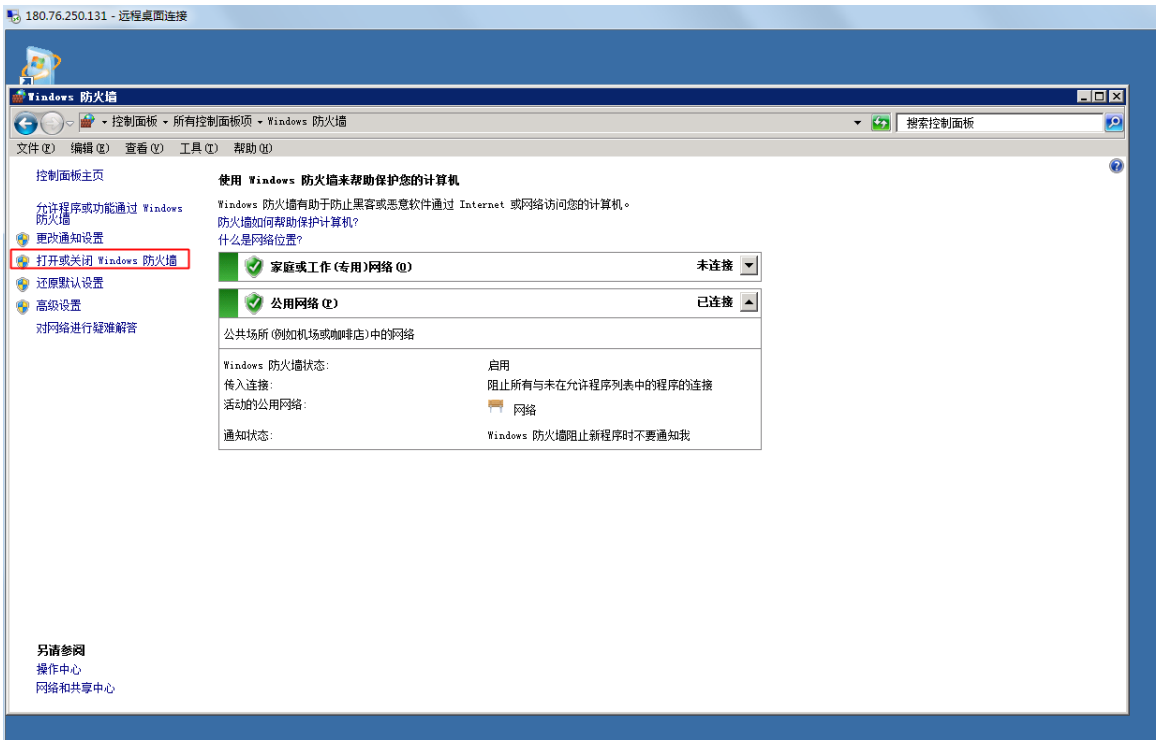
1.点击开始-控制面板。



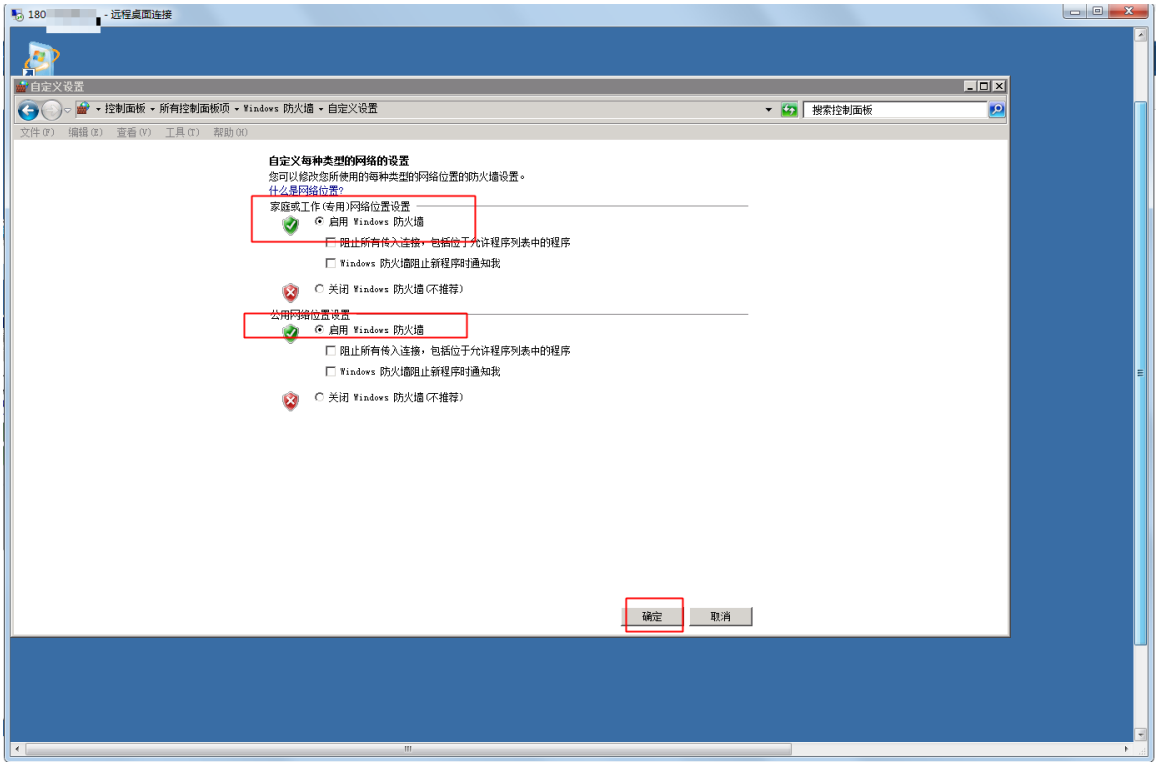
2.点击windows防火墙。



3.点击打开或关闭防火墙。



4.选择启动windows防火墙点击确定。



Windows修改默认连接远程端口

本文介绍百度智能云“Windows服务器修改默认远程端口”的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《Windows服务器修改默认远程端口》

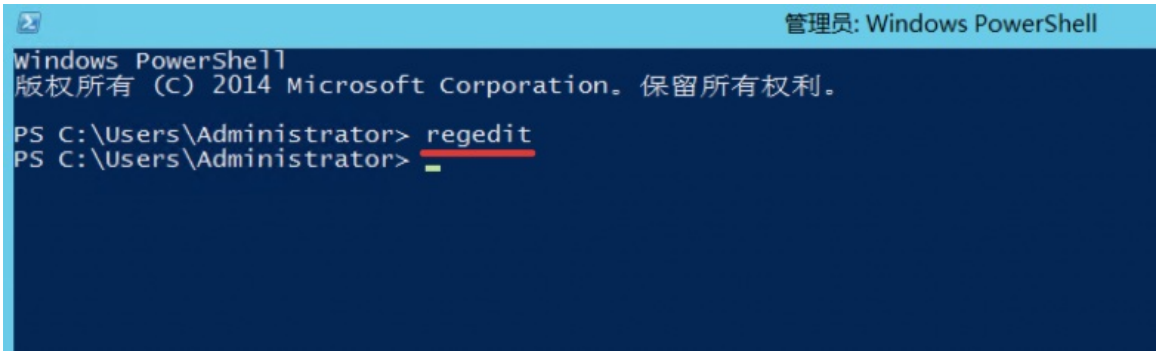
> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

本文以百度智能云公共镜像Windows server 2012为例，演示如何修改修改服务器默认远程端口（其他版本方式基本一致）。

步骤一 登录服务器

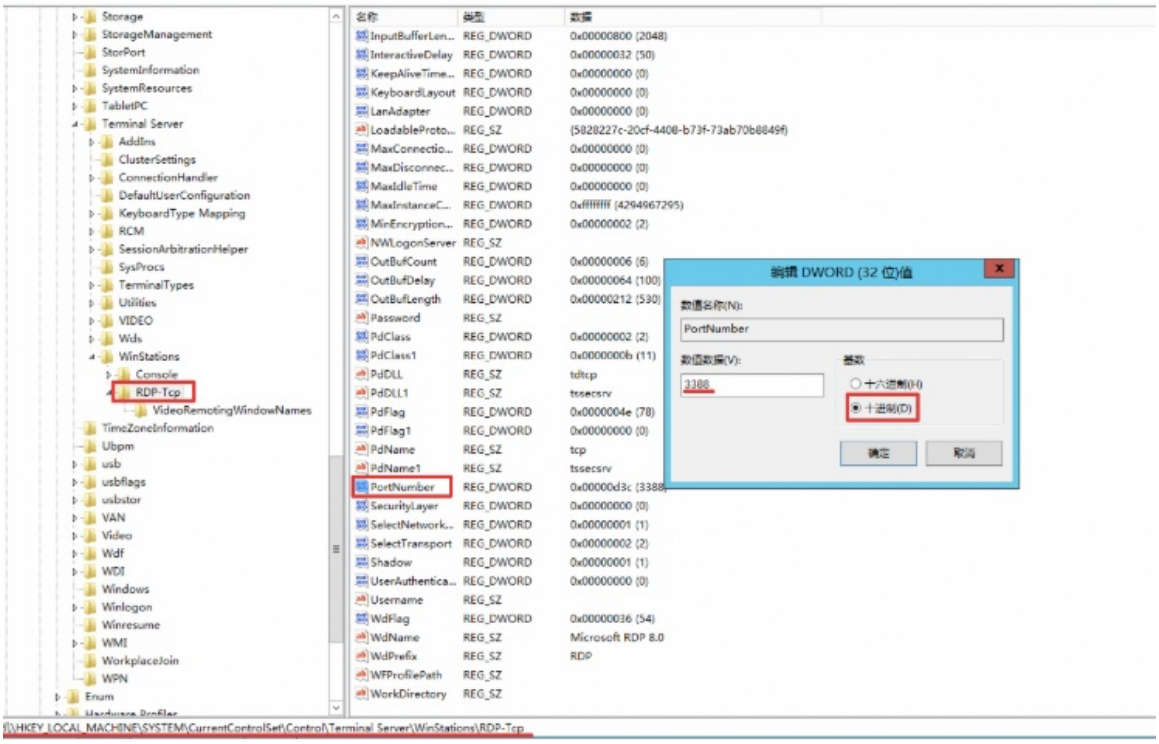
可以通过Windows自带的远程连接工具连接服务器，如果无法远程连接，使用VNC登录。

步骤二 打开powershell，输入regedit



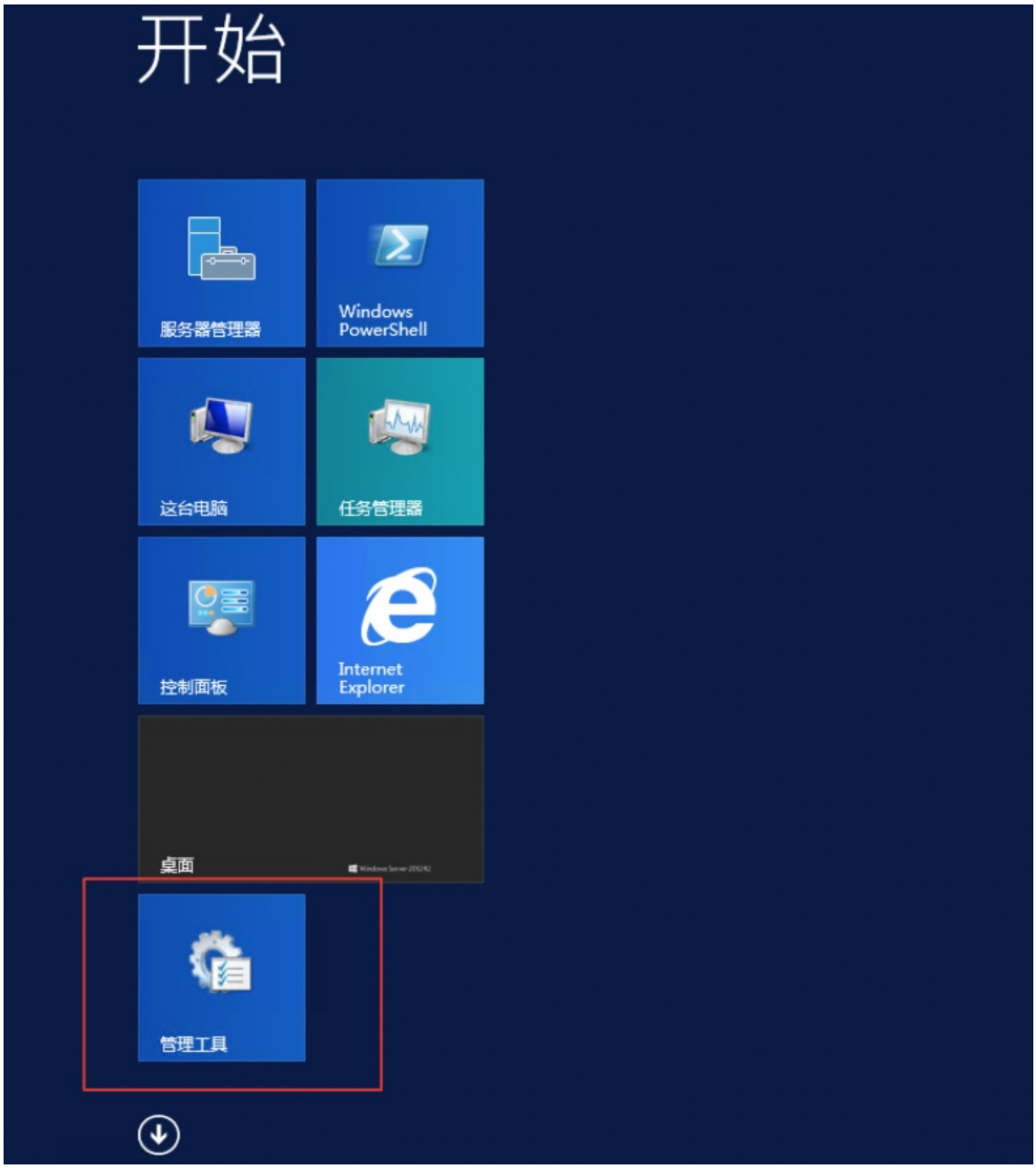
步骤三 更改注册表信息

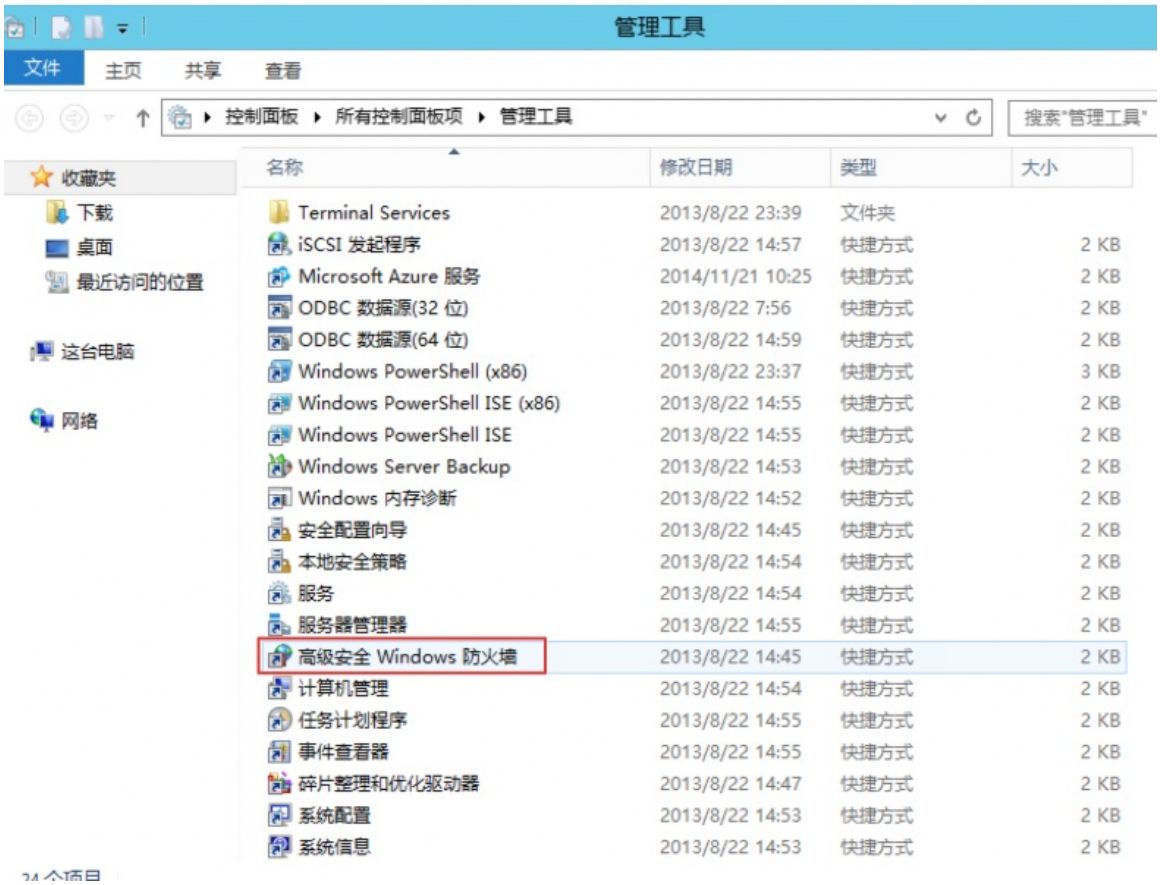
找到HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp路径下的"PortNumber"用十进制方式显示，默认为3389，改为自己需要的端口，比如修改为3388（前提是修改的端口未被其他服务占用）。



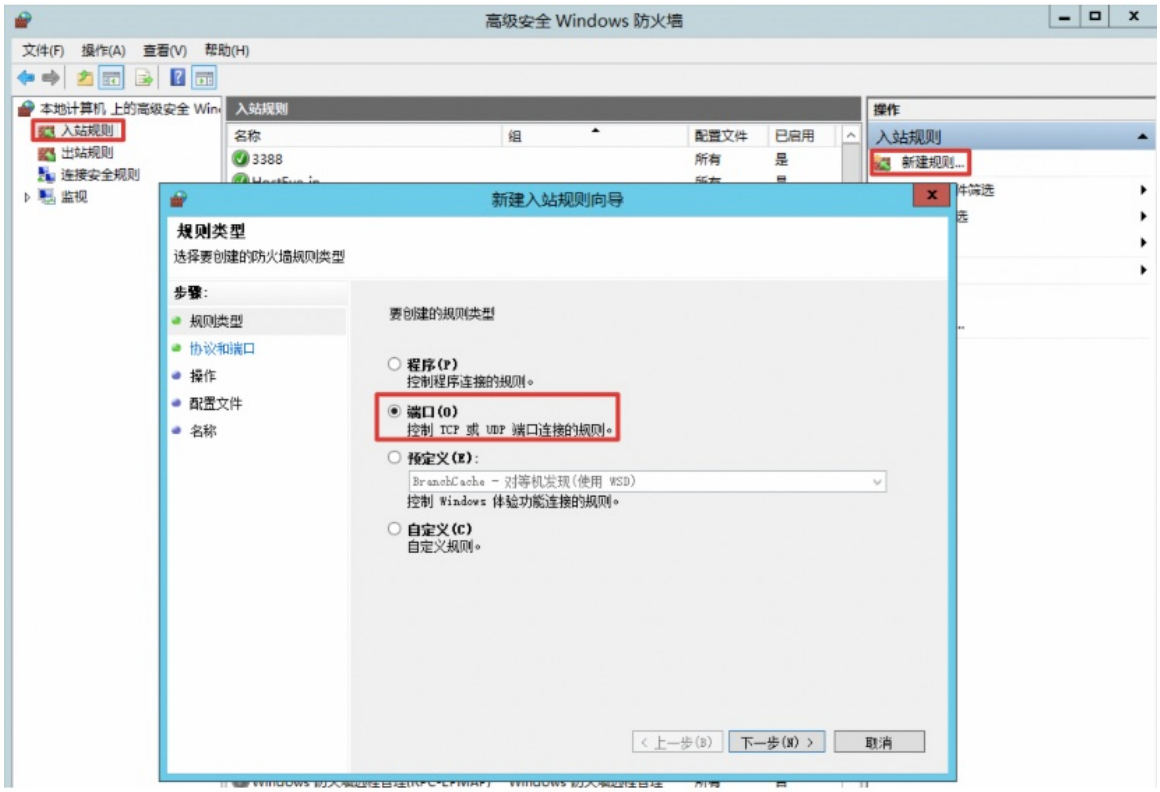
步骤四 设置防火墙

1.点击"管理工具"，双击打开"高级安全Windows防火墙"。


开始
服务器管理器
Windows PowerShell
这台电脑
任务管理器
控制面板
Internet Explorer
桌面
管理工具



2.点击"入站规则"，点击"新建规则"，编辑新建入站规则，开放自定义的端口。



步骤五 重启服务器

修改端口后，重启服务器，可以设备内部直接重启，也可以直接在控制台重启设备。

步骤六 验证连接

使用IP地址+端口的方式（ip:port）进行连接即可。

Windows快速登录

本地远程登录Windows时，需要每次都输入账号与密码信息，使用操作不方便。本文介绍BCC Windows实例如何设置记录登录账号信息，从而可以快速远程登录BCC实例。

步骤一 选择mstsc连接工具

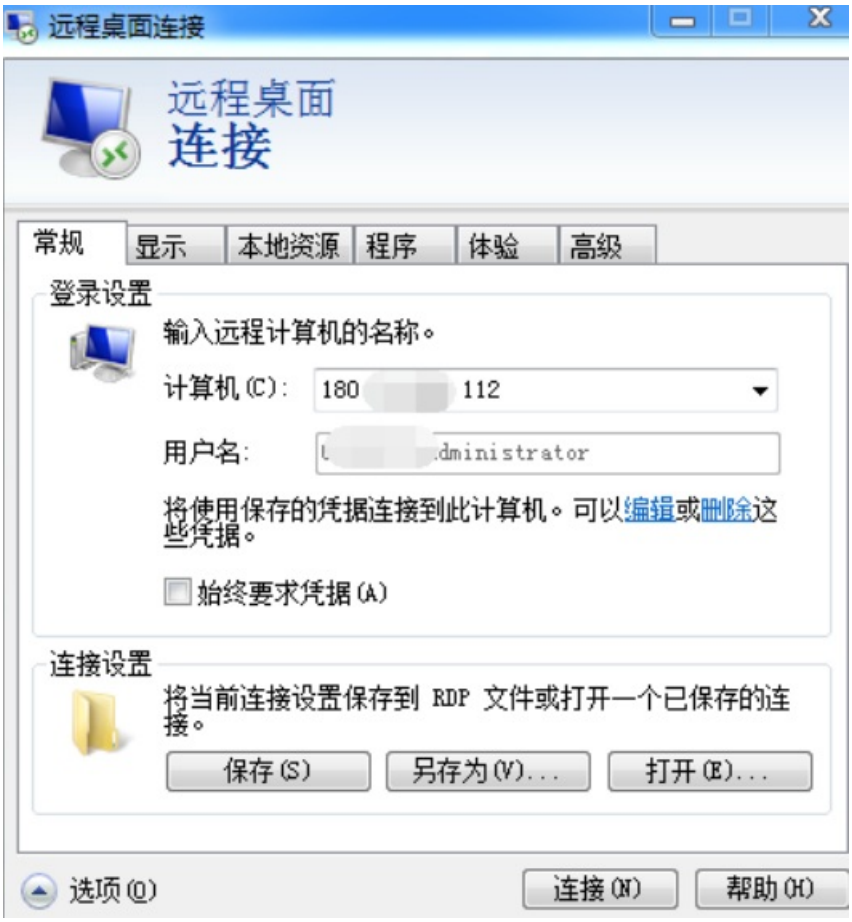
在本地电脑选择远程桌面连接应用，若是无法找到可以在搜索栏搜索远程桌面。

步骤二 设置保存用户名密码信息

点击远程桌面连接窗口左下角的选项–常规，在常规窗口的计算机中填写主机IP地址，用户名中填写登录名称。

注意：把始终要求凭据复选框取消勾选，若是勾选上，需要输入密码。

点击保存，最后点击连接，第一次登录需要输入账号密码，后续不需要在输入账号密码，直接可以登录服务器。



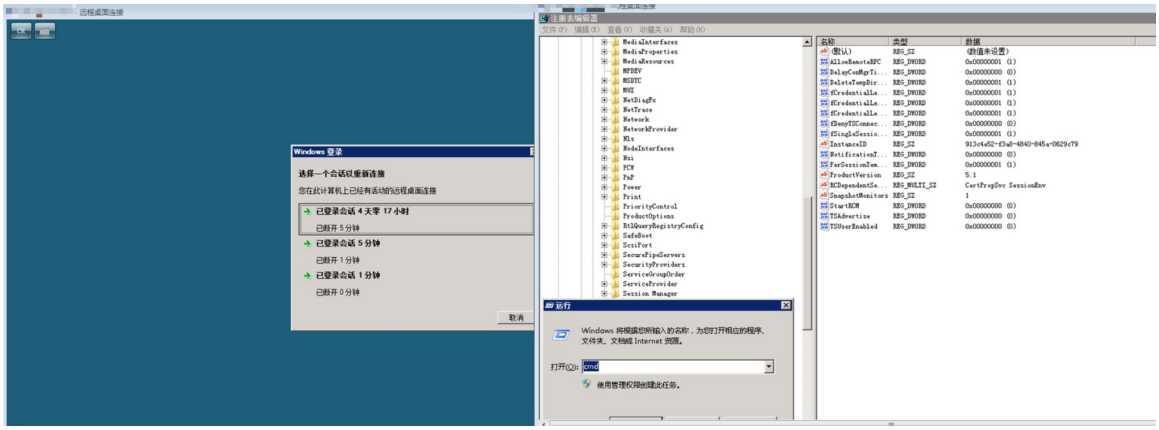
Windows允许单个用户连接多个会话的方法

本文介绍了如何使用 Windows 允许单个用户连接多个会话方式，以 Windows Server 2008 R2 为例。

问题描述

服务器默认只允许单个用户连接一个远程桌面会话，如果已有连接登录，另外的连接再登录会踢掉之前的连接。

即出现以下界面：

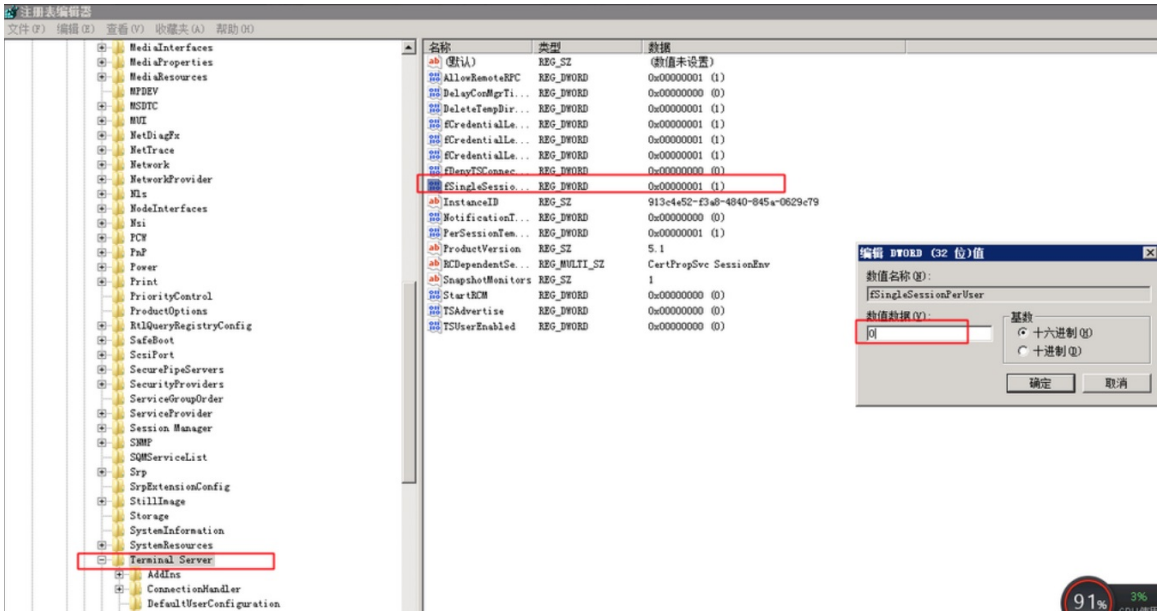


解决方法

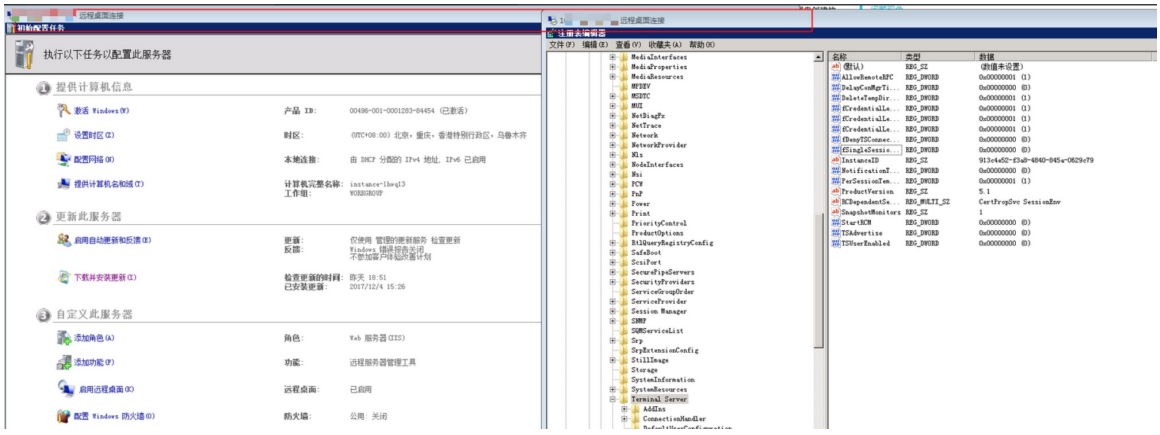
步骤一 通过注册表调整会话

1.WIN键＋R键打开运行，然后输入cmd，回车进入命令提示符，执行 regedit 可以打开注册表。

2.找到：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server，右侧找到fSingleSessionPerUser 键值，右键可以修改键值，默认键值是1，修改为0，不需要重启服务器。



3.测试开启两个远程。

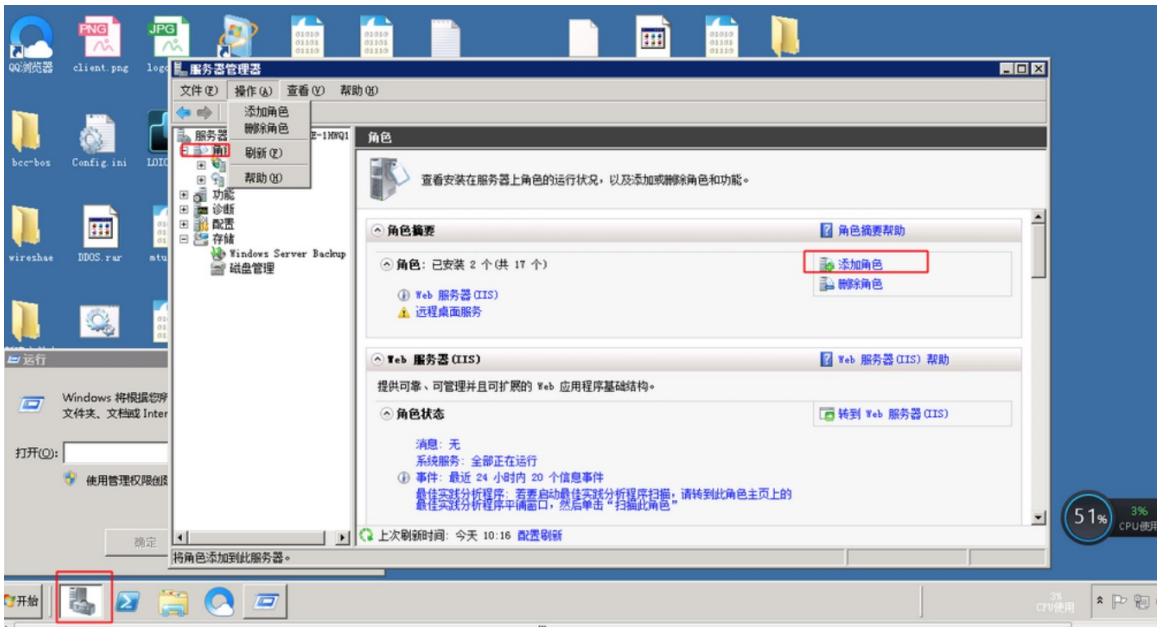


（注意：以上方法超过两个就会出现中断）

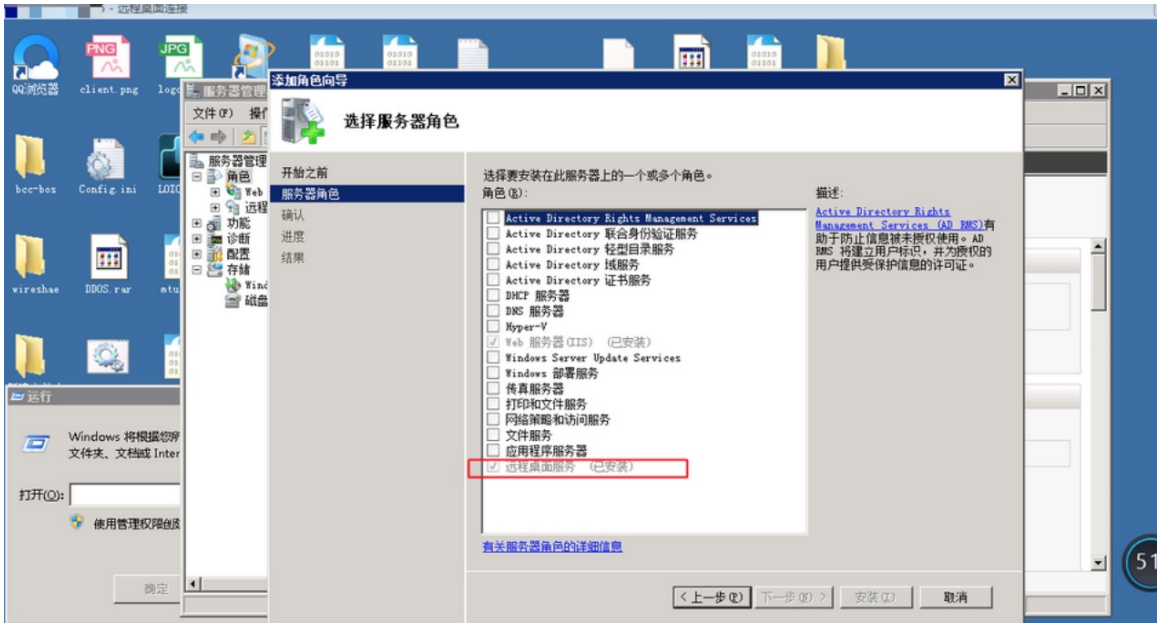
步骤二 部署远程桌面会话服务

超过两个或者两个以上连接，可以部署Windows 远程桌面会话主机配置，具体如下：

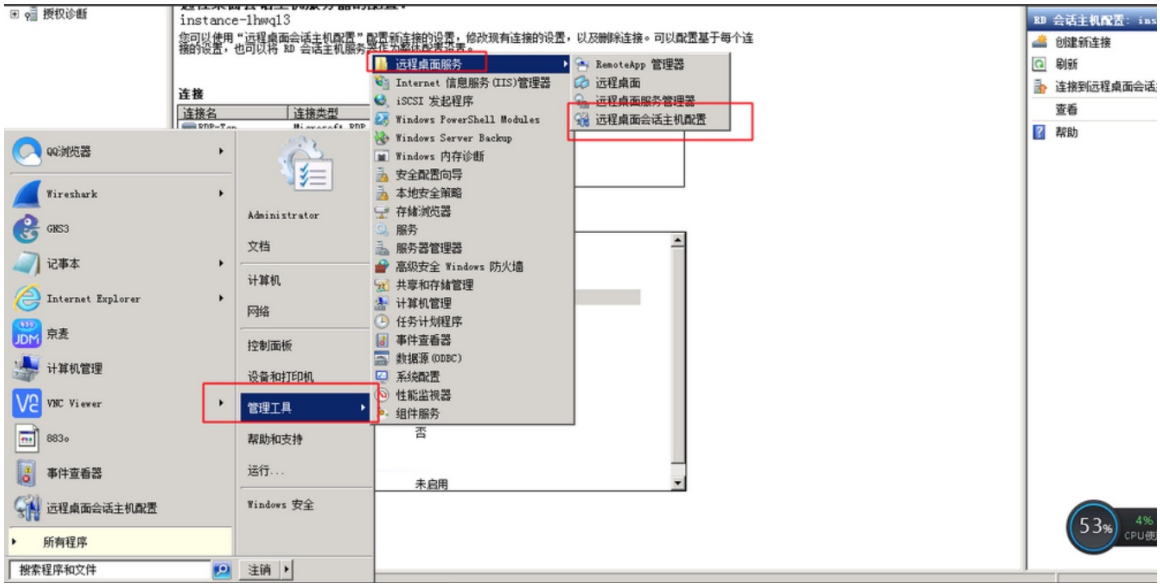
1.打开左下角的服务器管理器添加角色。



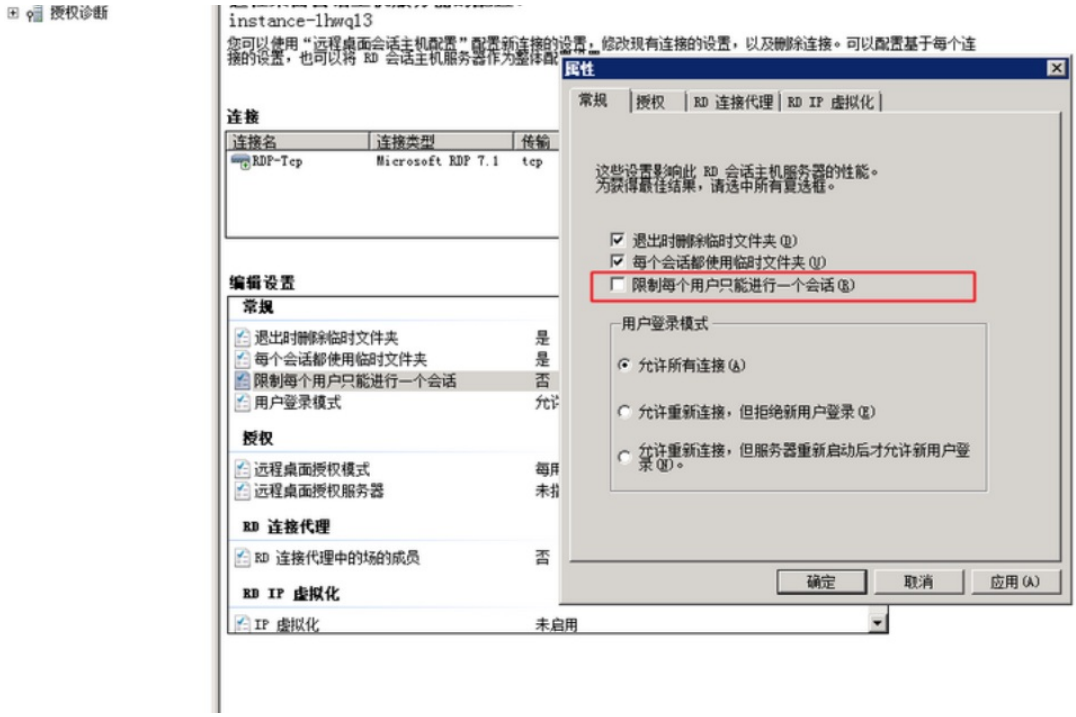
2.勾选远程桌面服务，之后下一步安装即可。



3.依次找到远程桌面会话主机配置。



4.取消限制每个用户只能进行一个会话（如果想重复通过administrator登入的话，请取消）。



5.验证。

注意：部署此方式只允许120天免费，如果想超出120天限制，建议购买微软的授权。

配置多用户访问Windows服务器

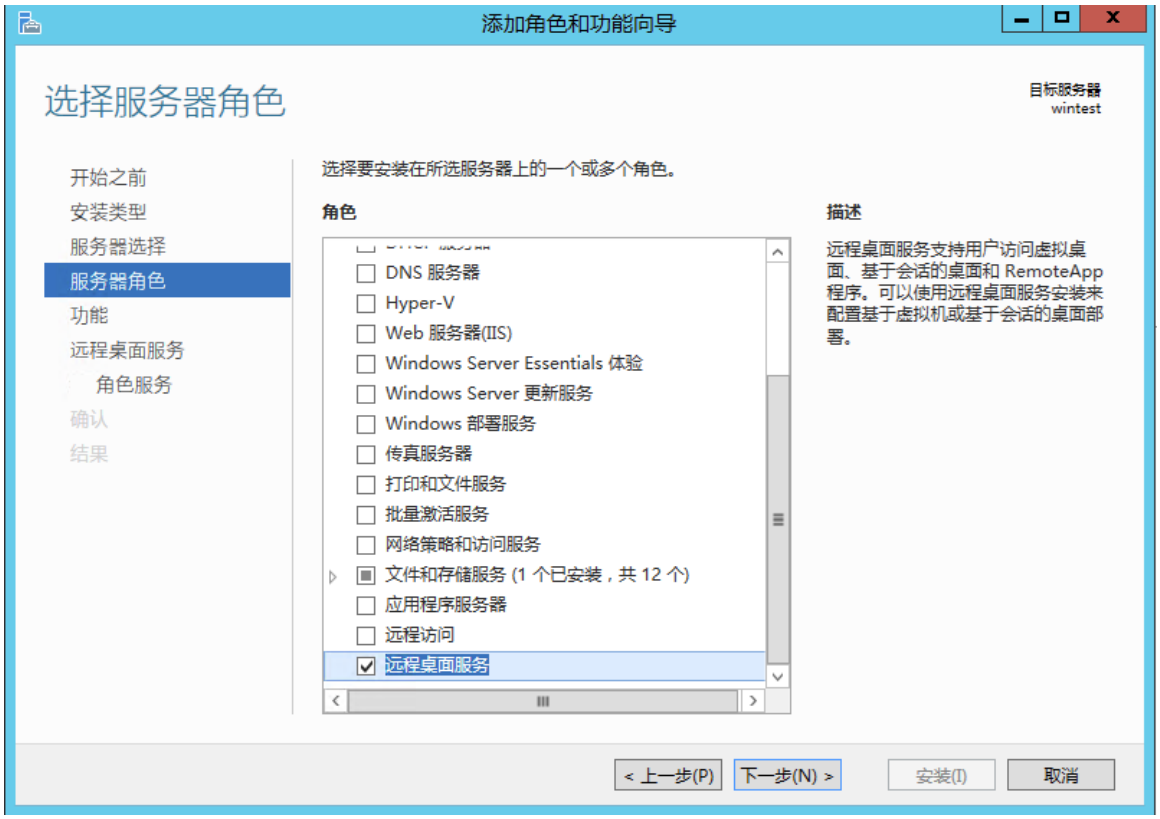
本文档以 Windows Server 2012 R2 操作系统为例，指导配置多用户远程登录BCC Windows云服务器实例。

步骤一 登录服务器

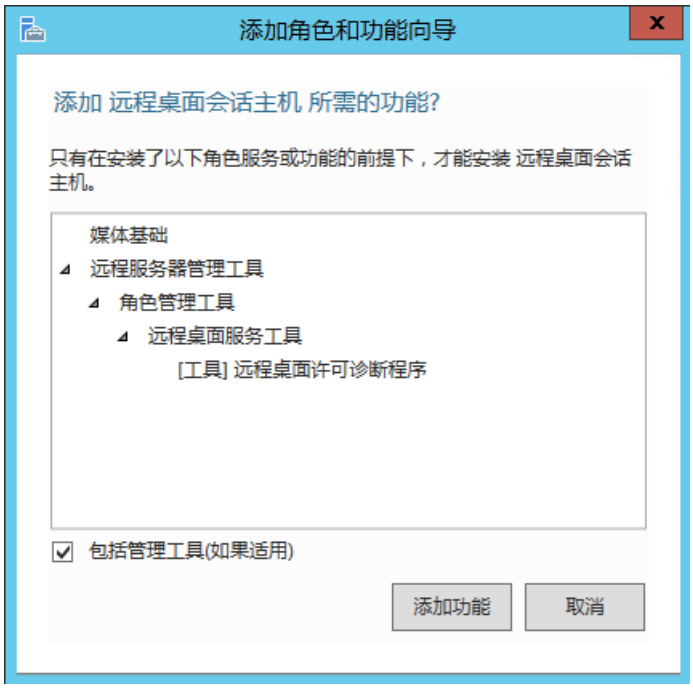
可以通过Windows自带的远程连接工具连接服务器，如果无法远程连接，使用VNC登录。

步骤二 安装远程桌面服务

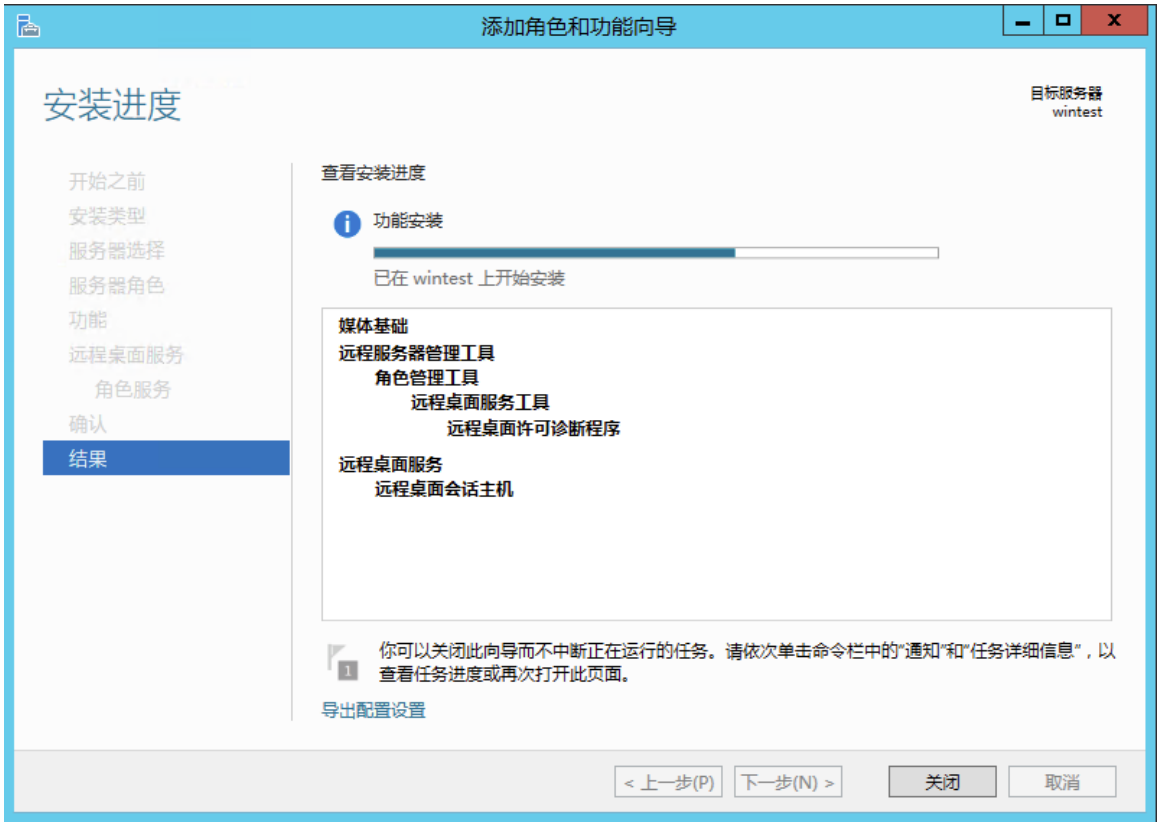
1.打开服务器管理器 ，选择 “添加角色和功能向导”，选择 “远程桌面服务”，点击下一步。



2.接下来不用选择，连续3次点击下一步，在角色服务页面勾选 “远程桌面会话主机”，然后在弹出的对话框中点击 “添加功能”。

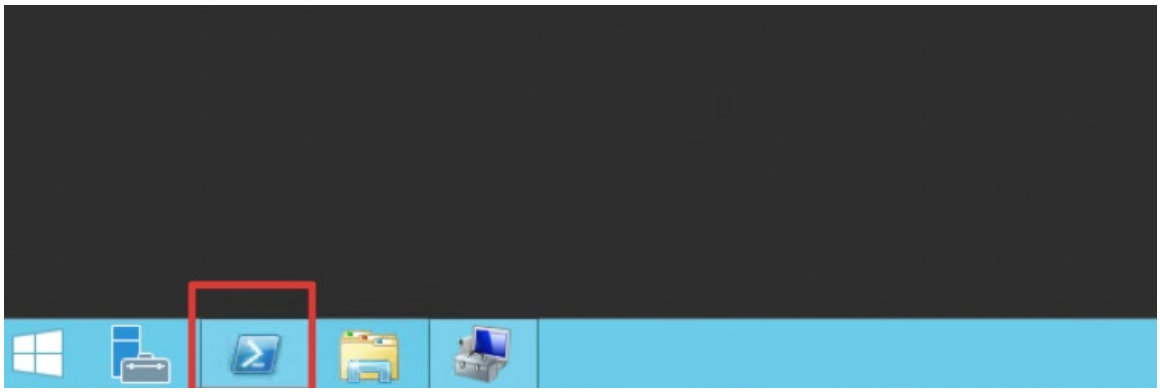


3.继续点击下一步，点击安装，等待安装完成后点击关闭。

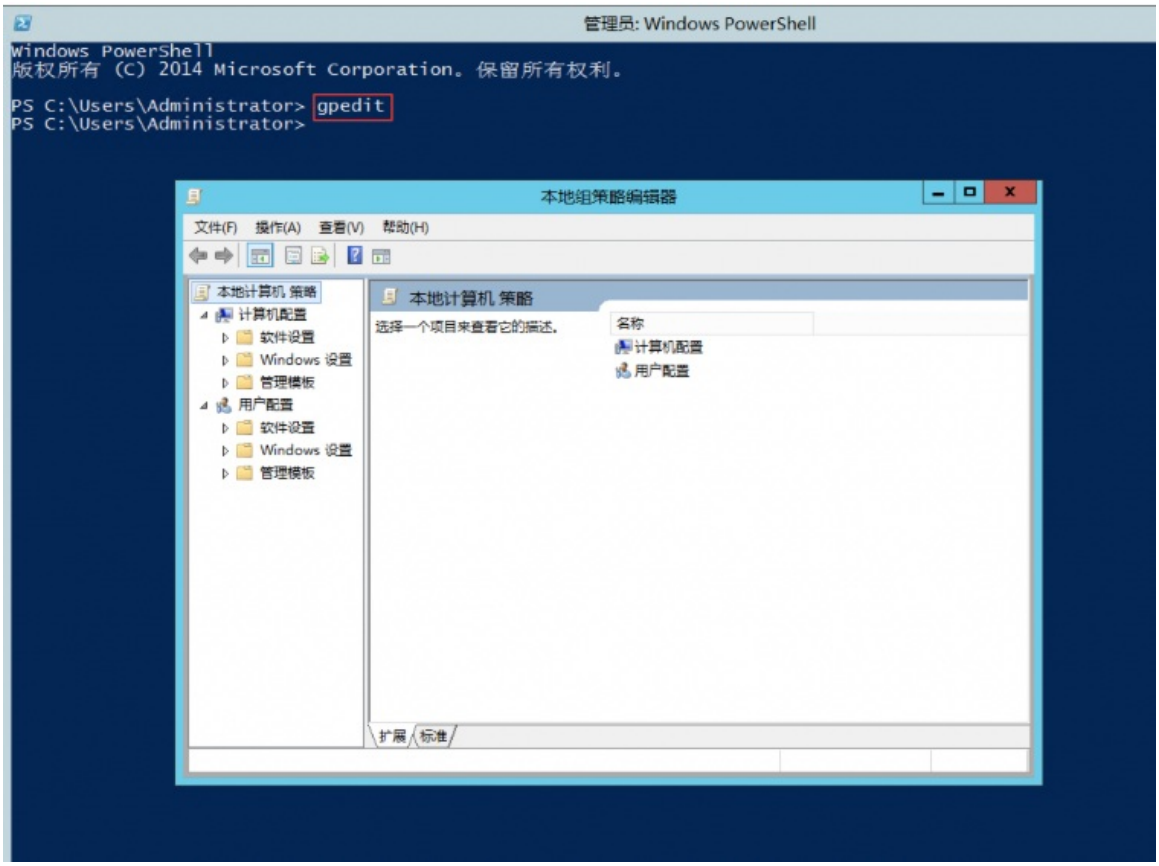


步骤三 设置本地组策略编辑器

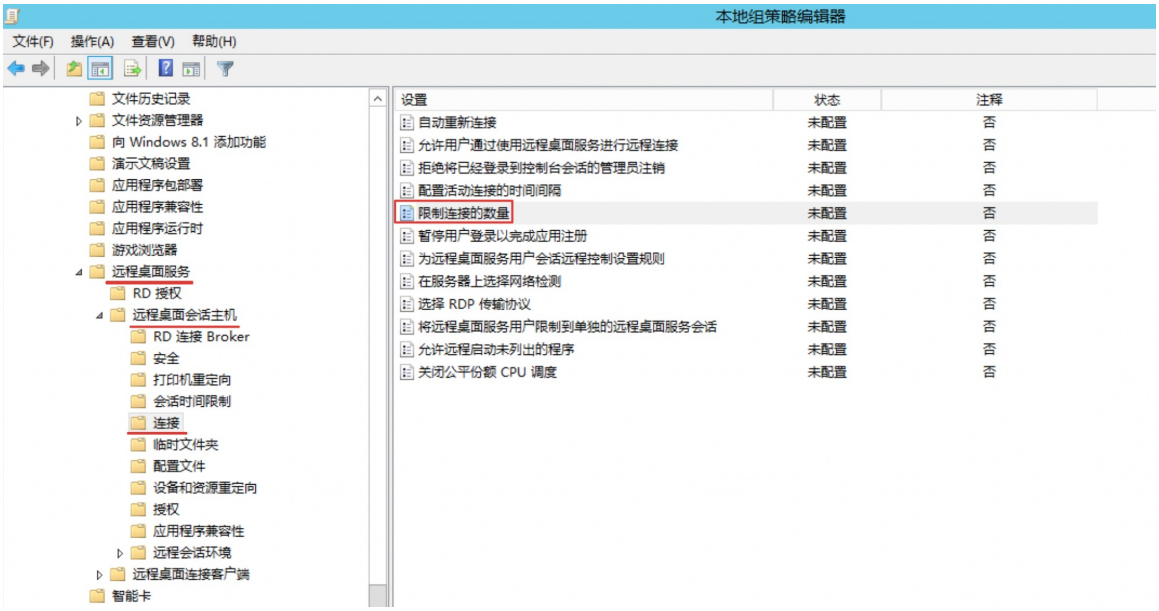
1.点击左下角PowerShell，打开PowerShell窗口。



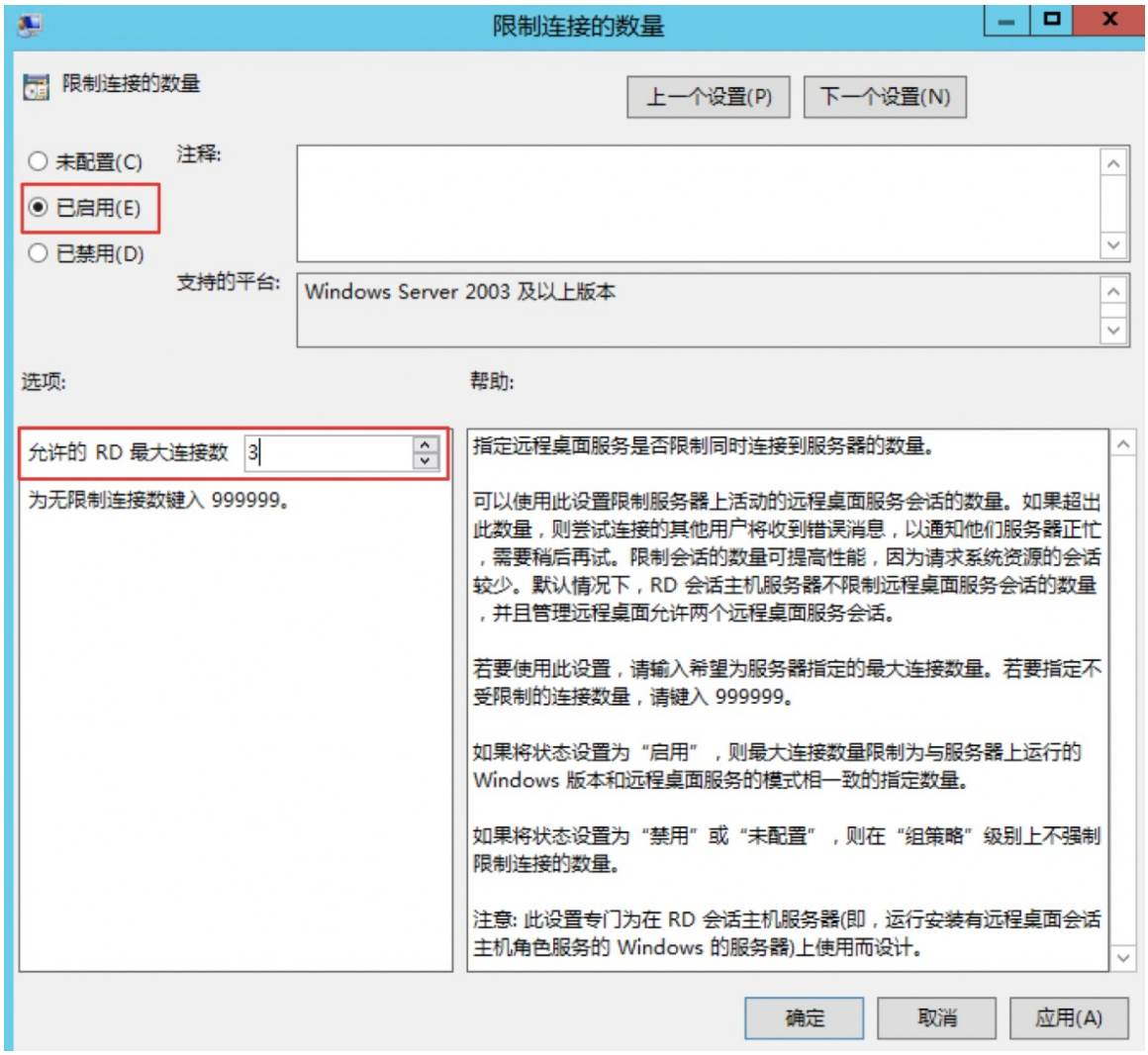
2. 在PowerShell 窗口中，输入 gpedit.msc，按 Enter，会自动打开 “本地组策略编辑器”。



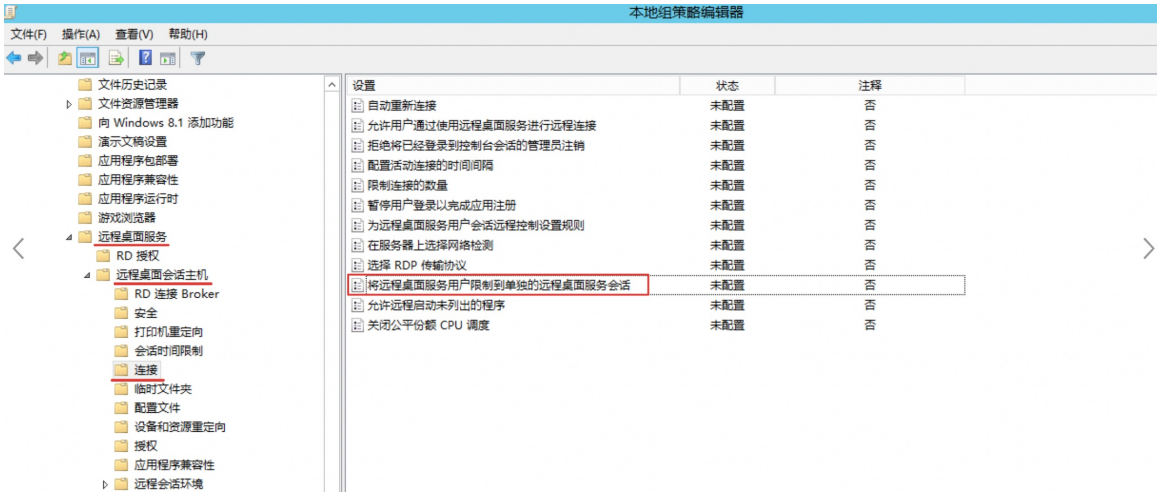
3.选择"计算机配置">"管理模板">"Windows 组件">"远程桌面服务">"远程桌面会话主机">"连接"，双击打开"限制连接的数量"。



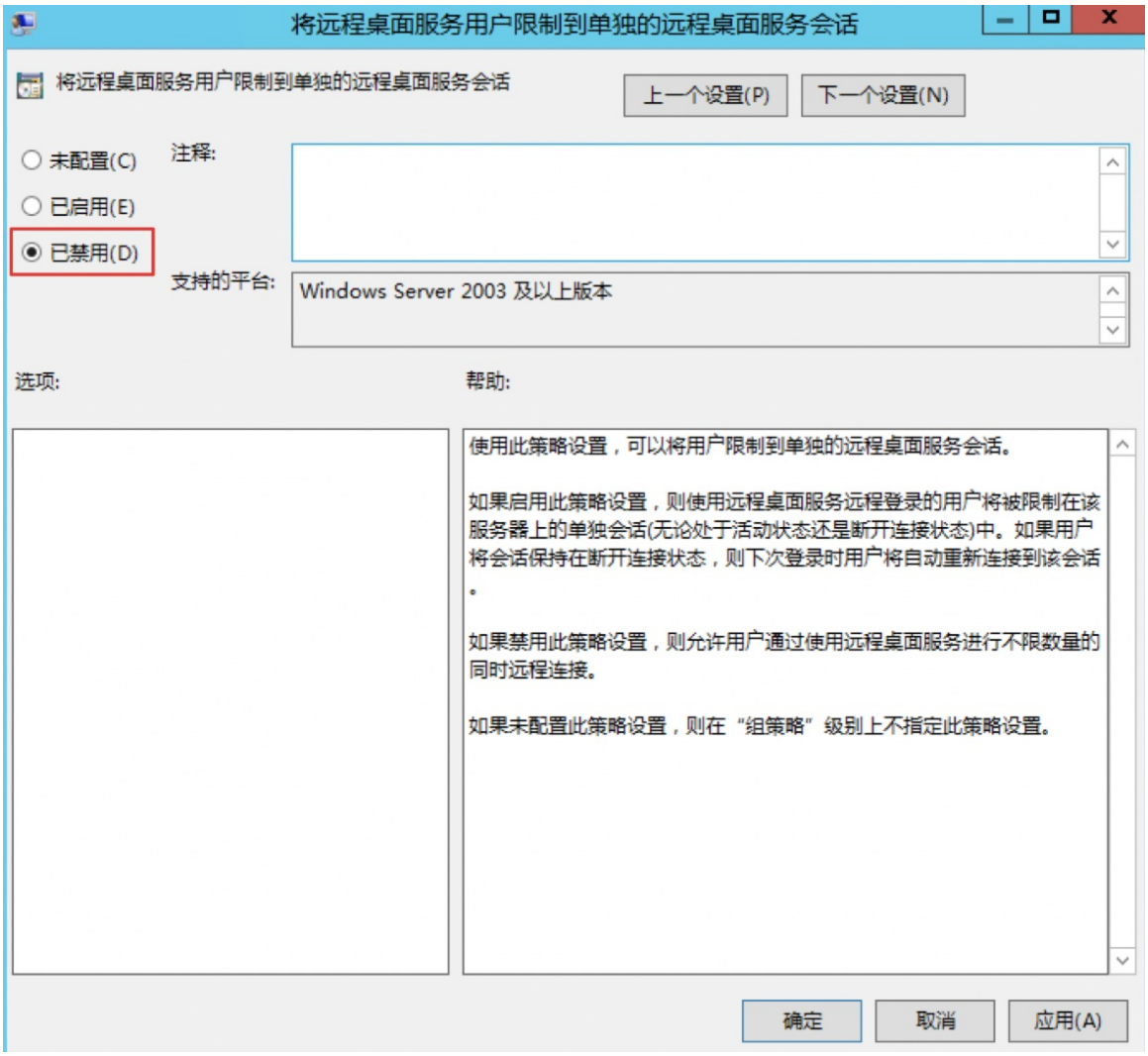
4.在弹出的"限制连接的数量"窗口中，"选择"已启用"，并在"允许的RD最大连接数"处输入最大的同时远程连接用户数，完成设置后，点击"应用"，点击"确定"。



5.选择"计算机配置">"管理模板">"Windows 组件">"远程桌面服务">"远程桌面会话主机">"连接"，双击打开"将远程桌面服务用户限制到单独的远程桌面服务会话"。



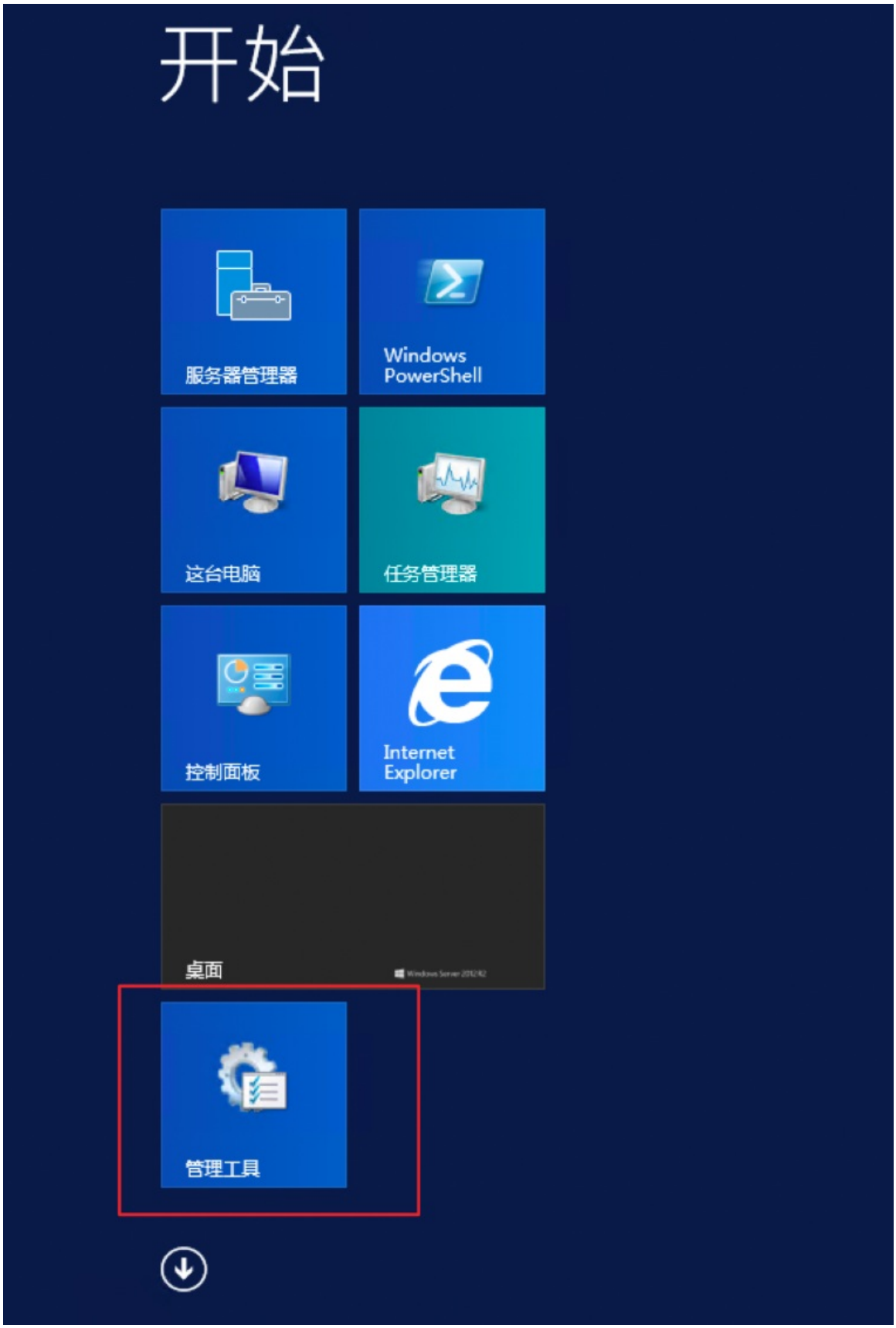
6.在弹出的"将远程桌面服务用户限制到单独的远程桌面服务会话"窗口中，选择"已禁用"，完成设置后点击"应用"，点击"确定"。



7.设置完毕后，可以关闭本地组策略编辑器。

步骤四 新建用户并加入到Administrators组

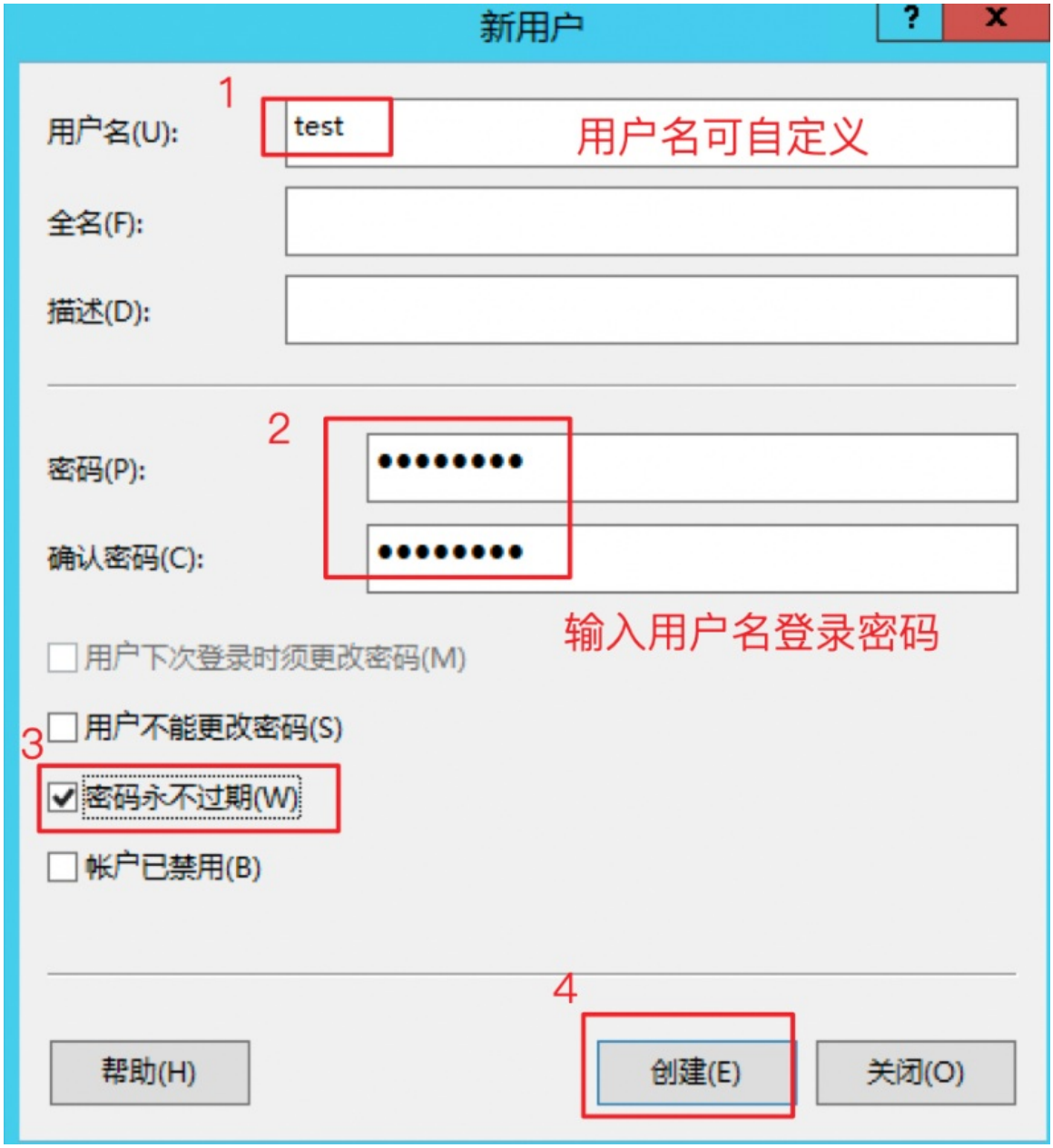
1.点击"管理工具"。



双击"计算机管理"。

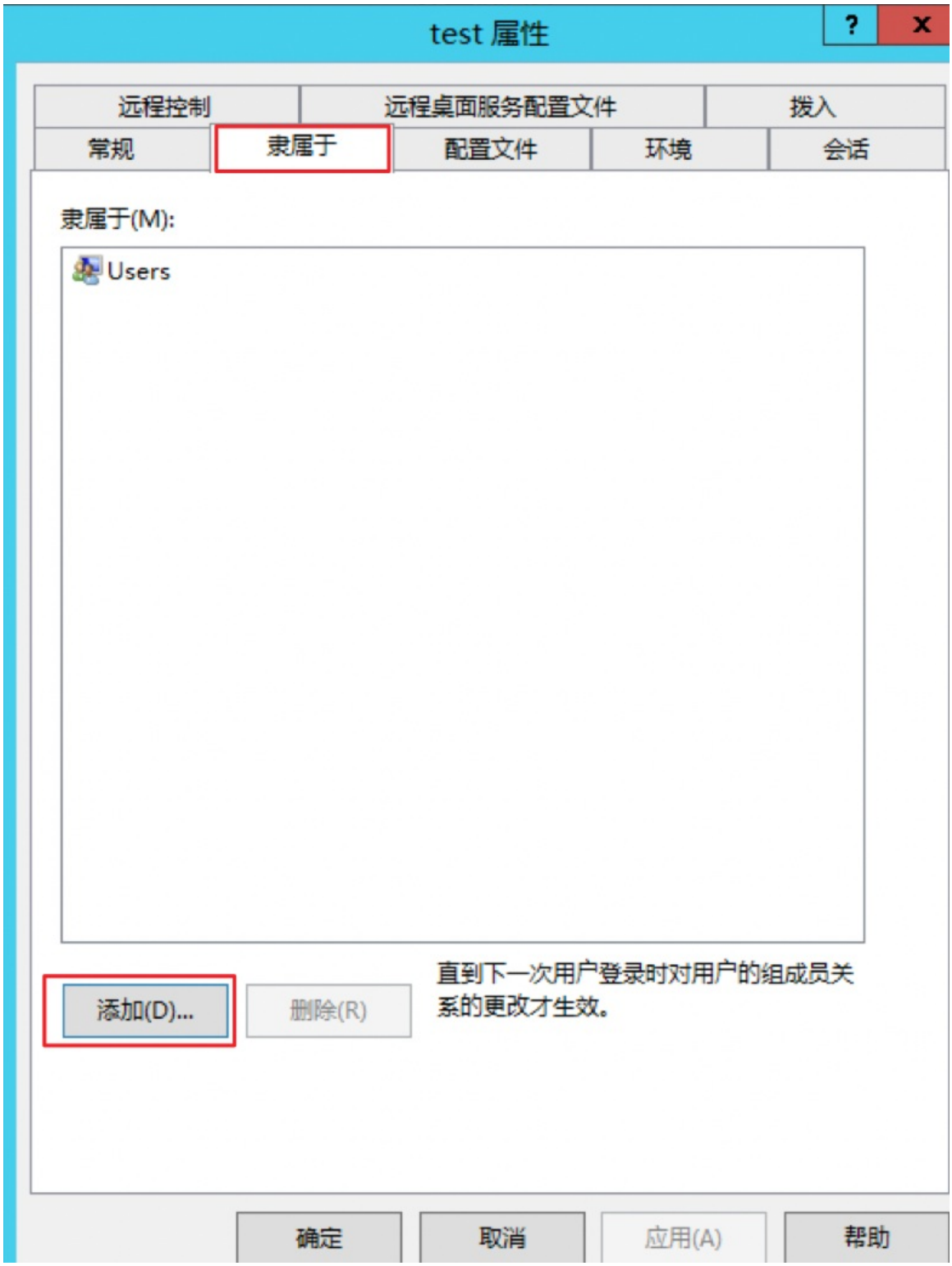
2.弹出"计算机管理"窗口后，点击"本地用户和组"→"用户"，鼠标右键点击右侧空白处，点击"新用户"。

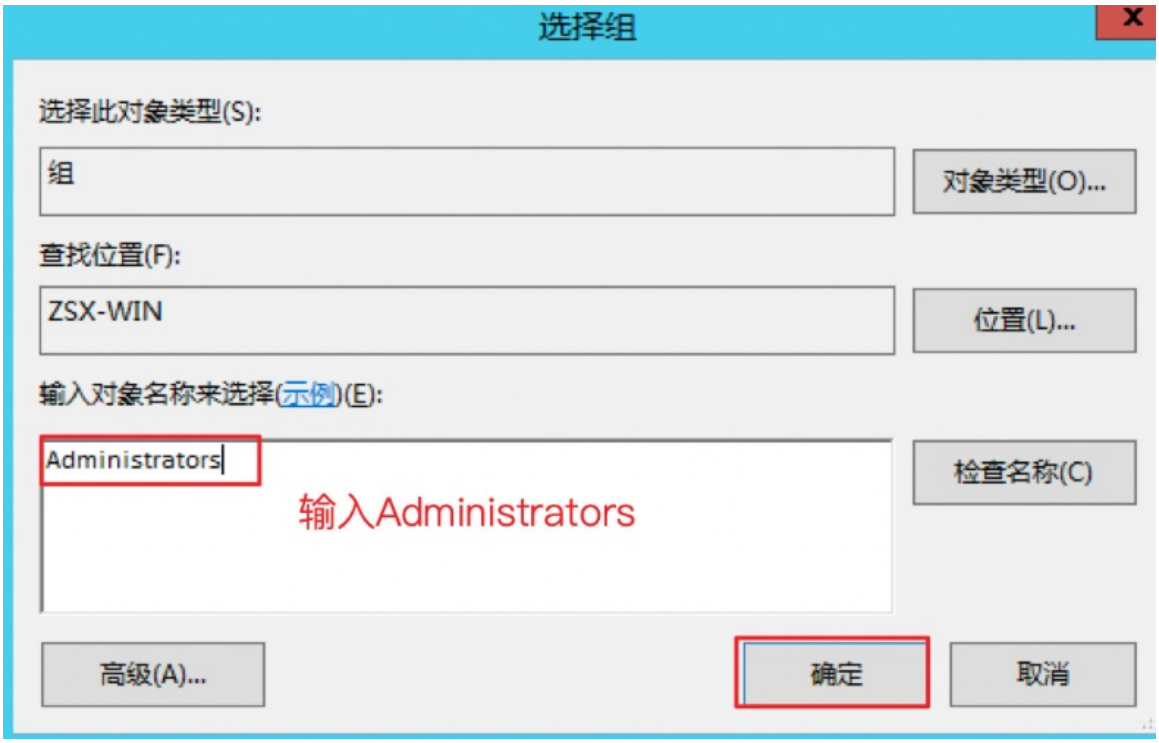
3.输入用户名称，输入用户名的登录密码，然后取消勾选"用户下次登录时须更改密码"，勾选"密码永不过期"，最后点击"创建"。



4.创建完成后，鼠标右键点击新创建的用户，点击"属性"。

5.选择"隶属于"，点击左下角的"添加"按钮，将该用户添加到Administrators组中，点击"应用"、"确定"。


test 属性
?
x
远程控制
远程桌面服务配置文件
拨入
常规
隶属于
配置文件
环境
会话
隶属于(M):
Users
添加(D)...
删除(R)
直到下一次用户登录时对用户的组成员关系的更改才生效。
确定
取消
应用(A)
帮助



步骤五 重启服务器

重启BCC实例，重启完毕后验证即可。

Windows CPU内存跑满排查步骤

本文以Windows Server 2012为例，介绍了如果在使用服务器的过程中出现卡顿、登录服务器后异常退出等情况，针对服务器内部资源的排查步骤。

步骤一 登录服务器

可以通过Windows自带的远程连接工具连接服务器，如果无法远程连接，使用VNC登录。

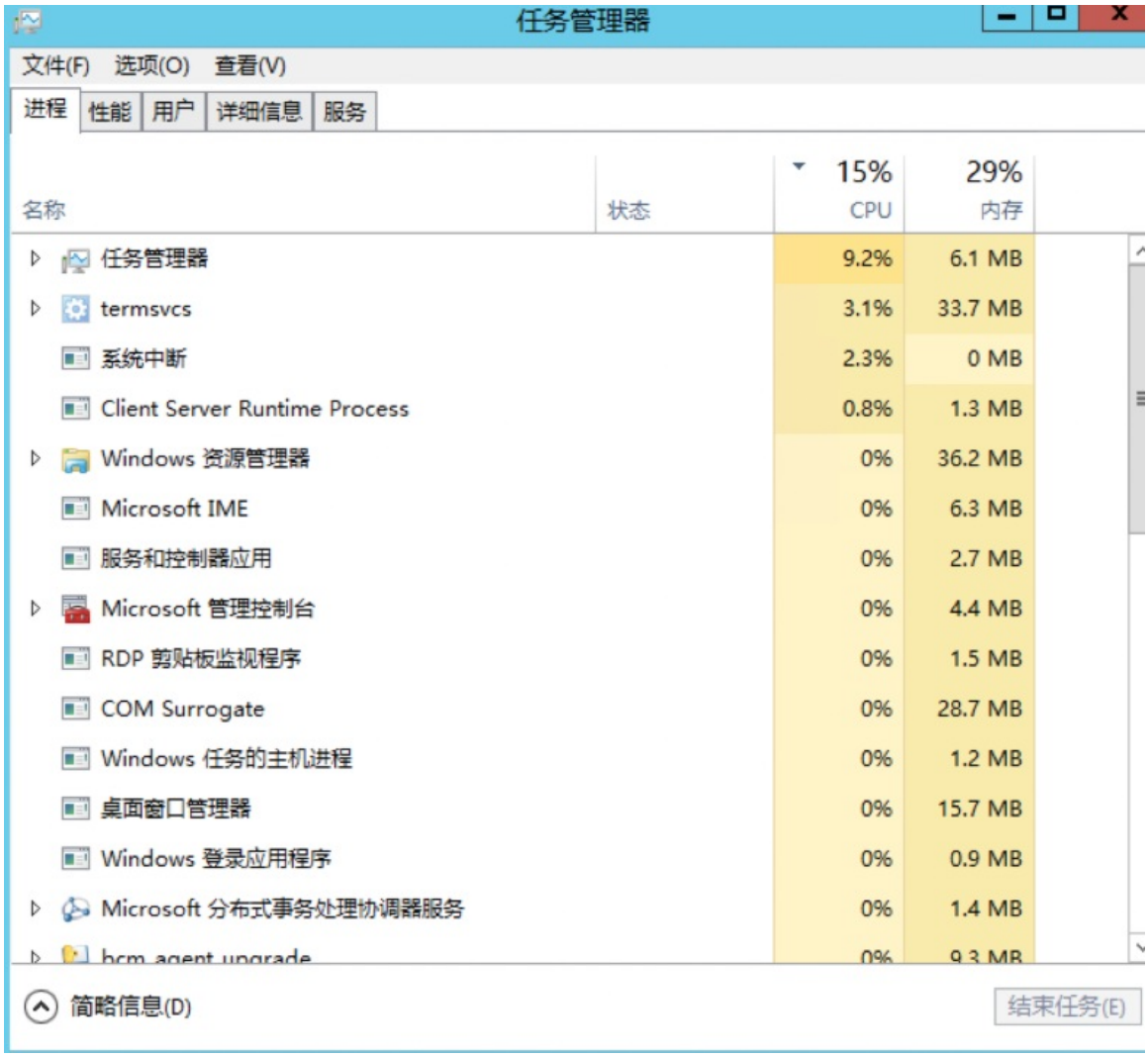
步骤二 查看进程占用资源情况

1.鼠标右键点击任务栏，点击"任务管理器"。

2.查看当前CPU、内存的使用情况。

如果是正常进程占用大量资源，可以考虑对CPU、内存进行扩容；

如果是异常进程，可以在做好快照的前提下，先结束该进程，终止进程后观察是否可以正常使用。



本地播放Windows服务器中的声音

本教程以百度智能云公共镜像2008 R2 x86_64 (64bit) 中文版操作系统为例，介绍如何在本地能播放BCC Windows服务器中的声音，需要BCC实例绑定公网IP。

步骤一 登录服务器

可以通过Windows自带的远程连接工具mstsc连接服务器，详情步骤可参考Windows快速登录。

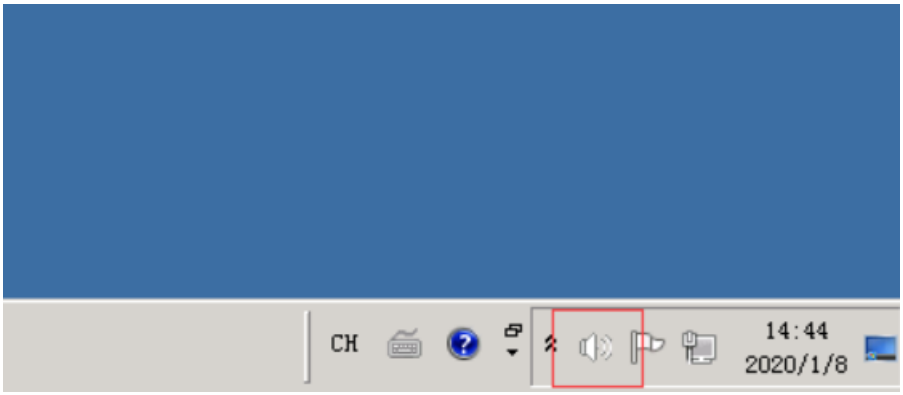
步骤二 远程连接设置

1.在远程桌面连接界面的计算机栏输入远程连接的IP地址，用户名：Administrator。

2.选择本地资源-远程音频-设置，勾选在此计算机上播放，然后点击确认。

步骤三 进行连接并且验证声音效果

1.远程连接后查看右下角音频显示正常。

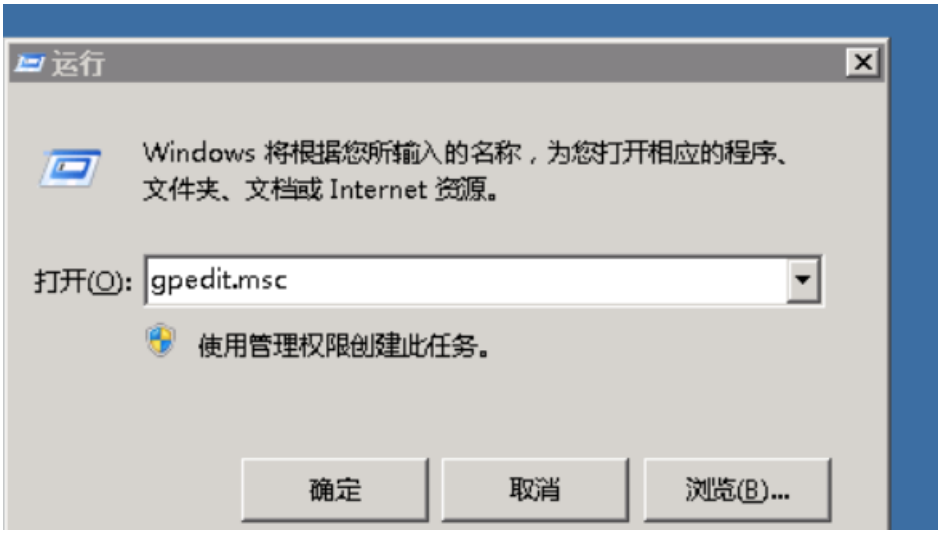


2.播放声音，进行验证。

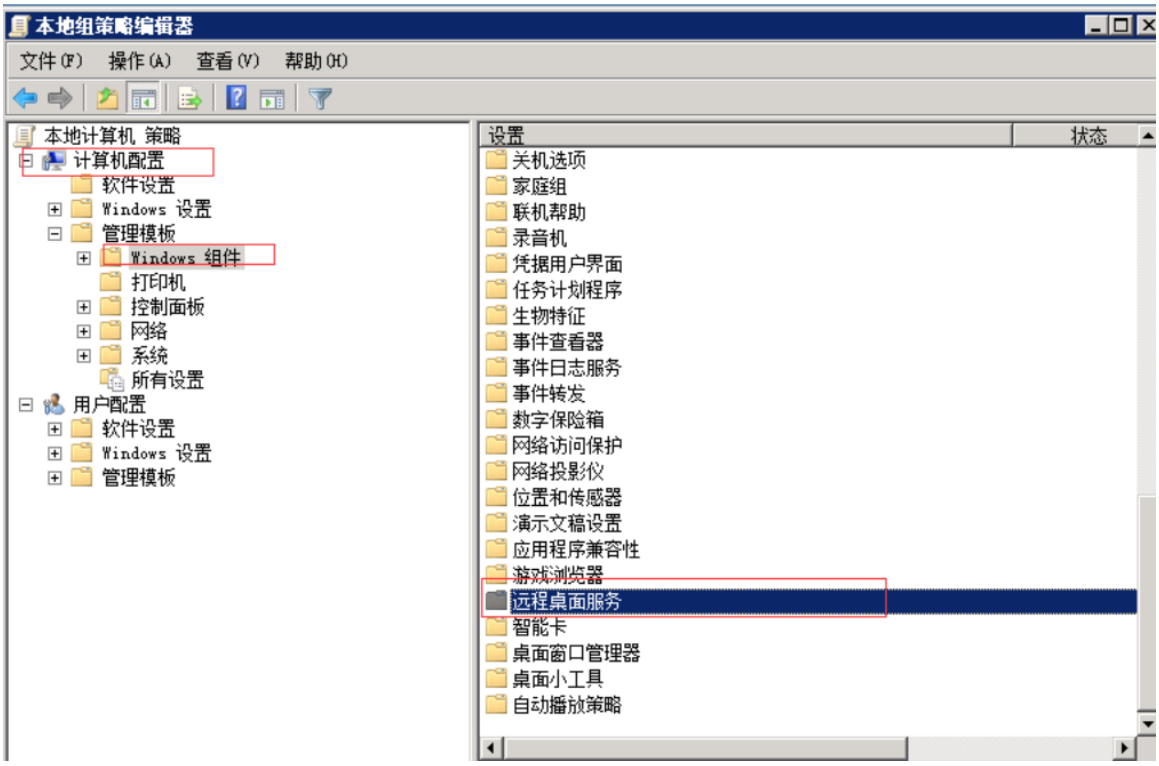
步骤四 若是使用上述方案，还是无法播放声音，请参考下列步骤进行操作

1.使用vnc或者远程连接方式进入Windows系统。

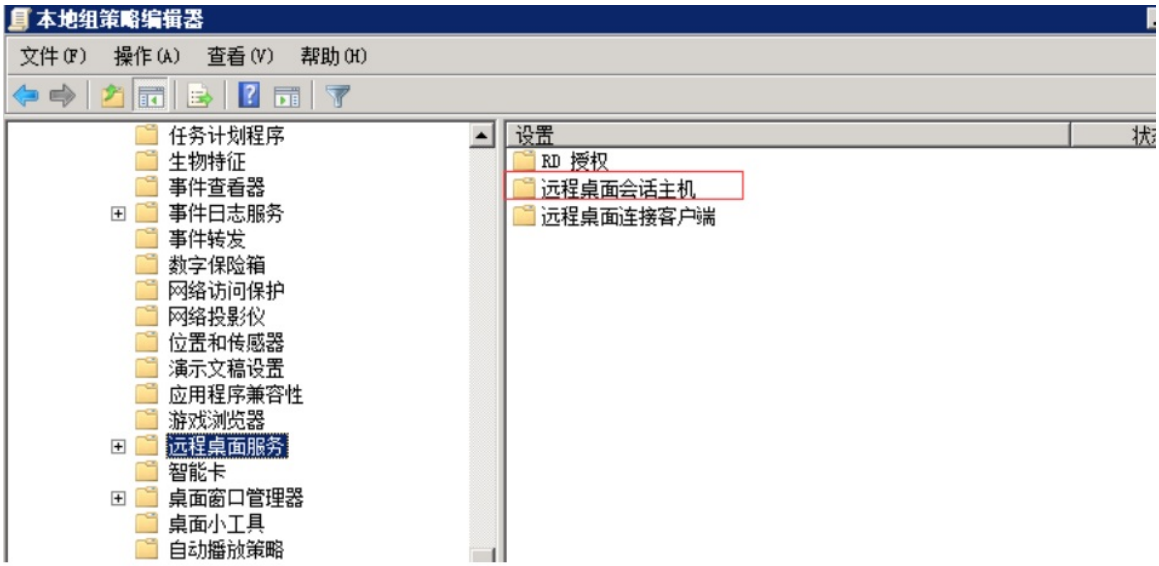
2.使用Win+R，输入命令gpedit.msc或者在搜索栏输入编辑组策略，进入本地组策略编辑器。



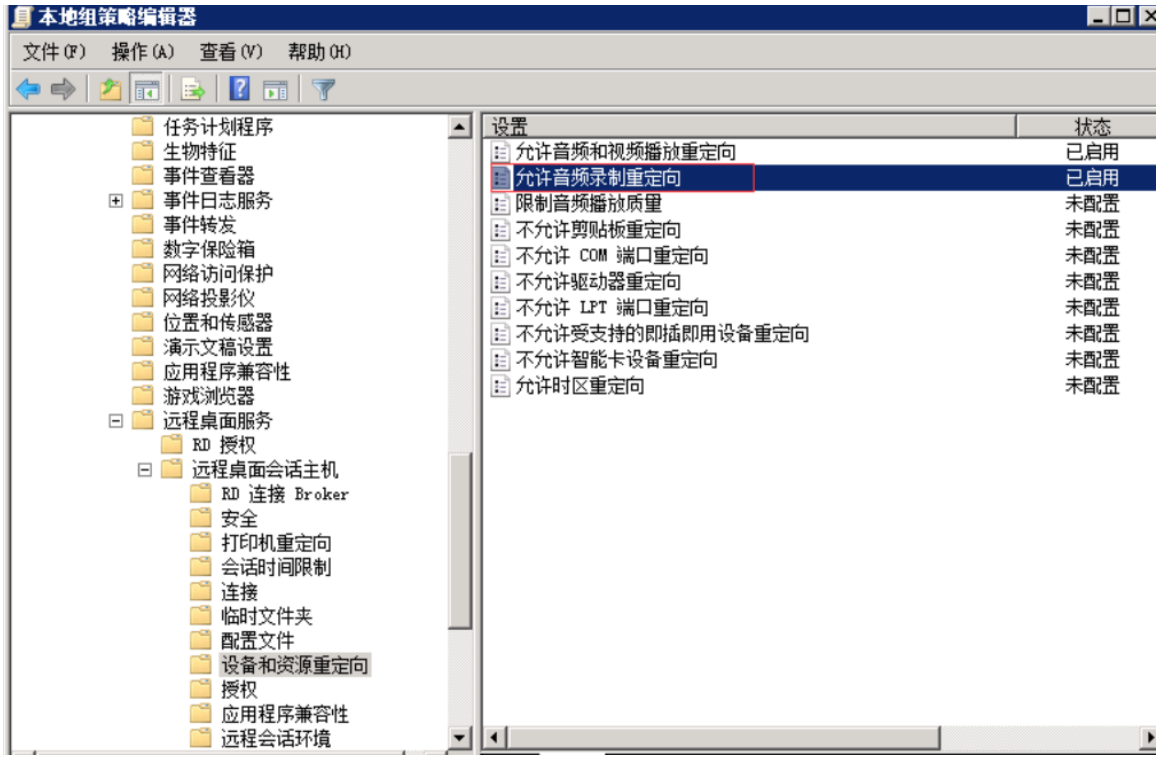
3.依次点击计算机配置->管理模板->Windows组件->远程桌面服务。



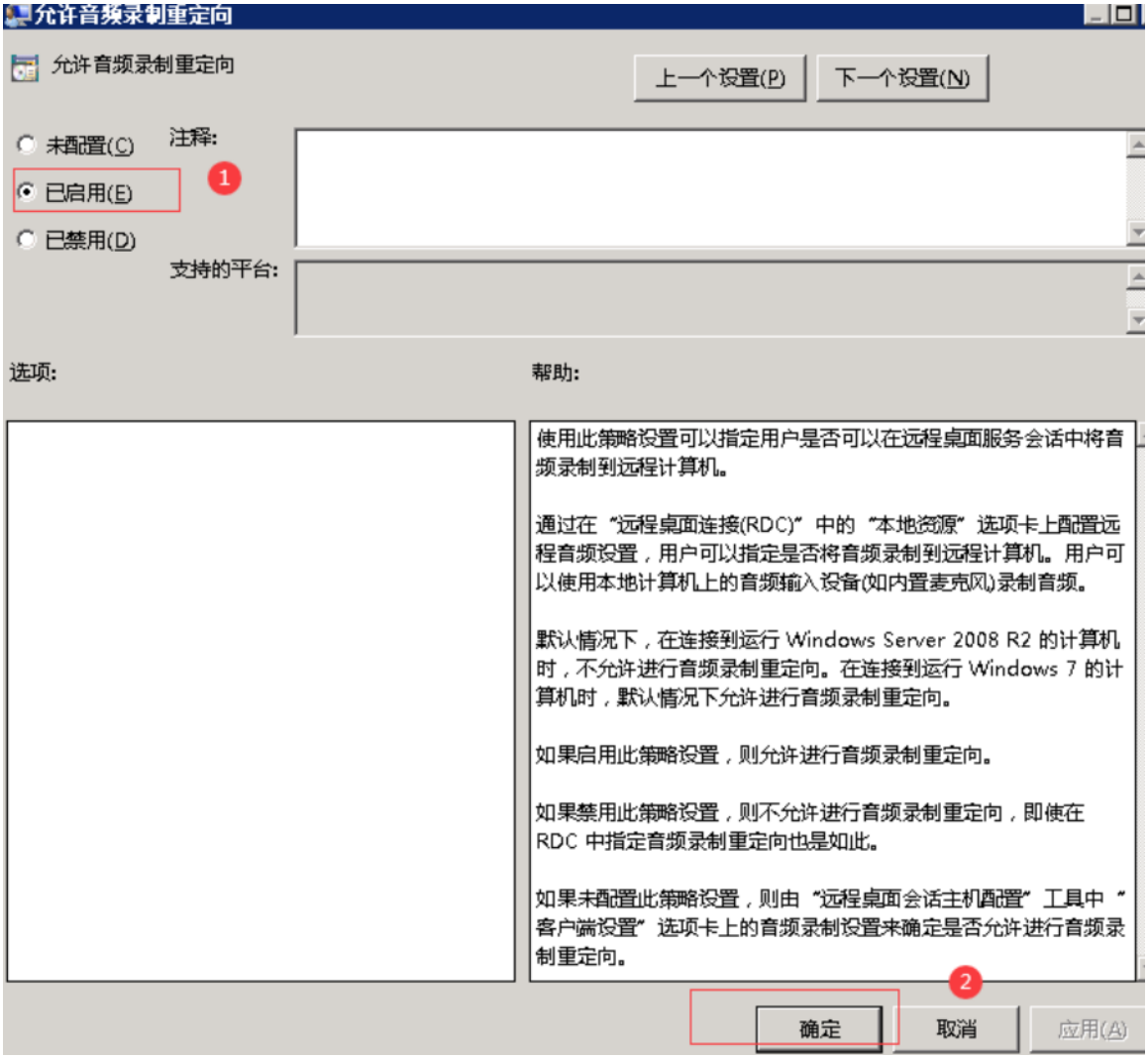
4.选择远程桌面会话主机->设备和资源重定向。



5.左键双击允许音频录制重定向 和 允许音频和视频播放重定向。



6.选择已启用后，点击应用、确定。



7.使用Win+R，输入：services.msc。

或者点击左下角的服务器管理器，在配置->服务中进入服务管理界面。

8.找到服务WindowsAudio与WindowsAudioEndpointBuilder，把启动类型修改成自启动，也检查下是否已经是运行状态。

9.重启服务器，然后远程连接至服务器，检测声音设置是否正常。

远程连接Windows后无法使用剪切板问题处理

本文介绍了当远程连接BCC Windows实例时，无法将本地复制的内容粘贴到服务器中的解决方案（以Windows2016版本为例）。

步骤一 登录服务器

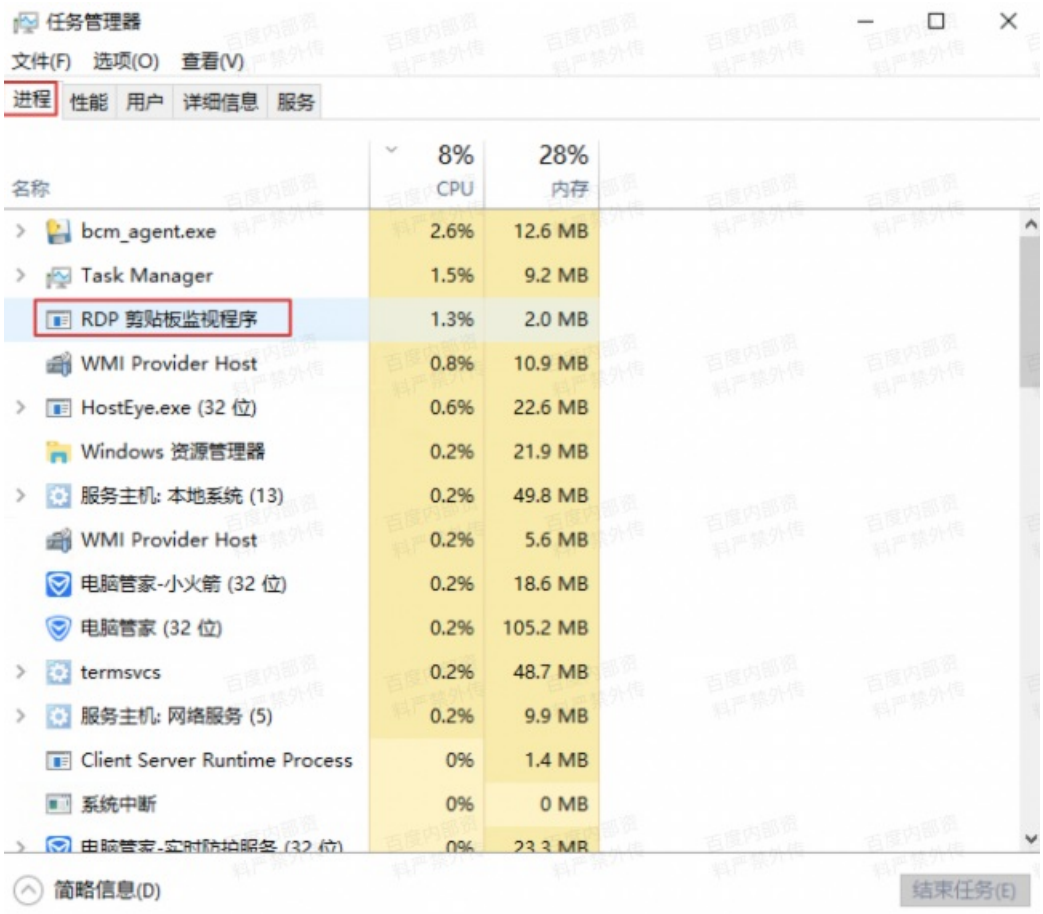
可以通过Windows自带的远程连接工具连接服务器，如果无法远程连接，使用VNC登录。

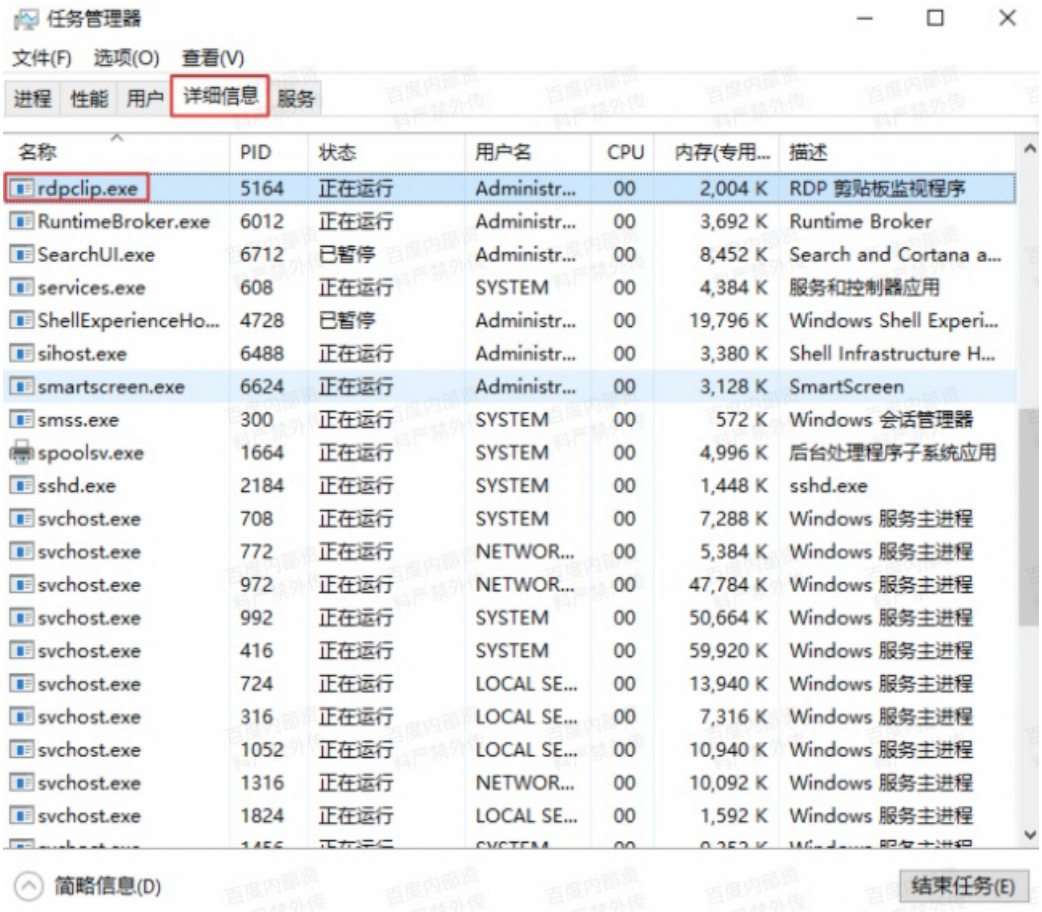
步骤二 打开任务管理器

直接鼠标右键点击任务栏，然后鼠标左键点击"任务管理器"。

步骤三 找到rdpclip.exe进程，结束任务

目前可以直接在"进程"板块，选择"RDP剪切板监视程序"，或者在"详细信息"板块，找到rdpclip.exe进程。

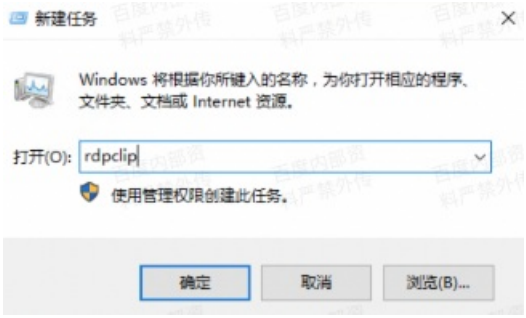




选中后点击右下角的"结束任务"。

步骤四 新建rdpclip任务

点击任务管理器左上角的"文件"，选择运行新任务，输入rdpclip，点击确定。



步骤五 验证是否可以粘贴内容

首先看下任务管理器中是否有"RDP剪切板监视程序"在运行，如果正常运行的话，直接复制本机内容，然后在服务器中进行粘贴，测试功能是否正常。

桌面及应用颜色变黑处理方法

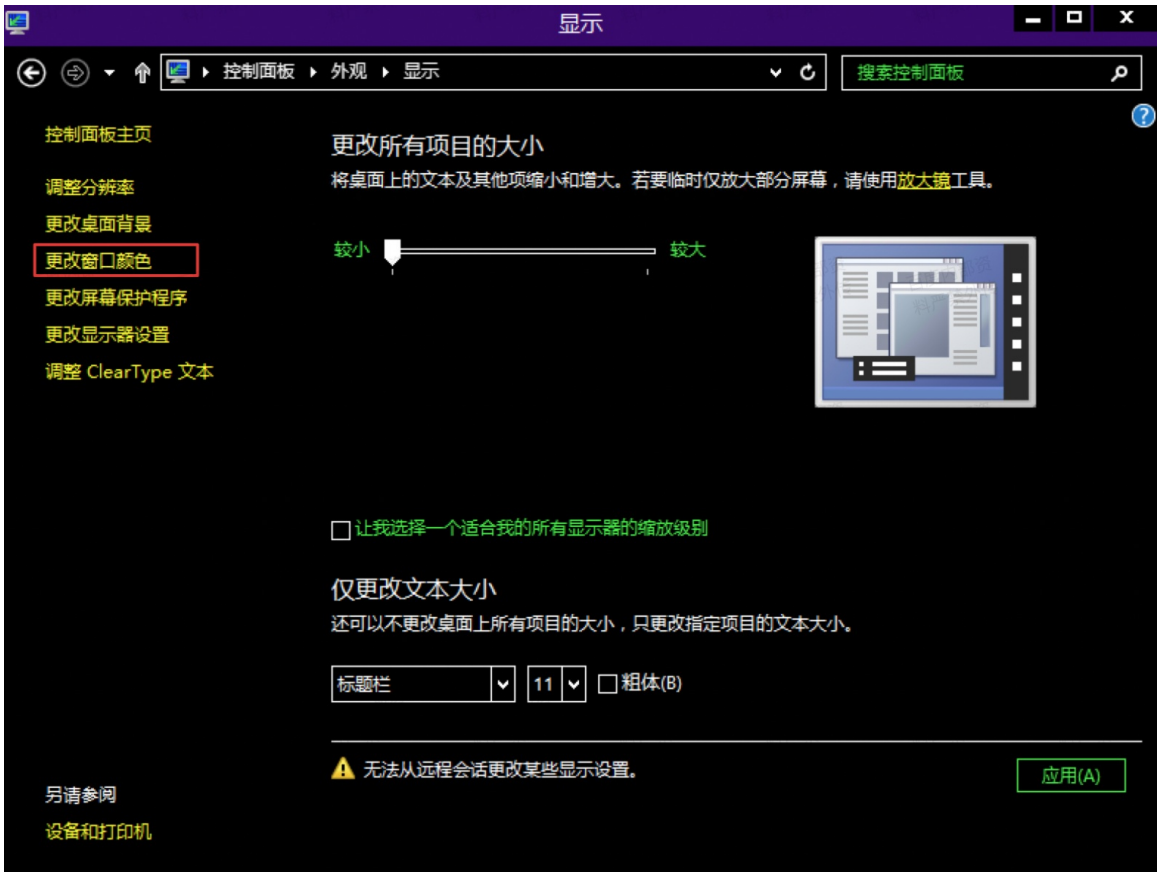
本文以Windows 2012版本为例，介绍了当桌面以及应用的背景色发生变化时，如何更改背景颜色的具体步骤和方法。

1、登录服务器，可以通过Windows自带的远程连接工具连接服务器，如果无法远程连接，使用VNC登录。

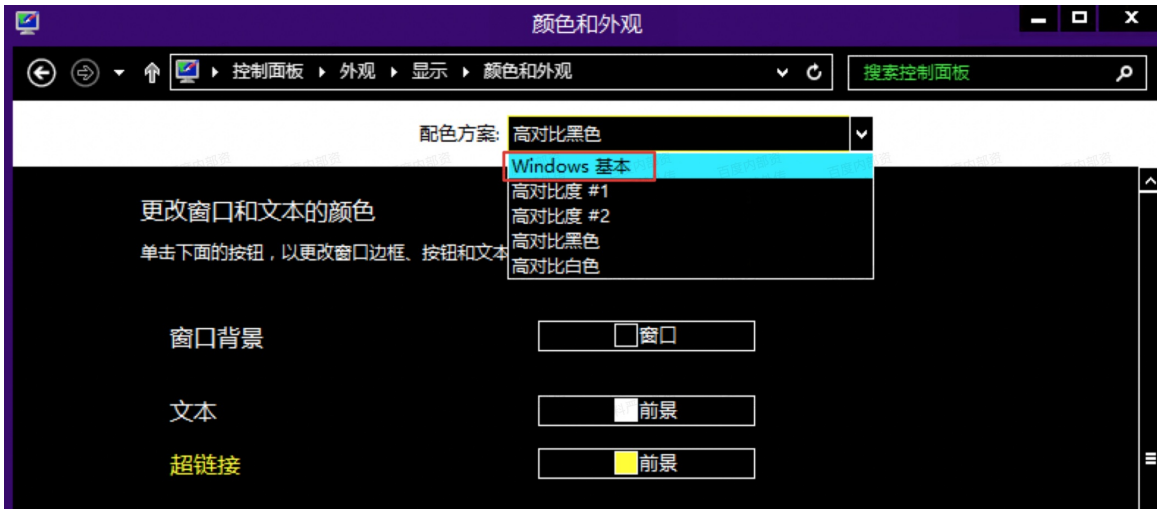
2、打开控制面板，在"控制面板"页点击"外观"，进入外观配置页面：

3、点击"显示"图标，修改系统显示配置页面：

4、点击左侧的"更改窗口颜色"：



5、将"配色方案"选择为"Windows基本"，点击"保存"，修改完成后能看到桌面以及应用的背景色已经恢复正常：



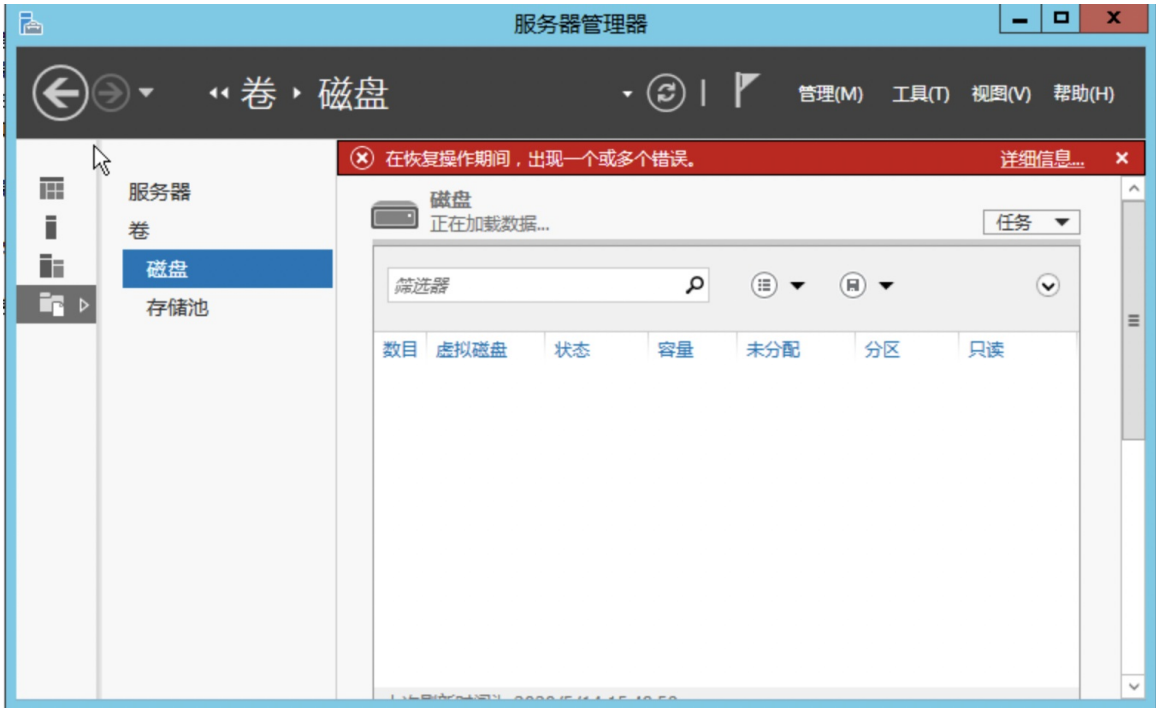
服务器重启后无法显示数据盘

问题描述

服务器重启后无法显示数据盘，按照如下步骤查看磁盘情况的报错信息：

1. 打开服务器管理器，点击服务器管理-文件和存储服务。

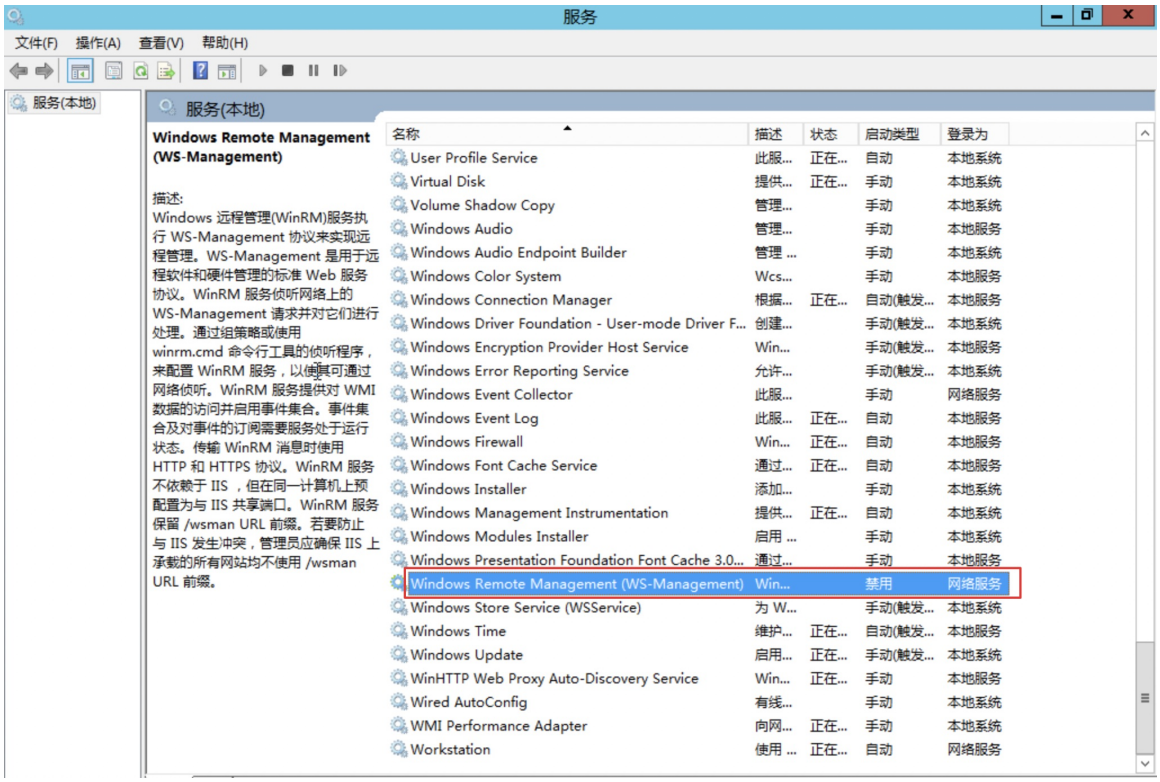
2. 点击"磁盘"后显示报错如下。



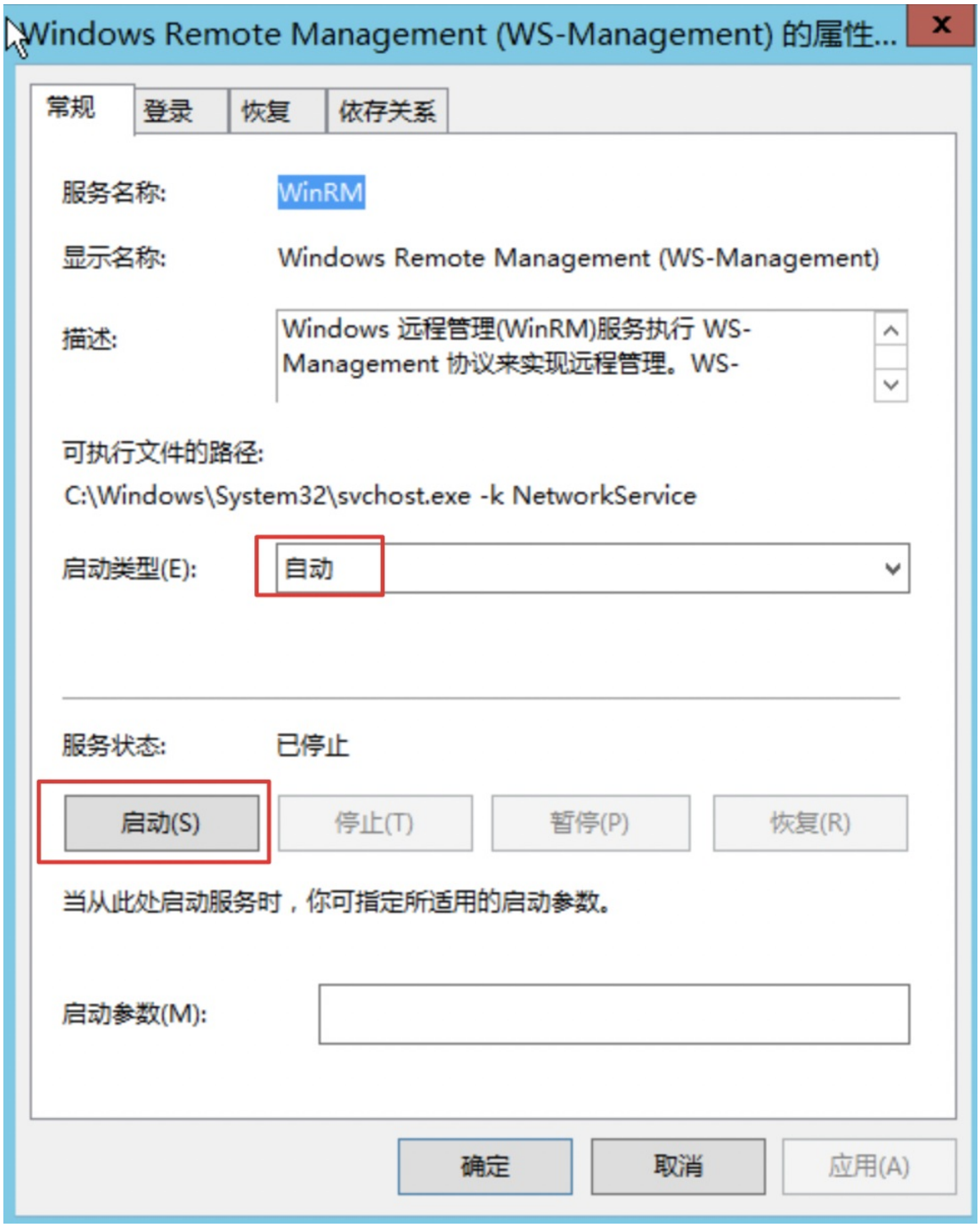
点击右上角的"详细信息"，具体报错信息如下：

解决方案

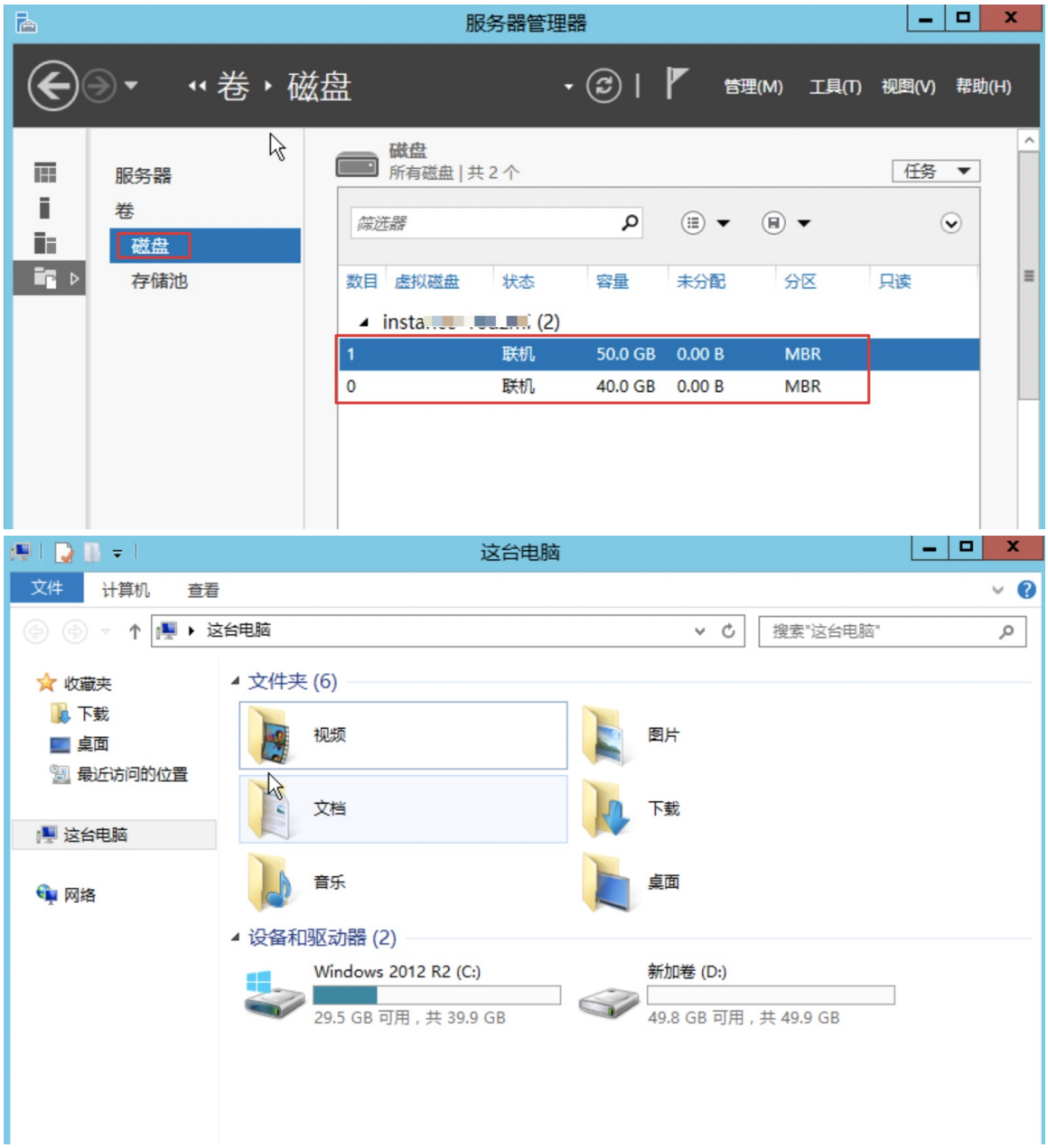
1. 打开"服务"列表，找到"Windows Remote Management (WS-Management)"服务，查看服务状态（该服务默认是启动的）。



2. 如果是禁用状态，需要设置为"自动"，并点击"启动"，以此来启动该服务。



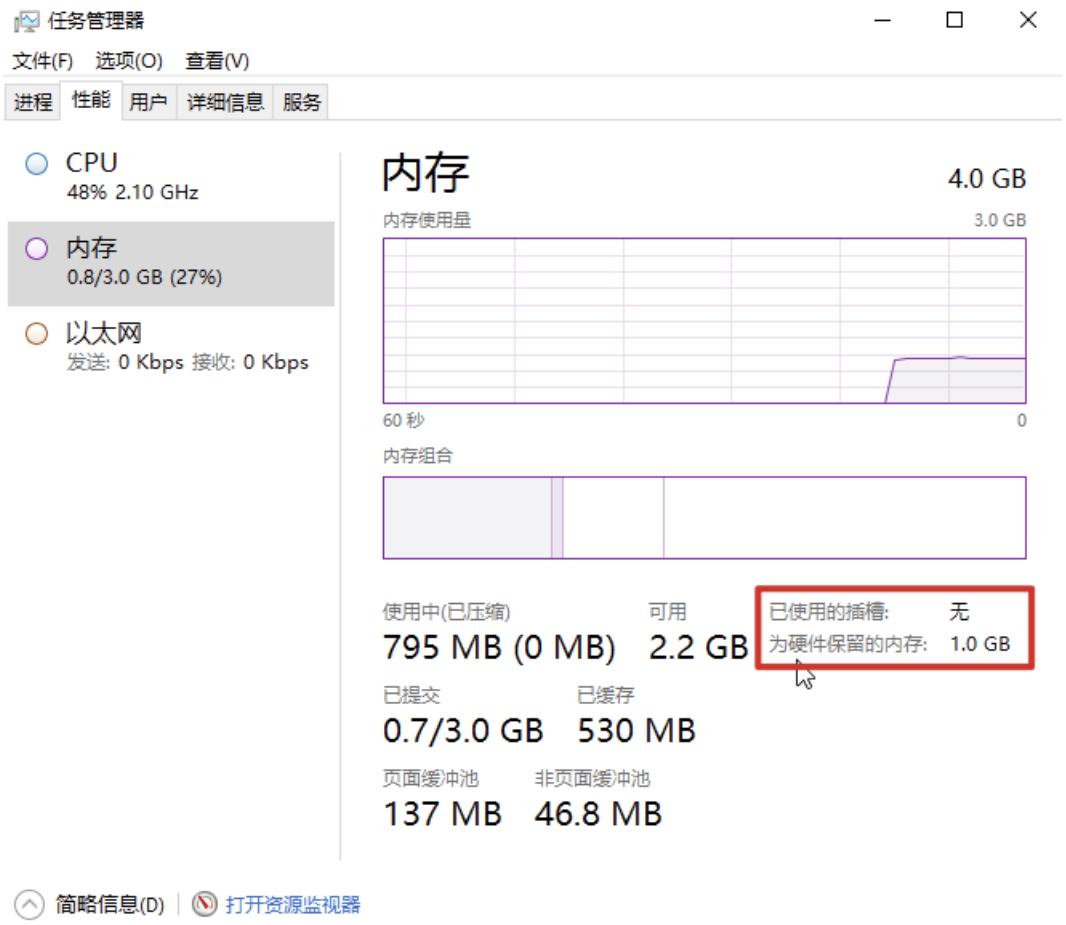
3. 启动Windows Remote Management (WS-Management)"服务后，重新打开服务器管理器，查看磁盘情况以及文件资源管理器中已恢复正常。



Windows释放为硬件保留内存

问题描述

任务管理器检测到物理内存总数与实际安装的内存总数相差过大，**为硬件保留内存** 占用大部分内存资源。



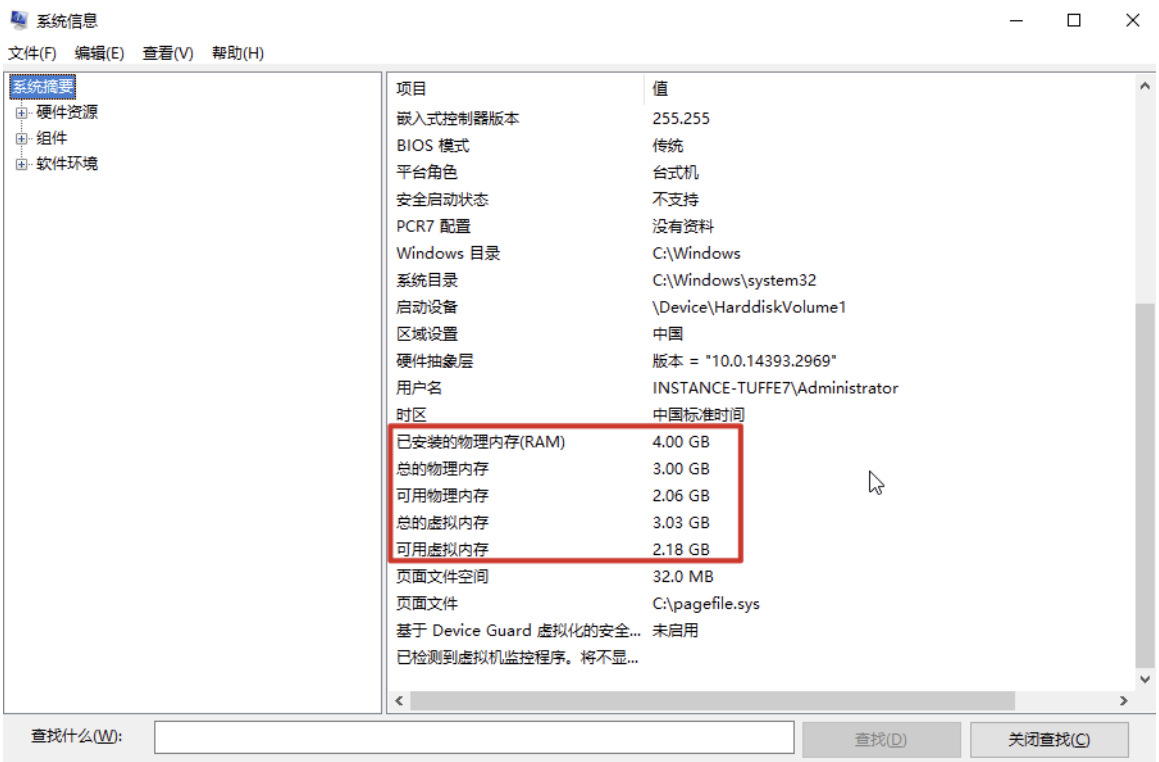
解决方案

方法一 msconfig设置最大内存

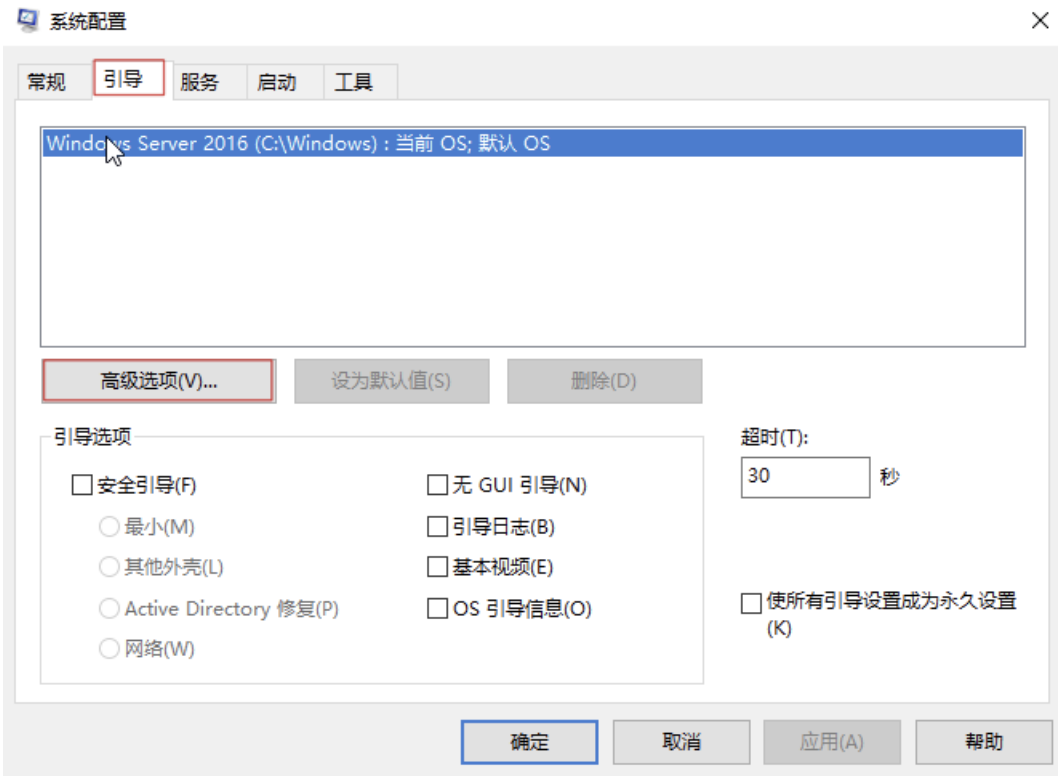
1. 检查Windows是否激活，如果未激活请先激活。

   激活方法请参考：系统激活。

2. 在cmd命令窗口输入msinfo32，查看系统摘要，可以看到可用内存和总内存相差很大。

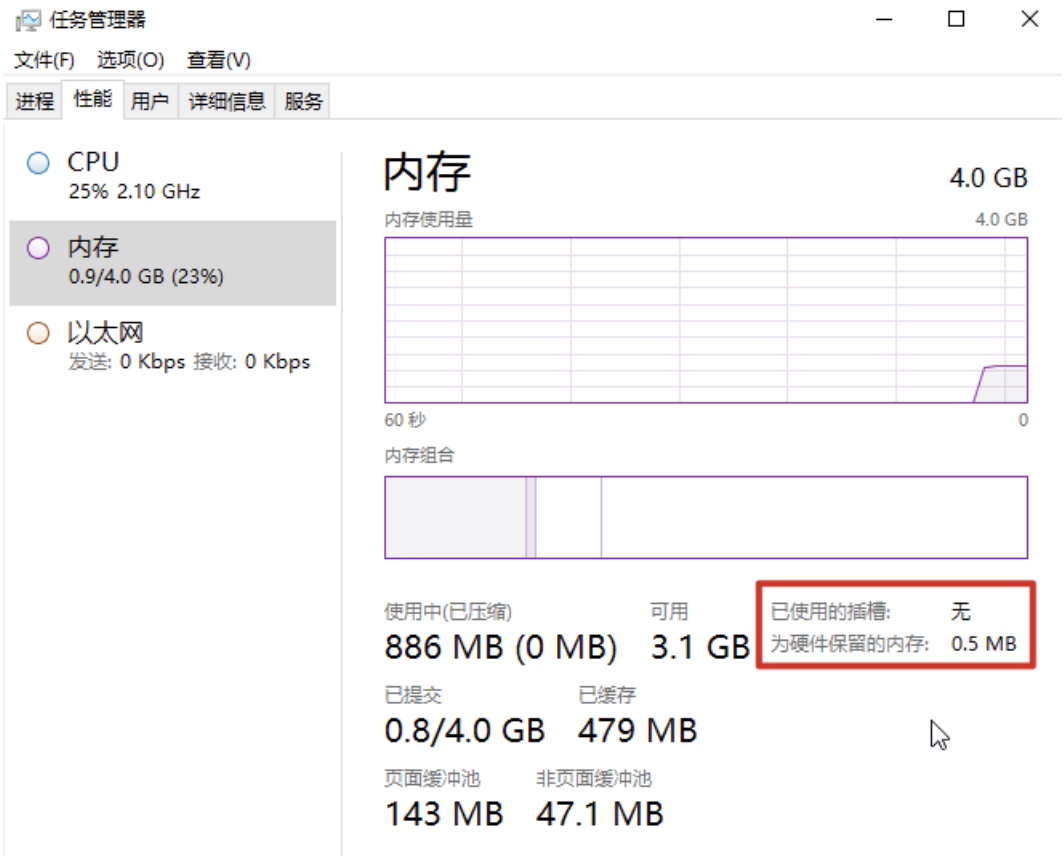


3. 在cmd命令窗口输入msconfig，进入 **系统配置** 页面，选择 **引导** 页，点击 **高级选项**。



4. 取消勾选 **最大内存**，点击确定。

5. 再次确认之后重启系统，问题即可解决。系统启动后，可以看到任务管理器中为硬件保留内存已恢复正常（解决之后硬件保留内存会在128M以内）。



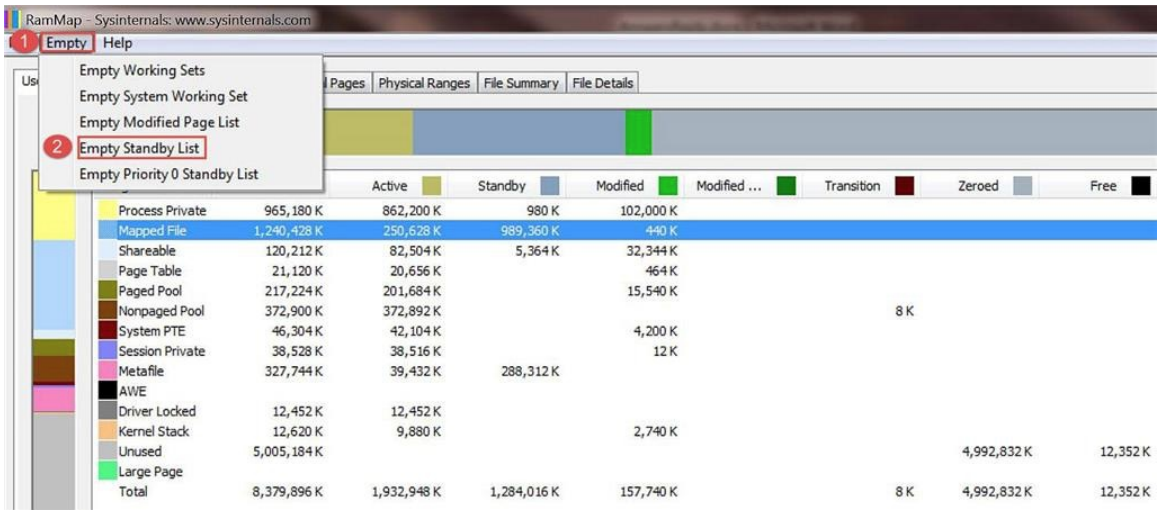
6. 如果按照如上步骤操作后，硬件保留内存还是有预留，再从步骤3开始操作一遍即可。

方法二 RamMap

需下载 RamMap 工具，清理备用内存，RamMap 工具微软官方下载地址如下：

https://docs.microsoft.com/zh-cn/sysinternals/downloads/rammap

点击菜单中的 **Empty**，然后点击 **Empty Standby List** 释放备用内存。



Windows实例磁盘空间不足清理教程

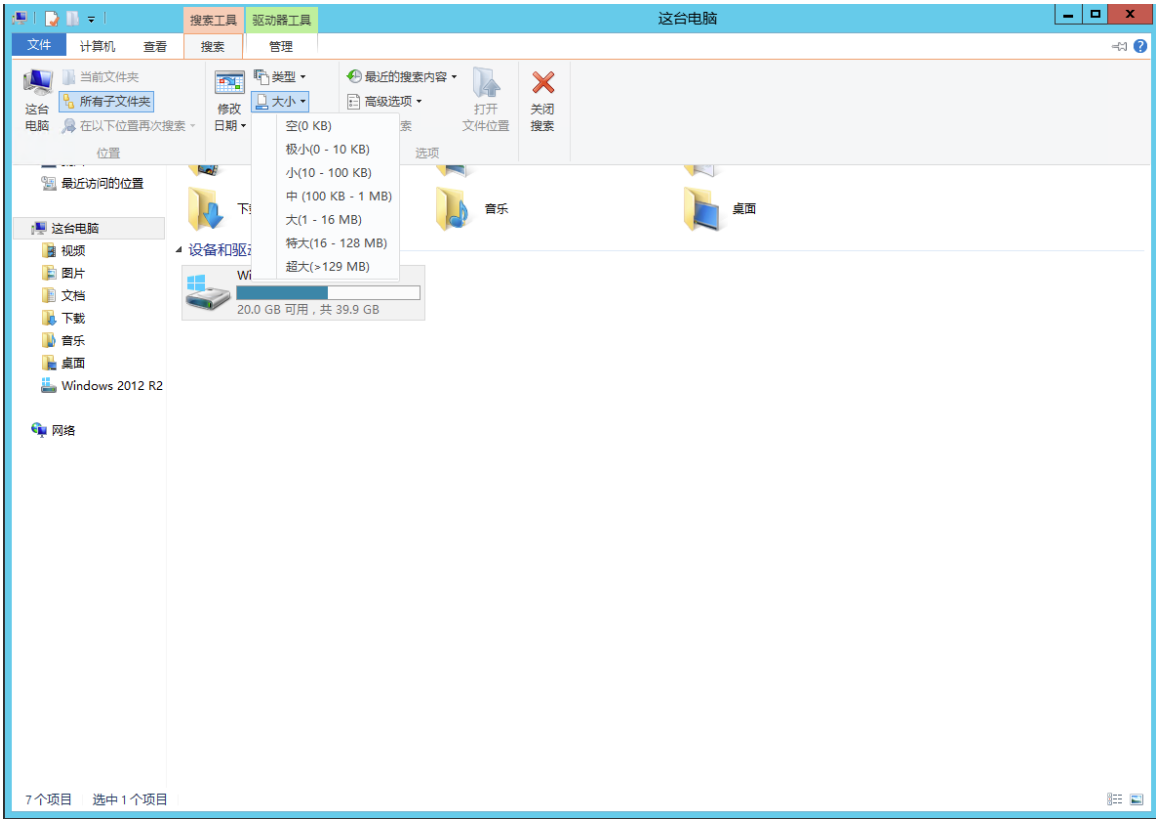
本文主要介绍BCC Windows实例磁盘空间不足时的清理方法以及磁盘日常维护方法，以百度智能云操作系统为 Windows Server 2012 R2 的云服务器为例进行演示。

操作步骤

释放磁盘空间

删除较大的文件

1. 使用远程桌面登录到Windows实例，详细操作请参考：使用远程桌面登录Windows实例
2. 双击桌面上的 “这台电脑”。
3. 在 “这台电脑” 中，单击需要清理的磁盘，按 Crtl + F 打开搜索工具。
4. 单击顶部导航栏 “搜索” > “大小”，然后根据系统定义大小筛选指定磁盘的大文件。

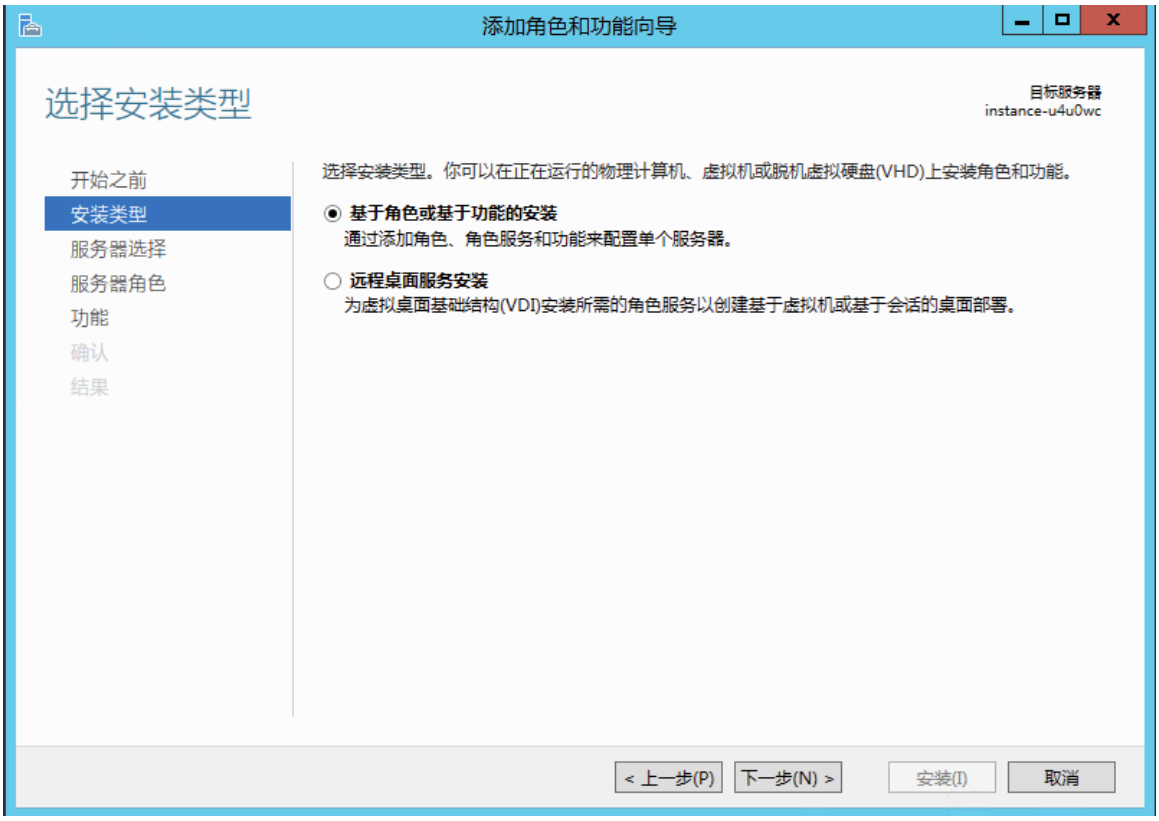


> 说明：您还可以在 “这台电脑” 右上角的搜索框中，自定义文件大小进行检索。例如： 输入“大小：>500M”，则会检索该磁盘中大于500M的文件。 输入“大小：> 100M < 500M”，则会检索该磁盘中大于100M但小于500M的文件。

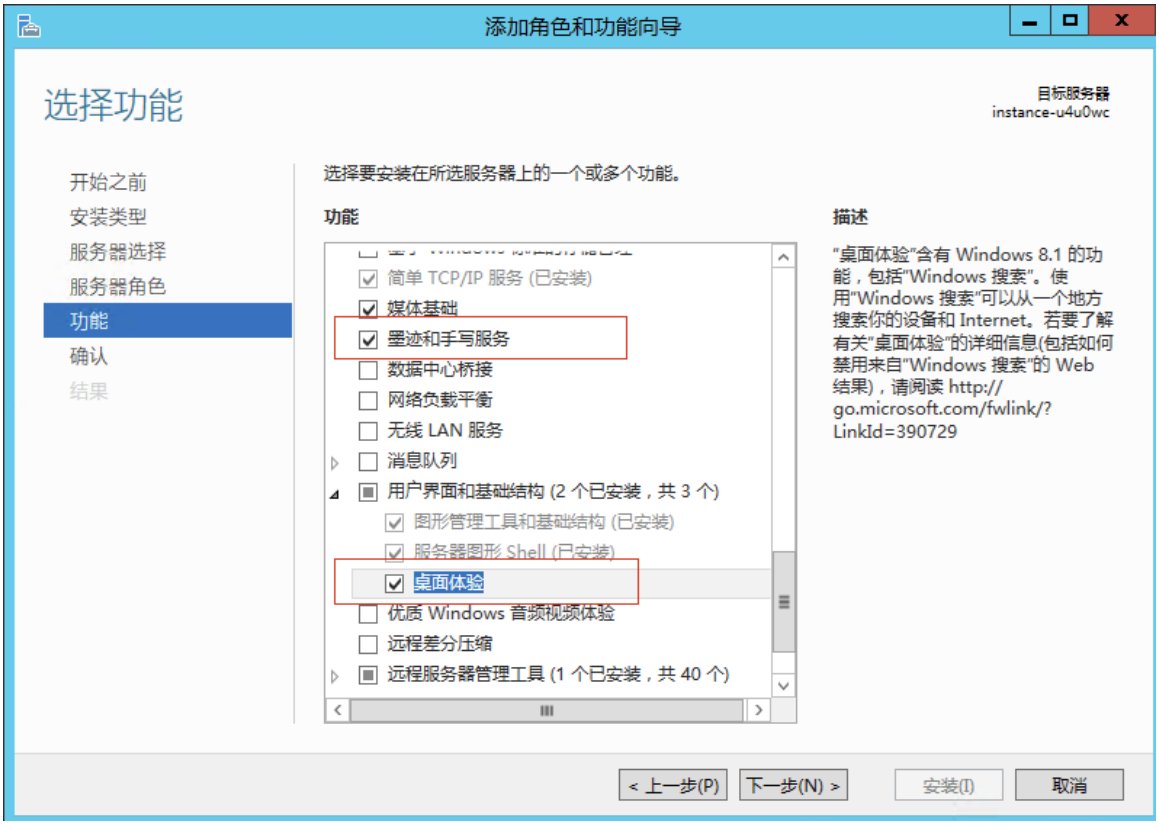
删除不用的文件

推荐您使用系统自带的磁盘清理工具进行文件删除，该工具需要在服务器上手动安装使用，具体步骤如下：

1. 在底部任务栏单击服务器管理器图标，打开服务器管理器。
2. 单击 “添加角色和功能”，弹出 “添加角色和功能向导” 窗口。
3. 在 “选择安装类型” 界面，选择 “基于角色或基于功能的安装”，并连续单击3次【下一步】，如下图所示：



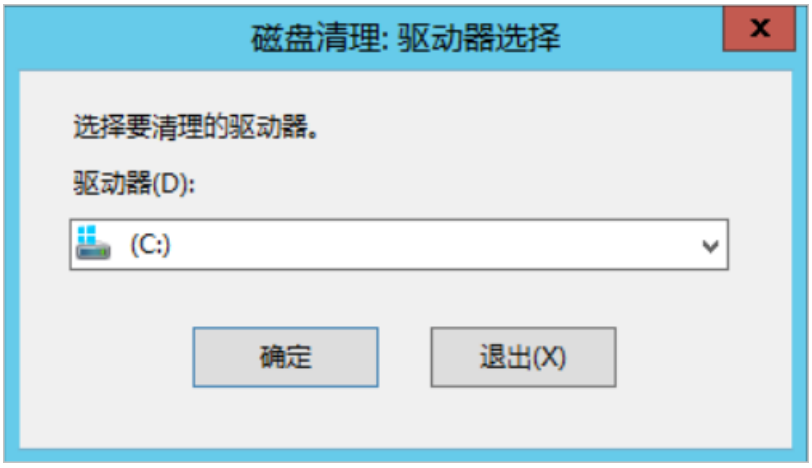
4. 在 “选择功能“ 界面勾选 “墨迹手写服务” 和 “桌面体验”，并在弹出的提示框中单击【确认】，如下图所示：

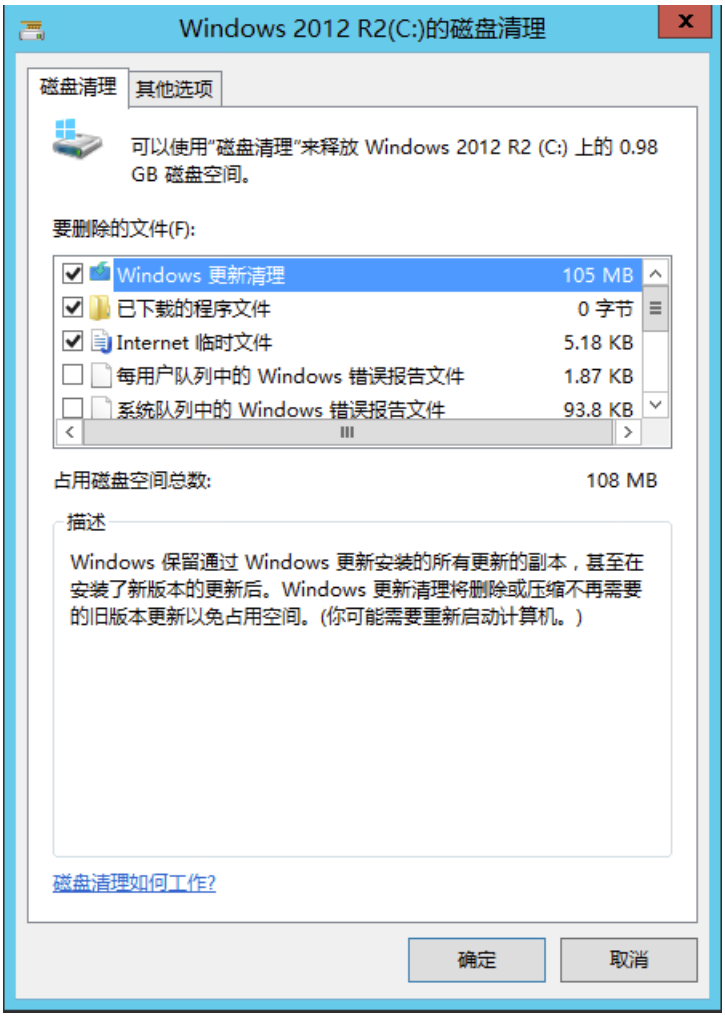


5. 选择【下一步】，并单击【安装】，安装完成后，系统将提示您手动重启服务器生效，您可自行选择合适的时间重启服务器。

6. 单击开始，在顶部搜索栏搜索磁盘清理，进入清理工具。

7. 选择要清理的磁盘，单击确定，开始清理，如下图所示：



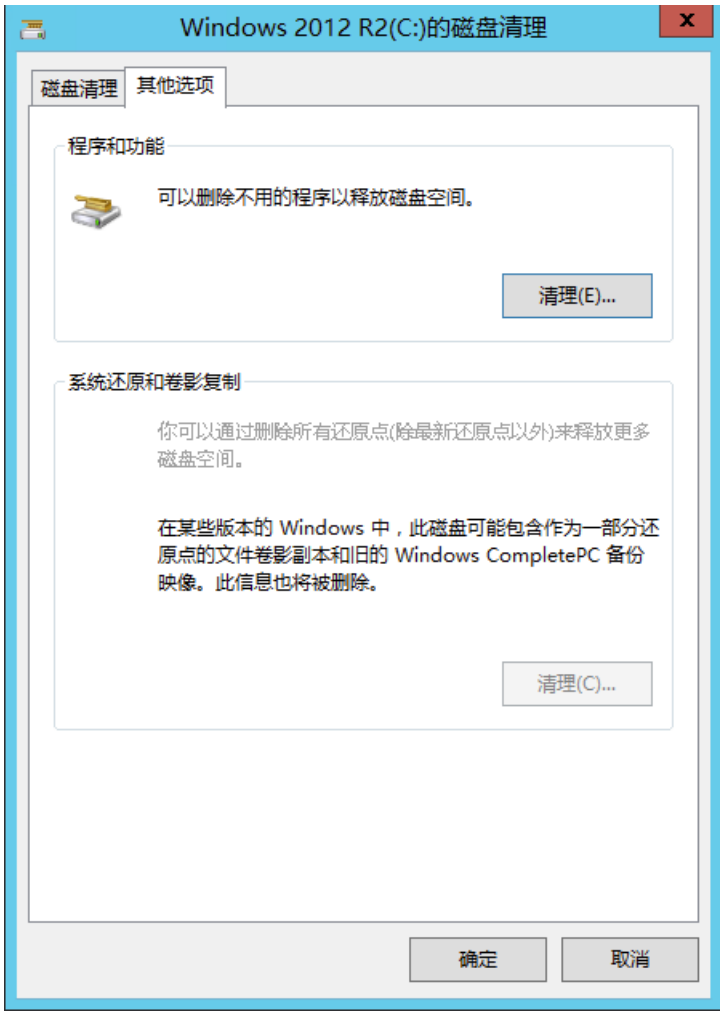


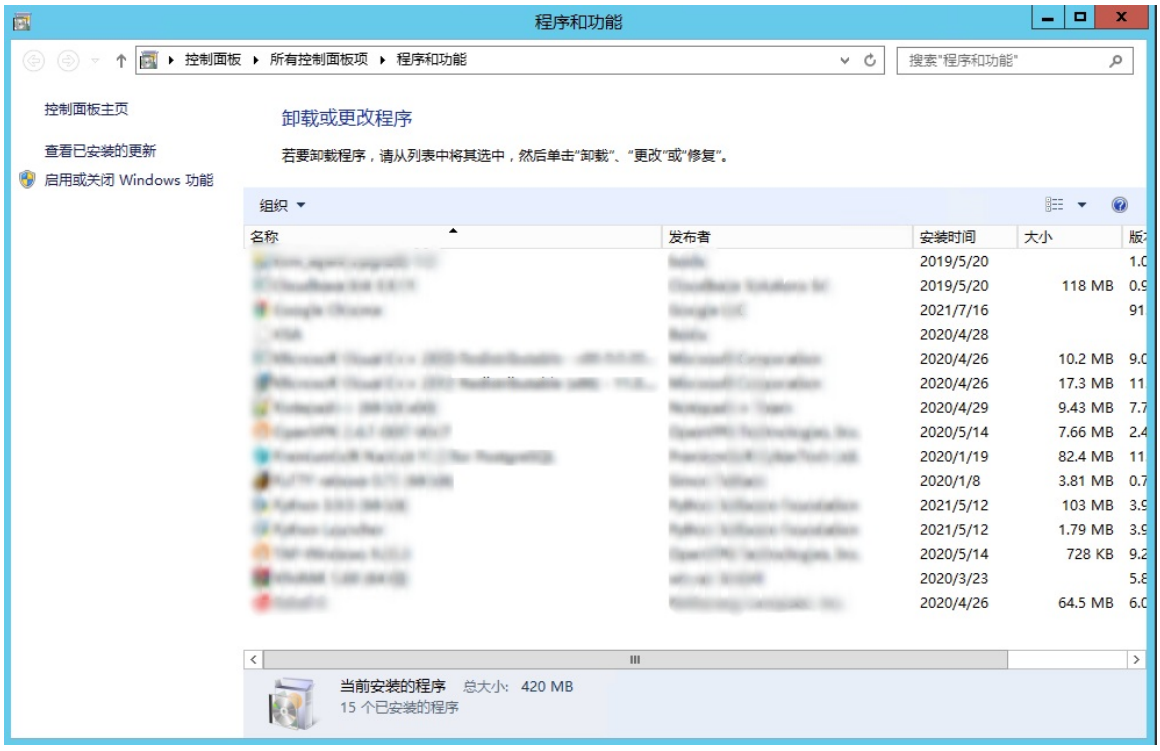
> 说明：如有特殊需求，也可以考虑使用第三方工具，比如 Dism++ 等专业清理软件，由于第三方软件非百度智能云产品，具体使用教程还请参考软件官网。 对于硬盘文件的查看，也可以考虑使用第三方软件，比如 TreeSize Free 等硬盘文件整理软件，具体使用教程同样请参考软件的官方网站。

磁盘日常维护

定期清理应用程序

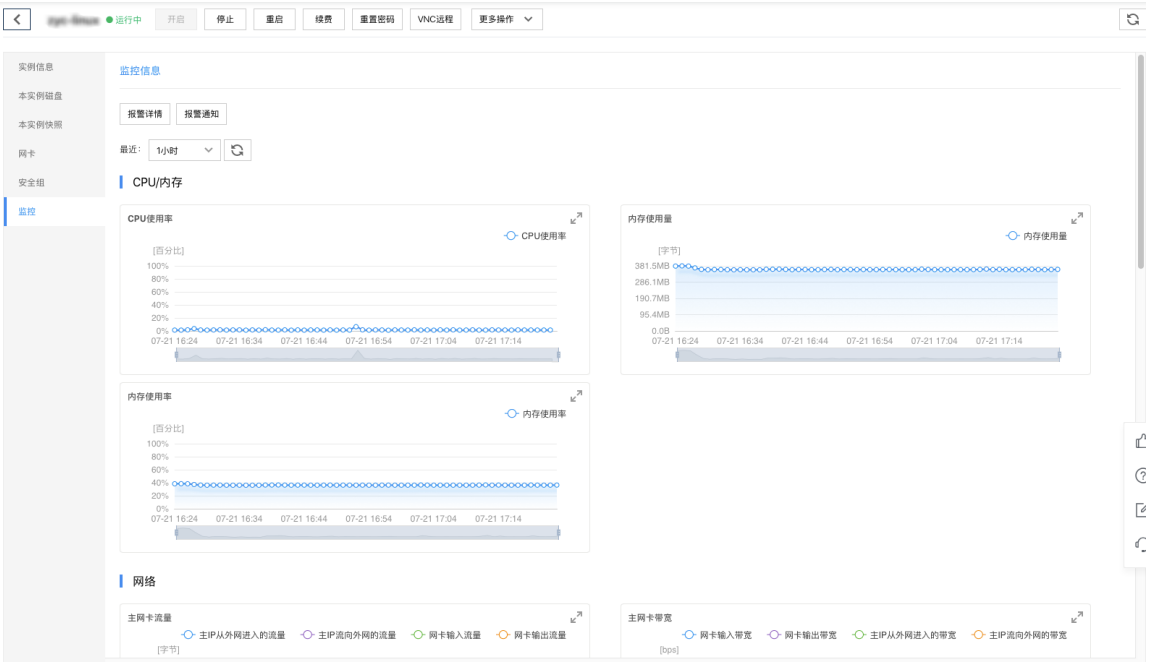
您可以从磁盘清理工具 “其他选项” 里点击 “清理” 进入应用程序卸载，也可以从 “控制面板” 中的 “卸载程序” 进入，定期清理不再使用的应用程序。



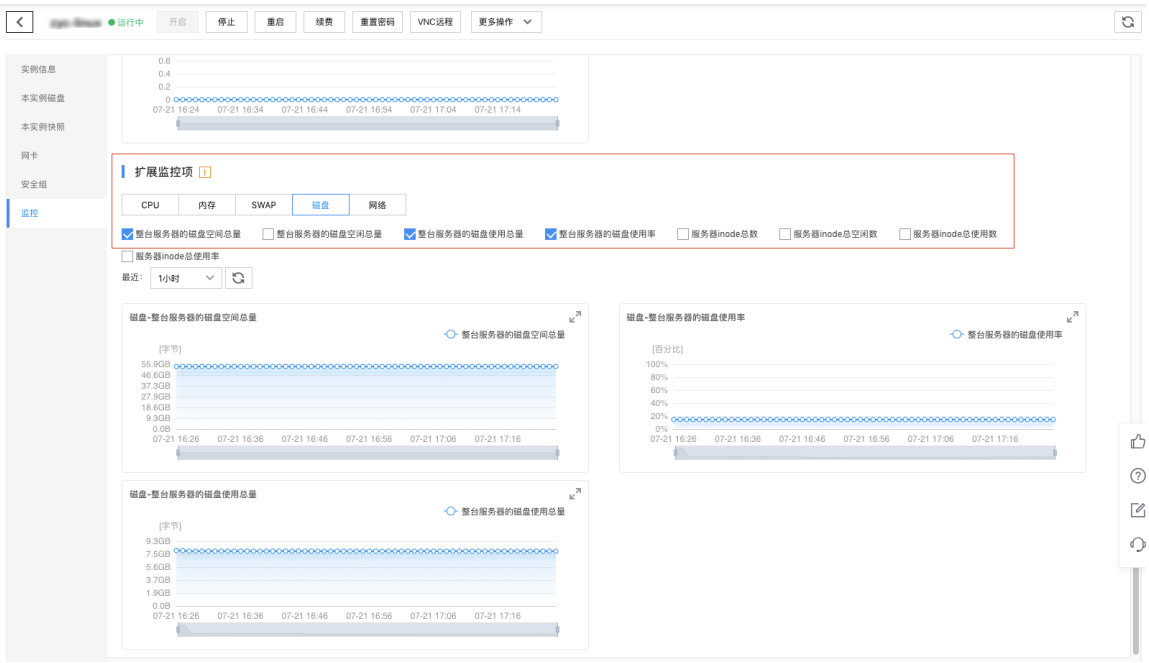


使用云服务器监控查看磁盘使用情况

1. 在BCC云服务器实例详情页面，点击监控进入服务器监控页面。



2. 在扩展监控项里面，可以选择 “整台服务器的磁盘空间总量”、“整台服务器的磁盘使用总量”、“整台服务器的磁盘使用率”这3个指标，对服务器磁盘空间使用情况进行查看。

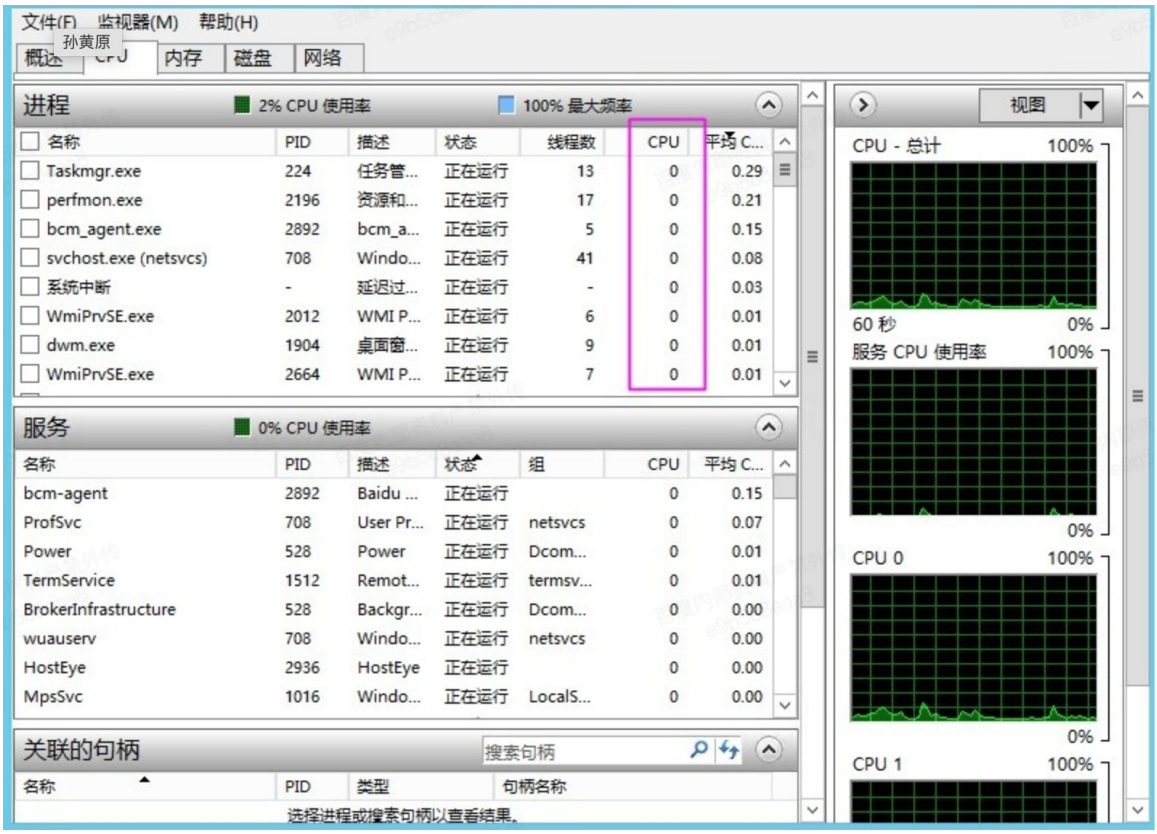


WindowsBCC实例卡顿问题

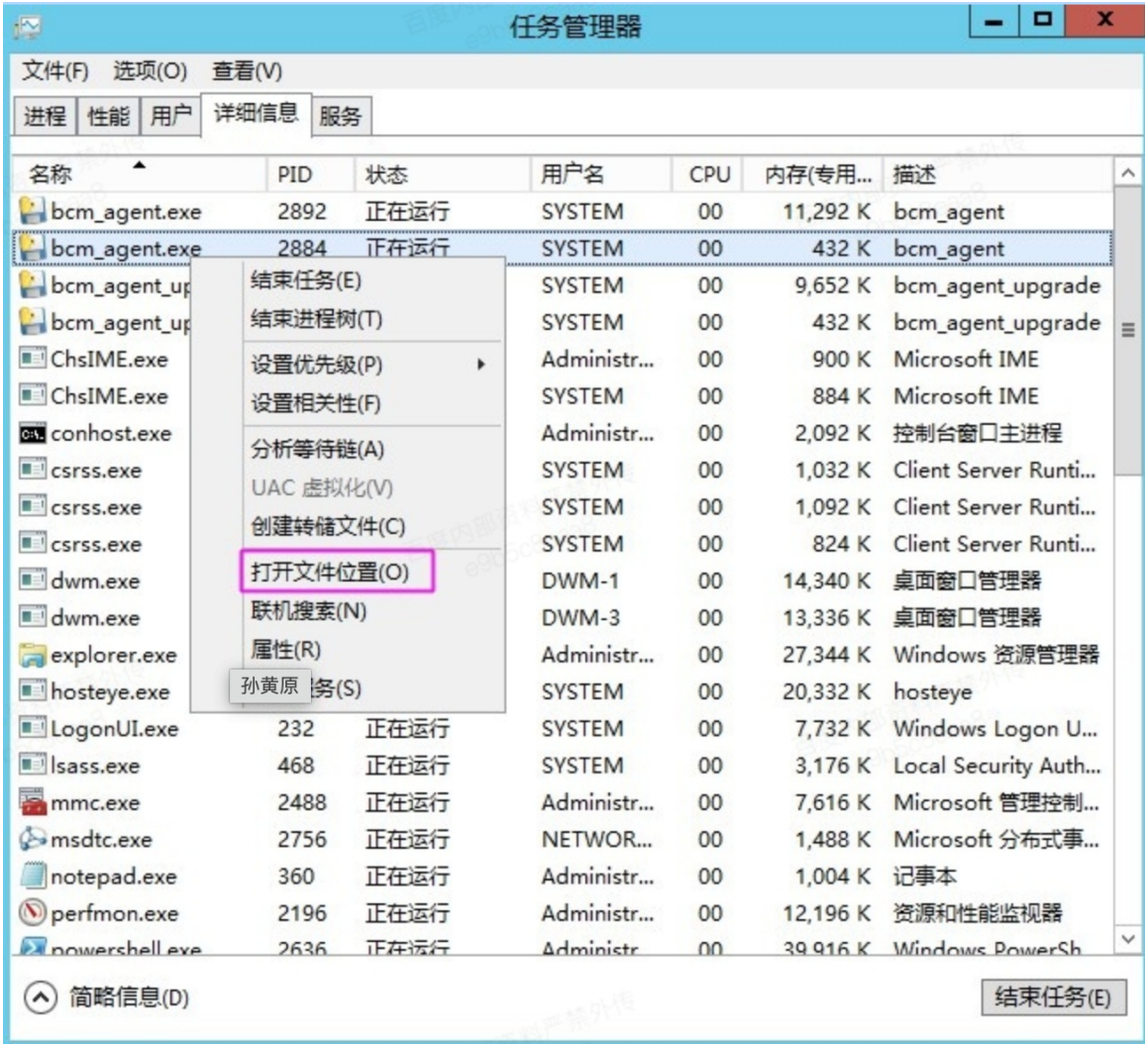
当发现云服务器实例运行速度变慢或突然出现网络断开的情况，可能是云服务器的CPU、内存或带宽使用率过高导致。

问题定位

1. 通过VNC方式登录云服务器。
2. 在命令行窗口，输入“perfmon -res”，打开资源监视器。
3. 在资源监视器中，查看CPU、内存或网络的使用情况。



4. 查看资源使用率较高的进程名称或进程ID。
5. 在命令行窗口，输入“taskmgr”，打开任务管理器。
6. 在任务管理器“详细信息”项中，根据进程名称或进程ID，找到对应进程，右键选择“打开文件位置”，即可定位程序所在目录。



7. 判断进程是否为正常程序。

分析处理

正常进程处理建议

1. 检查后台是否正在执行Windows Update，进行系统更新。
2. 重新判断业务层面是否对CPU、内存、网络等有高的需求，如果是，参考 实例配置变更。
3. 参考 Windows Server硬件要求。

异常进程处理建议

如果CPU或带宽利用率高是由于病毒、木马入侵导致的，需要手动结束进程并对程序文件进行清理，并进行如下操作：

1. 使用Microsoft 安全扫描程序，在安全模式下进行病毒扫描。
2. 及时更新Windows安全补丁。
3. 使用msconfig禁用所有非微软自带服务驱动，检查问题是否再次发生，参考 如何在 Windows中执行干净启动。

WindowsBCC实例入域后实现SID唯一

本文以Windows Server 2012 R2为例，指导您在Windows系统上执行Sysprep，保证系统SID唯一。

操作场景

对于需要加入域的Windows云服务器，必须保证SID唯一，否则会导致加入域失败。

基于这个原因，根据已有实例创建自定义镜像前，必须执行Sysprep，来保证SID唯一。

前提条件

Windows云服务器必须为正版系统，且已激活。

必须使用镜像自带的Sysprep版本，所在目录%WINDIR%\System32\Sysprep。

剩余Windows重置计数必须大于1。

通过执行slmgr.vbs /dlv可以查看剩余计数值。

确保已安装cloudbase-init，参考安装cloudbase-init。

以Administrator账号登录云服务器。

操作步骤

1. 进入cloudbase-init的conf目录，复制Unattend.xml，并命名为Unattend_sysprep.xml，默认路径C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf。
2. 修改Unattend_sysprep.xml文件内容。

```
 **Sysprep应答文件**
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
 <settings pass="oobeSystem">
  <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
    <OOBE>
     <HideEULAPage>true</HideEULAPage>
     <NetworkLocation>Work</NetworkLocation>
     <ProtectYourPC>1</ProtectYourPC>
     <SkipMachineOOBE>true</SkipMachineOOBE>
     <SkipUserOOBE>true</SkipUserOOBE>
    </OOBE>
  </component>
 </settings>
</unattend>
```

3. 打开命令行界面。
4. cd "C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf"。
5. 执行命令： C:\Windows\System32\Sysprep\sysprep.exe /quiet /generalize /oobe /shutdown /unattend:Unattend_sysprep.xml
6. 等待Sysprep执行完毕，并会自动关机，等待完成。
7. 制作自定义镜像，参考创建自定义镜像。

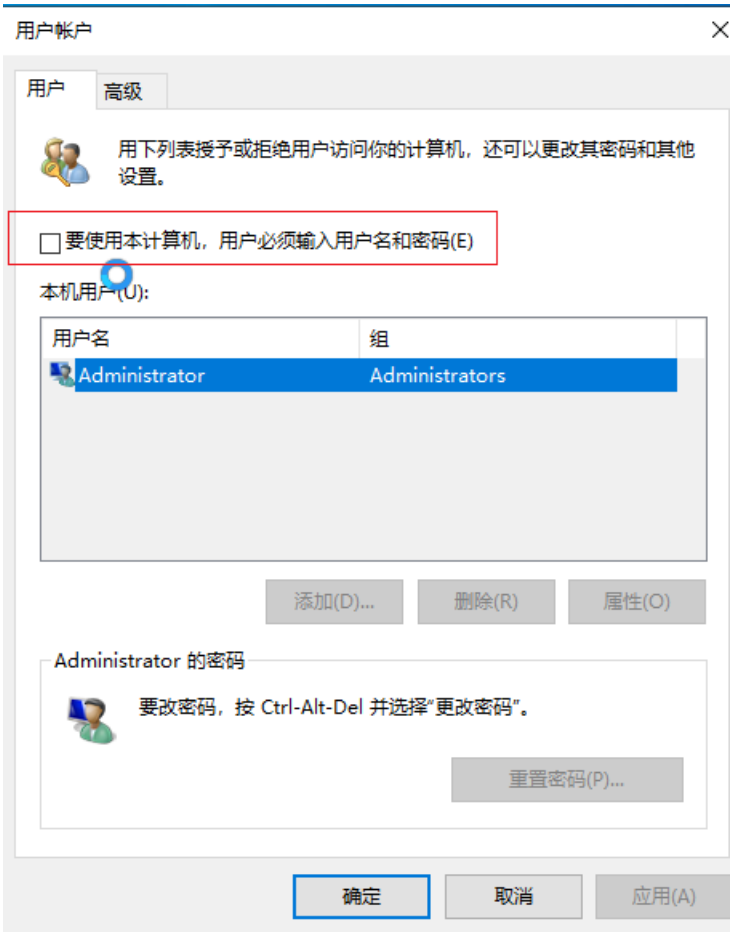
> 注意事项
>
> 为了保证使用此镜像创建云服务器SID唯一，在创建自定义镜像之前，不要重新启动该云服务器实例。
>
> 执行Sysprep后，该云服务器实例的密码会被清除，重启实例后，请重置密码。
>
> 可以通过执行 whoami /user 查看实例SID。
>
> 更多Sysprep信息参考 Sysprep微软官网。

Windows虚机自动登录失败处理办法

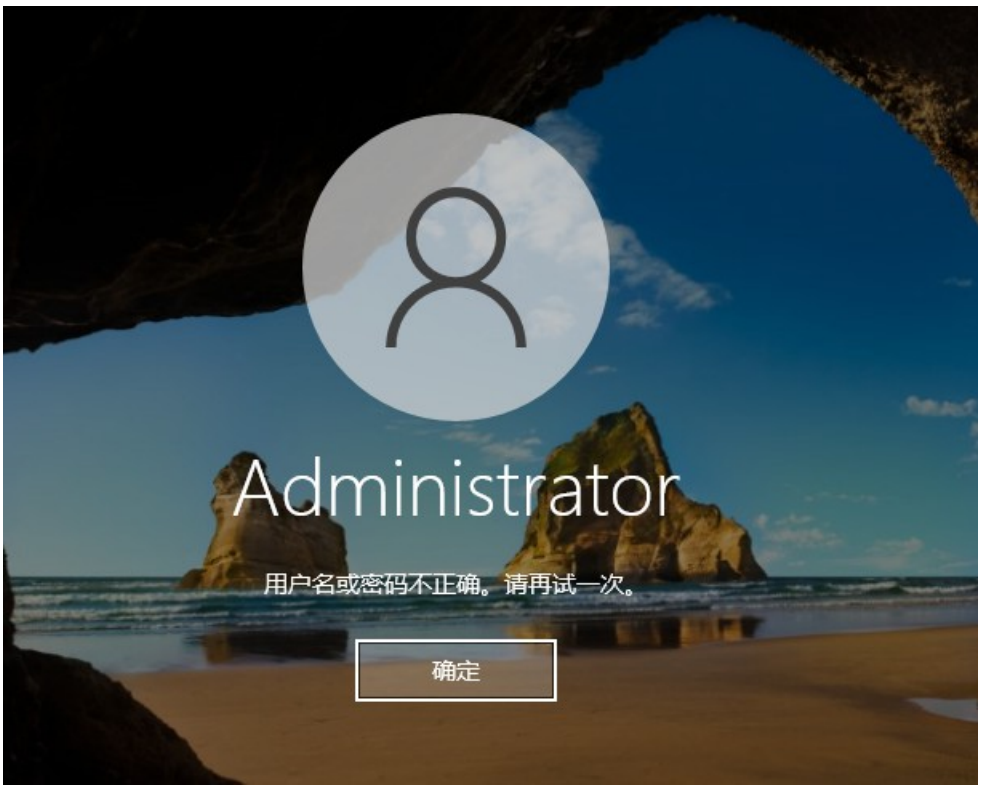
问题现象

部分百度智能云的渲染类客户会基于百度智能云的Windows公共镜像修改Administrator账号密码，并开启免密自动登录功能，

配置方法如下：



然后创建自定义镜像。用户再基于该自定义镜像创建虚机(创建虚机时指定的密码和自定义镜像密码一致)。但是在虚机第一次启动时有很大的概率会出现自动登录失败，现象如下：



问题原因

新创建的虚机在第一次启动时会先注入一个随机密码，之后再注入创建虚机时指定的密码。这样做的原因是为了安全，防止密码注入失败时所有虚机都用同一个密码。但是新密码注入的时机和Windows自动登录的时机的先后顺序不固定，如果注入密码在自动登录之后，那自动登录就会失败。 解决办法就是修改cloudbase-init的createuser.py脚本，不注入随机密码，而是直接注入Windows用户制作自定义镜像时指定的密码。

处理步骤

1. 备份C:\Program Files\Cloudbase Solutions\Cloudbase-Init\Python\Lib\site-packages\cloudbaseinit\plugins\common\createuser.py。
2. 用本文的createuser.py替换C:\Program Files\Cloudbase Solutions\Cloudbase-Init\Python\Lib\site-packages\cloudbaseinit\plugins\common\createuser.py。

createuser.py

3. 重新制作自定义镜像。

Linux运维参考

Linux服务器CPU内存跑满问题排查

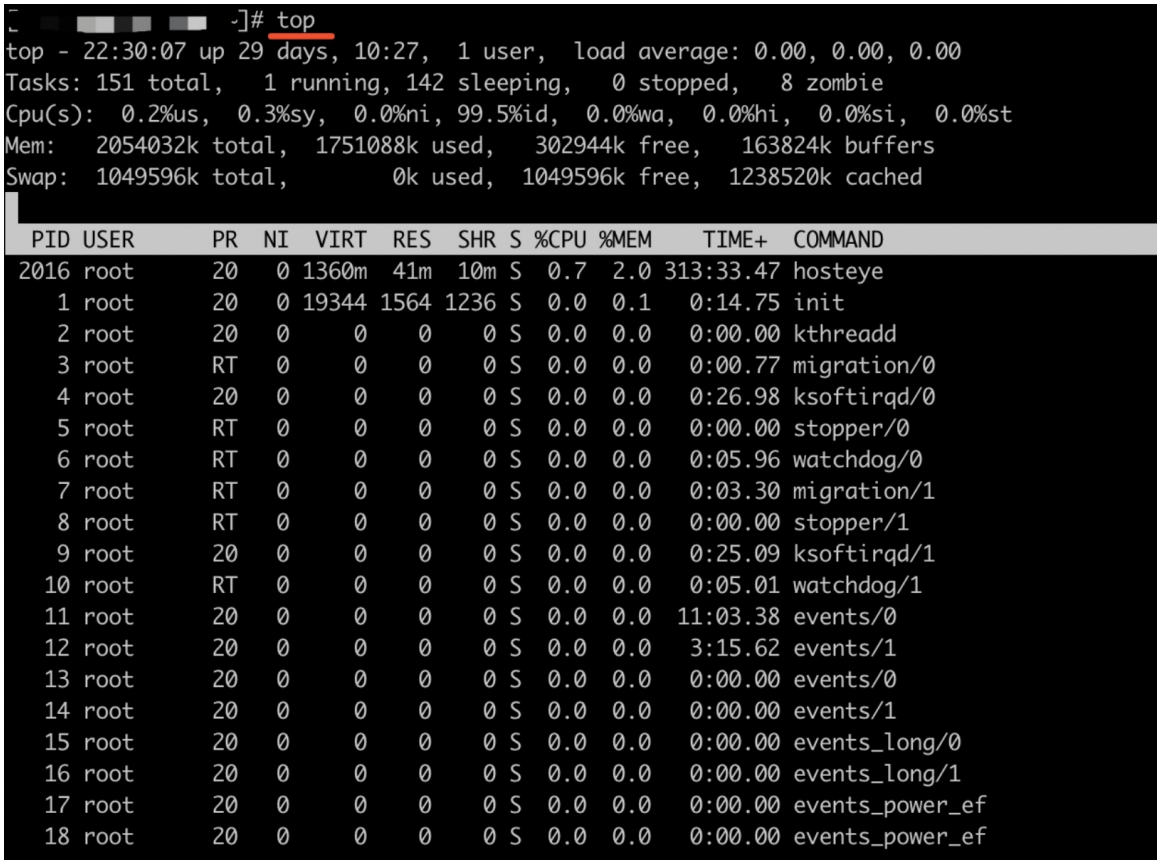
本文介绍了如果在使用服务器的过程中出现卡顿、登录服务器后异常退出等情况，针对服务器内部资源的排查步骤，以Centos7为例。

步骤一 登录服务器

可以通过ssh远程连接服务器，如果无法远程连接，可以使用VNC登录。

步骤二 查看进程占用资源情况

1.执行"top"查看当前CPU、内存资源使用情况。



```
[ ]# top
top - 22:30:07 up 29 days, 10:27,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 151 total,   1 running, 142 sleeping,   0 stopped,   8 zombie
Cpu(s):  0.2%us,  0.3%sy,  0.0%ni, 99.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2054032k total,  1751088k used,   302944k free,   163824k buffers
Swap:  1049596k total,        0k used,  1049596k free,  1238520k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2016 root      20   0 1360m  41m  10m S  0.7  2.0 313:33.47 hosteye
    1 root      20   0 19344 1564 1236 S  0.0  0.1   0:14.75 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.77 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:26.98 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:05.96 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:03.30 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/1
    9 root      20   0     0    0    0 S  0.0  0.0   0:25.09 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:05.01 watchdog/1
   11 root      20   0     0    0    0 S  0.0  0.0  11:03.38 events/0
   12 root      20   0     0    0    0 S  0.0  0.0   3:15.62 events/1
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events/0
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events/1
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_long/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_long/1
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_power_ef
   18 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_power_ef
```

2.如果是正常进程占用大量资源，可以考虑对CPU、内存进行扩容；如果是异常进程占用大量资源，可以在做好快照的前提下，先结束该进程，具体步骤如下：

1）执行top后显示的第一列为进程ID，以第一行的hosteye进程为例，进程ID为2016。

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2016 root      20   0 1360m  41m  10m S  0.7  2.0 313:33.91 hosteye
    1 root      20   0 19344 1564 1236 S  0.0  0.1   0:14.75 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.77 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:26.98 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:05.96 watchdog/0
```

2）执行ps -ef | grep 2016 ，查看进程的具体信息，如果确认占用大量资源的进程是异常的，不是自己启动的，可以执行kill -9 ID杀掉进程。终止进程后观察服务器状态是否恢复正常。

```
[ ]# ps -ef |grep 2016
root      2016  2013  1 Nov05 ?        05:13:34 /opt/hosteye/bin/hosteye --is_child_mode=true --is_console_mode=false --start_mode=0
root      5825  5791  0 22:31 pts/9    00:00:00 grep 2016
[ ]# kill -9 2016
```

Linux修改默认远程连接端口

本文介绍使用百度智能云公共镜像Linux系统如何修改SSH默认远程端口的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《Linux修改SSH默认端口号》

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

步骤一 登录服务器

可以通过ssh远程连接服务器，如果无法远程连接，可以使用VNC登录。

步骤二 找到ssh服务配置文件

一般都是在 /etc/ssh/这个目录下面的sshd_config 文件。

```
[root@instance ssh]# cd /etc/ssh/
[root@instance ssh]# ls -l
总用量 604
-rw-r--r--. 1 root root     581843 8月    9 09:40 moduli
-rw-r--r--. 1 root root       2276 8月    9 09:40 ssh_config
-rw-------  1 root root       3927 12月  24 16:41 sshd_config
-rw-r-----  1 root ssh_keys    227 12月  24 14:09 ssh_host_ecdsa_key
-rw-r--r--  1 root root        162 12月  24 14:09 ssh_host_ecdsa_key.pub
-rw-r-----  1 root ssh_keys    387 12月  24 14:09 ssh_host_ed25519_key
-rw-r--r--  1 root root         82 12月  24 14:09 ssh_host_ed25519_key.pub
-rw-r-----  1 root ssh_keys   1675 12月  24 14:09 ssh_host_rsa_key
-rw-r--r--  1 root root        382 12月  24 14:09 ssh_host_rsa_key.pub
```

步骤三 将端口修改为自定义端口

以root身份编辑文件 /etc/ssh/sshd_config文件，找到 #Port 22，最好是在#Port 22 下面添加一个新的端口，比如Port 2333，修改后保存文件。

```
#       $OpenBSD: sshd_config,v 1.100 2016/08/15 12:32:04 naddy Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
Port 2333
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
```

步骤四 设置防火墙规则

如果防火墙是开启的，需要新增防火墙规则，如果防火墙是关闭的，可以忽略该步骤。

```
 #查看防火墙状态，如果是running的状态，可以按照如下步骤进行操作
[root@insta...sq4 ~]#  systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: active (running) since 二 2019-12-24 16:48:44 CST; 17min ago
     Docs: man:firewalld(1)
 Main PID: 31543 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─31543 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

12月 24 16:48:44 instance-yg8jrsq4 systemd[1]: Starting firewalld - dynamic firewall daemon...
12月 24 16:48:44 instance-yg8jrsq4 systemd[1]: Started firewalld - dynamic firewall daemon.
12月 24 16:49:16 instance-yg8jrsq4 firewalld[31543]: WARNING: ALREADY_ENABLED: 2333:tcp
#防火墙规则新增2333端口
[root@insta...sq4 ~]# firewall-cmd --zone=public --add-port=2333/tcp --permanent
#查看2333端口是否新增成功
[root@insta...sq4 ~]# firewall-cmd --zone=public --query-port=2333/tcp
#重载防火墙规则
[root@insta...sq4 ~]# firewall-cmd --reload
#查看防火墙已放行的端口
[root@insta...sq4 ~]# firewall-cmd --zone=public --list-ports
```

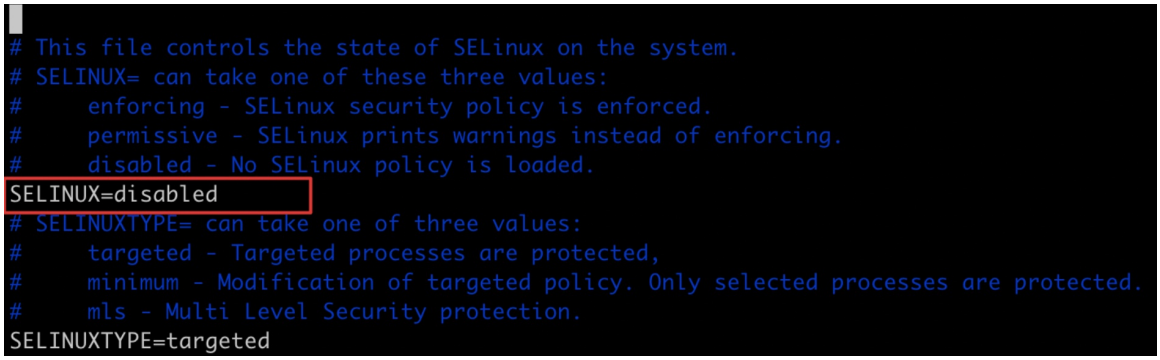
步骤五 重启ssh服务，查看ssh服务状态和端口占用情况

```
 #重启ssh服务
[root@insta...sq4 ~]# systemctl restart sshd
#查看ssh占用的端口
[root@insta...sq4 ~]# netstat -ntlp  | grep ssh
tcp        0      0 0.0.0.0:2333            0.0.0.0:*               LISTEN      31881/sshd
tcp6       0      0 :::2333                 :::*                    LISTEN      31881/sshd
```

步骤六 使用自定义端口连接

```
 ssh -p 2333 root@ip
```

如果使用新端口无法远程连接服务器的话，可以通过VNC登录到服务器，检查/etc/sysconfig/selinux配置文件，将SELINUX参数改为disabled：SELINUX=disabled，修改后保存文件，然后再次尝试。



Centos7安装docker

本文介绍如何在Linux操作系统上的安装docker，以CentOS 7为例，其他Linux系统的发行版本，如Ubuntu，Debian，Fedora等，可以自行查询Docker的官方文档

1. 删除旧的Docker版本

可能有些Linux已经预先安装了Docker，但一般版本比较旧，所以可以先执行以下代码来删除旧版本的Docker。

```
 $ yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-selinux docker-engine-selinux docker-engine
```

2. 指定安装源地址

可以使用官方源地址或者国内源地址，这里给出官方源地址配置命令。

```
$ yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

3. 使用yum安装Docker

```
$ yum install docker-ce docker-ce-cli containerd.io
```

4. 启动Docker服务器

```
# 启动docker守护进程
$ systemctl start docker
```

5. 测试安装是否成功

通过上面几种方式安装了Docker之后，我们可以通过下面的方法来检测安装是否成功。

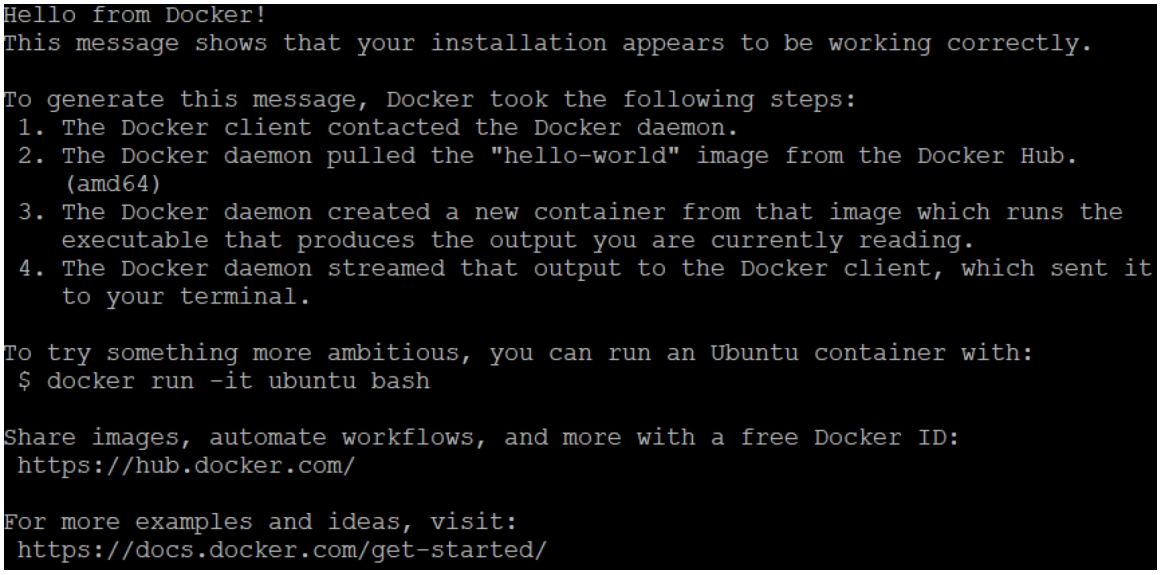
- 打印Docker版本

```
##### 打印Docker版本
$ docker version
```

- 拉取镜像并运行容器

```
##### 拉取hello-world镜像
$ docker pull hello-world
##### 使用hello-world运行一个容器
$ docker run hello-world
```

运行上面的命令之后，如果有如下图所示的输出结果，则说明安装已经成功了。



Linux系统安装图形化界面并远程连接

CentOS

CentOS6.5安装KDE桌面图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像Centos6.5环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对 BCC 实例**创建手工快照**备份，以备发生异常时能回滚恢复。

本文分为三步骤：

步骤一：安装KDE桌面环境。

步骤二：安装vncserver服务。

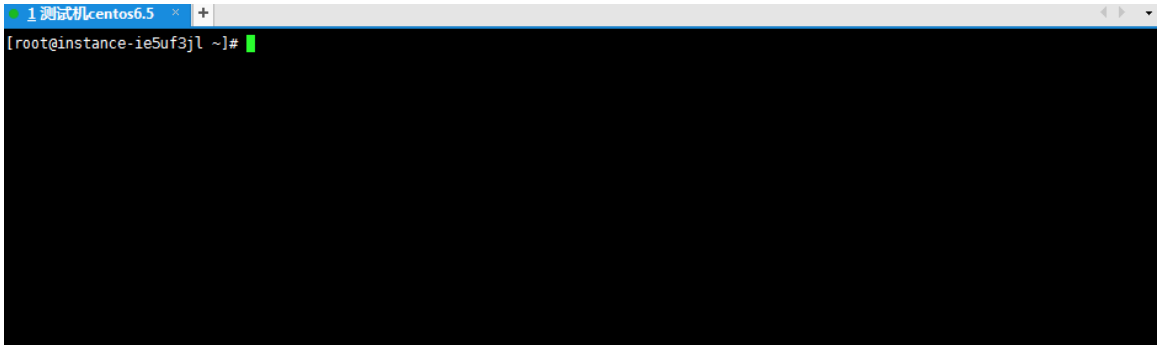
步骤三：使用viewer客户端连接软件连接该实例。

前提条件

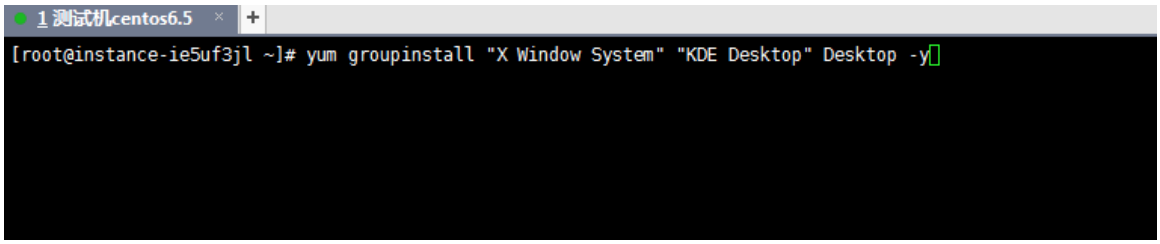
- 安全组和防火墙需要放行5901端口（vncserver默认第一个窗口的端口）。
- 下载VNC viewer客户端连接软件。

安装KDE桌面环境

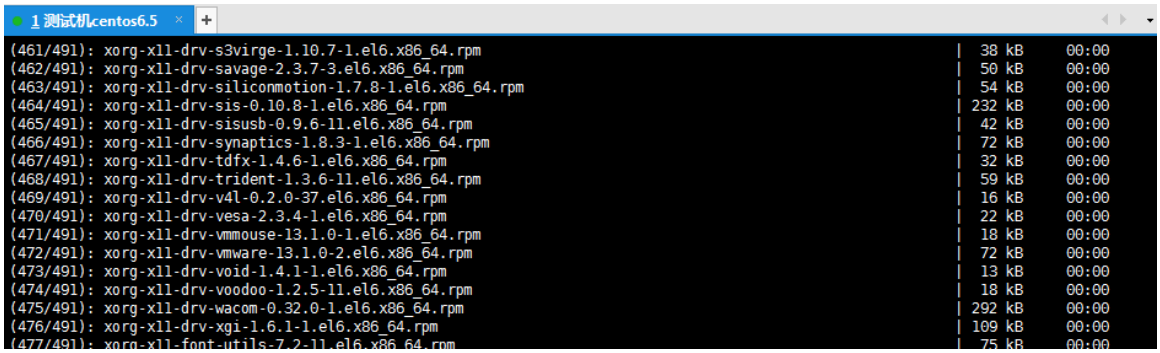
1.首先远程连接登录服务器。



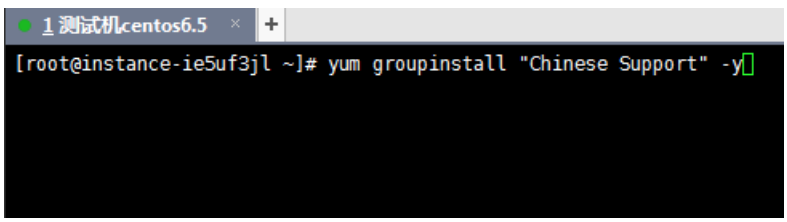
2.执行命令：yum groupinstall "X Window System" "KDE Desktop" Desktop -y 说明：该指令同时安装了 3 个软件包。注意，因为 KDE Desktop 和 X Window System 两个软件包名称中间都包含空格，需要用引号引起来才行。



执行后等待安装完毕。



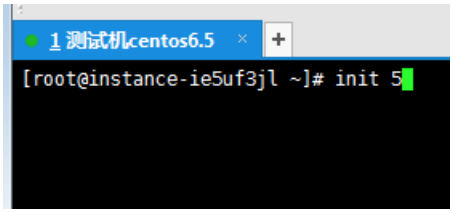
3.执行命令：yum groupinstall "Chinese Support" 说明：该指令为安装中文插件，使该桌面支持中文（不使用中文可以不安装）。



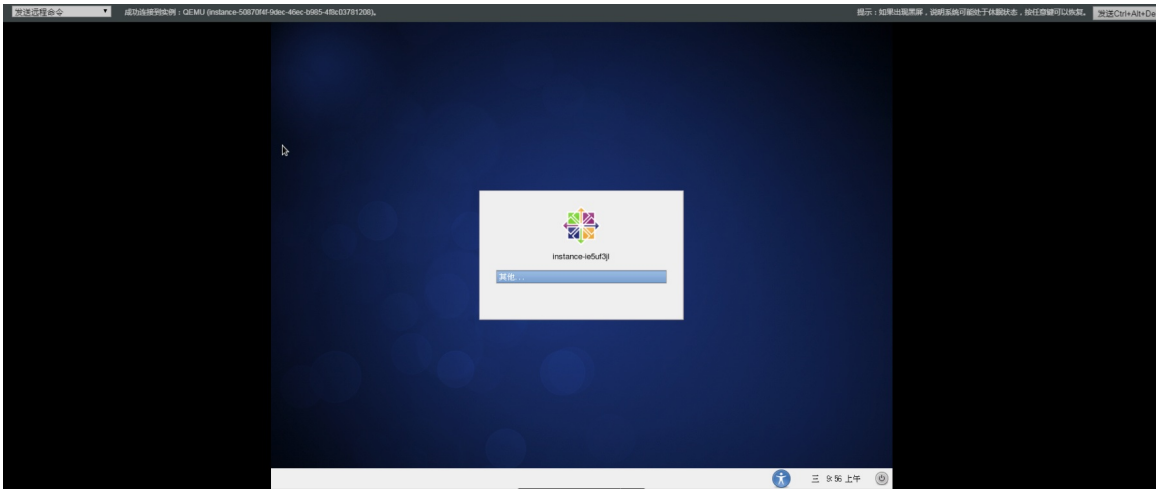
4.将语言，LANG修改为LANG=”zh_CN.UTF-8” 执行命令：sed -i 's/LANG=.*/LANG="zh_CN.UTF-8"/' /etc/sysconfig/i18n （此操作会使所有界面和命令字符均为中文显示，如不需要可以不执行该步骤）。

5.重启服务器使其生效。执行命令：reboot

6.到此为止，图形界面安装成功，待重启后重新连接执行命令:init 5 启动图形界面



注：通过VNC可以看到图形化桌面已经安装完毕。

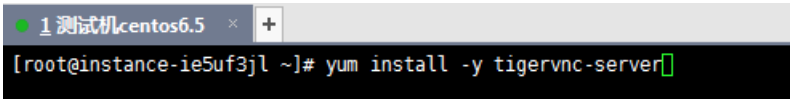


安装vncserver服务

实现本地远程访问图形化桌面。

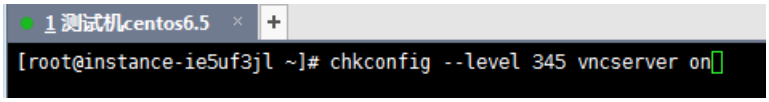
1.安装vncserver。

命令：`yum install -y tigervnc-server`



2.配置vncserver。

(1)配置为开机自启动。 命令：`chkconfig --level 345 vncserver on`

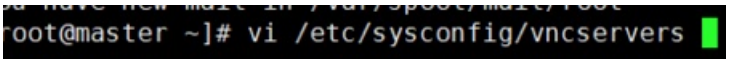


(2)配置vnc密码（初次启动的时候需要输入两次密码）。

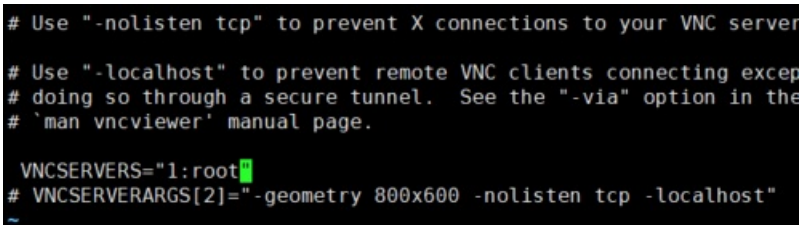
命令：

```
vncserver

You will require a password to access your desktop.
Password:（输入并记住该密码）
Verify:（重复输入并记住该密码）
```

3.编辑配置文件 `vi /etc/sysconfig/vncservers`



增加内容`VNCSERVERS="1:root"`。退出并保存。



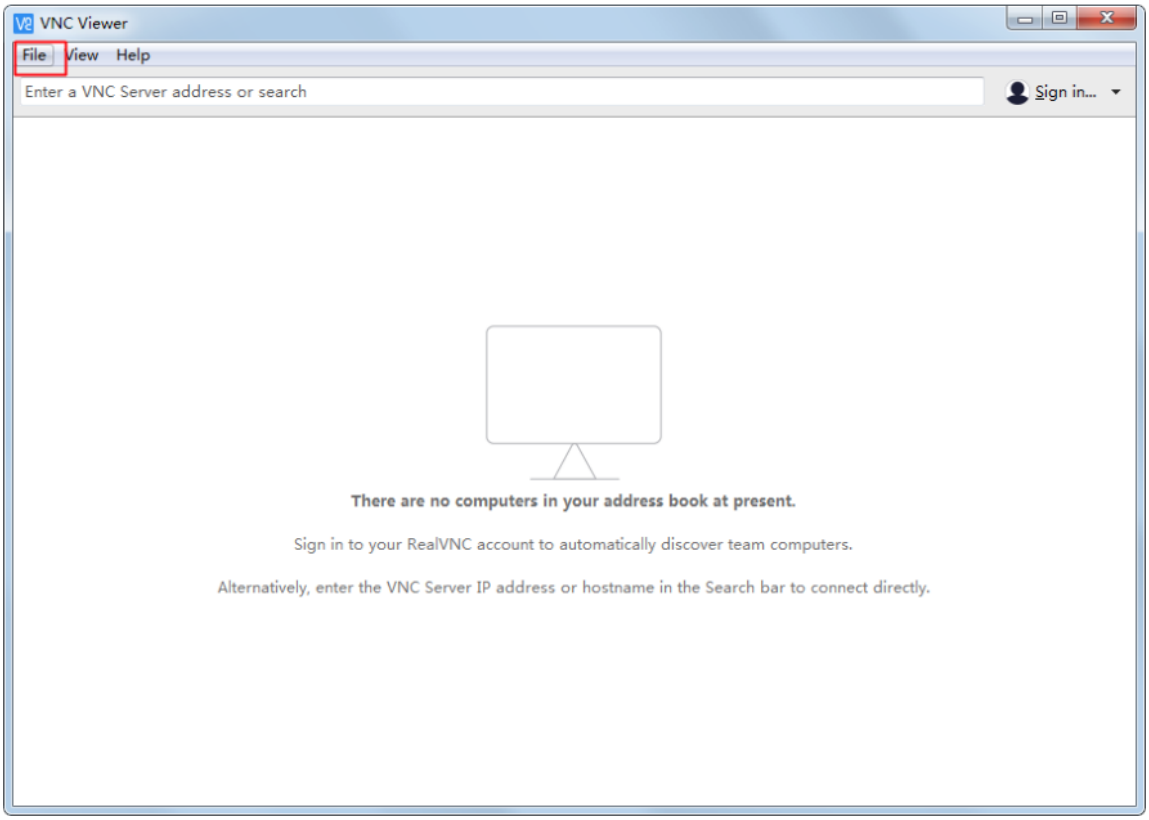
重启vnc服务。 命令：`service vncserver restart`

```
[root@master ~]# service vncserver restart
Shutting down VNC server: 1:root                           [  OK  ]
Starting VNC server: 1:root
New 'master:1 (root)' desktop is master:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/master:1.log

                                                           [  OK  ]
[root@master ~]#
```

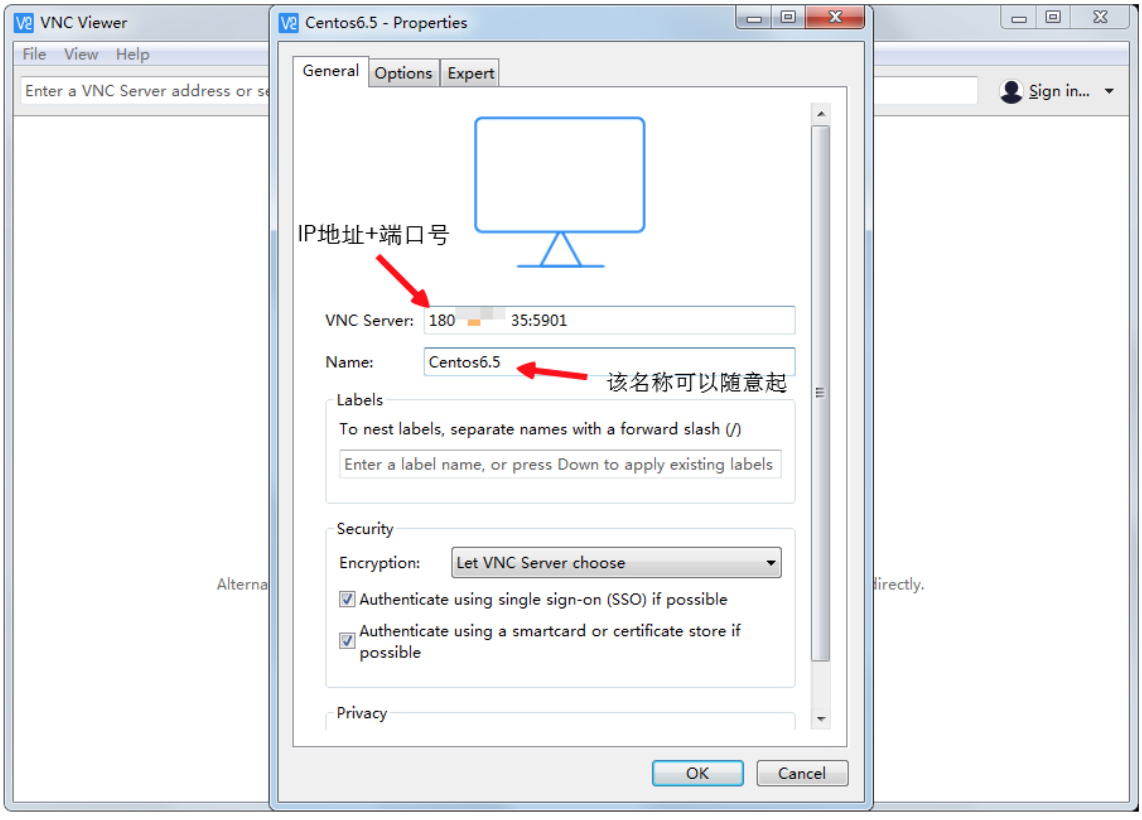
使用viewer客户端连接软件连接该实例

1.使用前提条件中提供的viewer客户端连接软件连接该实例：

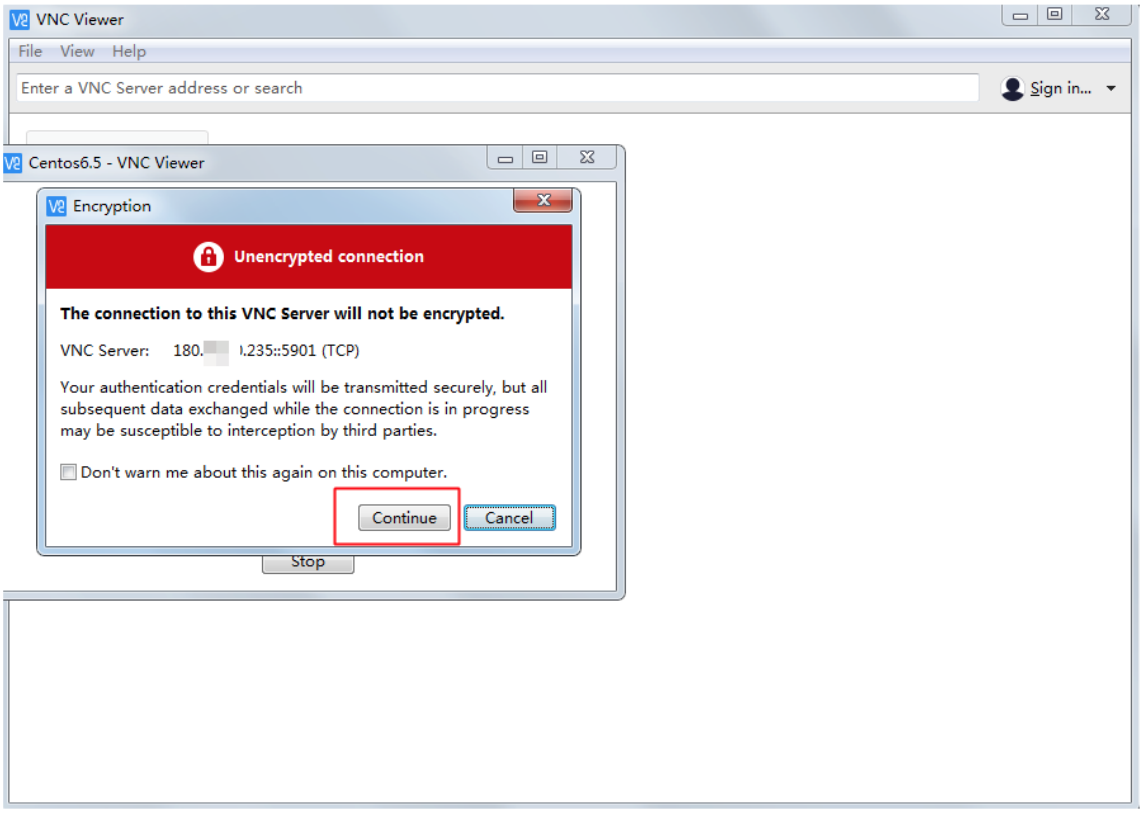
点击file-New connection..创建新连接。



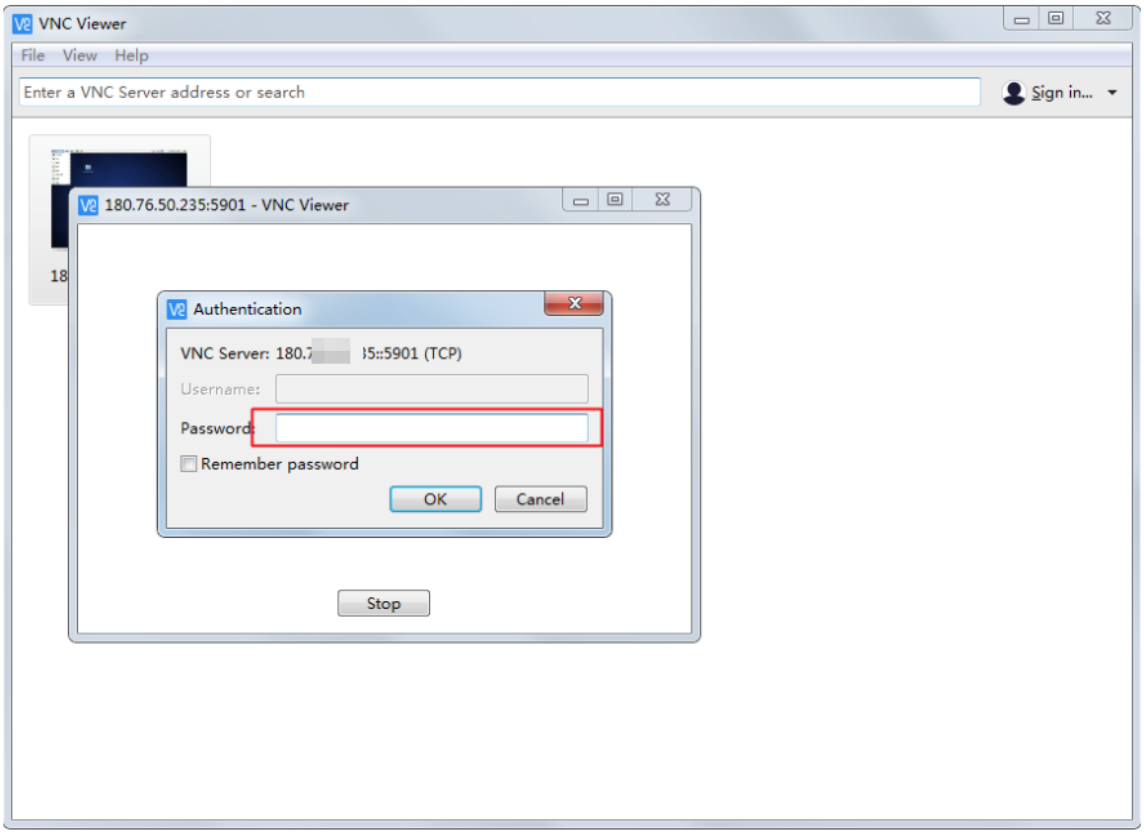
2.输入对应的信息。



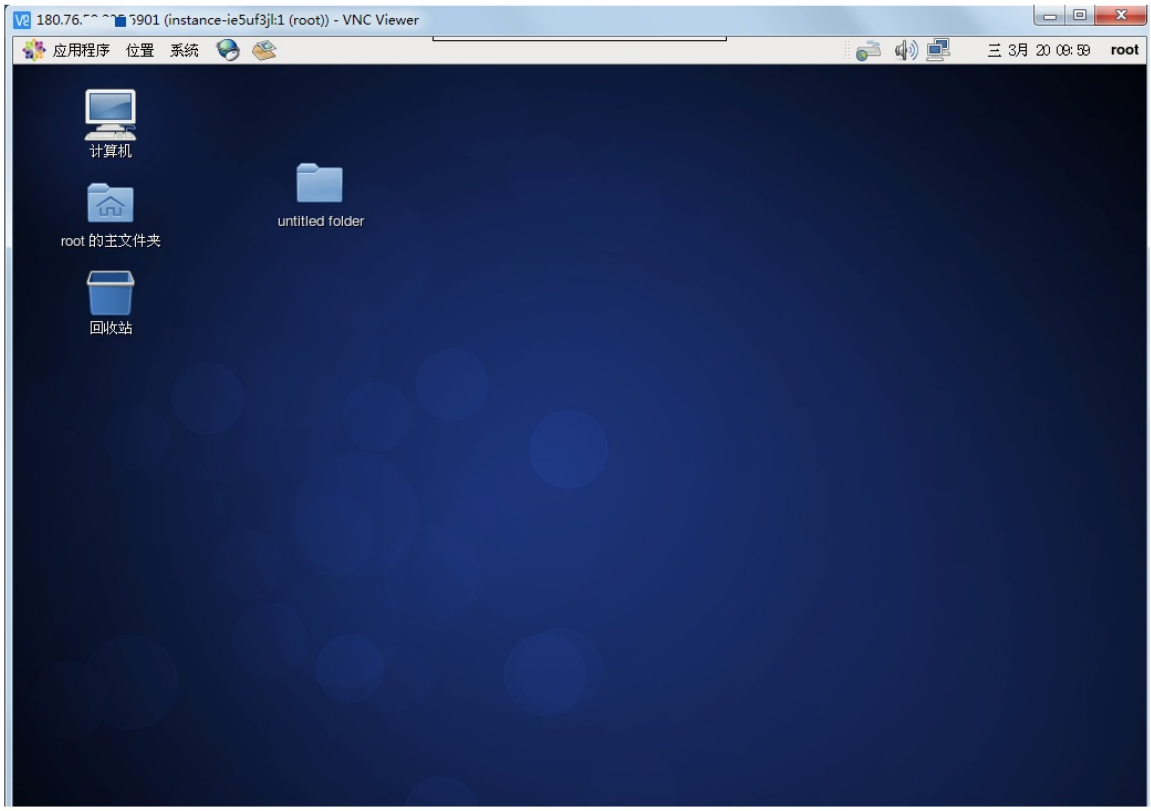
3.双击之前创建的连接，点击进行连接。



4.此处输入前文第(2)步时设置的密码。



5.连接成功。



CentOS7.2安装gnome图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像Centos7.2环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对 BCC 实例**创建手工快照**备份，以备发生异常时能回滚恢复。

步骤一：给BCC安装gnome桌面环境。

步骤二：安装VNC Server实现本地远程图形化访问。

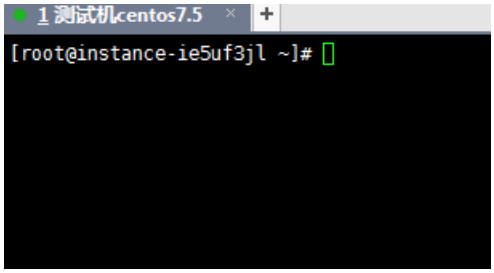
步骤三：使用viewer客户端连接软件连接该实例。

前提条件

- 安全组和防火墙需要放行5901端口（vncserver端口）。
- 下载VNC viewer客户端连接软件 。

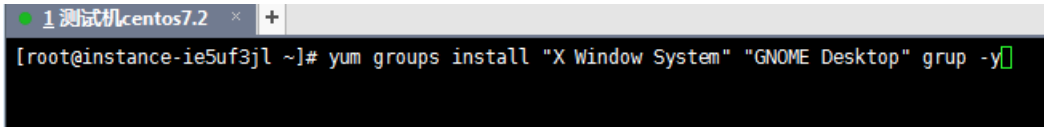
给BCC安装gnome桌面环境

1.首先远程连接登录服务器。

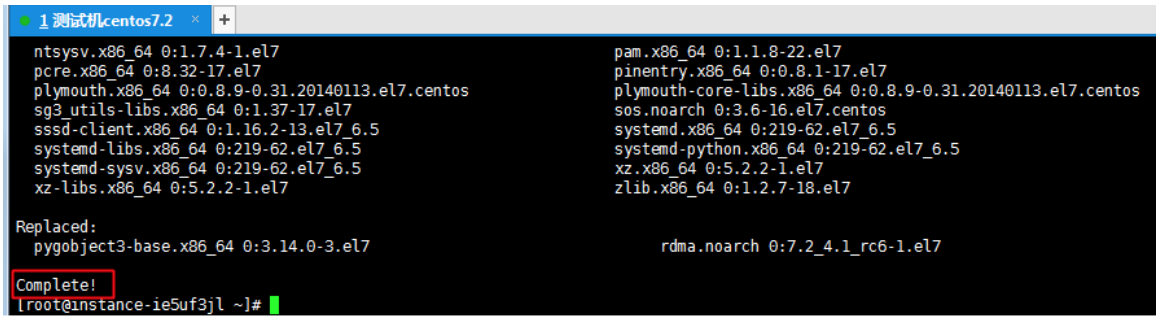


2.执行命令：`yum groups install "X Window System" "GNOME Desktop" -y`

安装X Window System 环境 和GNOME桌面，时间较长，请耐心等待。



3.执行后等待安装完毕，以下界面表示安装完成：

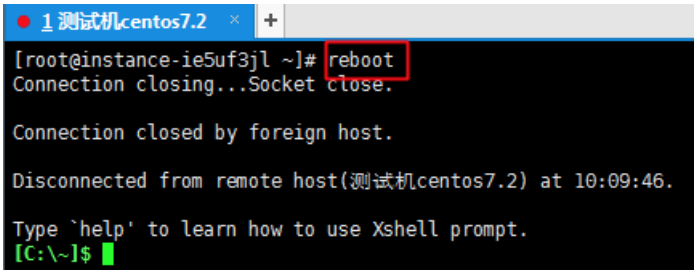


4.执行命令systemctl set-default graphical.target设置默认进入图形界面。 后期可根据需要更改默认启动方式：

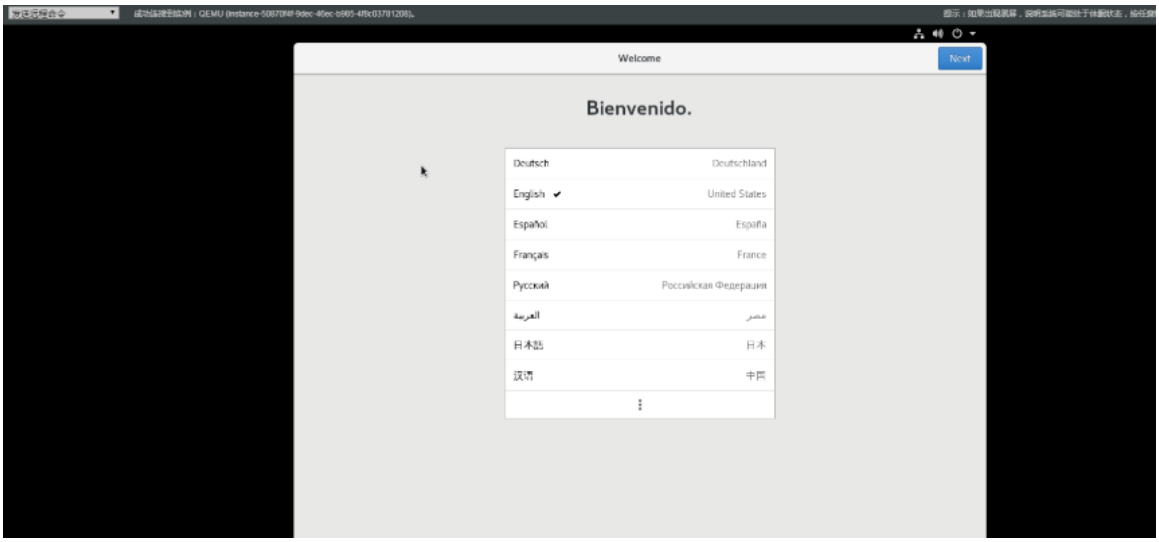
```
systemctl set-default multi-user.target //设置成命令模式

systemctl set-default graphical.target //设置成图形模式
```

```
[root@instance-ie5uf3jl ~]# systemctl set-default graphical.target
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to /usr/lib/systemd/system/graphical.target.
```

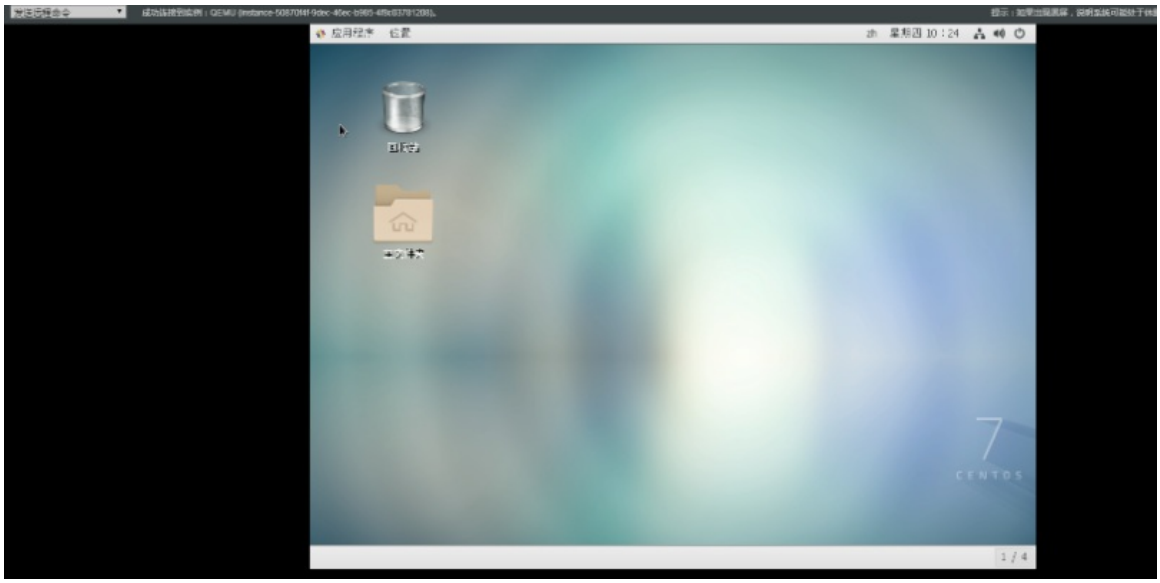
5.执行命令：reboot 重启服务器



6.此时通过VNC查看已经进入图形化界面。



7.经过设置后进入桌面。

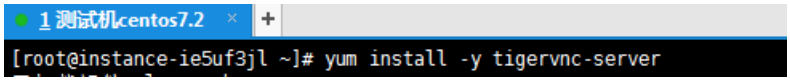


到此为止，桌面安装完成。

安装VNC Server实现本地远程图形化访问

1.安装vncserver。

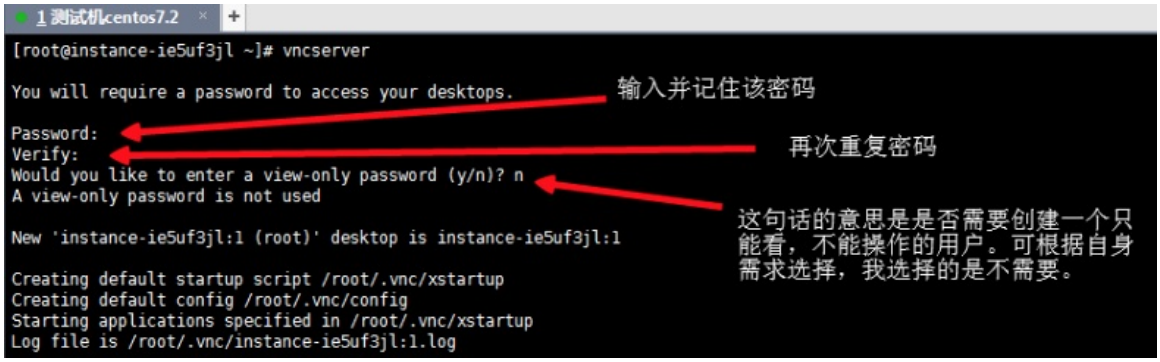
命令：`yum install -y tigervnc-server`



2.配置vncserver。

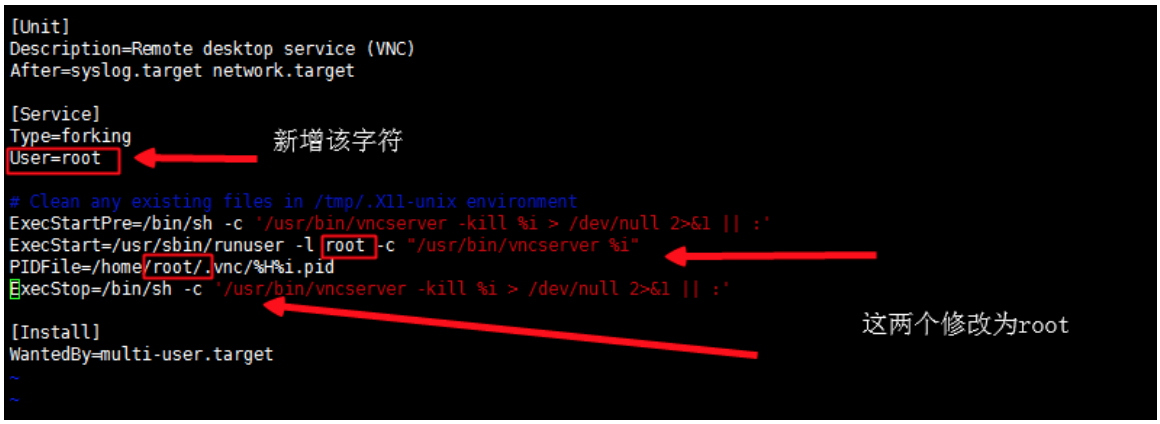
(1)配置vnc密码（初次启动的时候需要输入两次密码）。 命令：

```
 vncserver
You will require a password to access your desktop.

Password:（输入并记住该密码）

Verify:（重复输入并记住该密码）
```



(2)生成并编辑配置文件。

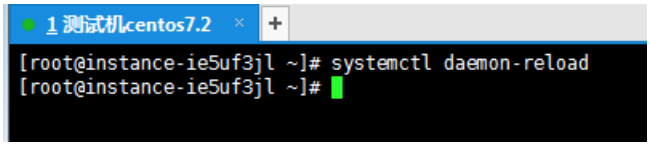
执行：`cp -p /lib/systemd/system/vncserver@.service /lib/systemd/system/vncserver@:1.service` 再执行：`vim /lib/systemd/system/vncserver@:1.service`



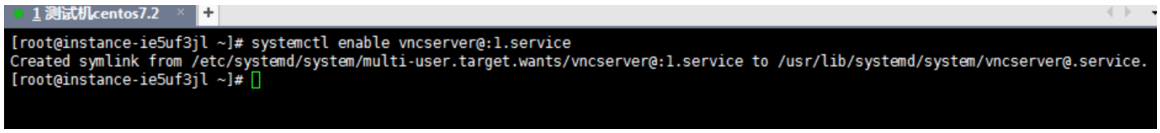
(3)按Esc键退出编辑模式，再输入:wq保存并退出。

3.运行以下命令重启systemd。

命令：`systemctl daemon-reload`



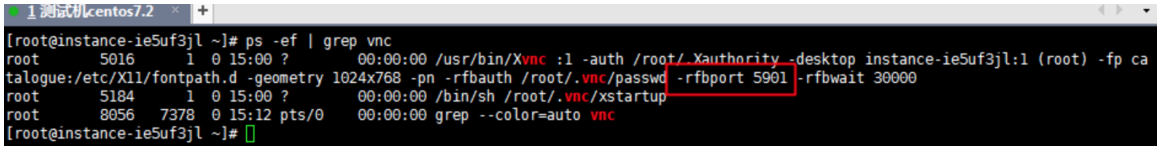
4.配置为开机自启动。 命令：`systemctl enable vncserver@:1.service`



5.运行以下命令启动VNC Server。

命令：`systemctl start vncserver@:1.service`

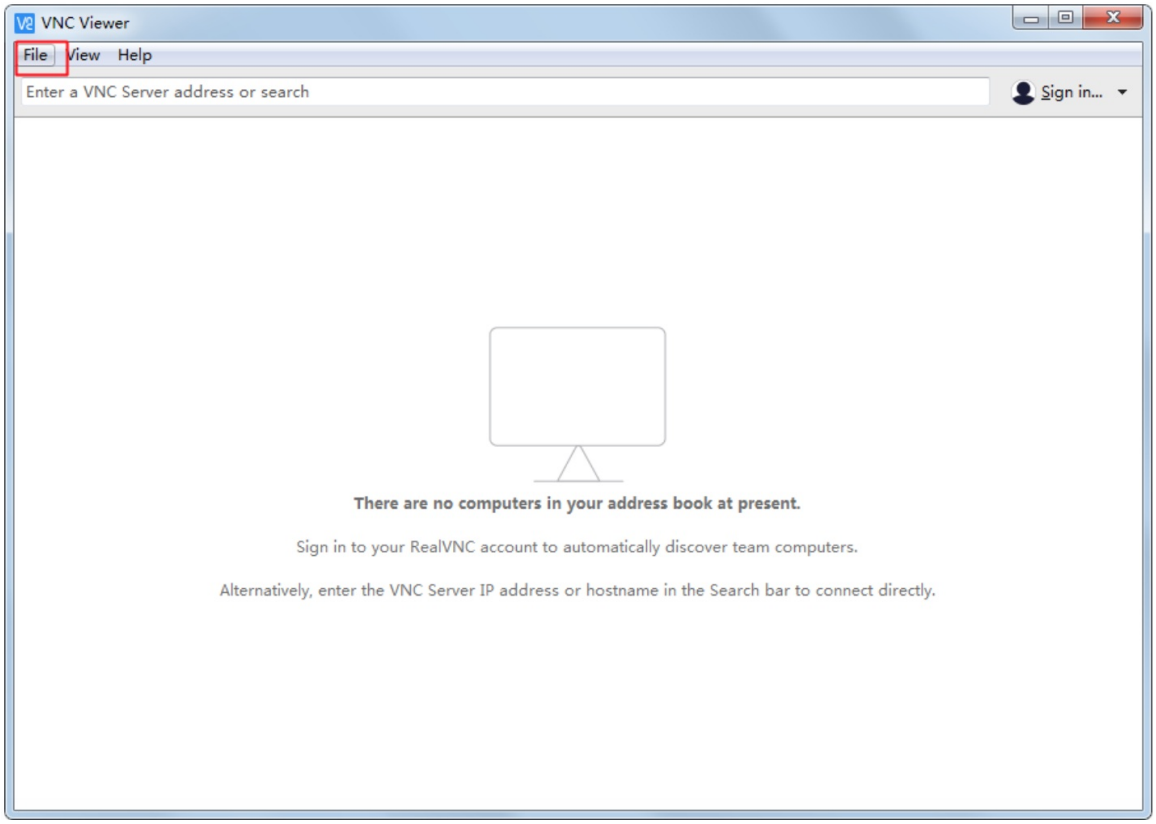
6.运行命令 ps -ef | grep vnc 确认服务是否已经启动。如果返回以下类似结果，说明服务已经启动。



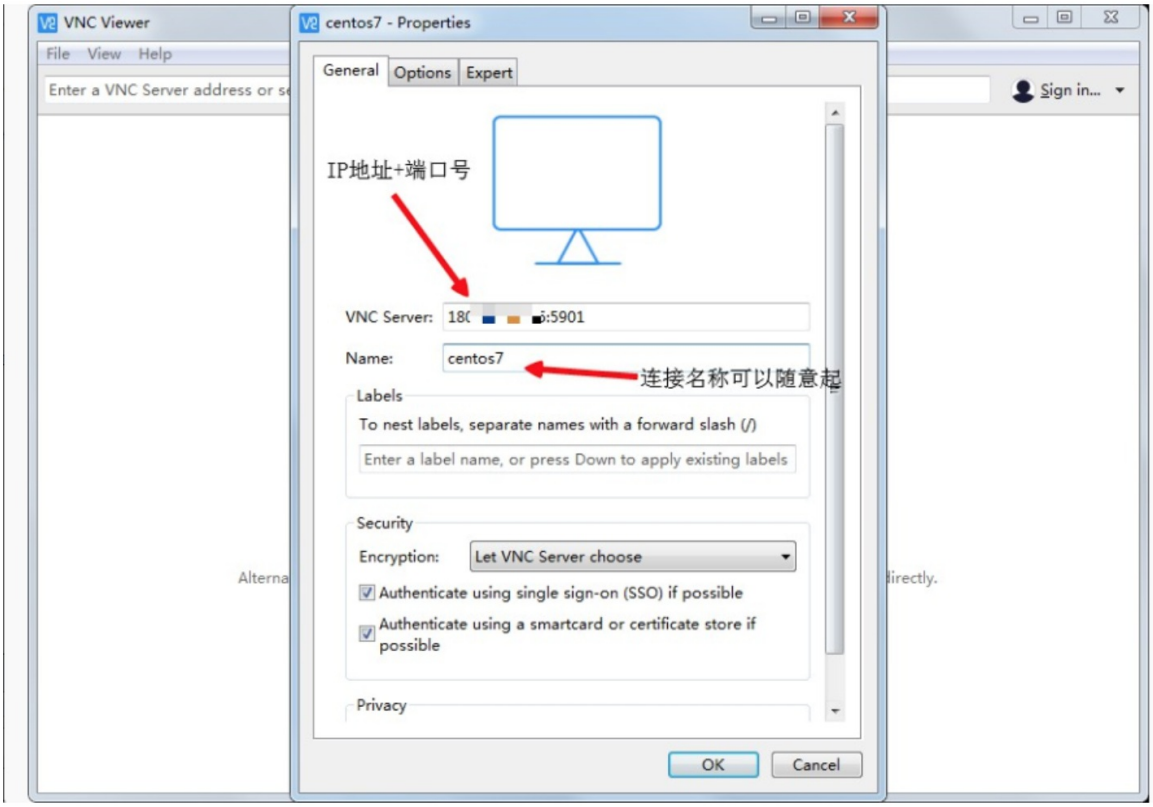
使用viewer客户端连接软件连接该实例

1.使用前提条件中提供的viewer客户端连接软件连接该实例：

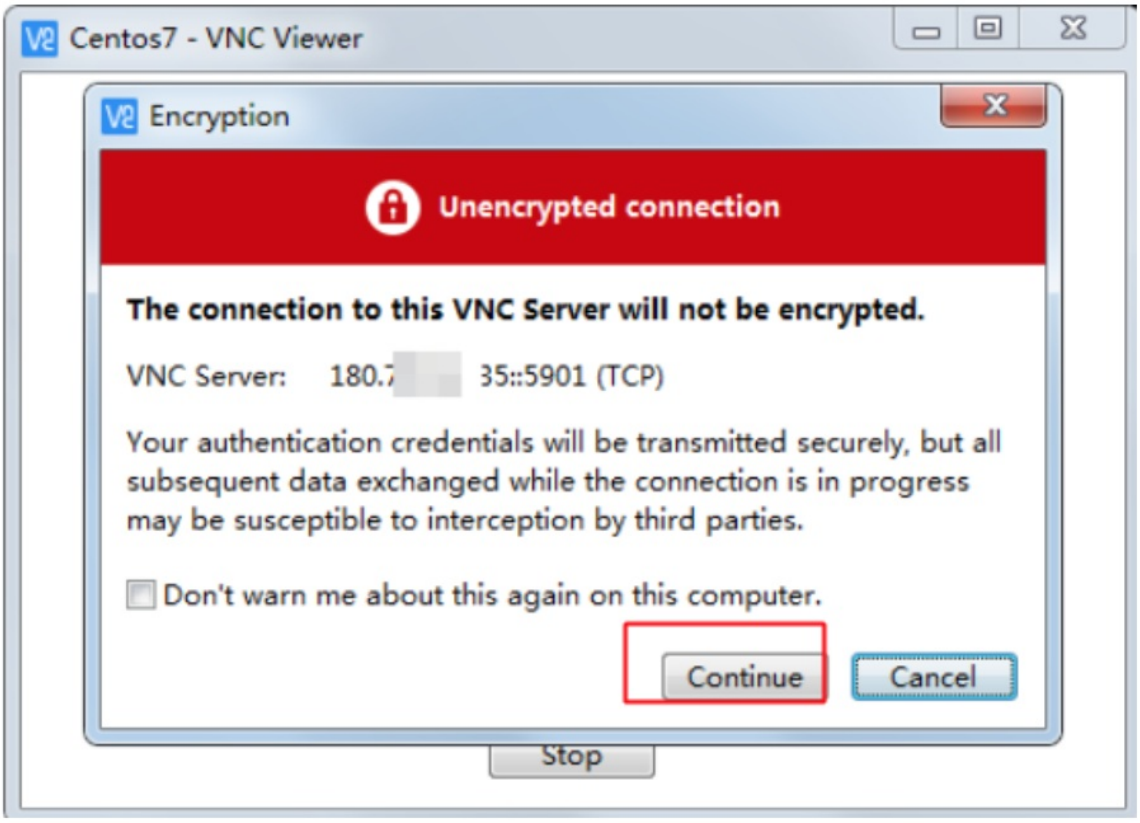
点击file------New connection..创建新连接。



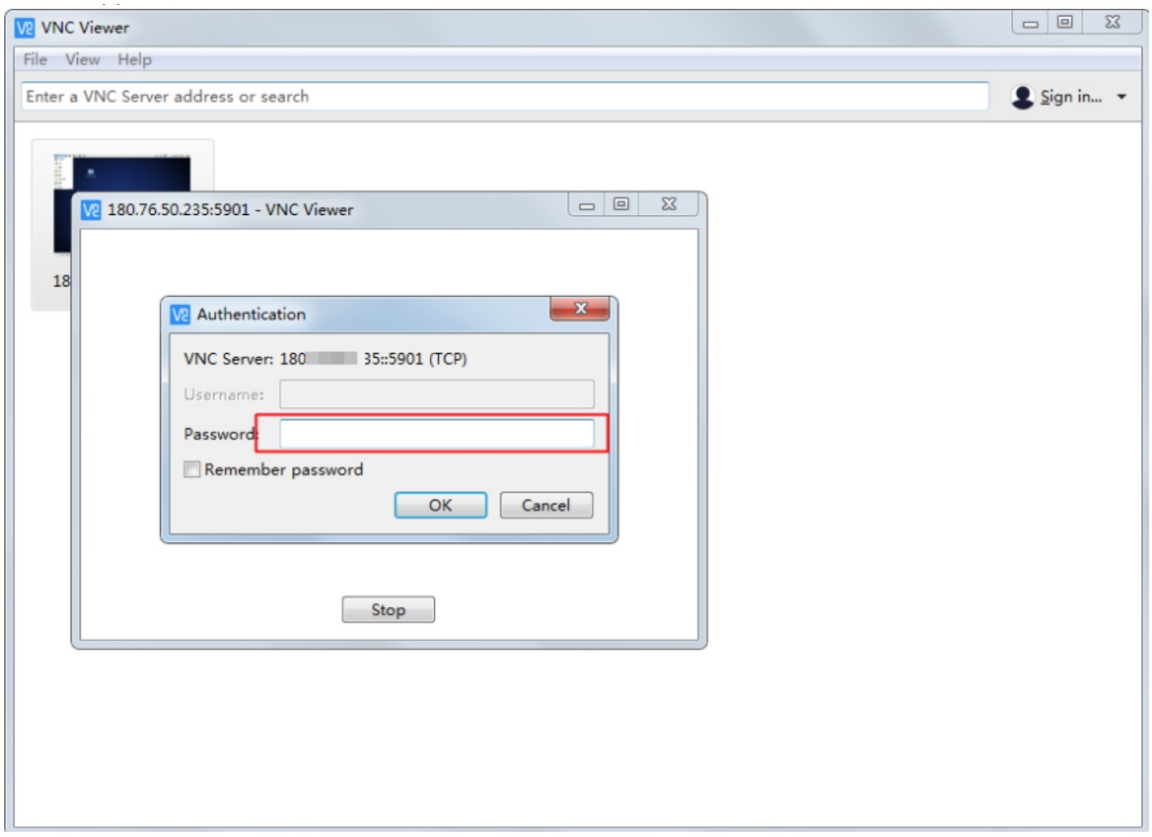
2.输入对应的信息。



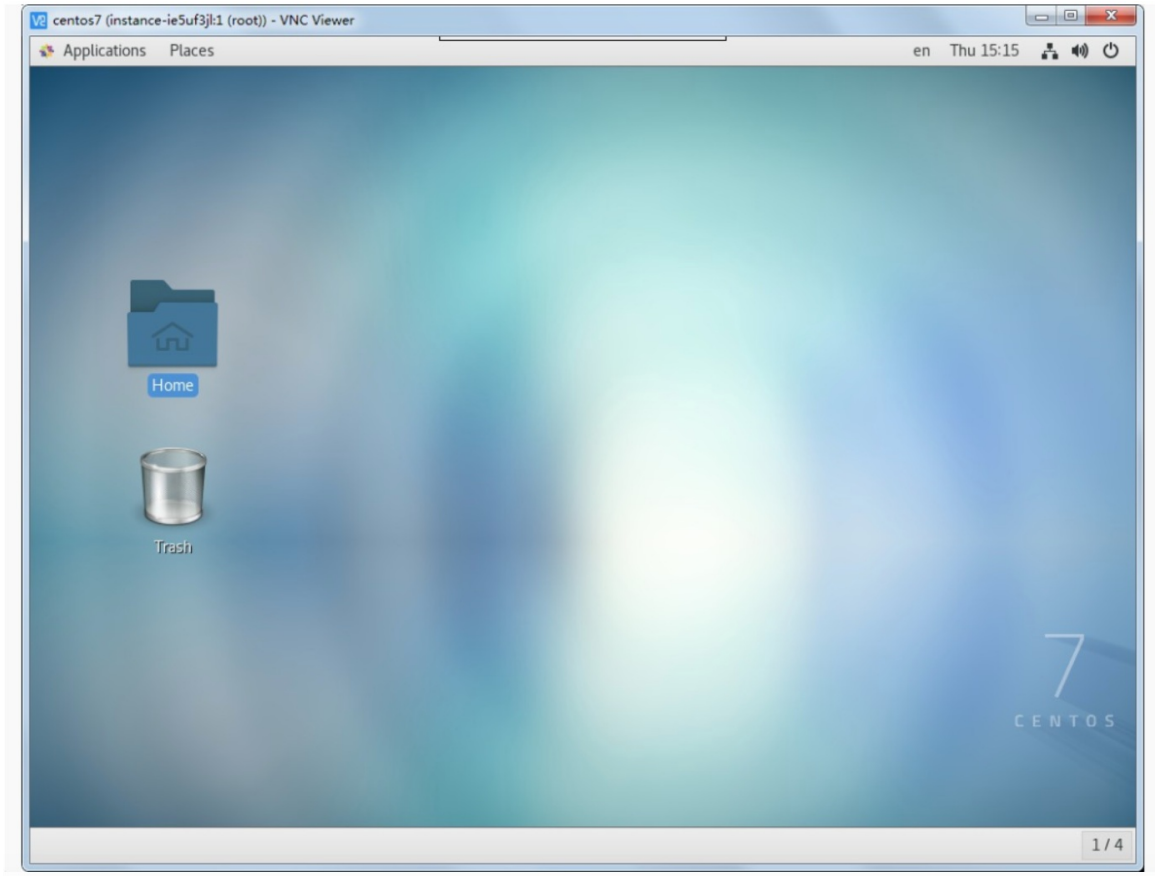
3.双击之前创建的连接，点击进行连接。



4.此处输入第(1)步骤时设置的密码。



连接成功。



CentOS8安装图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像CentOS8环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对BCC实例手动创建快照备份，以备发生异常时能回滚恢复。

步骤一：BCC安装桌面环境。
步骤二：安装VNCserver服务。
步骤三：使用viewer客户端连接软件连接该实例。

前提条件

1. 安全组和防火墙需要放行5901端口（vncserver端口）。
2. 下载 VNC viewer客户端连接软件。

BCC安装桌面环境

1. 远程或者VNC连接服务器。
2. 安装图形桌面的软件包。

   `yum groupinstall "Server with GUI" -y`

```
  usermode-1.113-1.el8.x86_64
  userspace-rcu-0.10.1-2.el8.x86_64
  wpa_supplicant-1:2.7-1.el8.x86_64
  xml-common-0.6.3-50.el8.noarch
  xmlrpc-c-1.51.0-5.el8.x86_64
  xmlrpc-c-client-1.51.0-5.el8.x86_64
  imlib2-1.4.9-8.el8.x86_64
  openbox-3.6.1-14.el8.x86_64
  openbox-libs-3.6.1-14.el8.x86_64

Complete!
[root@instance-w9dn8qnk ~]#
```

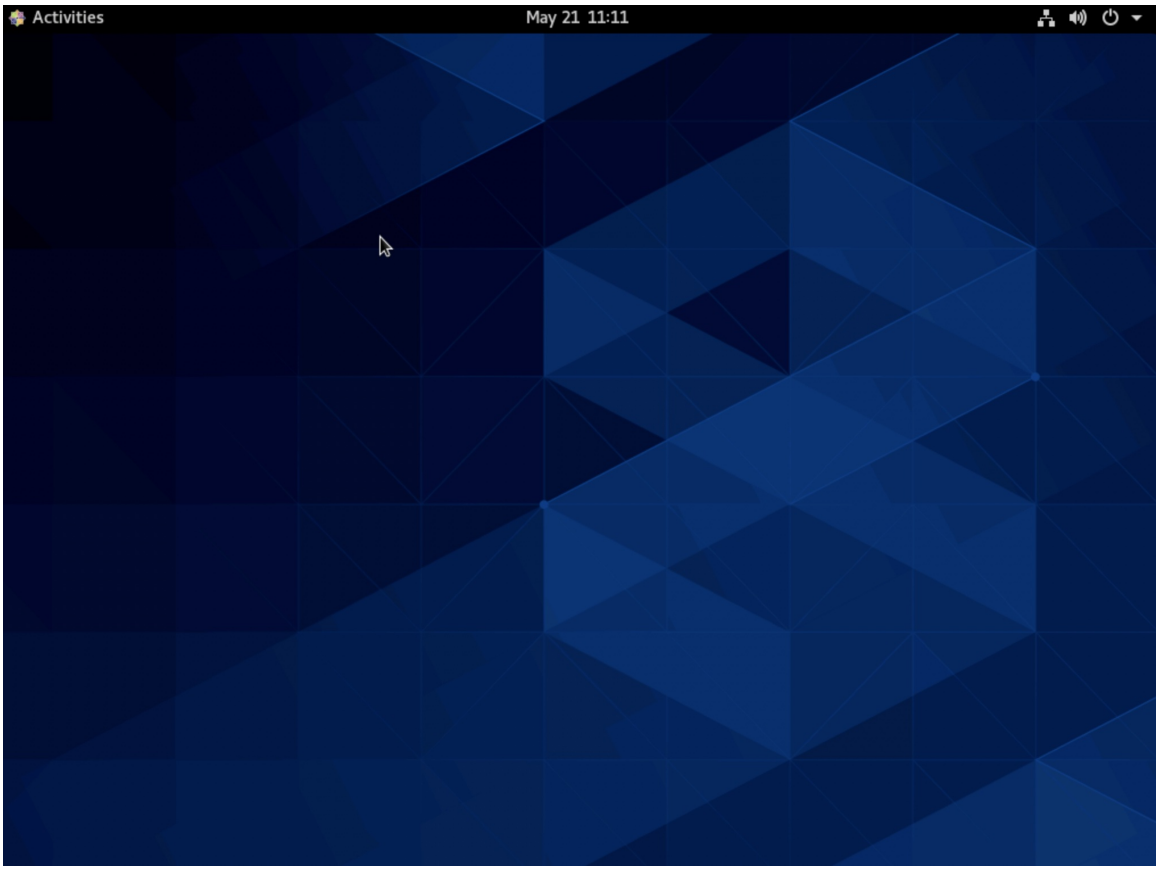
3. 设置图形模式为默认模式启动。

   `systemctl set-default graphical`

```
[root@instance-w9dn8qnk ~]# systemctl set-default graphical
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
```

4. 重启后，即可登录图形化界面。

   `reboot`



安装VNCserver服务

1. 安装软件包。

   `yum install tigervnc-server tigervnc-server-module -y`

```
Running transaction
  Preparing        :                                                        1/1
  Installing       : tigervnc-server-module-1.9.0-14.el8_1.x86_64           1/2
  Installing       : tigervnc-server-1.9.0-14.el8_1.x86_64                  2/2
  Running scriptlet: tigervnc-server-1.9.0-14.el8_1.x86_64                  2/2
  Verifying        : tigervnc-server-1.9.0-14.el8_1.x86_64                  1/2
  Verifying        : tigervnc-server-module-1.9.0-14.el8_1.x86_64           2/2

Installed:
  tigervnc-server-1.9.0-14.el8_1.x86_64
  tigervnc-server-module-1.9.0-14.el8_1.x86_64

Complete!
```

2. 配置VNC密码。

`Vncpasswd`

```
[root@instance-w9dn8qnk ~]# vncpasswd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
A view-only password is not used
```

> 注释：
>
> Password:第一次输入密码。
>
> Verify:确认密码。
>
> Would you like to enter a view-only password (y/n)?：是否创建一个只读用户。

3. 运行 `vncserver` 命令启动VNC服务端。

```
[root@instance-w9dn8qnk ~]# vncserver

Warning: instance-w9dn8qnk:1 is taken because of /tmp/.X11-unix/X1
Remove this file if there is no X server instance-w9dn8qnk:1

New 'instance-w9dn8qnk:2 (root)' desktop is instance-w9dn8qnk:2

Creating default startup script /root/.vnc/xstartup
Creating default config /root/.vnc/config
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/instance-w9dn8qnk:2.log
```

4. 使用 `ps -ef | grep vnc` 查看是否启动并查看使用端口号。

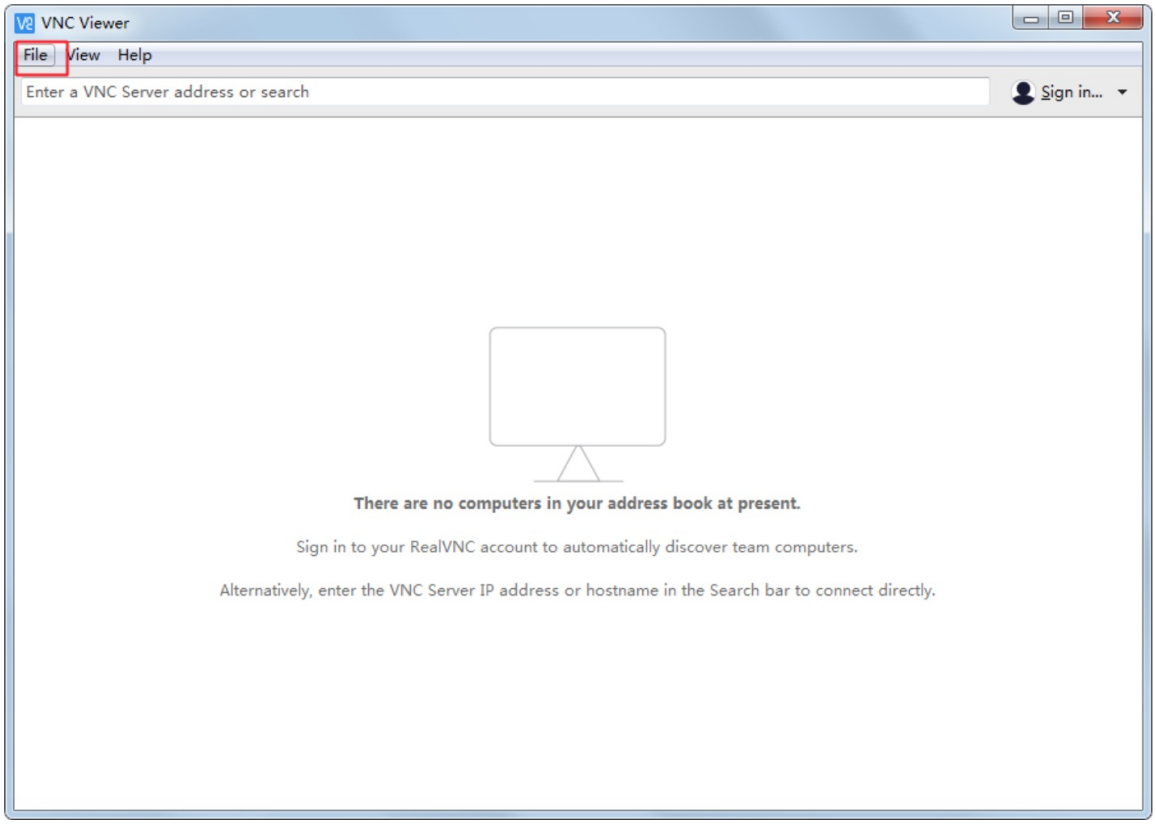
```
[root@instance-w9dn8qnk ~]# ps -ef | grep vnc
root      3553  2038  0 11:18 pts/0    00:00:00 /usr/bin/Xvnc :2 -auth /run/user
/0/gdm/Xauthority -desktop instance-w9dn8qnk:2 (root) -fp catalogue:/etc/X11/fon
tpath.d -geometry 1024x768 -pn -rfbauth /root/.vnc/passwd -rfbport 5902 -rfbwait
 30000
root      3562  2038  0 11:18 pts/0    00:00:00 /bin/sh /root/.vnc/xstartup
root      4075  3026  0 11:19 pts/0    00:00:00 grep --color=auto vnc
```

可以看到服务已正常启动，并且使用的是5902端口。

使用viewer客户端连接软件连接该实例

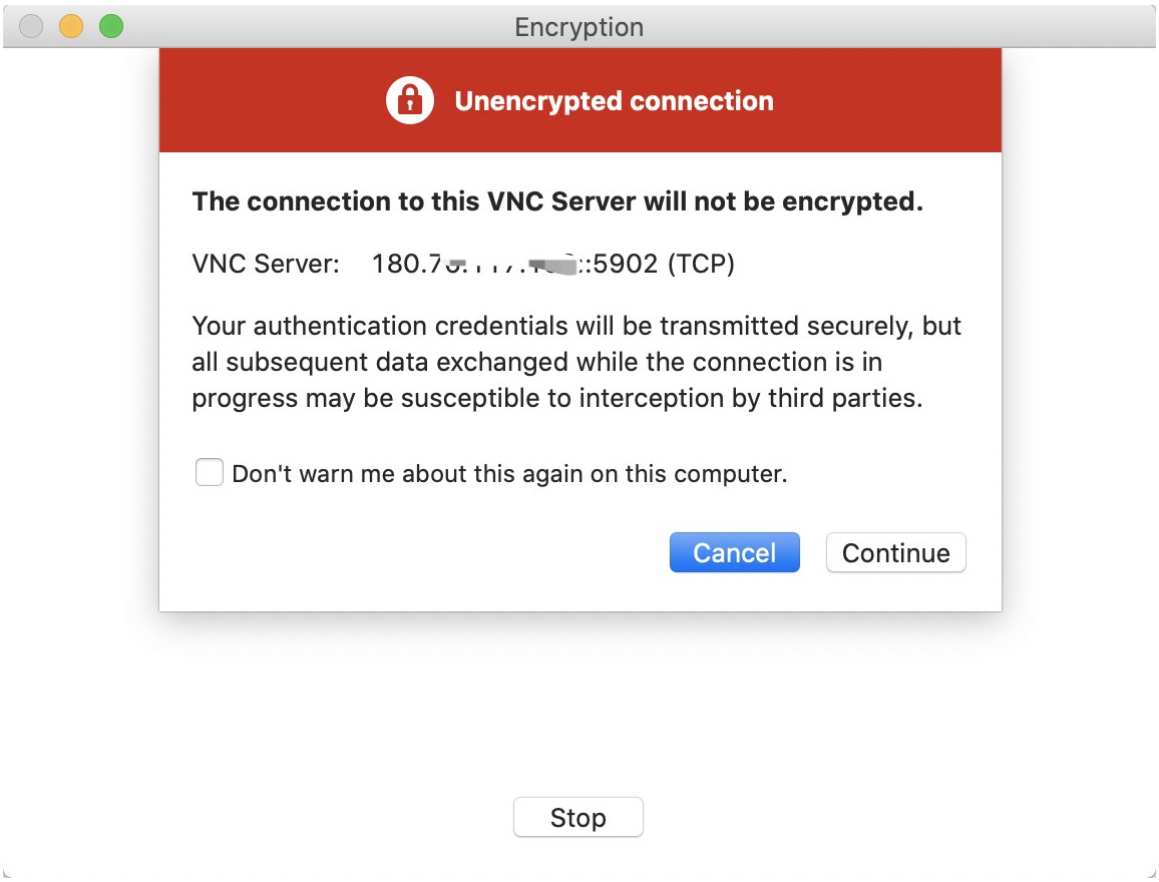
1. 使用前提条件中提供的viewer客户端连接软件连接该实例：

点击file------New connection..创建新连接。

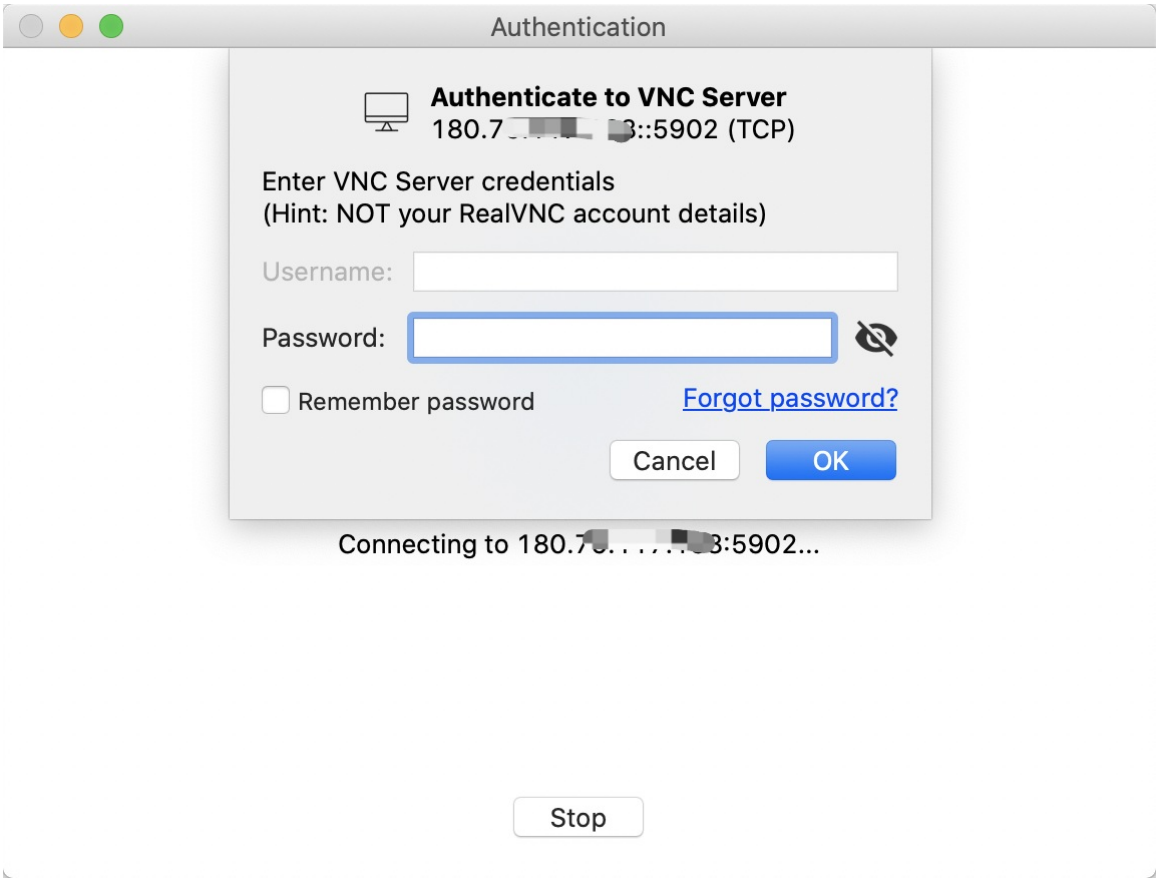

VNC Viewer
File
View
Help
Enter a VNC Server address or search
Sign in...
There are no computers in your address book at present.
Sign in to your RealVNC account to automatically discover team computers.
Alternatively, enter the VNC Server IP address or hostname in the Search bar to connect directly.

2. 输入对应的信息。

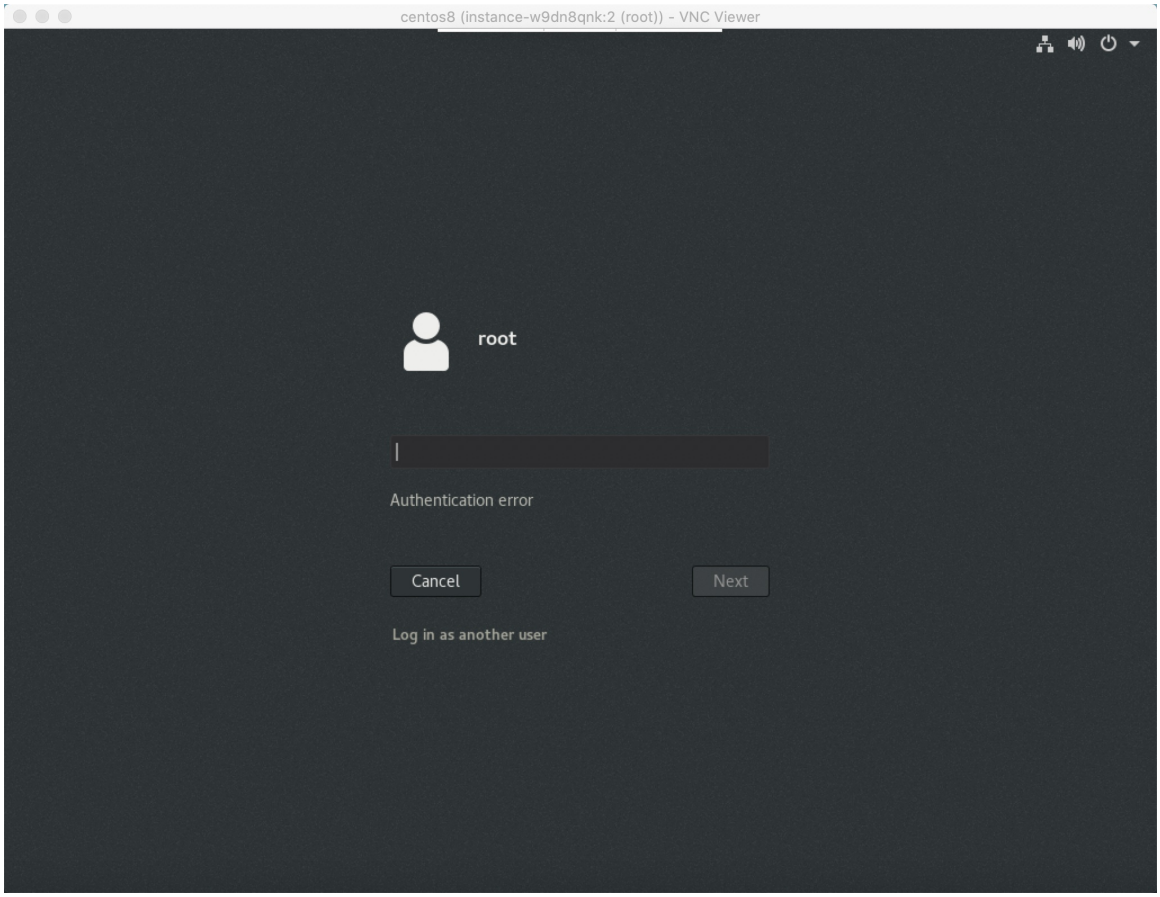
3. 点击"Continue"。



4. 此处输入 安装VNCserver服务 时第2步骤时设置的密码。



5. 连接成功。



Ubuntu

Ubuntu14.04安装图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像Ubuntu14.04环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对BCC实例 创建手工快照备份，以备发生异常时能回滚恢复。

步骤一：BCC安装gnome桌面环境。

步骤二：安装VNCserver服务。

步骤三：使用viewer客户端连接软件连接该实例。

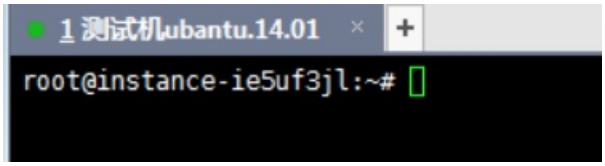
前提条件

1.安全组和防火墙需要放行5901端口（vncserver端口）。

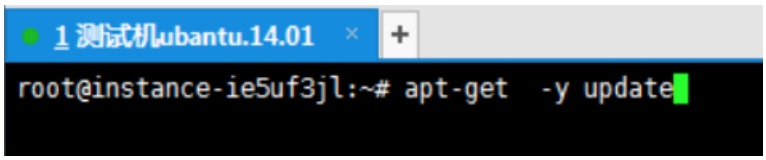
2.下载VNC viewer客户端连接软件。

BCC安装gnome桌面环境

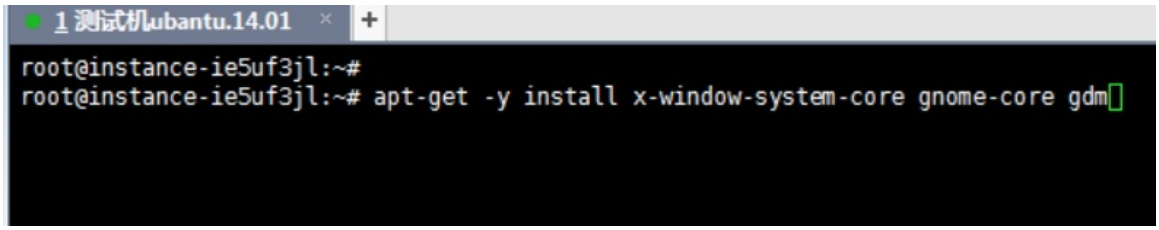
1.首先远程连接登录服务器。



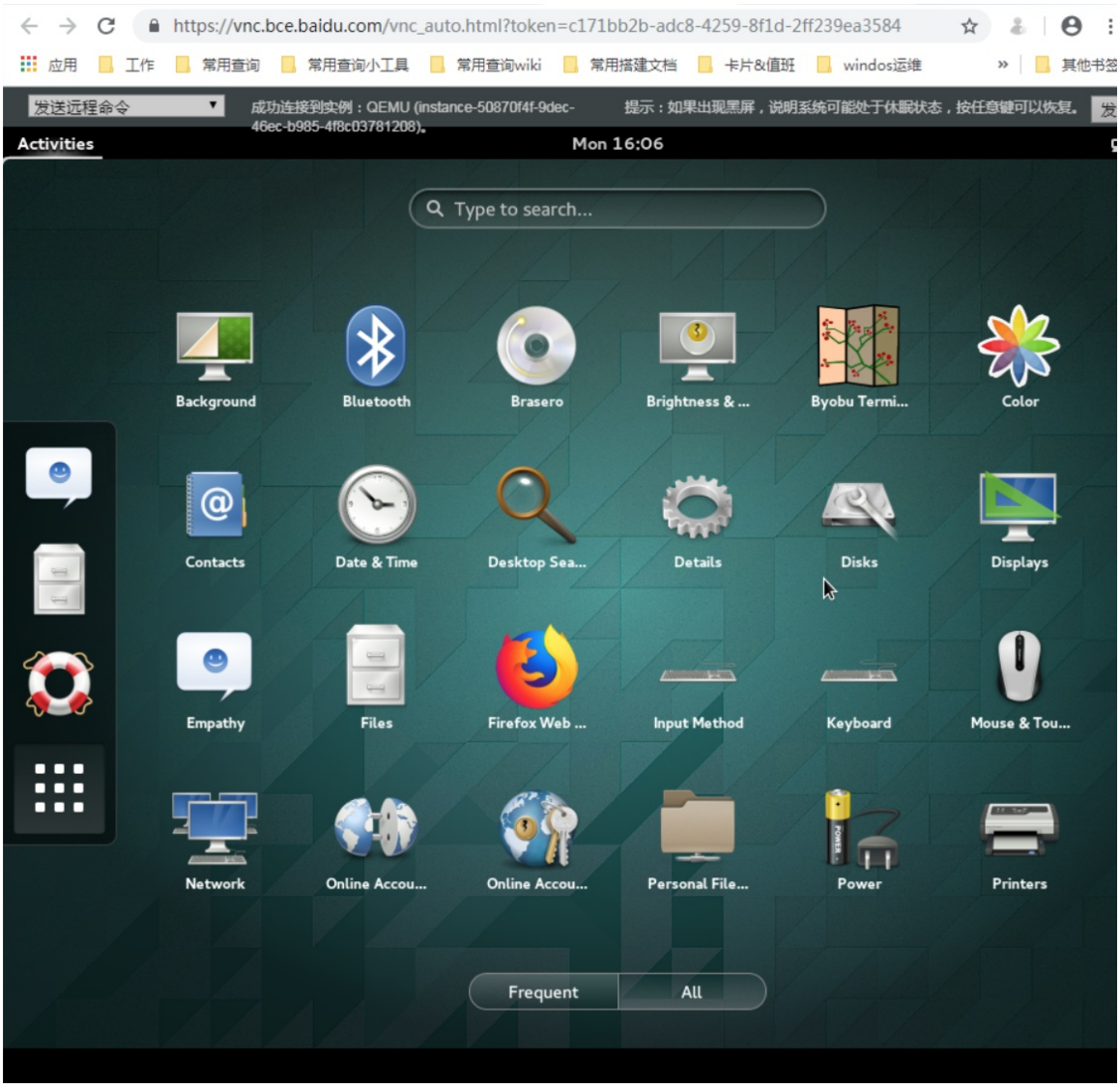
2.执行命令：apt-get -y update更新源。



3.执行命令 apt-get -y install x-window-system-core gnome-core gdm 安装环境。

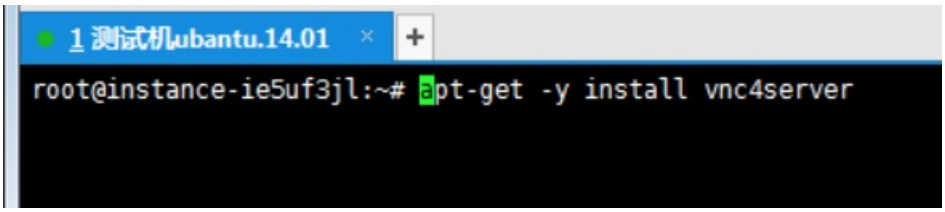


4.执行命令 startx 启动图形化桌面。 效果如下：

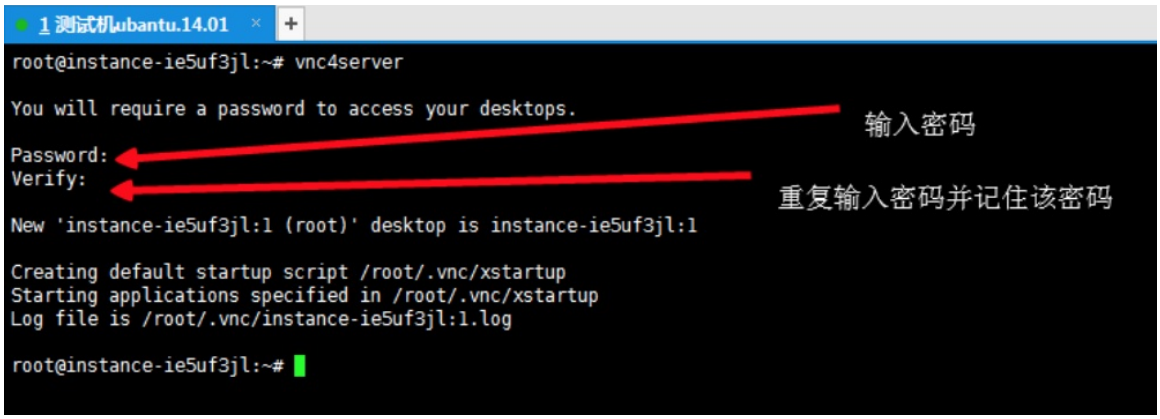


安装VNCserver服务

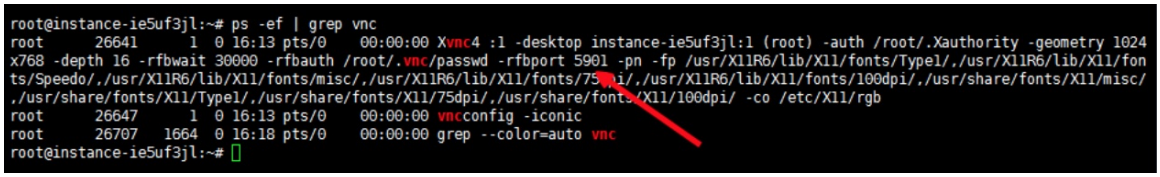
1.执行命令 apt-get -y install vnc4server 安装vncserver。



2.运行vnc4server 命令开启VNC服务并按界面提示设置连接密码。



3.运行命令 ps -ef | grep vnc 确认服务是否已经启动。如果返回以下类似结果，说明服务已经启动。



由返回结果可知，服务使用了TCP 5901端口。

4.运行 cp -p ~/.vnc/xstartup ~/.vnc/xstartup.bak 命令备份原有xstartup文件。

5.按以下步骤修改vnc4server启动文件。

（1）运行 vim ~/.vnc/xstartup 命令打开文件。

（2）按i键进入编辑模式。

（3）将文件内容替换为以下内容。

```
 #!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#x-window-manager &
gnome-panel &
gnome-settings-daemon &
metacity &
nautilus &
gnome-terminal &
```

（4）按Esc键退出编辑模式，再输入:wq保存并退出。

5.依次运行以下命令生成新的会话。

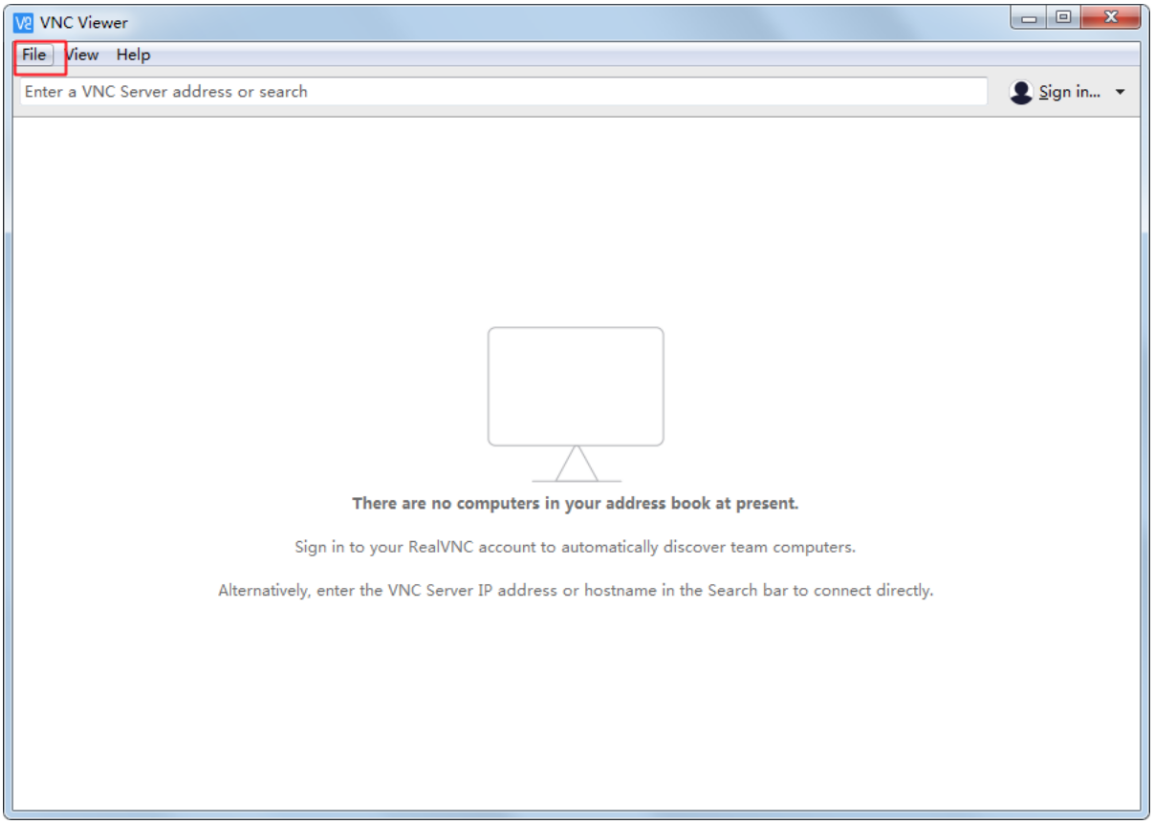
vncserver -kill :1 #杀掉原来的桌面进程（假设桌面号为:1）

vncserver :1 # 生成新的会话

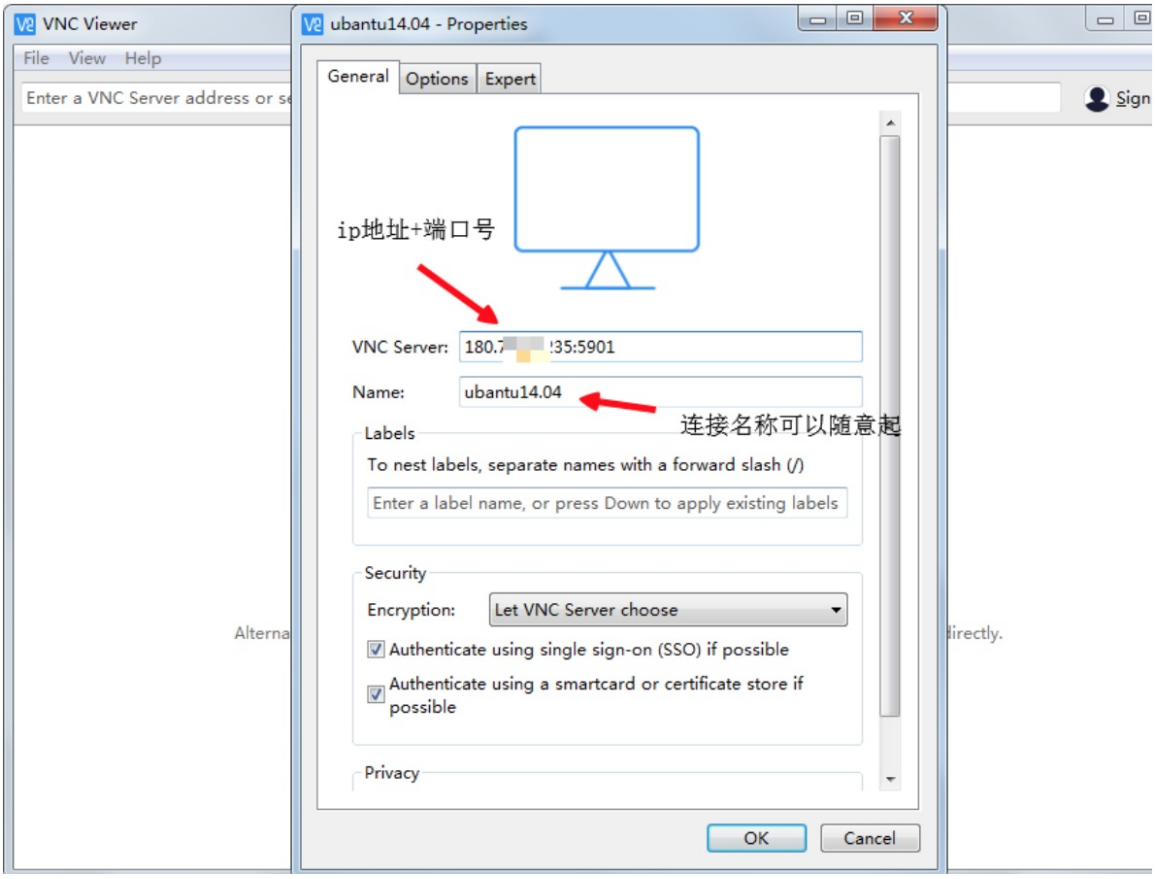
使用viewer客户端连接软件连接该实例

1.使用前提条件中提供的viewer客户端连接软件连接该实例：

点击file------New connection..创建新连接

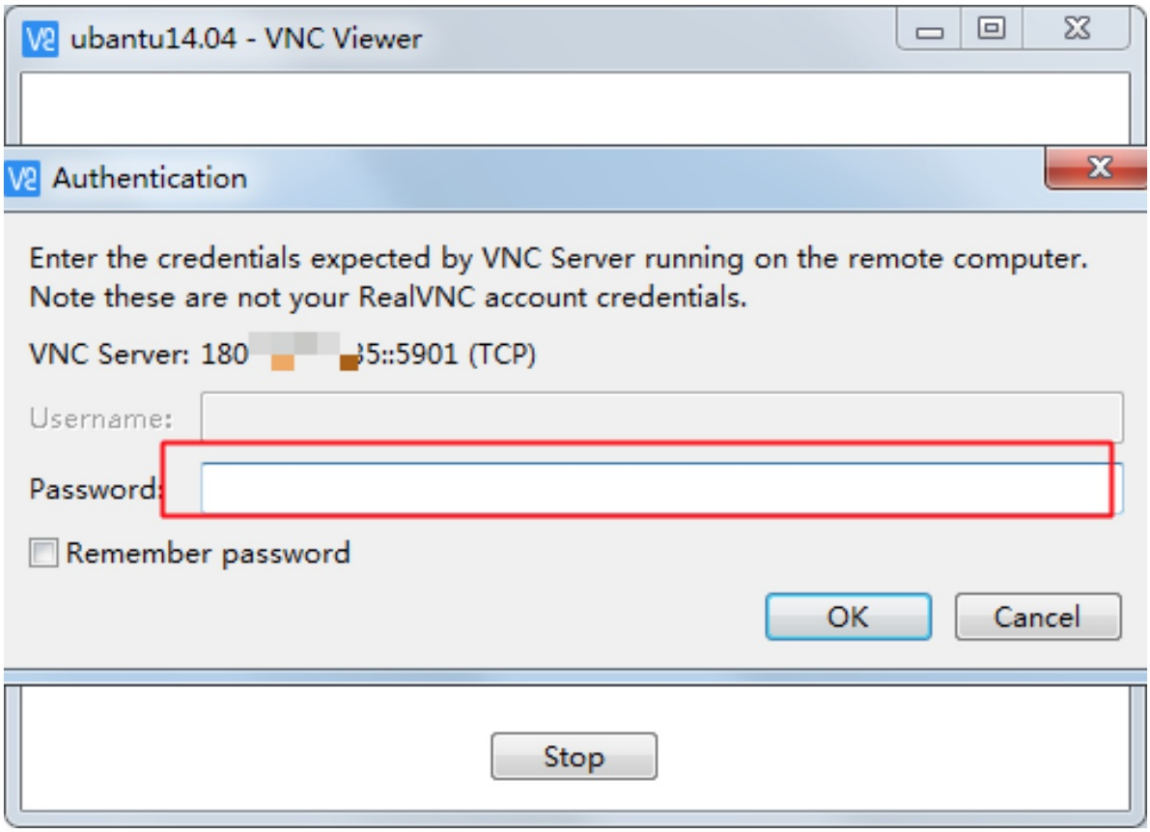


2.输入对应的信息。

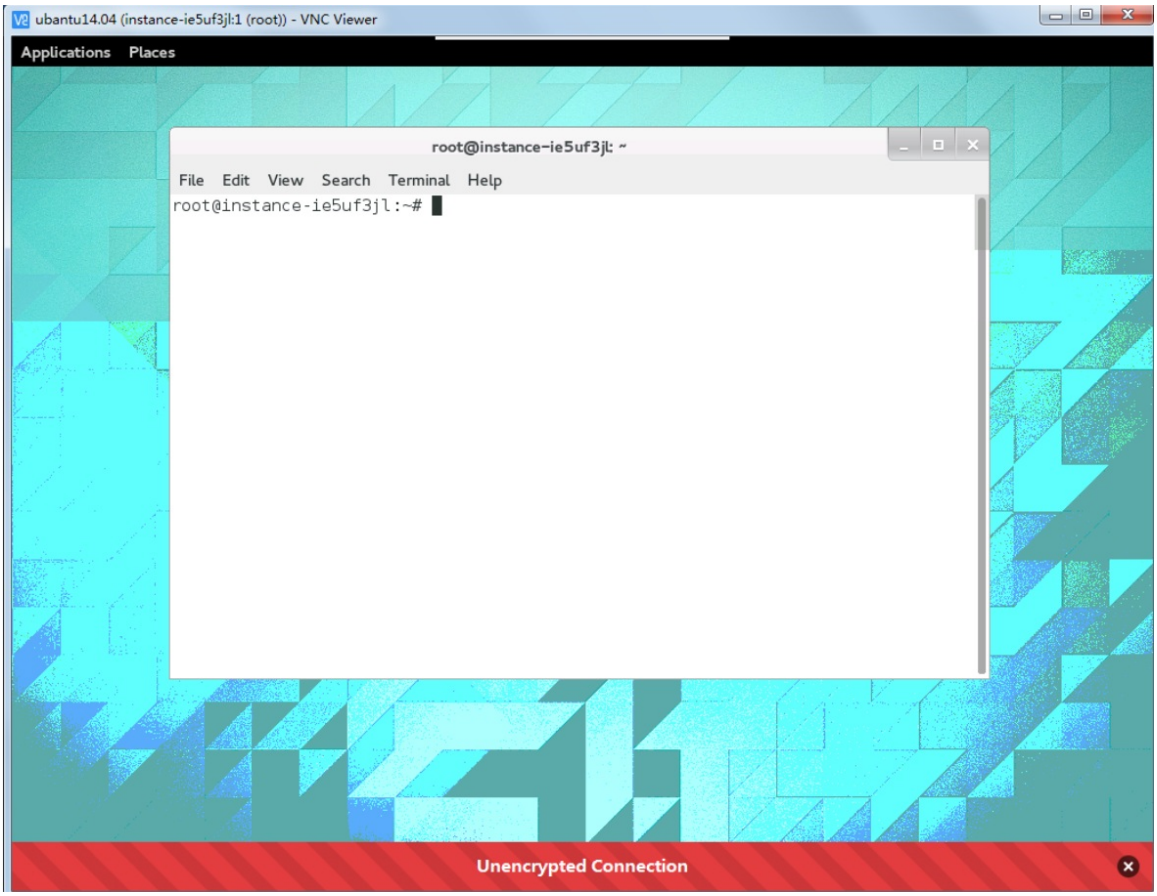


3.双击之前创建的连接，点击进行连接。

4.此处输入第2步骤时设置的密码。



5.连接成功。



Ubuntu18.04安装图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像Ubuntu18.04环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会 显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对BCC实例手动创建快照备份，以备发生异常时能回滚恢复。

步骤一：BCC安装桌面环境。
步骤二：安装VNCserver服务。
步骤三：使用viewer客户端连接软件连接该实例。

前提条件

1. 安全组和防火墙需要放行5901端口（vncserver端口）。
2. 下载 VNC viewer客户端连接软件。

BCC安装桌面环境

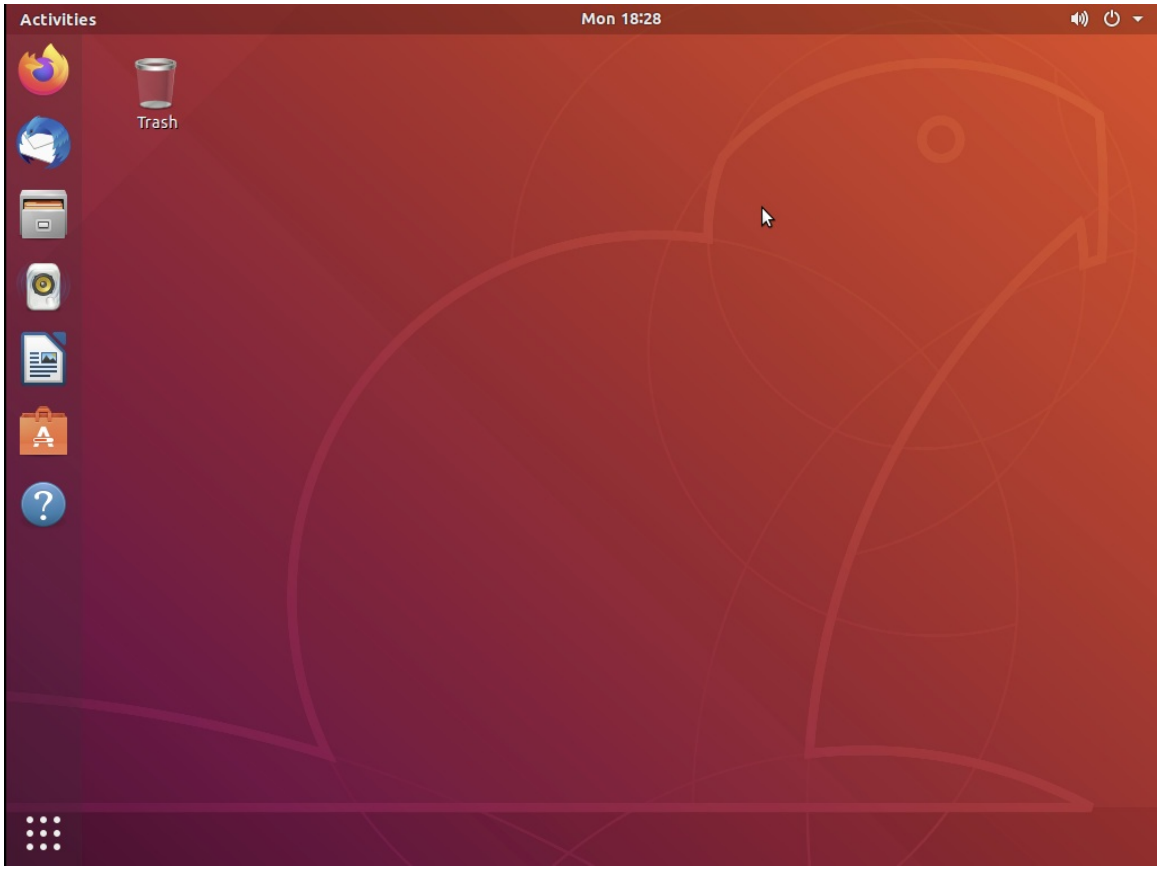
1. 远程或者VNC连接服务器。
2. 执行命令：`apt-get update && apt-get upgrade -y` 更新源。

```
[redacted] # apt-get update && apt-get upgrade -y
Hit:1 http://mirrors.baidubce.com/ubuntu bionic InRelease
Get:2 http://mirrors.baidubce.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://mirrors.baidubce.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://mirrors.baidubce.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://mirrors.baidubce.com/ubuntu bionic-updates/main amd64 Packages [951 kB]
Get:6 http://mirrors.baidubce.com/ubuntu bionic-updates/main i386 Packages [685 kB]
```

3. 执行命令 `apt-get install ubuntu-desktop gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal` 安装环境。

```
[redacted].~# apt-get install ubuntu-desktop gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

4. 重启后，即可使用VNC登录图形化界面 `reboot`。



安装VNCserver服务

1. 执行命令 `apt-get -y install vnc4server` 安装vncserver。

```
[redacted]:~# apt-get -y install vnc4server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  xbase-clients
Suggested packages:
  vnc-java x11-xfs-utils
The following NEW packages will be installed:
  vnc4server xbase-clients
0 upgraded, 2 newly installed, 0 to remove and 167 not upgraded.
Need to get 1,626 kB of archives.
After this operation, 5,517 kB of additional disk space will be used.
Get:1 http://mirrors.baidubce.com/ubuntu bionic-updates/universe amd64 xbase-clients all 1:7.7+19ubuntu7.1 [4,280 B]
Get:2 http://mirrors.baidubce.com/ubuntu bionic/universe amd64 vnc4server amd64 4.1.1+xorg4.3.0-37.3ubuntu2 [1,622 kB]
Fetched 1,626 kB in 0s (12.5 MB/s)
Selecting previously unselected package xbase-clients.
(Reading database ... 165244 files and directories currently installed.)
Preparing to unpack .../xbase-clients_1%3a7.7+19ubuntu7.1_all.deb ...
Unpacking xbase-clients (1:7.7+19ubuntu7.1) ...
Selecting previously unselected package vnc4server.
Preparing to unpack .../vnc4server_4.1.1+xorg4.3.0-37.3ubuntu2_amd64.deb ...
Unpacking vnc4server (4.1.1+xorg4.3.0-37.3ubuntu2) ...
Setting up xbase-clients (1:7.7+19ubuntu7.1) ...
Setting up vnc4server (4.1.1+xorg4.3.0-37.3ubuntu2) ...
update-alternatives: using /usr/bin/vnc4server to provide /usr/bin/vncserver (vncserver) in auto mode
update-alternatives: using /usr/bin/Xvnc4 to provide /usr/bin/Xvnc (Xvnc) in auto mode
update-alternatives: using /usr/bin/x0vnc4server to provide /usr/bin/x0vncserver (x0vncserver) in auto mode
update-alternatives: using /usr/bin/vnc4passwd to provide /usr/bin/vncpasswd (vncpasswd) in auto mode
update-alternatives: using /usr/bin/vnc4config to provide /usr/bin/vncconfig (vncconfig) in auto mode
Processing triggers for man-db (2.8.3-2) ...
```

2. 运行 `vncserver` 命令启动VNC服务端。

```
root@instance-csyypsna:~# vnc4server

You will require a password to access your desktops.

Password:
Verify:
xauth:  file /root/.Xauthority does not exist

New 'instance-csyypsna:1 (root)' desktop is instance-csyypsna:1

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/instance-csyypsna:1.log
```

3. 使用 `ps -ef | grep vnc` 查看是否启动并查看使用端口号。

```
root@instance-csyypsna:~# ps -ef | grep vnc
root      3014     1  0 18:34 pts/0    00:00:00 Xvnc4 :1 -desktop instance-csyypsna:1 (root) -auth /root/.Xauthority -geometry 1024x768 -depth 16 -rfbwait 30000 -rfbauth /root/.vnc/passwd -rfbport 5901 -pn -fp /usr/X11R6/lib/X11/fonts/Type1/,/usr/X11R6/lib/X11/fonts/Speedo/,/usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fonts/100dpi/,/usr/share/fonts/X11/misc/,/usr/share/fonts/X11/Type1/,/usr/share/fonts/X11/75dpi/,/usr/share/fonts/X11/100dpi/ -co /etc/X11/rgb
root      3020     1  0 18:34 pts/0    00:00:00 vncconfig -iconic
root      3035  2700  0 18:35 pts/0    00:00:00 grep --color=auto vnc
```

可以看到服务已正常启动，并且使用的是5901端口。

4. 运行 cp -p ~/.vnc/xstartup ~/.vnc/xstartup.bak 命令备份原有xstartup文件。

5. 按以下步骤修改vnc4server启动文件。

（1）运行 vim ~/.vnc/xstartup 命令打开文件。

（2）按i键进入编辑模式。

（3）将文件内容替换为以下内容。

```
 #!/bin/sh
**Uncomment the following two lines for normal desktop:**
**unset SESSION_MANAGER**
**exec /etc/X11/xinit/xinitrc**
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
x-window-manager &
gnome-panel &
gnome-settings-daemon &
metacity &
nautilus &
```

（4）按Esc键退出编辑模式，再输入:wq保存并退出。

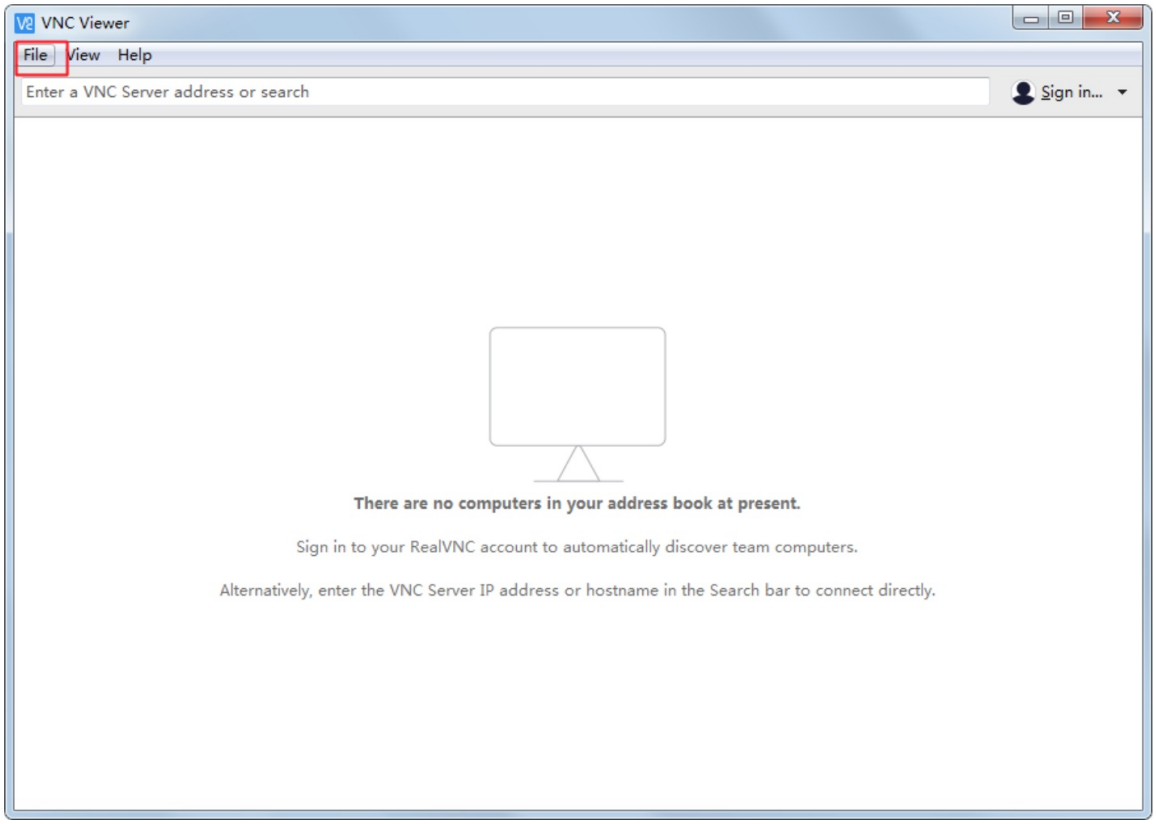
6. 依次运行以下命令生成新的会话。

vncserver -kill :1 #杀掉原来的桌面进程（假设桌面号为:1）

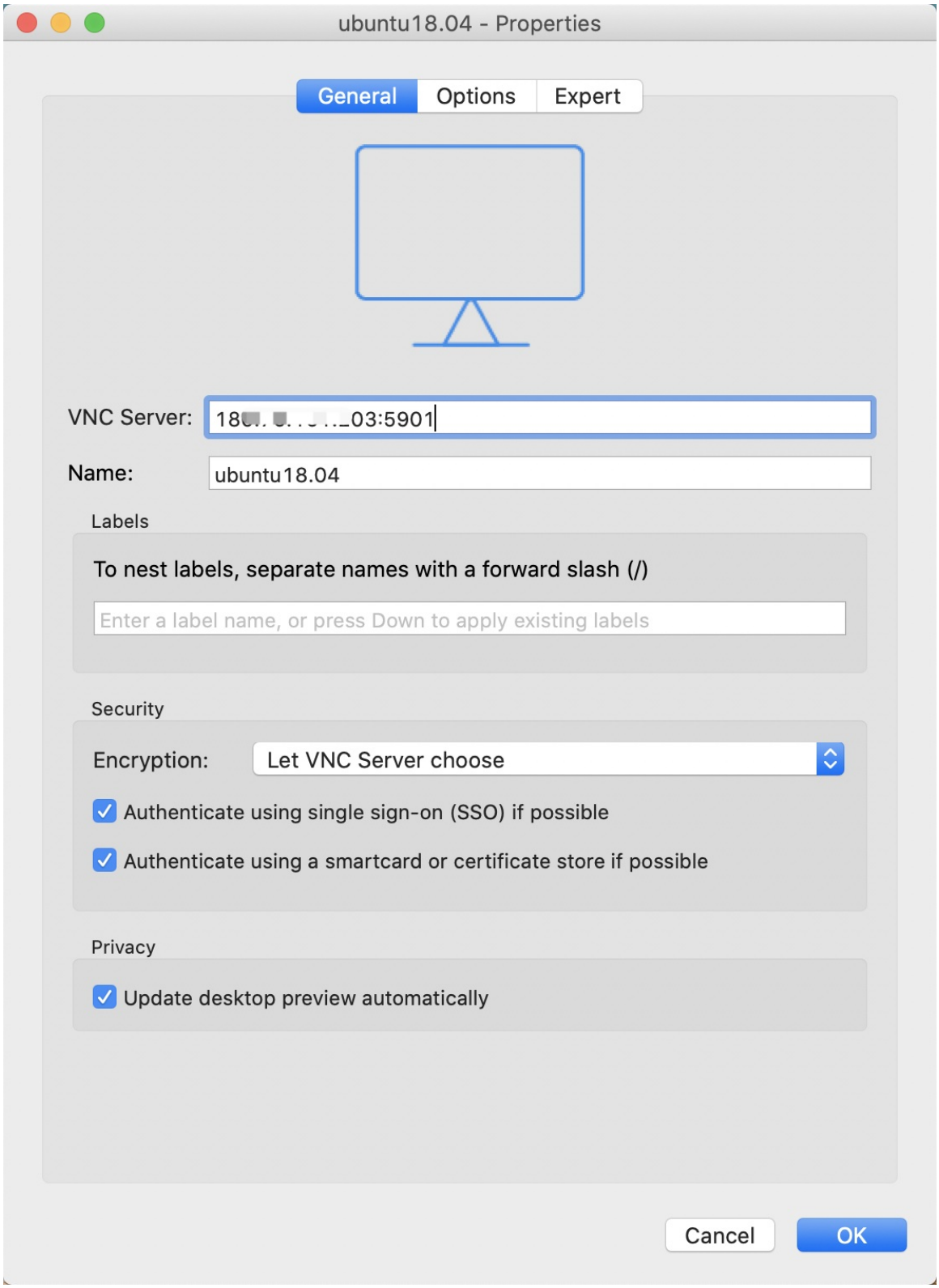
vncserver :1 # 生成新的会话

使用viewer客户端连接软件连接该实例

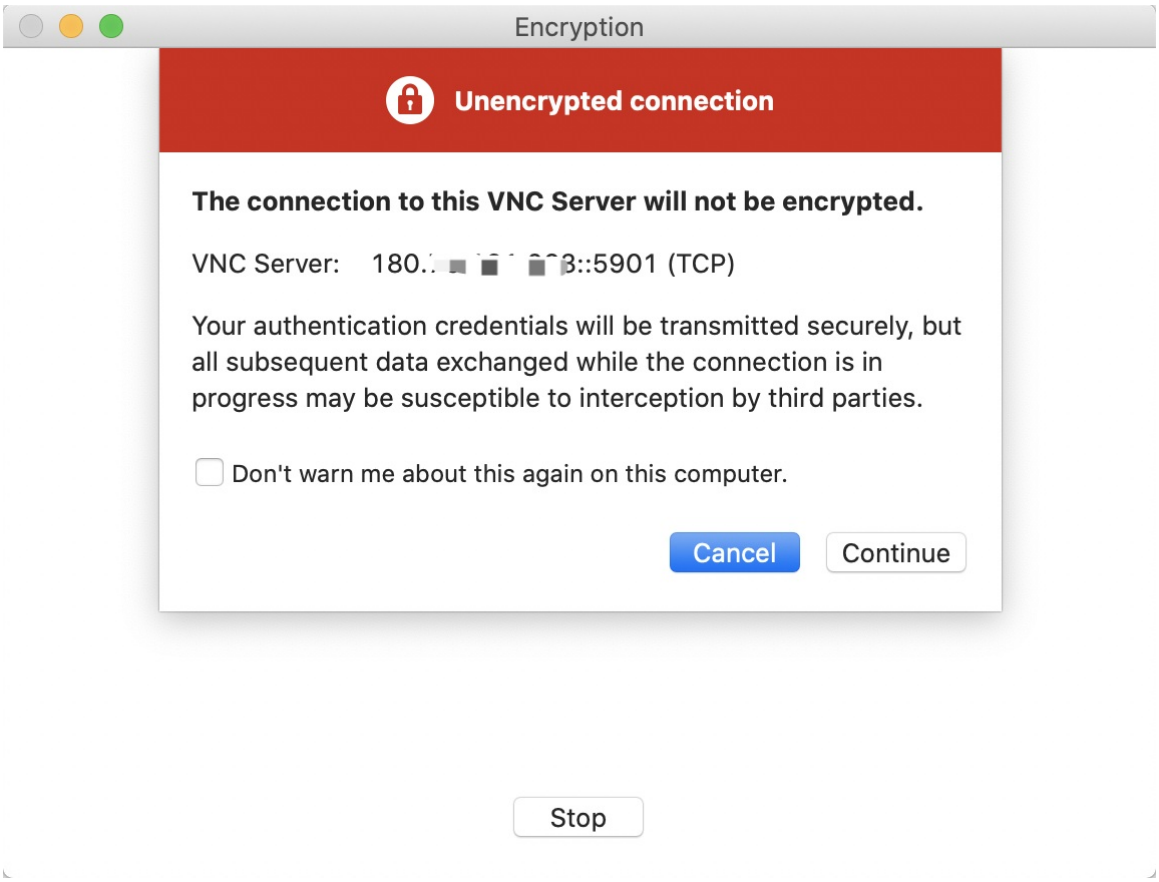
1. 使用前提条件中提供的viewer客户端连接软件连接该实例： 点击file-------New connection..创建新连接。


VNC Viewer
File View Help
Enter a VNC Server address or search
Sign in...
There are no computers in your address book at present.
Sign in to your RealVNC account to automatically discover team computers.
Alternatively, enter the VNC Server IP address or hostname in the Search bar to connect directly.

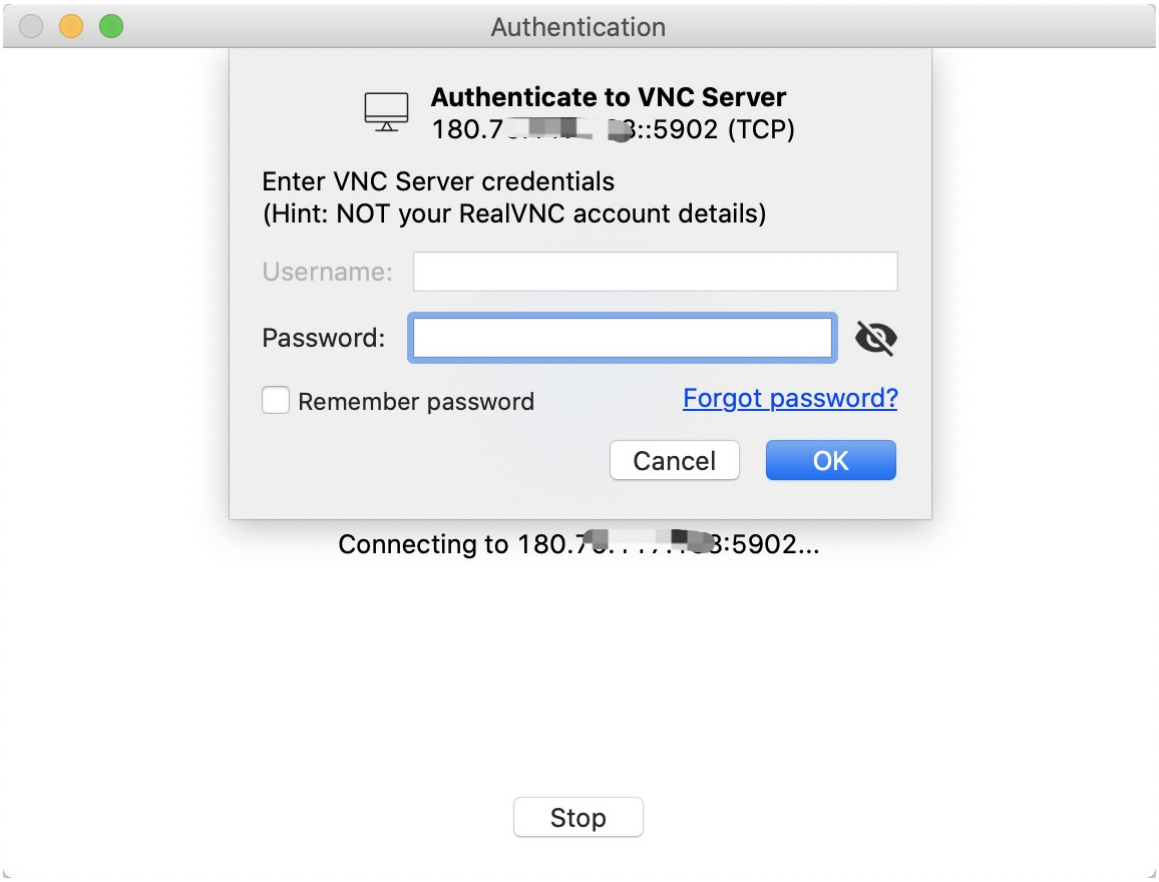
2. 输入对应的信息。



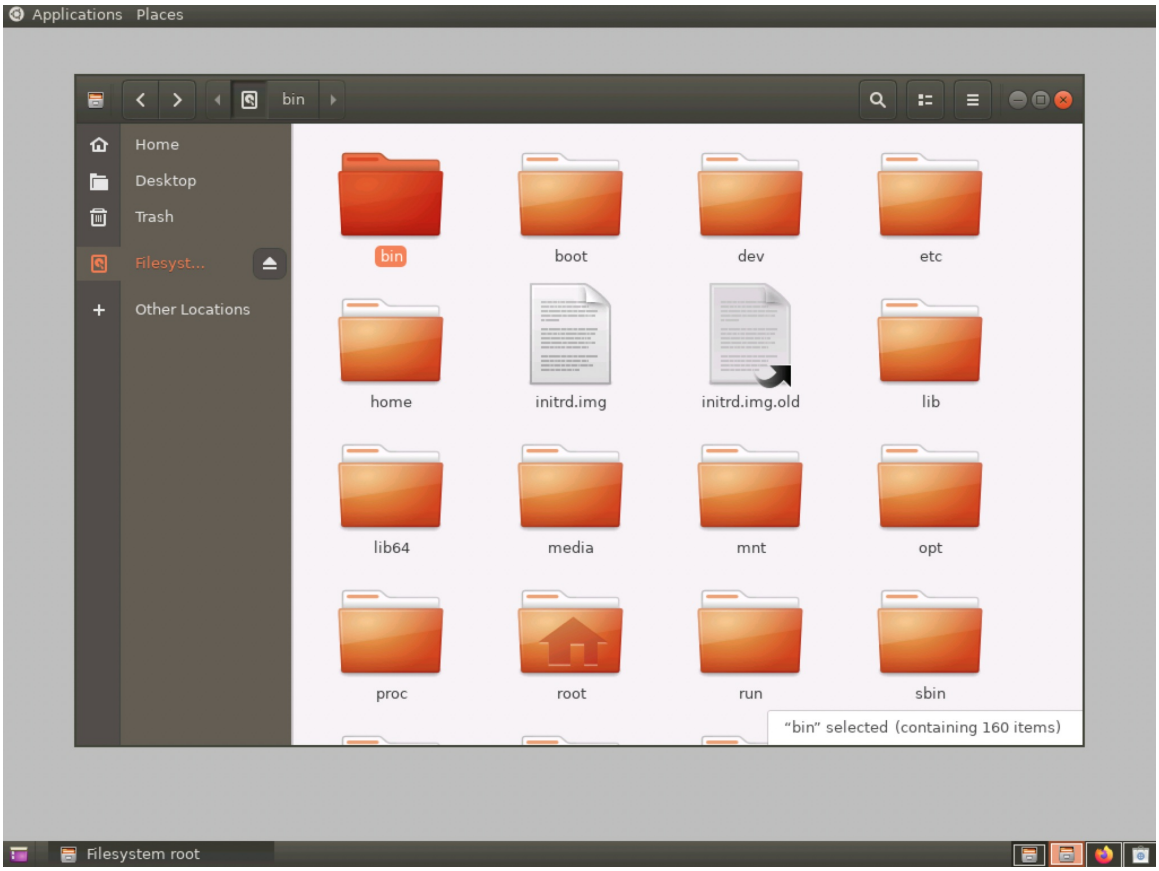
3. 点击"Continue"。



4. 此处输入"安装vncserver"时第2步骤时设置的密码。



5. 连接成功。



opensuse

opensuse42.3安装图形化界面并使用VNCViewer连接

本教程指导如何在百度智能云公共镜像opensuse42.3环境下给服务器安装图形化界面并在本地使用VNC客户端连接。

> 注意：图形化桌面可能会显著降低服务器的性能，如果安装不当，可能会导致操作系统无法正常启动。在安装之前，请您先对BCC 实例创建手工快照备份，以备发生异常时能回滚恢复。

步骤一：安装桌面环境。

步骤二：运行VNCserver服务。

步骤三：使用viewer客户端连接软件连接该实例。

安装桌面环境

1. 远程连接登录服务器。

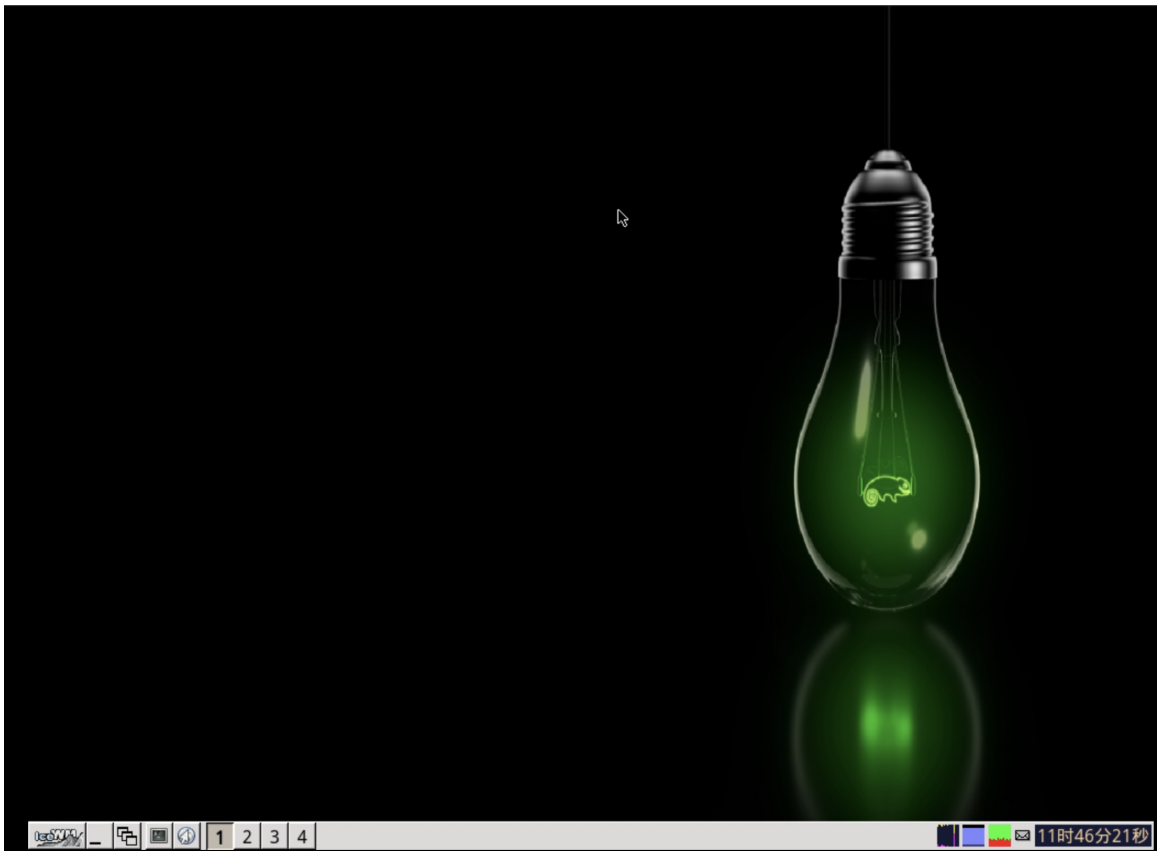
2. 执行命令：`zypper install -t pattern kde kde_plasma` ，该命令会自动安装依赖项，等待所有依赖项依次安装，进度显示为`done`，且没有提示错误，图形应用安装完毕。

```
instance-2f2m592e:~ # zypper install -t pattern kde kde_plasma
```

3. 安装完成后，编辑 /etc/sysconfig/displaymanager 文件并设定 DISPLAYMANAGER="kdm"。

```
## Path:        Desktop/Display manager
## Description: settings to generate a proper displaymanager config
DISPLAYMANAGER="kdm"
## Type:        string(Xorg)
## Path:        Desktop/Display manager
## Default:     "Xorg"
#
DISPLAYMANAGER_XSERVER="Xorg"

## Type:        yesno
## Default:     no
#
# Allow remote access (XDMCP) to your display manager (xdm/kdm/gdm). Please note
# that a modified kdm or xdm configuration, e.g. by KDE control center
# will not be changed. For gdm, values will be updated after change.
# XDMCP service should run only on trusted networks and you have to disable
# firewall for interfaces, where you want to provide this service.
#
DISPLAYMANAGER_REMOTE_ACCESS="no"
```

4. 到此为止，图形界面安装成功，直接执行startx，然后通过VNC可以看到图形化桌面已经安装完毕。



运行VNCserver服务

1. 运行 `vncserver` 命令开启VNC服务并按界面提示设置连接密码。

```
instance-2f2m592e:~ # vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? y
Password:
Verify:

New 'instance-2f2m592e:1 (root)' desktop is instance-2f2m592e:1

Creating default startup script /root/.vnc/xstartup
Creating default config /root/.vnc/config
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/instance-2f2m592e:1.log
```

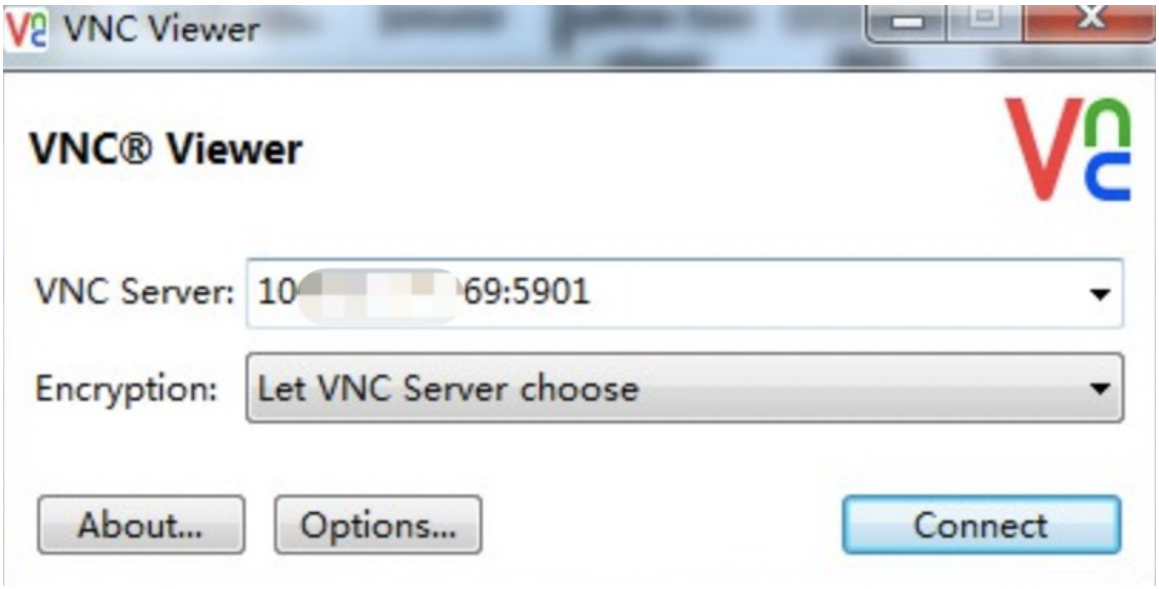
2. 运行命令 `ps -ef | grep vnc` 确认服务是否已经启动。如果返回以下类似结果，说明服务已经启动。

```
instance-2f2m592e:~ # ps -ef | grep vnc
root     17173     1  1 11:52 pts/0    00:00:00 /usr/bin/Xvnc :1 -auth /root/.Xauthority -desktop instance-2f2m592e:1 (root) -fp /usr/share/fonts/misc,/usr/share/fonts/75dpi,/usr/share/fonts/100dpi,/usr/share/fonts/Speedo,/usr/share/fonts/Type1 -geometry 1024x768 -httpd /usr/share/vnc/classes -pn -rfbauth /root/.vnc/passwd -rfbport 5901 -rfbwait 30000
root     17348     1  0 11:52 pts/0    00:00:00 /usr/bin/vncconfig -nowin
root     18669 11097  0 11:52 pts/0    00:00:00 grep --color=auto vnc
```

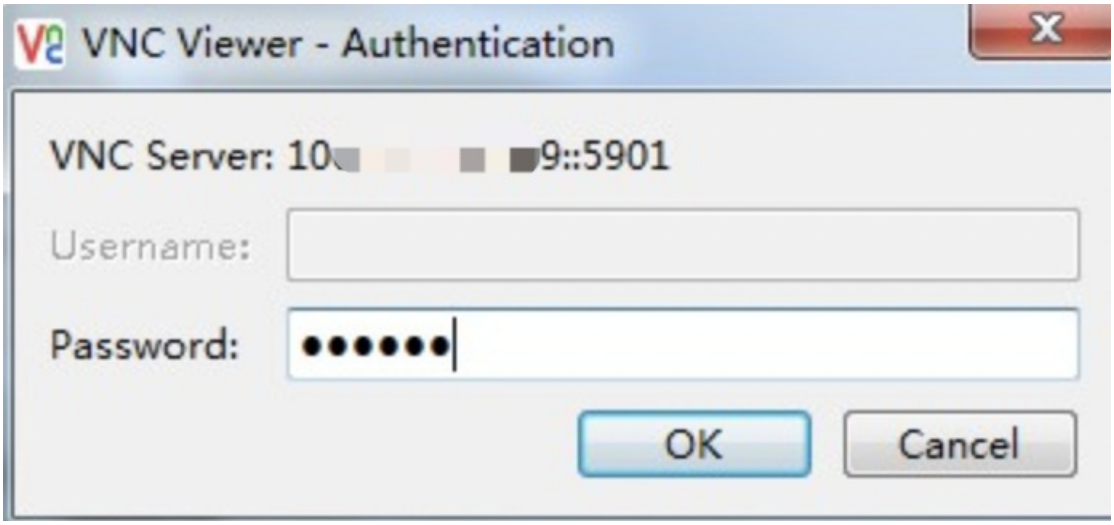
由返回结果可知，服务使用了TCP 5901端口。

使用viewer客户端连接软件连接该实例

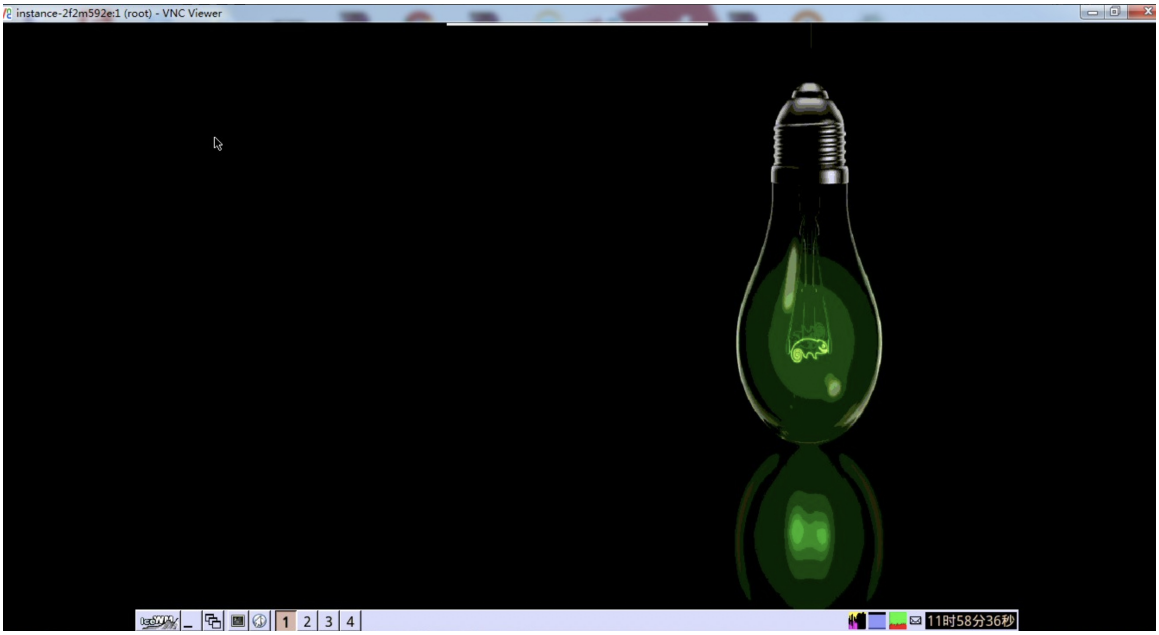
1. 打开vnc viewer，输入对应的信息。



2. 点击"Connect"，输入启动vnc时设置的密码。



3. 连接成功。



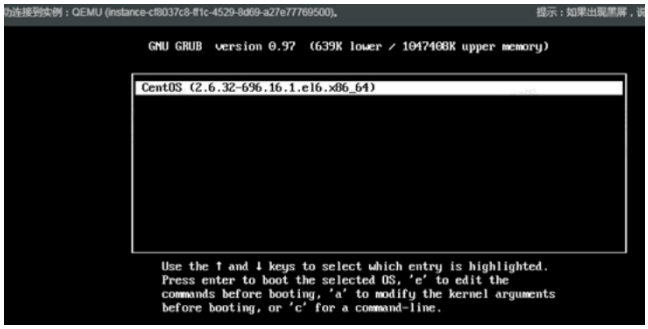
进入单用户模式

CentOS6进入单用户

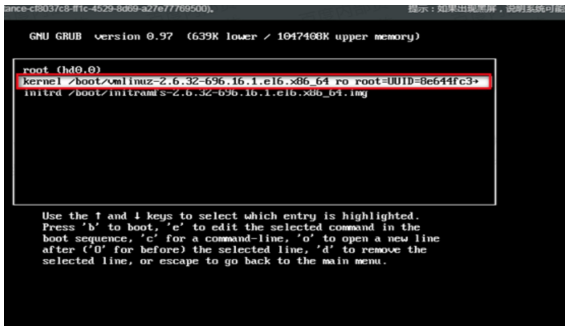
1．VNC连接服务器后，在VNC右上方点击 “发送Ctrl+Alt+Del”进行重启服务器，进入重启状态，此时及时按Esc键，进入下一界

面。

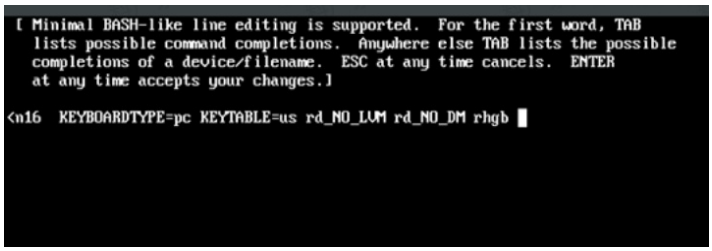
2.按Esc键后，进入Grub界面，此时按e键。



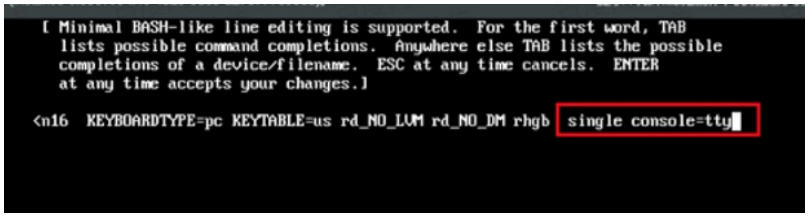
3.之后进入如下界面，上下键选中如下红色标记的Kernel,按e键。



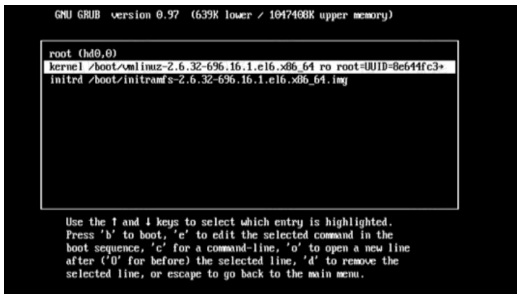
4.之后进入如下界面：



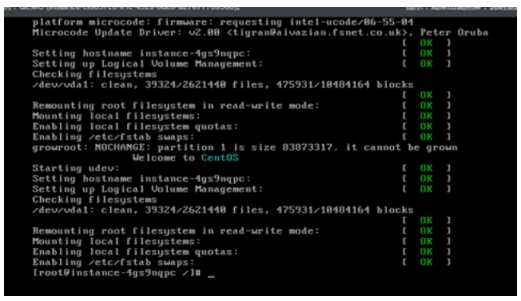
5.在末尾增加： single console=tty， 修改后如下所示：



6.按Enter 键 ，会返回之前界面如下，再按b键进行启动。



7.之后会启动系统，进入单用户模式，不需要密码进入，在此界面可执行修改密码等操作。

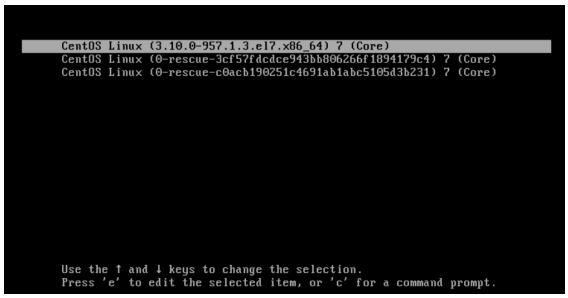


CentOS7进入单用户

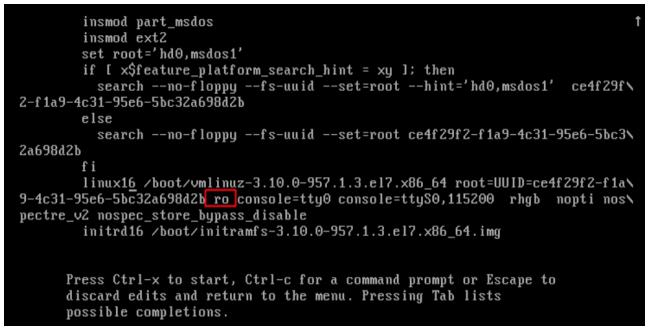
本文介绍百度智能云“CentOS 7如何进入单用户模式”的方法（含操作指导视频），您可以参考视频中的关键步骤指导，使用智能云产品或是自助解决问题，请点击此处观看视频《CentOS 7如何进入单用户模式并重置密码》。

> **说明：**
>
> 如视频指导中的部分界面元素或操作方式发生变化，请以智能云控制台界面为准。本视频重点演示功能操作流程，界面的细微变化不影响流程的整体呈现。

1.点击界面右上角的按钮“发送Ctrl+Alt+Del”进行重启服务器，进入重启状态，显示如下页面快速按e。

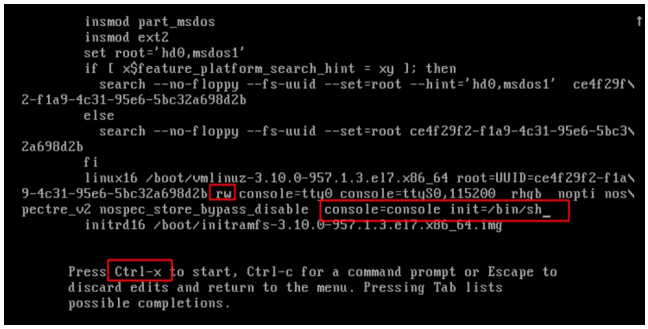


显示此页面：



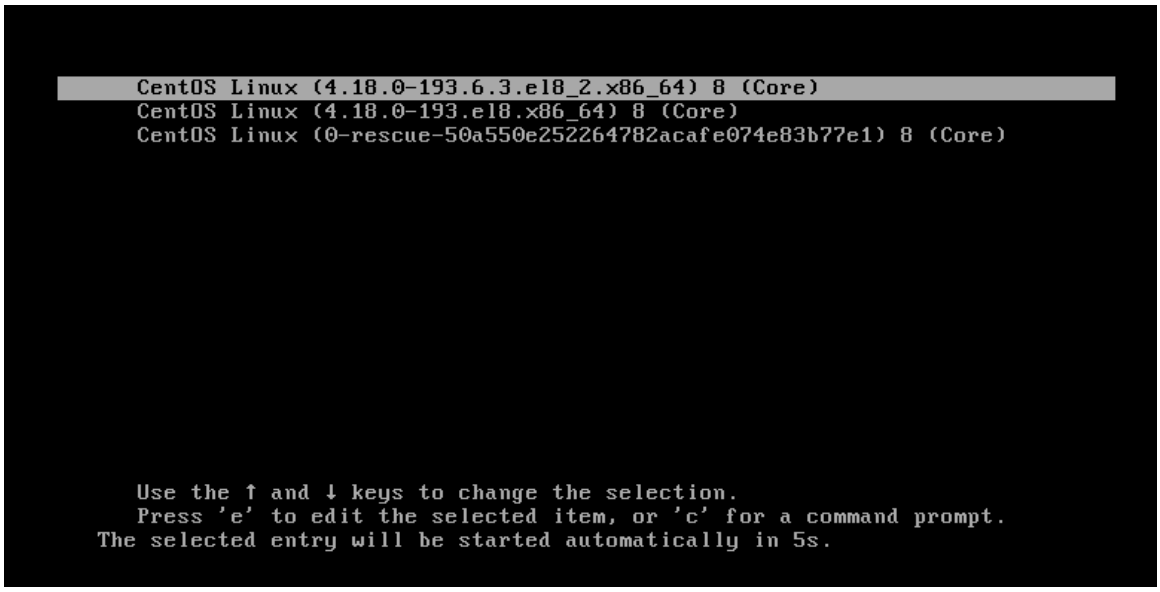
2.找到第一个linux16部分，修改ro为rw，在末尾后面添加`console=console init=/bin/sh`

按Ctrl+X重启 进入单用户模式。



Centos8进入单用户

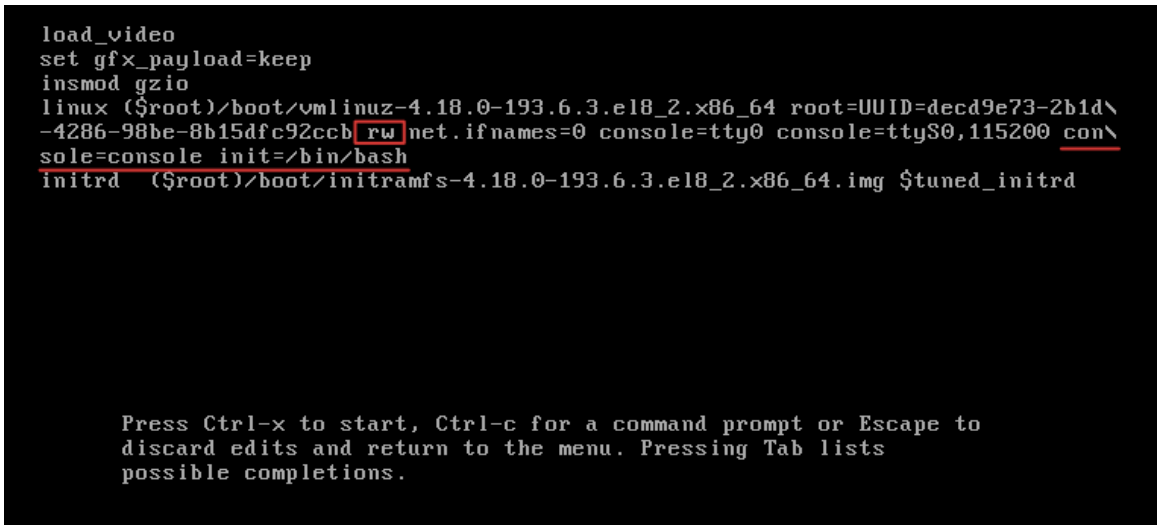
1. 点击界面右上角的按钮 **发送Ctrl+Alt+Del** 进行重启服务器，进入重启状态，显示如下页面快速按e。



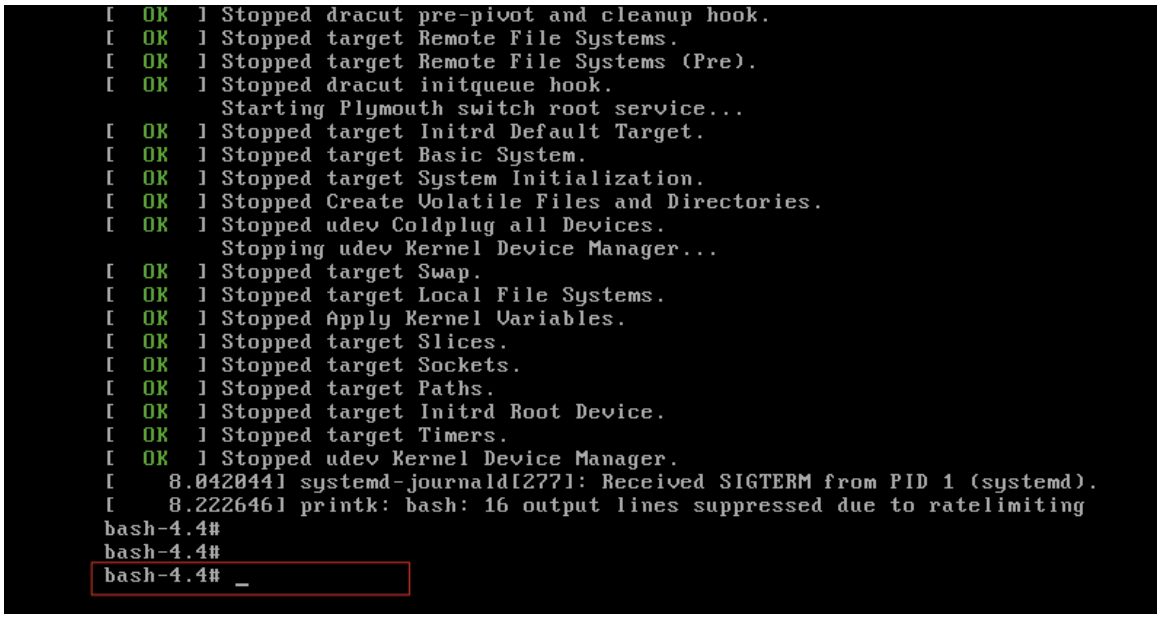
显示此页面：

2. 找到第一个linux部分，修改ro为rw，在末尾后面添加 `console=console init=/bin/sh`。

按 Ctrl+X 重启，进入单用户模式。



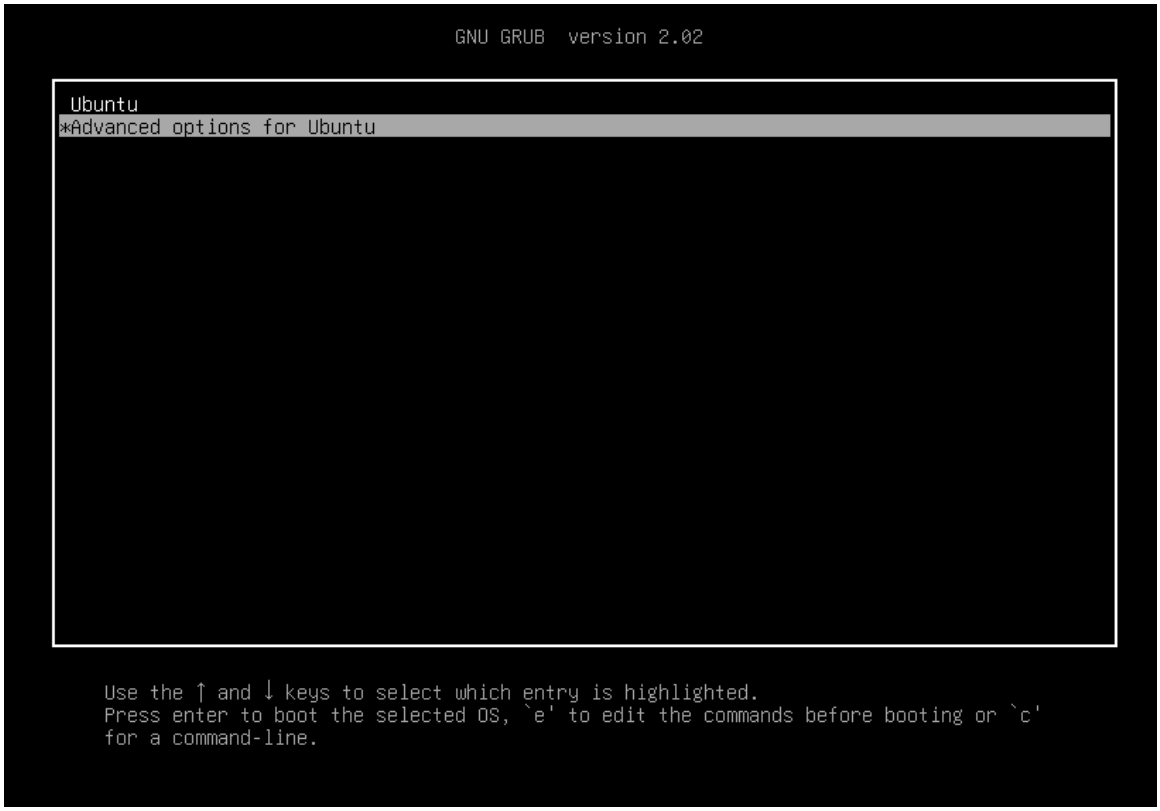
3. 显示如下界面，即为正常进入了单用户模式。



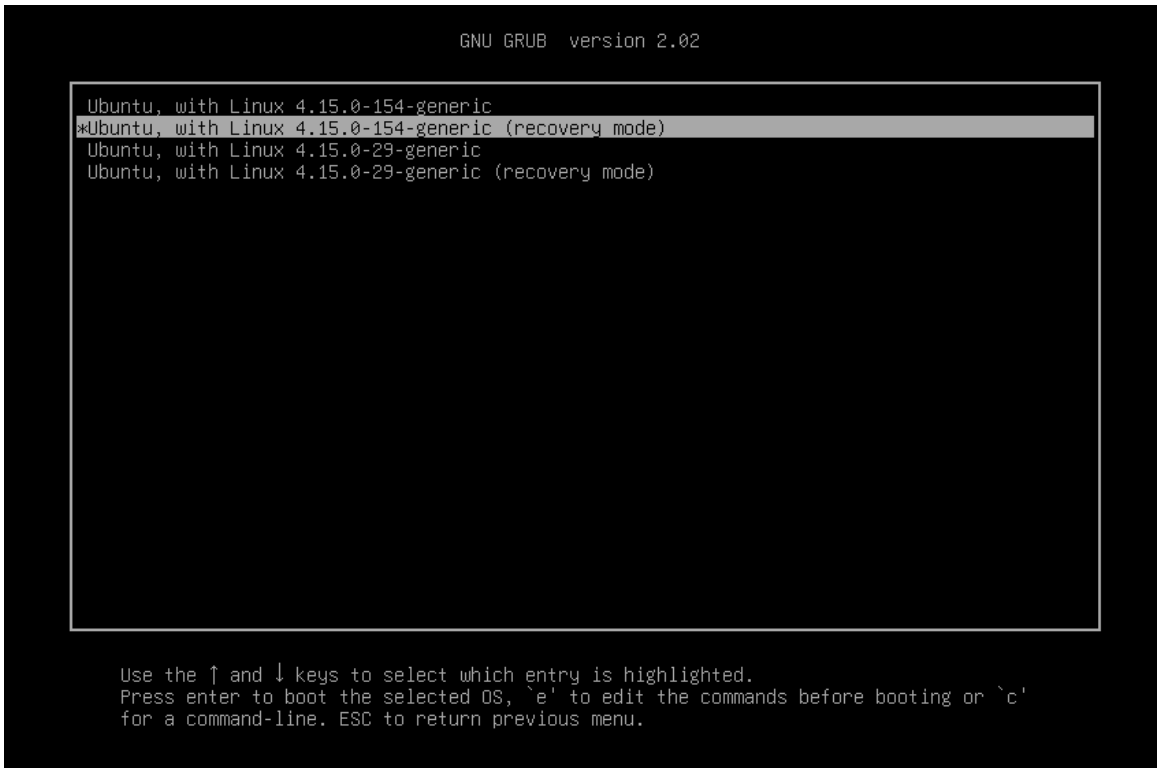
Ubuntu进入单用户

1.在vnc登录窗口，点击右上角按钮，发送命令重启系统。

2.启动系统的时候等待一段时间或者长按 shift 键，进入grub菜单，然后选择Ubuntu高级选项，按 回车 键。



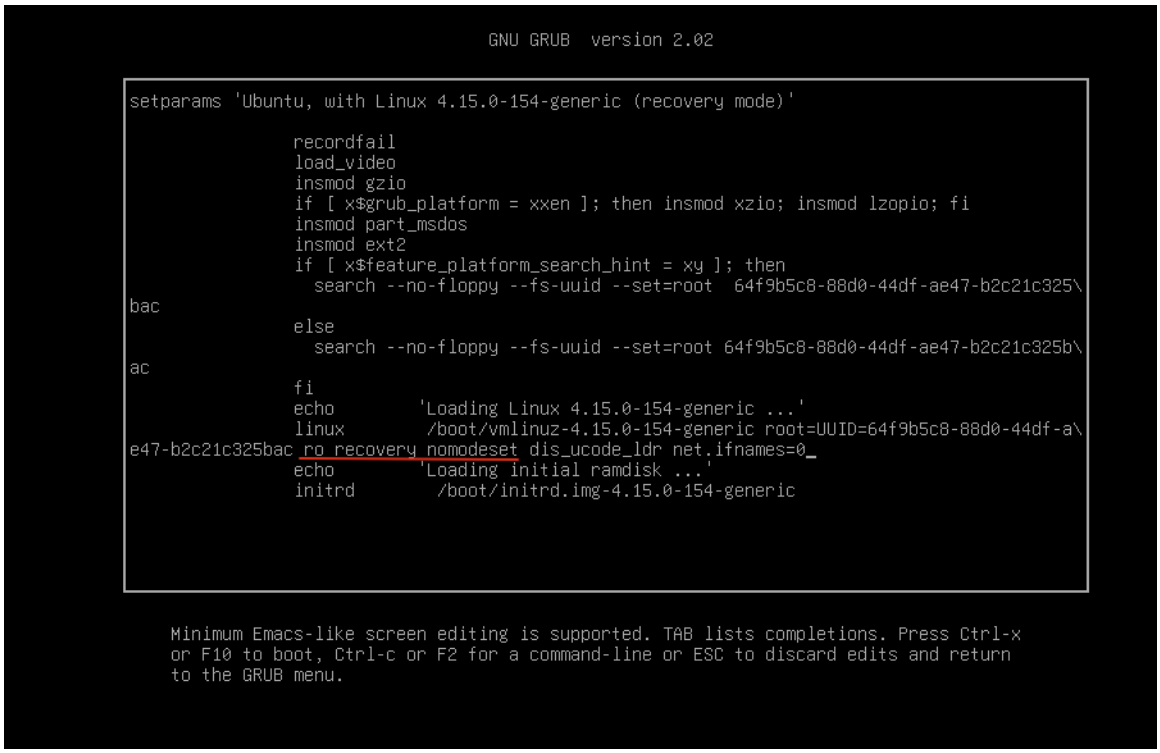
3.选择recovery模式，按 e 进入编辑。



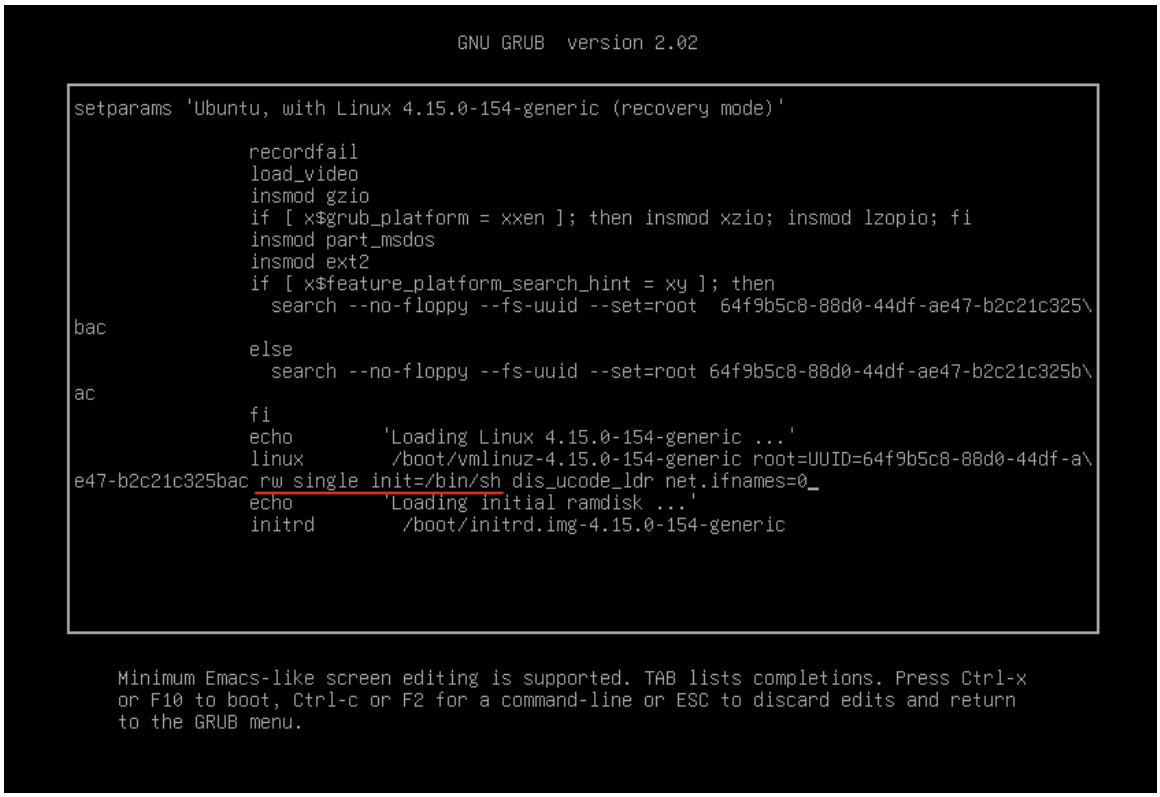
修改里面的linux行：将原来的 `ro recovery nomodeset` 删除掉，并修改为 `rw single init=/bin/sh` 如下图所示：

> 请注意，对于Ubuntu18.04、Ubuntu20.04系统，需要将 `ro recovery nomodeset dis_ucode_ldr` 删除掉，并修改为上述字段。

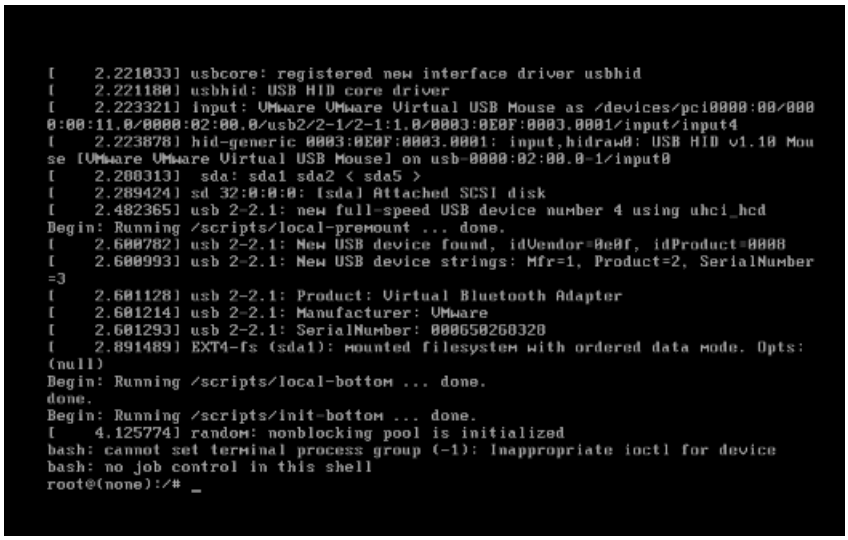
修改前：



修改后：

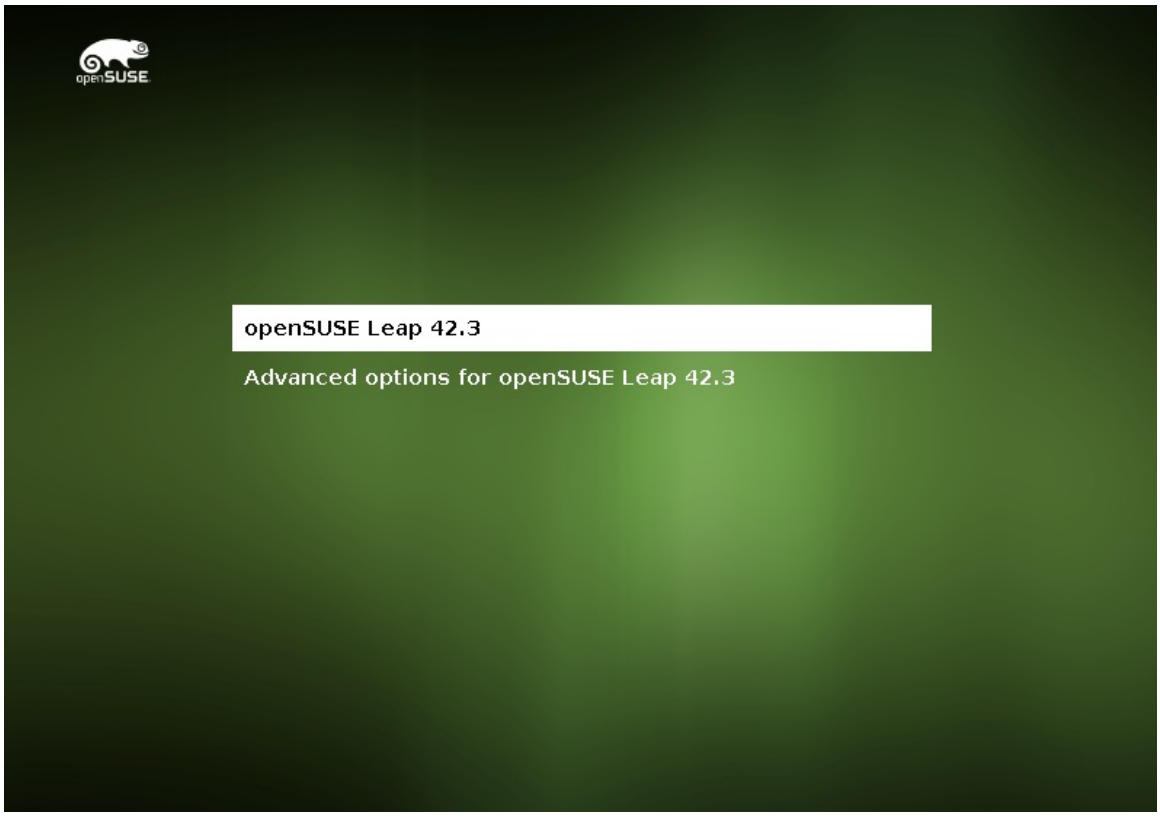


4.按ctrl + x或者F10退出并启动，进入单用户模式。



opensuse42.3进入单用户

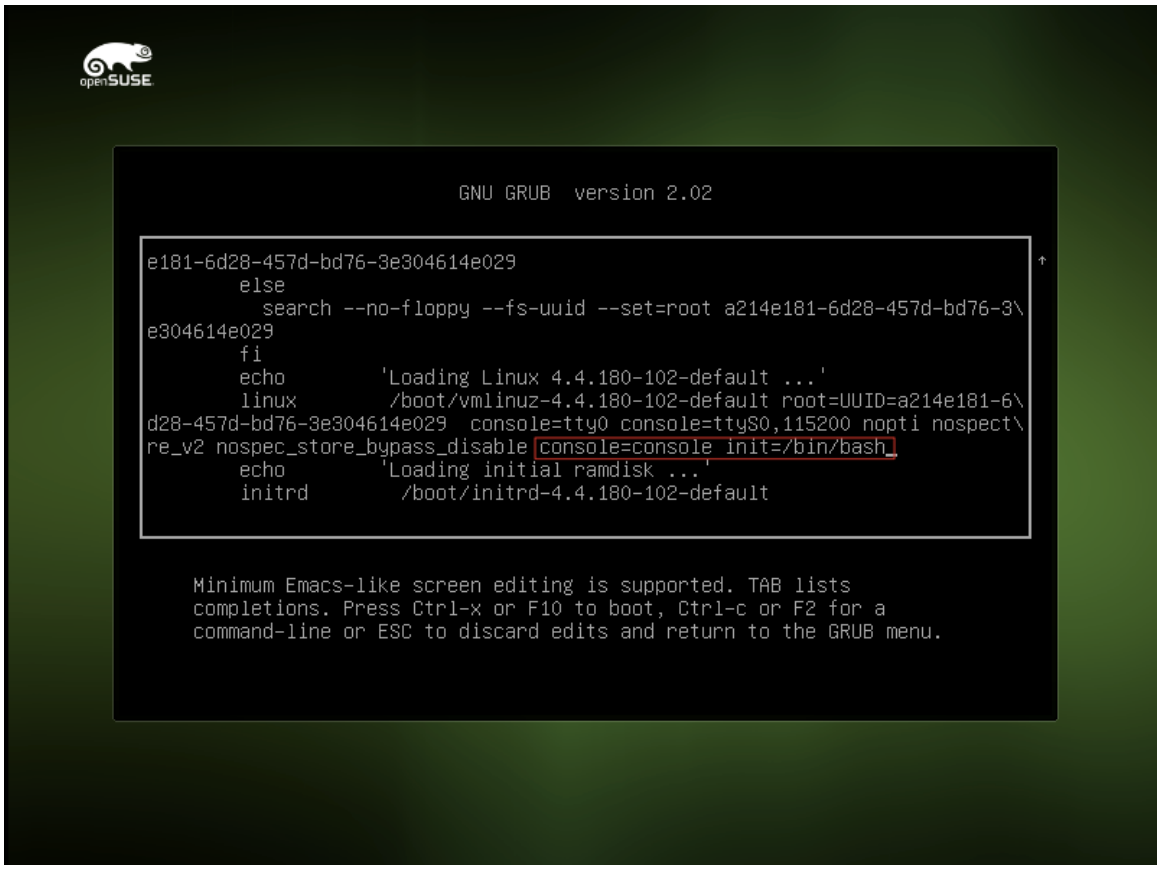
1. 点击界面右上角的按钮 **发送Ctrl+Alt+Del** 进行重启服务器，进入重启状态，显示如下页面快速按e。



显示此页面：

2. 找到第一个linux部分，在末尾后面添加 `console=console init=/bin/sh`。

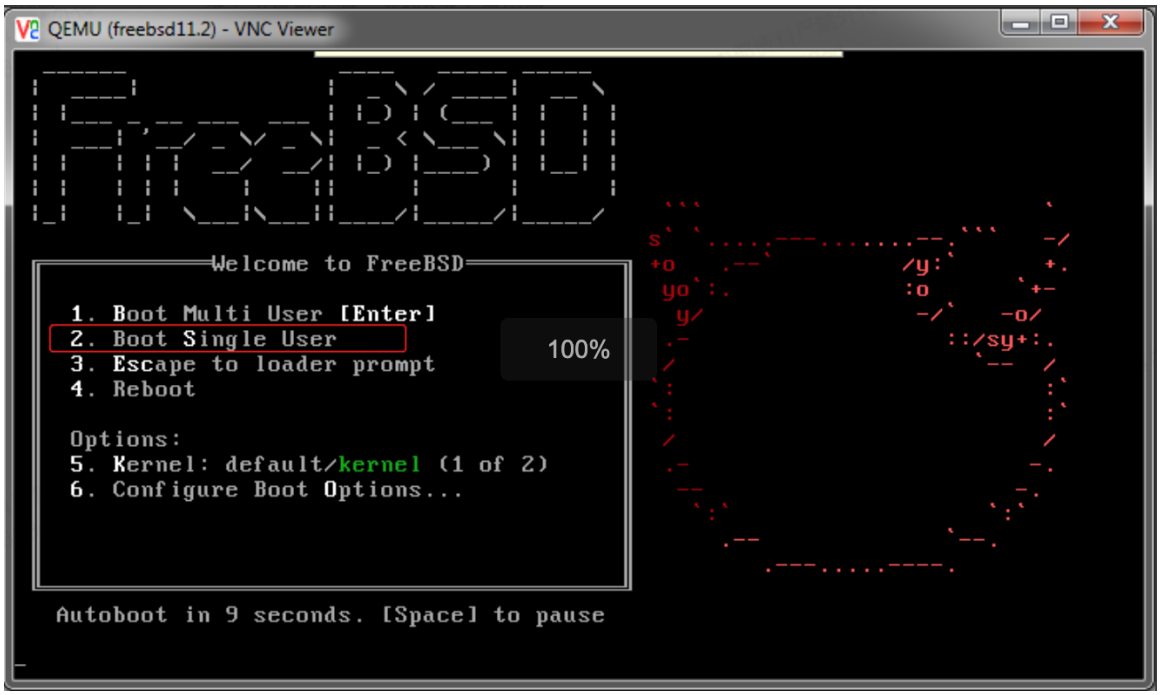
按 **Ctrl+X** 重启，进入单用户模式。



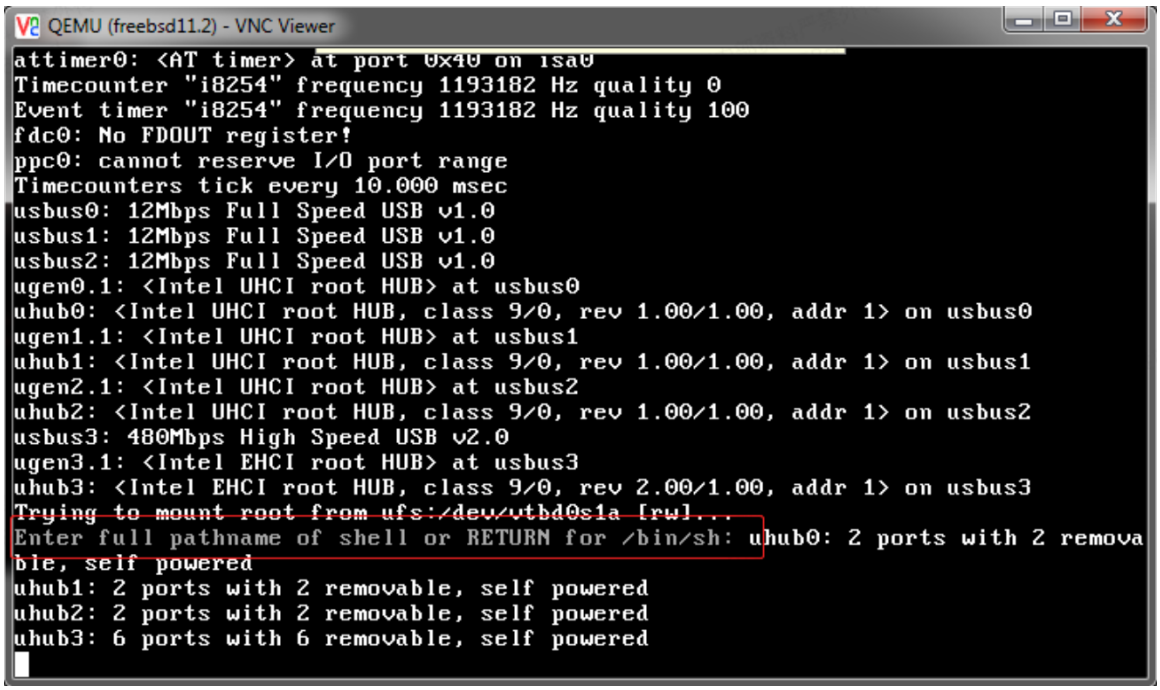
3. 显示如下界面，即为正常进入了单用户模式。

freebsd11单用户修改密码SOP

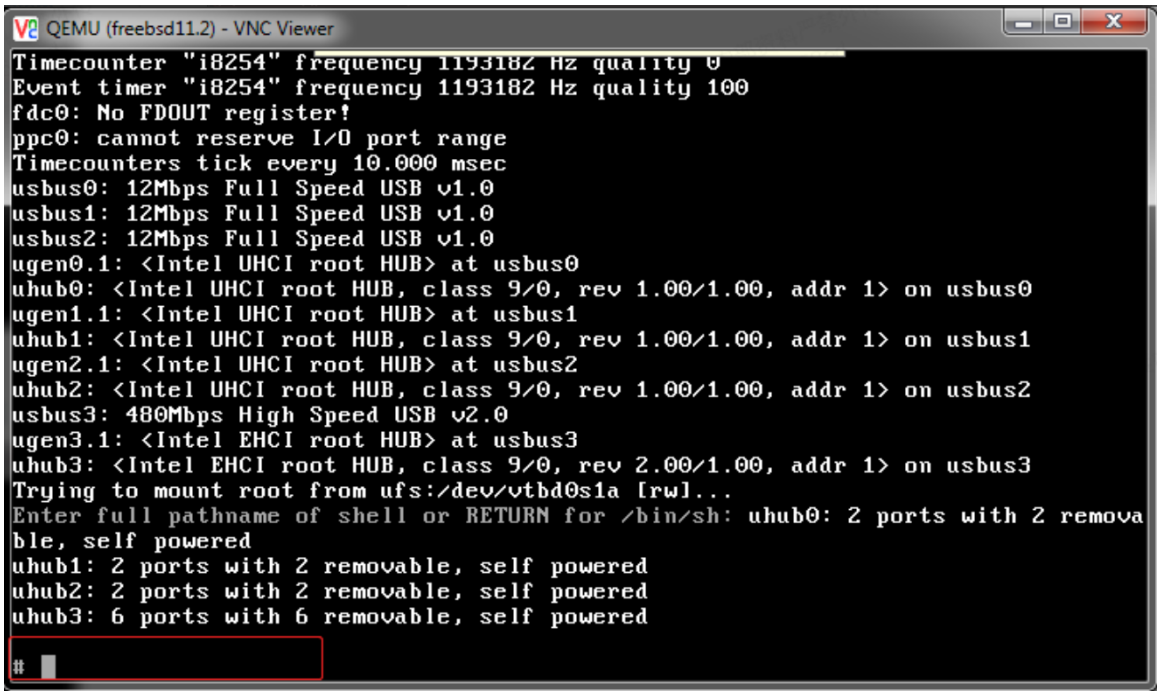
1. 进入引导界面时，选择2进入单用户。



2. 当出现"Enter full pathname of shell or ENTURN for /bin/sh"时说明已经进入单用户模式。

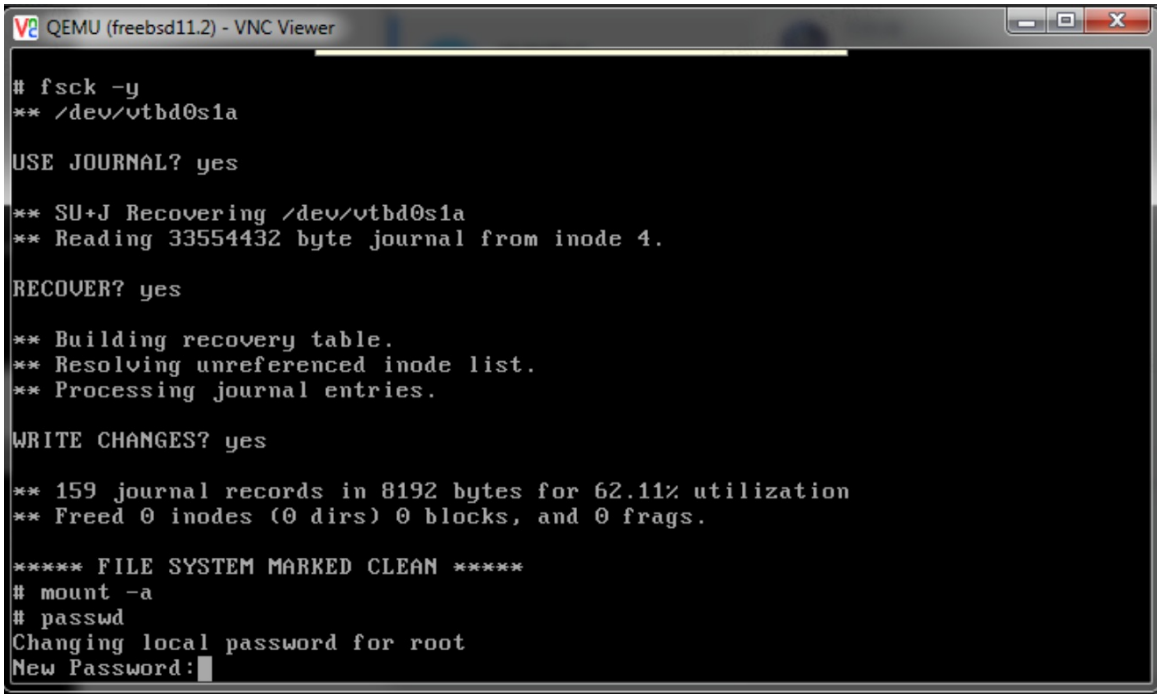


3. 按回车，出现 # 符号。



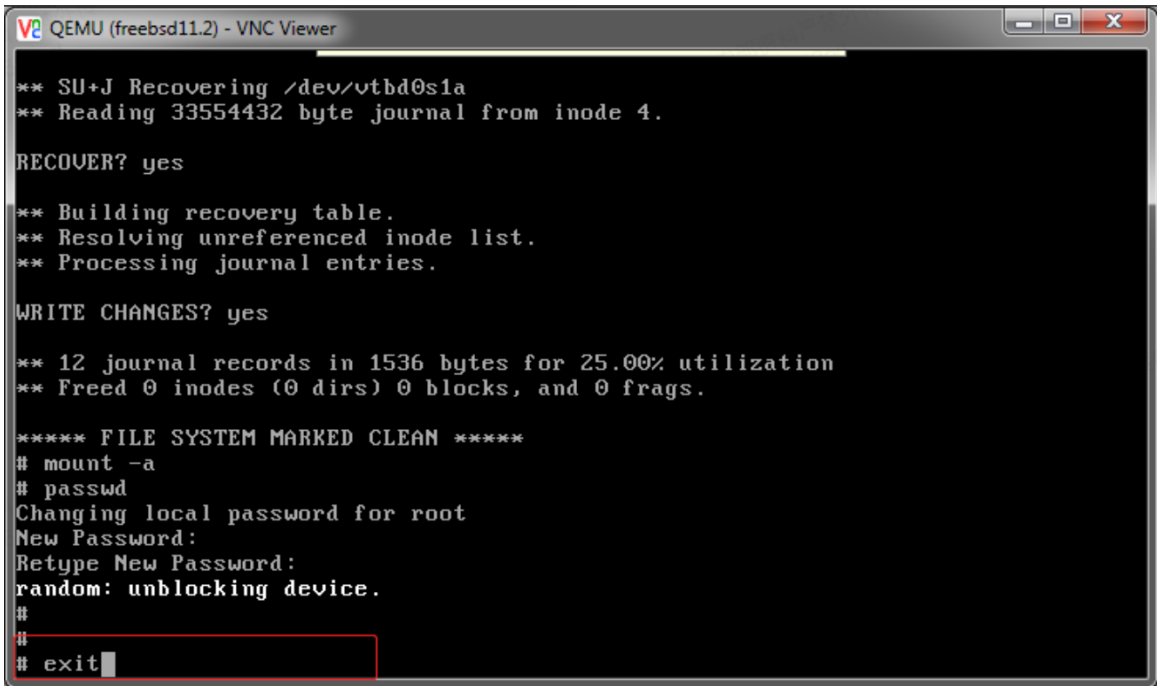
在光标处输入依次输入以下命令：

- fsck -y
- mount -a
- passwd



4. 修改密码完成后，在命令行输入：exit （退出单用户模式，直接进入系统启动流程）/或者输入 init 6（重启系统）

然后就可以使用新的密码登录系统了。



性能测试

Linux-磁盘IO性能测试

本文主要介绍BCC Linux系统中IO性能测试工具的使用，以Centos7.6为例，主要说明如下（本次测试的参数结果仅作为参考，具体还需结合服务器自身业务情况综合判断）。

1. **衡量指标**

一般使用以下指标衡量云磁盘的性能：

- IOPS：每秒读/写次数，单位为次（计数）。存储设备的底层驱动类型决定了不同的 IOPS。
- 吞吐量：每秒的读写数据量，单位为MB/s。
- 时延：I/O 操作的发送时间到接收确认所经过的时间，单位为秒。

2. **测试工具**

FIO 是测试磁盘性能的工具，用来对硬件进行压力测试和验证，本文以 FIO 为例。

> 警告：
>
> - 请不要在系统盘上进行 FIO 测试，避免损坏系统重要文件。
> - 为避免底层文件系统元数据损坏导致数据损坏，请不要在业务数据盘上进行测试。
> - 请确保/etc/fstab文件配置项中没有被测硬盘的挂载配置，否则将导致云服务器启动失败。

3. **工具安装**

本文以 CentOS 7.6 操作系统的云服务器为例，可以直接yum安装测试工具fio和libaio。

```
 $ yum install libaio -y
$ yum install libaio-devel -y
$ yum install fio -y
```

4. **测试示例**

1）参数说明：

参数名	说明	取值样例
bs	每次请求的块大小。取值包括4k、8k及16k等。	4k
ioengine	I/O 引擎。推荐使用 Linux 的异步 I/O 引擎。	libaio
iodepth	请求的 I/O 队列深度。	1
direct	指定 direct 模式。 • True（1）表示指定 O_DIRECT 标识符，忽略 I/O 缓存，数据直写。 • False（0）表示不指定 O_DIRECT 标识符。 默认为 True（1）。	1
rw	读写模式。取值包括顺序读（read）、顺序写（write）、随机读（randread）、随机写（randwrite）、混合随机读写（randrw）和混合顺序读写（rw，readwrite）。	read
time_based	指定采用时间模式。无需设置该参数值，只要 FIO 基于时间来运行。	N/A
runtime	指定测试时长，即 FIO 运行时长，以秒为单位。	600
refill_buffers	FIO 将在每次提交时重新填充 I/O 缓冲区。默认设置是仅在初始时填充并重用该数据。	N/A
norandommap	在进行随机 I/O 时，FIO 将覆盖文件的每个块。若给出此参数，则将选择新的偏移量而不查看 I/O 历史记录。	N/A
randrepeat	随机序列是否可重复，True（1）表示随机序列可重复，False（0）表示随机序列不可重复。默认为 True（1）。	0
group_reporting	多个 job 并发时，打印整个 group 的统计值。	N/A
name	job 的名称。	fio-read
size	I/O 测试的寻址空间。	100GB
filename	测试对象，即待测试的磁盘设备名称。	/dev/sdb

2）测试场景

- bs = 4k iodepth = 1：随机读/写测试，能反映硬盘的时延性能（filename需要指定设备中的具体磁盘盘符，如下以/dev/vda1为例）
  执行以下命令，测试硬盘的随机读时延。

```
$ fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randread -time_based -runtime=600  -randrepeat=0 -name=fio-randread-lat --size=10G -filename=/dev/vda1
```

- 执行以下命令，测试硬盘的随机写时延。

```
$ fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randwrite -time_based -runtime=600 -randrepeat=0 -name=fio-randwrite-lat --size=10G -filename=/dev/vda1
```

- 执行以下命令，测试随机混合读写时延性能。

```
$ fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randrw -time_based --runtime=600 -randrepeat=0 -name=fio-read --size=1G --filename=/dev/vda1
```

Linux-CPU性能测试

本文主要介绍BCC Linux系统中CPU性能测试工具的使用，以Centos7.6为例，主要说明如下（本次测试的参数结果仅作为参

考，具体还需结合服务器自身业务情况综合判断）。

1. **衡量指标**

   一般使用以下指标衡量CPU的性能：

   - Dhrystone 2 using register variables
   - Double-Precision Whetstone

2. **测试工具**

UnixBench是一款开源的测试 Unix 系统基本性能的工具，是比较通用的测试VPS性能的工具，该工具会测试系统各个方面一系列的性能，然后将每个测试结果和一个基准值进行比较,得到一个索引值，所有测试项目的索引值结合在一起形成一个测试分数值，测试结果不仅仅取决于硬件，也取决于系统、开发库、甚至是编译器，测试执行完大约需要10-30分钟。

同时，UnixBench也支持多CPU系统的测试，默认的行为是测试两次，第一次是单个进程的测试，第二次是多个进程的测试。

3. **工具安装**

1）下载软件包，解压后进入unixbench目录，以unixbench-5.1.3为例

```
 $ wget https://soft.laozuo.org/scripts/UnixBench5.1.3.tgz
$ tar -zxvf UnixBench5.1.3.tgz
$ cd UnixBench
```

2）make编译

```
 $ make
```

如果在make过程中出现报错，可以依照如下的参考，安装依赖包。

```
 ///1.make: gcc: Command not found
make: *** [pgms/arithoh] Error 127
如果出现gcc报错，我们需要安装gcc依赖包
///
$ yum install gcc
///2.Error: Please install /usr/bin/time提示错误，解决方法如下
$ yum install time
///3.如果出现bash: make: command not found问题
$ yum -y install gcc automake autoconf libtool make
```

3）执行./Run开始跑分

```
 $ ./Run
```

如果在run过程中出现报错，可以依照如下的参考，安装依赖包。

```
 ///3.Can't locate Time/HiRes.pm
有的时候在运行./Run时候会出现这样的错误
Can't locate Time/HiRes.pm in @INC (@INC contains: /usr/local/lib64/perl5 /usr/local/share/perl5
/usr/lib64/perl5/vendor_perl /usr/share/perl5/vendor_perl /usr/lib64/perl5 /usr/share/perl5 .) at ./Run line 6.
BEGIN failed--compilation aborted at ./Run line 6.
解决方案也简单，只需要安装组件
///
yum install perl-Time-HiRes
//运行安装之后我们再运行./Run就可以。
```

4. **参数说明**

测试过程中每个项目后面会有1 2 3 4 5 6 7 8 9 10 数字，意思是进行了10组测试，测试过程中部分内容及解释如下：

测试项目	项目说明
Dhrystone 2 using register variables	测试 string handling，该测试侧重字符串处理，没有浮点运算，结果受硬件设计和软件优化影响大
Double-Precision Whetstone	测试浮点数操作的速度和效率
Execl Throughput	此测试考察每秒钟可以执行的 execl 系统调用的次数
File Copy 1024 bufsize 2000 maxblocks	测试从一个文件向另外一个文件传输数据的速率
File Copy 256 bufsize 500 maxblocks	测试从一个文件向另外一个文件传输数据的速率
File Read 4096 bufsize 8000 maxblocks	测试从一个文件向另外一个文件传输数据的速率
Pipe-based Context Switching	测试两个进程（每秒钟）通过一个管道交换一个不断增长的整数的次数
Pipe Throughput	一秒钟内一个进程可以向一个管道写 512 字节数据然后再读回的次数
Process Creation	测试每秒钟一个进程可以创建子进程然后收回子进程的次数（子进程一定立即退出）
Shell Scripts (8 concurrent)	测试一秒钟内一个进程可以并发地开始一个shell 脚本的 n 个拷贝的次数，n 一般取值 1，2，4，8
System Call Overhead	测试进入和离开操作系统内核的代价，即一次系统调用的代价

Linux-网络性能测试

本文主要介绍BCC Linux系统中网络性能测试工具的使用，以Centos7.6为例，主要说明如下（本次测试的参数结果仅作为参考，具体还需结合服务器自身业务情况综合判断）。

1. **衡量指标**
   一般使用以下指标查看网络带宽的性能：
   - Bandwidth
   - Retr/Retransmits
2. **测试工具**
   iperf是一个网络性能测试工具，可以测试TCP和UDP带宽质量，可以测量最大TCP带宽，具有多种参数和UDP特性，可以报告带宽，延迟抖动和数据包丢失。主要功能如下：
   TCP
   - 测量网络带宽
   - 报告MSS/MTU值的大小和观测值
   - 支持TCP窗口值通过套接字缓冲
   - 当P线程或Win32线程可用时，支持多线程。客户端与服务端支持同时多重连接

   UDP
   - 客户端可以创建指定带宽的UDP流
   - 测量丢包
   - 测量延迟
   - 支持多播
   - 当P线程可用时，支持多线程。客户端与服务端支持同时多重连接（不支持Windows）
3. **工具安装**

1）下载软件包，解压后进入iperf目录，以iperf-3.1.3为例。

```
 $ wget https://iperf.fr/download/source/iperf-3.1.3-source.tar.gz
$ tar xzvf iperf-3.1.3-source.tar.gz
$ cd iperf-3.1.3/
```

2）make编译。

```
 $ ./configure
$ make && make install
```

如果在make过程中出现报错，可以依照如下的参考，手动同步依赖包。

```
 //报错信息：iperf3: error while loading shared libraries: libiperf.so.0: cannot open shared object file: No such file or directory
$ /sbin/ldconfig
```

4. **参数说明**

在完成iperf安装后，执行“iperf3 –h”即可显示iperf的详细用法，iperf的命令行选项共分为三类，分别是客户端与服务器端共用选项、服务器端专用选项和客户端专用选项，下面对常用的选项进行介绍。

-s 以server模式启动，eg：iperf-s

-chost以client模式启动，host是server端地址，eg：iperf -c 222.35.11.23

1）**通用参数**

-f [kmKM] 分别表示以Kbits,Mbits, KBytes, MBytes显示报告，默认以Mbits为单位,eg：iperf -c 222.35.11.23 -fK

-i sec以秒为单位显示报告间隔，eg：iperf -c 222.35.11.23 -i 2

-l 缓冲区大小，默认是8KB,eg：iperf-c 222.35.11.23 -l 16

-m 显示tcp最大mtu值

-o 将报告和错误信息输出到文件eg：iperf-c 222.35.11.23 -o ciperflog.txt

-p 指定服务器端使用的端口或客户端所连接的端口eg：iperf -s -p 9999;iperf -c 222.35.11.23 -p9999

-u 使用udp协议

-w 指定TCP窗口大小，默认是8KB

-B 绑定一个主机地址或接口（当主机有多个地址或接口时使用该参数）

-C 兼容旧版本（当server端和client端版本不一样时使用）

-M 设定TCP数据包的最大mtu值

-N 设定TCP不延时

-V 传输ipv6数据包

2）**server专用参数**

-D 以服务方式运行iperf，eg：iperf-s -D

-R 停止iperf服务，针对-D，eg：iperf -s -R

3）**client端专用参数**

-d 同时进行双向传输测试

-n 指定传输的字节数，eg：iperf -c222.35.11.23 -n 100000

-r 单独进行双向传输测试

-t 测试时间，默认10秒,eg：iperf -c222.35.11.23 -t 5

-F 指定需要传输的文件

-T 指定ttl值

5. **工具使用**

以两台服务器为例，进行TCP测试，具体信息如下：

Server端：192.168.44.9

Client端：192.168.44.12

1）Server端执行：iperf -s

```
 $ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.44.12, port 60228
[  5] local 192.168.44.9 port 5201 connected to 192.168.44.12 port 60230
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec   172 MBytes  1.44 Gbits/sec
[  5]   1.00-2.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   2.00-3.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   3.00-4.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   4.00-5.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   5.00-6.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   6.00-7.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   7.00-8.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   8.00-9.00   sec   179 MBytes  1.50 Gbits/sec
[  5]   9.00-10.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  10.00-11.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  11.00-12.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  12.00-13.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  13.00-14.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  14.00-15.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  15.00-16.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  16.00-17.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  17.00-18.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  18.00-19.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  19.00-20.00  sec   179 MBytes  1.50 Gbits/sec
[  5]  20.00-20.06  sec  9.85 MBytes  1.50 Gbits/sec
-----------------------------------------------------------
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-20.06  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-20.06  sec  3.50 GBytes  1.50 Gbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

2）client端执行：iperf -c 192.168.44.9 -t 20 -i 2 -d -m

```
 # iperf3 -c 192.168.44.9 -t 20 -i 2
Connecting to host 192.168.44.9, port 5201
[ 4] local 192.168.44.12 port 60230 connected to 192.168.44.9 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[ 4]   0.00-2.00   sec   362 MBytes  1.52 Gbits/sec    0   2.96 MBytes
[ 4]   2.00-4.00   sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]   4.00-6.00   sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]   6.00-8.00   sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]   8.00-10.00  sec   358 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]  10.00-12.00  sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]  12.00-14.00  sec   359 MBytes  1.50 Gbits/sec   81   3.01 MBytes
[ 4]  14.00-16.00  sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]  16.00-18.00  sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
[ 4]  18.00-20.00  sec   359 MBytes  1.50 Gbits/sec    0   3.01 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[ 4]   0.00-20.00  sec  3.51 GBytes  1.51 Gbits/sec   81             sender
[ 4]   0.00-20.00  sec  3.50 GBytes  1.51 Gbits/sec                  receiver

iperf Done.
```

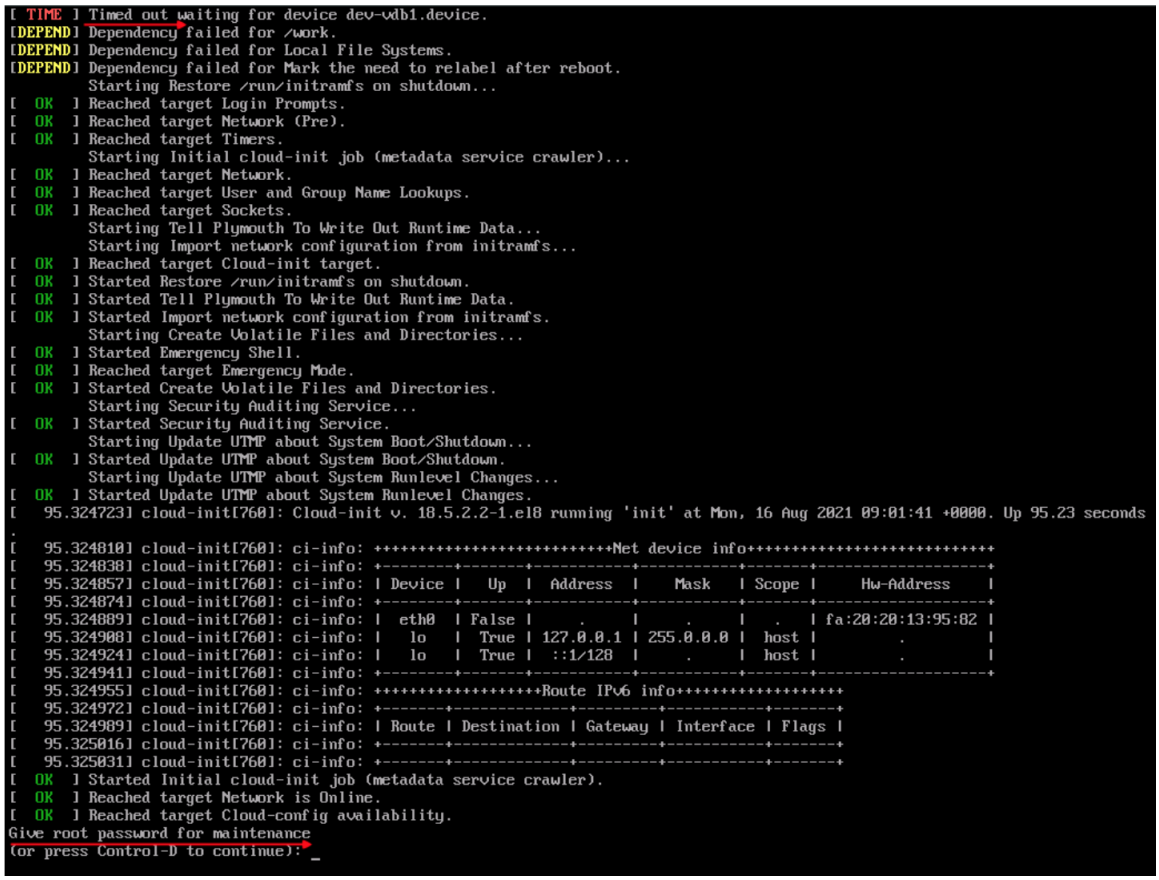
可以从测试结果中看到测试过程中的传输数据量以及带宽信息，关于UDP的测试以及调整测试参数，您可以依据自身测试需求进行更改。

fstab配置导致系统启动异常如何处理

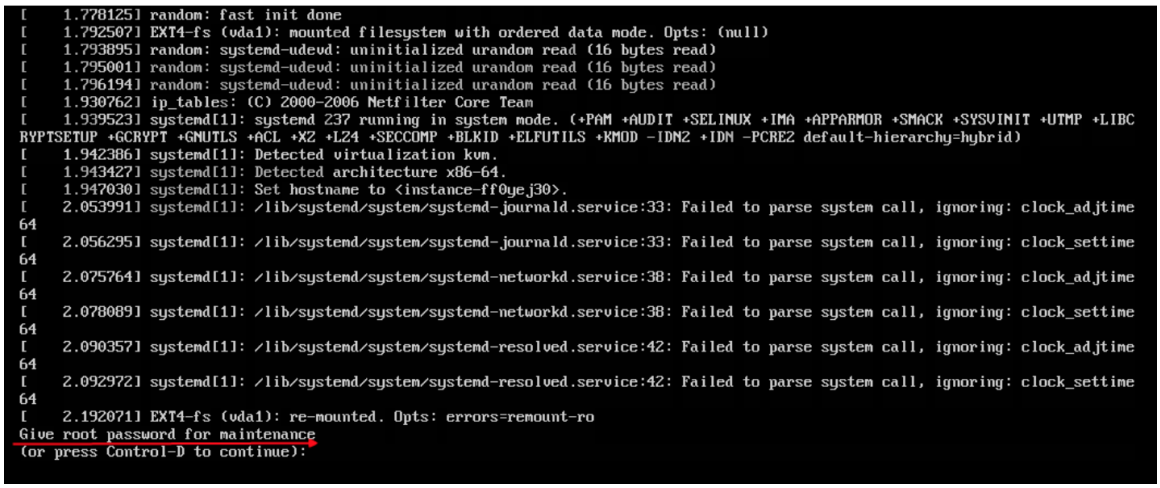
问题描述

使用远程连接软件SSH无法登录BCC Linux实例，登录百度智能云控制台使用VNC连接实例，发现操作系统进入维护模式。

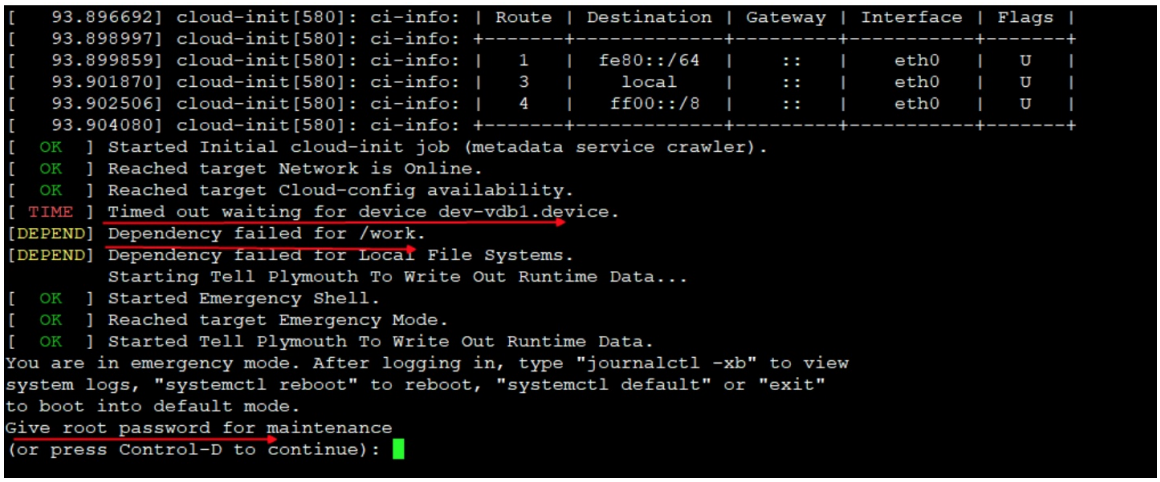
1. CentOS 7 / 8 错误信息（VNC）



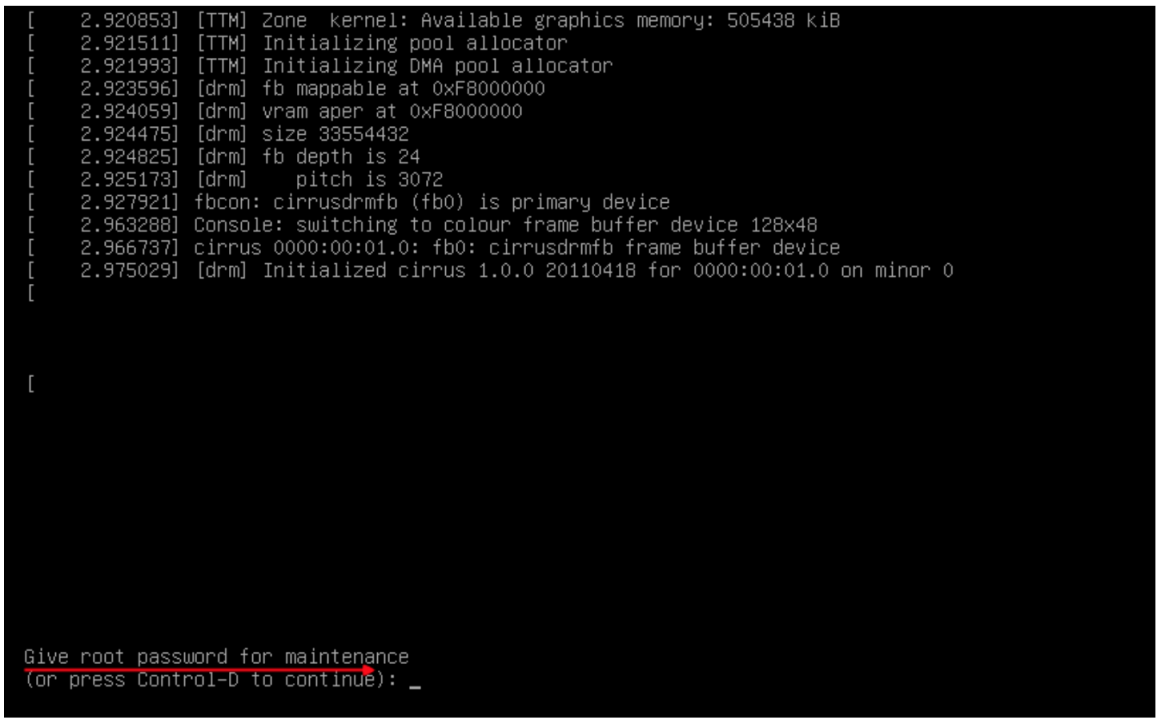
2. ubuntu 18.04错误信息（VNC）



3. ubuntu 18.04 错误信息（console.log）



4. debian 10.8 错误信息（VNC）



问题原因

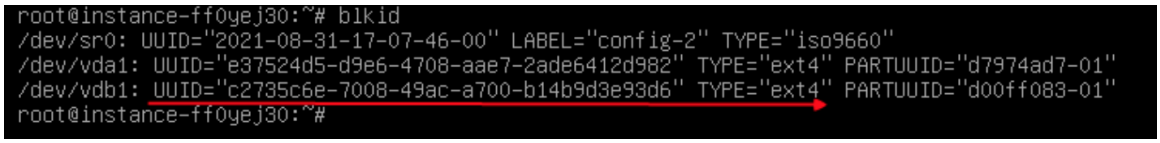
一般是因为/etc/fstab文件中磁盘挂载了错误的文件系统，或者磁盘错误造成的。

解决方法

通过VNC连接BCC实例，当出现Give root password for mintenance时，输入BCC实例的root密码，进入维护模式。

在终端上输入以下命令，以读写方式重新挂载根分区：

mount / -o remount,rw 执行blkid命令，查看BCC的磁盘的分区和文件系统，如图所示，数据盘为/dev/vdb1，文件系统类型为ext4。



查看/etc/fstab内容，检查其写入的磁盘信息是否与上图所示的一致。

如果一致，则查看挂载参数是否有nofail，如果没有建议添加该参数【Linux实例启动过程中，若设备不存在会直接忽略它，从而不影响系统的启动。

如果不一致，修改/etc/fstab，写入正确的挂载信息。

```
 UUID=c2735c6e-7008-49ac-a700-b14b9d3e93d6 /work         ext4      defaults,nofail    0    0
或
/dev/vdb1                                 /work         ext4      defaults,nofail    0    0
```

Linux镜像如何开启或关闭intelCPU漏洞补丁

概述

本文主要介绍Linux镜像如何开启或关闭Meltdown与Spectre、MDS等漏洞修复的方法。

Meltdown与Spectre

漏洞详情

Meltdown（CVE-2017-5754）与Spectre（CVE-2017-5715/CVE-2017-5753）存在于英特尔（Intel）处理器芯片的安全漏洞，该漏洞源于芯片硬件层面的设计缺陷，可导致操作系统内核信息泄露、应用程序越权访问系统内核数据等问题。

漏洞编号：

CVE-2017-5753

CVE-2017-5715

CVE-2017-5754

Linux内核社区已经针对上述问题提出了缓解措施，但部分缓解措施存在副作用，会导致系统的运行效率有较大幅度下降，目前百度智能云在GuestOS中默认关闭Meltdown与Spectre漏洞的修复，以保证GuestOS的性能。

漏洞名称	官方补丁	百度云镜像默认处理方式	关闭漏洞修复的方法
Meltdown	centos、ubuntu、debian、opnesuse等社区已发布内核补丁	关闭漏洞修复	添加内核参数：nopti
Spectre Variant 2		关闭漏洞修复	添加内核参数：nospectre_v2
Spectre Variant 4（Speculative Store Bypass）		关闭漏洞修复	添加内核参数：nospec_store_bypass_disable

SRSO

漏洞详情

SRSO（CVE-2023-20569）是一个存在于AMD处理器芯片的安全漏洞。该漏洞与AMD处理器的推测执行机制、返回栈缓冲区溢出相关，可能会导致数据泄露、系统破坏等潜在安全风险。

当前许多最新版本的操作系统内核已经默认合入了AMD SRSO漏洞缓解措施SafeRET，并且默认为开启状态。然而，这些缓解措施会导致处理器性能有较为明显的下降。

根据AMD官方发布的针对公有云用户的SRSO漏洞缓解措施的选项与考虑因素文档，AMD认为，在典型的生产环境中，与SRSO相关的整体安全风险较低。对于使用常见侧信道道保护的典型生产环境中的云用户，AMD认为在配置操作系统时禁用SafeRET是合理的。

目前，百度智能云在公共镜像中默认关闭了SRSO漏洞的缓解措施(内核启动参数添加spec_rstack_overflow=off)，以保证GuestOS的性能。用户可以根据自身业务安全需求，决定是否使用安全缓解措施(在内核启动参数中移除spec_rstack_overflow=off)。

MDS

漏洞详情

Intel “Micro Architectural Data Sampling (MDS)“，该漏洞可能通过侧信道攻击的方式，导致未经授权的一些敏感数据泄露。

漏洞编号

CVE编号	漏洞名称	严重程度	漏洞描述
CVE-2018-12126	Microarchitectural Store Buffer Data Sampling (MSBDS)	中危	某些处理器的Store buffers允许授权用户本地利用预测执行通过侧信道获取敏感信息。
CVE-2018-12127	Microarchitectural Load Port Data Sampling (MLPDS)	中危	某些处理器的Load ports允许授权用户本地利用预测执行通过侧信道获取敏感信息。
CVE-2018-12130	Microarchitectural Fill Buffer Data Sampling (MFBDS)	中危	某些处理器的Fill buffers允许授权用户本地利用预测执行通过侧信道获取敏感信息。
CVE-2019-11091	Microarchitectural Data Sampling UncacheableMemory (MDSUM)	低危	某些处理器的 Uncacheable memory允许授权用户本地利用预测执行通过侧信道获取敏感信息。

影响范围

Intel相应型号的CPU芯片，详情请参见Intel安全公告。

如何开启或者关闭MDS漏洞修复

目前Linux发行版社区针对MDS漏洞，已通过更新内核补丁来修复处理器漏洞。

以下几种镜像为涉及的公共镜像：

- centos 7 / 8
- debian 9 /10
- ubuntu 16.04 / 18.04 【 ubuntu-20.04内核不受影响】
- opensuse 15

漏洞的修复可能会带来比较明显的性能回退，如果用户对性能需求比较敏感，可通过关闭修复的方式来提升性能。

OS发行版	社区修复情况	如何关闭/开启mds漏洞修复	备注
centos	已修复，需更新最新内核	**开启：内核默认已开启修复。** **关闭：添加内核参数：mds=off tsx_async_abort=off**	百度云centos 6使用了未修复漏洞的内核。
ubuntu			
debian			
opensuse			

社区修复情况

OS	发行版	修复后的内核版本	社区
CentOS	6	kernel-2.6.32-754.14.2.el6	https://access.redhat.com/errata/RHSA-2019:1169
	7	kernel-3.10.0-957.12.2.el7	https://access.redhat.com/errata/RHSA-2019:1168
	8	kernel-4.18.0-80.1.2.el8_0	https://access.redhat.com/errata/RHSA-2019:1167
Ubuntu	16.04 ESM (Xenial Xerus)	Released (4.4.0-148.174)	
	18.04 LTS (Bionic Beaver)	Released (4.15.0-50.54)	
	20.04 LTS (Focal Fossa)	Not vulnerable (5.4.0-9.12)	
Debian			

Ubunut16.04订阅扩展安全维护（ESM）

2021年04月，Ubuntu 16.04 LTS（Xenial Xerus）将停止为期5年的标准支持维护，百度智能云也将停止Ubuntu 16.04 LTS公共镜像的更新。

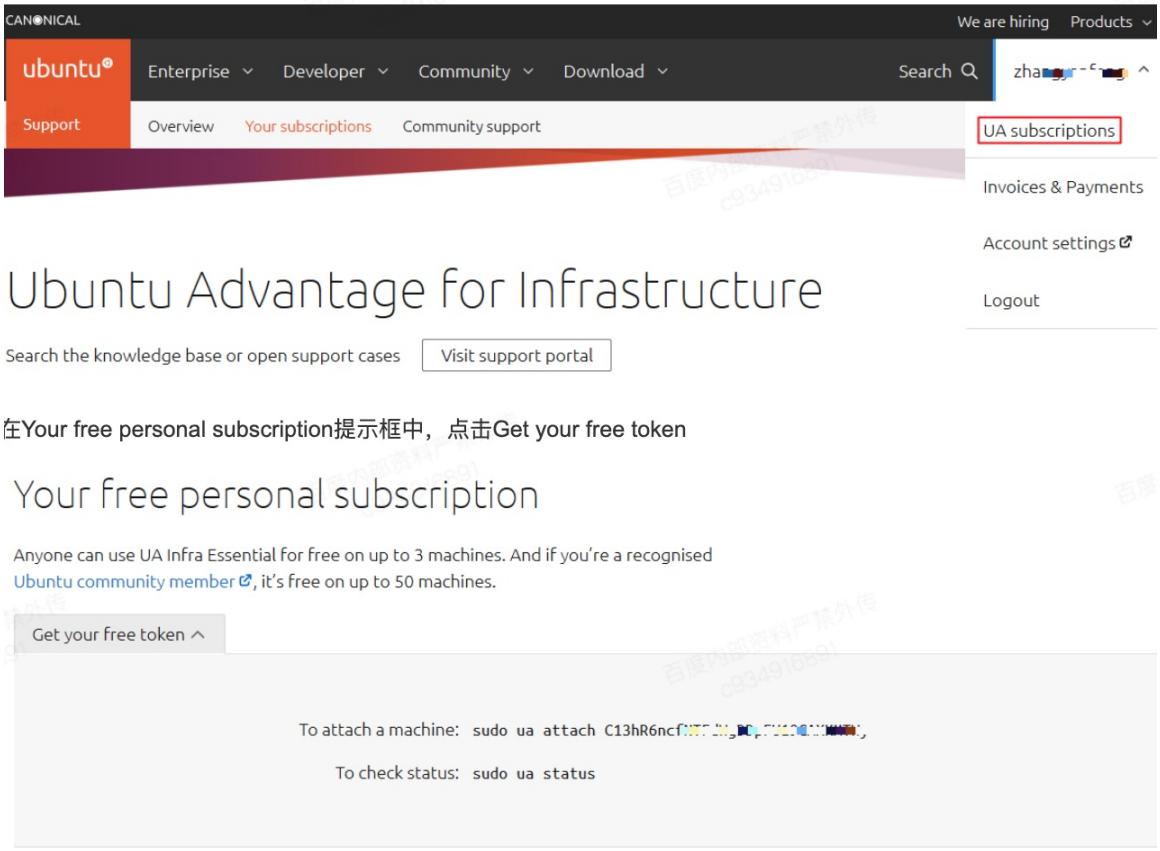
Ubuntu 16.04 LTS后续将过渡到扩展安全维护（ESM）的阶段。您可以参考本指导文档，通过Ubuntu订阅服务（UA-I）为Ubuntu 16.04实例获取ESM更新支持。

在Ubuntu官网订阅UA-I服务

1、使用可以访问外网的计算机，通过浏览器访问Ububtu官网。

2、在顶部菜单栏，单击Sign in，登录 的Ubuntu账号登录（如果您没有Ubuntu账号，需要进行注册并登录Ubuntu官网）。

3、找到并点击Ubuntu账号的用户名，然后再点击【UA subscriptions】。



在Your free personal subscription提示框中，点击Get your free token

在Ubuntu 16.04实例中添加UA-I并进行安全更新

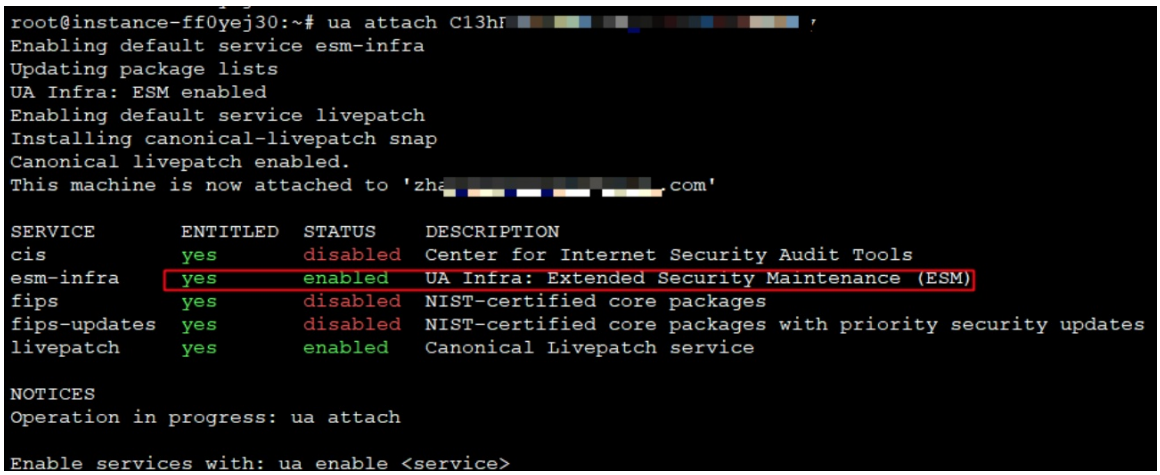
1、远程连接您的Ubuntu 16.04 BCC实例，安装Ubuntu最新的Ubuntu Advantage（UA）客户端。

2、升级软件包。

3、apt-get update 安装UA客户端。

4、apt-get install ubuntu-advantage-tools 添加UA令牌(token)

5、ua attach 如图所示，表示已启用ESM。



云服务器删除或者修改文件时提示文件系统只读

问题描述

Linux BCC 云服务器删除或者修改文件时提示文件系统只读：Read-only file system，导致操作失败。

```
[root@instance-0wazpxeh mnt]# rm -f openssh-*
rm: cannot remove 'openssh-8.4p1-2.el7.x86_64.rpm': Read-only file system
rm: cannot remove 'openssh-clients-8.4p1-2.el7.x86_64.rpm': Read-only file system
rm: cannot remove 'openssh-server-8.4p1-2.el7.x86_64.rpm': Read-only file system
[root@instance-0wazpxeh mnt]#
```

出现文件系统Read-only的原因可能有：

- 文件系统错误导致文件系统进入只读模式。
- 文件系统是以只读方式进行的挂载。
- 磁盘故障，例如出现坏道等。

解决方法

执行命令mount | grep 挂载点，查看文件文件的挂载属性。

```
[root@instance-0wazpxeh ~]# mount | grep /mnt
/dev/vdb1 on /mnt type ext4 (ro,relatime,data=ordered)
[root@instance-0wazpxeh ~]#
```

如果挂载选项显示为ro，说明以只读方式挂载了文件系统，需要以可读写方式进行挂载。

mount -o remount,rw 挂载点

如果挂载选项显示为rw，需要确认是否为文件系统错误导致。

通过以下命令（二选一），查看内核日志中是否存在文件系统错误。

dmesg | egrep -i "ext3|ext4|xfs" cat /var/log/messages | egrep -i "ext3|ext4|xfs"

如果内核日志中出现了文件系统错误，则需要进行修复文件系统。

如果是系统盘的文件系统修复需要进入单用户模式，进行文件系统的修复。

如果是数据磁盘的文件系统出错，需要先把数据磁盘umount。

示例命令：umount /dev/vdb1 或 umount /mnt

修复文件系统命令如下：

ext系列文件系统执行以下命令。

fsck /dev/vdb1

xfs系列文件执行以下命令。

xfs_repair /dev/vdb1

> 注：修复文件系统，可能会产生数据丢失，请先备份数据后进行操作。
>
> 如果以上操作都不能解决Read-only问题， 需要检查磁盘是否存在物理故障。

LinuxBCC实例卡顿问题

当BCC实例的运行速度变慢或出现网络中断时，可能是BCC的CPU和网络的使用率过高导致。本文档将引导您针对使用率高的问题进行定位，并尝试自主处理相关问题，提高自运维效率。整体处理思路：

1. 找到影响CPU使用率或CPU负载过高的具体进程。
2. 排查影响CPU使用率或CPU负载过高的进程是否正常，并分类进行处理。
   - 正常进程：考虑进行实例规格升配，或调整、优化对应的进程/程序。
   - 异常进程：可以手动对进程进行查杀，或考虑使用第三方安全工具。

CPU使用率高问题定位

首先，您需要登录BCC实例。如果无法通过SSH或SmartTerm登录，请参考本产品文档中实例登录部分的相关文档，通过VNC等方式登录。

您可以考虑使用`top`命令来分析进程。

```
top - 16:54:50 up 10 min,  2 users,  load average: 0.00, 0.03, 0.04
Tasks: 167 total,   2 running, 165 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  5.0 sy, 92.4 ni,  1.0 id,  0.3 wa,  0.3 hi,  0.3 si,  0.0 st
MiB Mem :    971.0 total,    165.6 free,    221.7 used,    583.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    602.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1726 root      39  19  641360  94872  30324 R  96.3   9.5   0:03.66 dnf
   1524 root      20   0  995344  54024  27092 S   1.0   5.4   0:03.27 hosteye
    559 root      20   0       0      0      0 S   0.3   0.0   0:00.02 jbd2/vda1-8
    658 root      20   0  102188  12392  11356 S   0.3   1.2   0:00.27 systemd-journal
   1725 root      20   0   64528   4440   3768 R   0.3   0.4   0:00.01 top
      1 root      20   0  176868  10788   8224 S   0.0   1.1   0:02.14 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.08 ksoftirqd/0
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.04 rcu_sched
     12 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
     13 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 watchdog/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
     17 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
     18 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kauditd
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 khungtaskd
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
     21 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 writeback
```

返回结果说明：

上方，静态展示：

第一行：系统当前的时间，包含该BCC实例已运行时间、登录用户情况，以及最近1分钟、最近5分钟和最近15分钟的CPU平均负载。
第二行：当前系统中任务运行情况。
第三行：CPU资源总体使用情况。
第四行：内存资源总体使用情况。

下方，动态显示：各进程的资源占用情况。可以通过以下命令可以查看每个进程ID对应的程序文件：

`ls -l /proc/PID/exe`

```
[root@instance-0wazpxeh ~]# ll /proc/1524/exe
lrwxrwxrwx 1 root root 0 Sep  9 16:50 /proc/1524/exe -> /opt/hosteye/bin/hosteye
[root@instance-0wazpxeh ~]#
```

CPU使用率高问题处理

异常进程处理

- 对于CPU使用率高的具体进程，如果是异常进程，可以直接通过 `kill` 或 `top` 命令终止进程。

使用kill命令终止CPU使用率高的进程

`kill -9 PID`

使用top命令终止CPU使用用率高的进程

- 在top命令运行的同时，按下小写的“k”键，输入要终止进程的PID

常用进程处理

- 对于**kswapd0**进程导致的CPU使用率高的问题，需要对应用程序进行优化，或者通过增加内存进行系统规格的升级。kswapd0是系统的虚拟内存管理程序，如果物理内存不够用，系统就会唤醒kswapd0进程，由kswapd0分配磁盘交换空间用作缓存，因而占用大量的CPU资源。

当您发现实例中出现kswapd0进程占用导致CPU使用率高的情况时，您可通过以下步骤排查进程的内存占用情况：

- 通过`top`命令查看kswapd0进程的资源使用情况。

  如果kswapd0进程持续处于非睡眠状态且运行时间较长，可以初步判定系统在持续的进行换页操作，需将问题转向内存不足的原因来排查。

- 通过vmstat命令进一步查看系统虚拟内存的使用情况。如果si和so的值也比较高，说明系统存在频繁的换页操作，系统物理内存不足。

  si：每秒从交换区写到内存的大小，由磁盘调入内存。
  so：每秒写入交换区的内存大小，由内存调入磁盘。

对于内存不足问题，可以通过free、ps等命令进一步查询系统及系统内进程的内存占用情况，做进一步排查分析，可通过在业务空闲期重启应用或者系统释放内存。

网络使用率高问题分析

当网络使用率过高时，可通过流量监控工具nethogs来实时监测统计各进程的网络使用情况，并进行问题定位。以centos 8 为例：

1. 安装 nethogs

`yum install -y nethogs`

2. 查看网卡的网络使用情况

`nethogs eth0 【网卡名为eth0为例】`

无法通过SSH方式登录LinuxBCC实例怎么办

使用 SSH 登录 Linux 实例 时，提示无法连接或者连接失败，导致无法正常登录 Linux 实例。

根据返回的错误信息，您可记录报错信息

User root not allowed because not listed in AllowUsers

问题原因

该问题通常是由于 SSH 服务启用了用户登录控制参数，对登录用户进行了限制。参数说明如下：

AllowUsers：允许登录的用户白名单，只有该参数标注的用户可以登录。

DenyUsers：拒绝登录的用户黑名单，该参数标注的用户都被拒绝登录。

AllowGroups：允许登录的用户组白名单，只有该参数标注的用户组可以登录。

DenyGroups：拒绝登录的用户组黑名单，该参数标注的用户组都被拒绝登录。

> 注：拒绝策略优先级高于允许策略。

解决方法

1. 通过VNC登录BCC实例，编辑/etc/ssh/sshd_config，注释或删除以下内容：

AllowUsers username

DenyUsers username

DenyGroups groupname

AllowGroups groupname

重启sshd服务，以centos 7 为例：

systemctl restart sshd

2. Disconnected:No supported authentication methods available 问题现象：

`Permission denied (publickey,gssapi-keyex,gssapi-with-mic)`

问题原因

sshd_config配置文件中禁用了密码验证登录。

解决方法

编辑/etc/ssh/sshd_config，修改PasswordAuthentication no 为 PasswordAuthentication yes 重启sshd服务

ssh_exchange_identification: read: Connection reset by peer 使用 SSH 登录时，出现以下等错误信息

"ssh_exchange_identification: Connection closed by remote host"

"kex_exchange_identification: read: Connection reset by peer"

"kex_exchange_identification: Connection closed by remote host"

问题原因

访问控制限制了连接

更改了防火墙规则

sshd 配置中最大连接数限制

网络存在问题

解决方法

检查及调整访问策略设置。

以CentOS 7 为例，可通过 /etc/hosts.allow 和 /etc/hosts.deny 文件设置访问策略，分别对应允许和阻止的策略。

例如，在hosts.allow 文件中设置信任主机规则，在hosts.deny 文件中设置拒绝主机规则。

阻止策略如下【示例】

```
 sshd:180.76.100.35:deny  #表示拒绝180.76.100.35这个ip的sshd请求！

sshd:all:deny                  #表示拒绝了所有sshd远程连接。

注 :deny可以省略。
```

如果配置了阻止策略，请根据需要删除相应的设置。

检查及调整 sshd 配置。

检查 MaxStartups 值是否需调整。sshd_config 配置文件中通过 MaxStartups 设置允许的最大连接数，如果短时间需建立较多连接，则需适当调整该值。

MaxStartups 10:30:100为默认配置，指定 SSH 守护进程未经身份验证的并发连接的最大数量。10:30:100表示从第10个连接开始，以30%的概率（递增）拒绝新的连接，直到连接数达到100。

检查 iptables 防火墙规则

运行以下命令，查看查看防火墙是否阻止 SSH 连接。

iptables -L --line-number 如果SSH被阻止，请更新防火墙规则，取消对SSH的阻止。

Permission denied, please try again

问题原因

sshd_config配置文件中配置了：PermitRootLogin no

解决方法

编辑/etc/ssh/sshd_config，将 PermitRootLogin no 修改为 PermitRootLogin yes。

重启sshd服务

Too many authentication failures

问题原因

多次连续输入错误密码后，触发了 SSH 服务密码重置策略。

解决方法

检查/etc/ssh/sshd_config配置文件，查看是否开设置了【MaxAuthTries】参数。

如果出现以下设置

MaxAuthTries 6 【SSH登录最大密码尝试失败次数6】 编辑/etc/ssh/sshd_config，删除或注释掉MaxAuthTries 设置，然后重启sshd服务。

配置Linux 分析工具atop和kdump

atop

atop是一个功能非常强大的linux服务器监控工具。

它以一定的频率记录系统的运行状态，所采集的数据包含系统资源（CPU、内存、磁盘、网络）的使用情况和进程运行情况。

并以日志文件的方式保存在磁盘中，服务器出现问题后，我们可以获取相应的atop日志文件进行分析。

安装atop

以centos 7 / 8 为例，运行以下命令：

yum install -y atop

atop配置

atop配置文件为/etc/sysconfig/atop，

参数说明：

LOGINTERVAL：监控周期，单位为秒，默认600s采集一次数据，建议修改为15s。

LOGGENERATIONS：日志保留时间，单位为天。默认保留时间为28天，建议修改为7天，避免大量占用磁盘空间。

LOGPATH：日志保存路径。默认路径为/var/log/atop/ 你可根据实际情况，修改监控周期和日志保留时间。

启动和停止atop

atop安装完成后，默认为停止状态，可执行以下命令启动atop：

systemctl start atop 不建议在生产环境中长时间运行 atop，您可在排查问题完成后停止 atop，命令如下：

systemctl stop atop

分析atop

atop启动后，会将采集到的记录存放在/var/log/atop目录中，通过执行如下命令，查看日志文件。

atop -r /var/log/atop/atop_20210910 atop常用指令如下所示。

c：按照进程CPU使用率进行降序筛选。

m：按照进程内存使用率进行降序筛选。

d：按照进程磁盘使用率进行降序筛选。

a：按照进程资源综合使用率进行降序筛选。

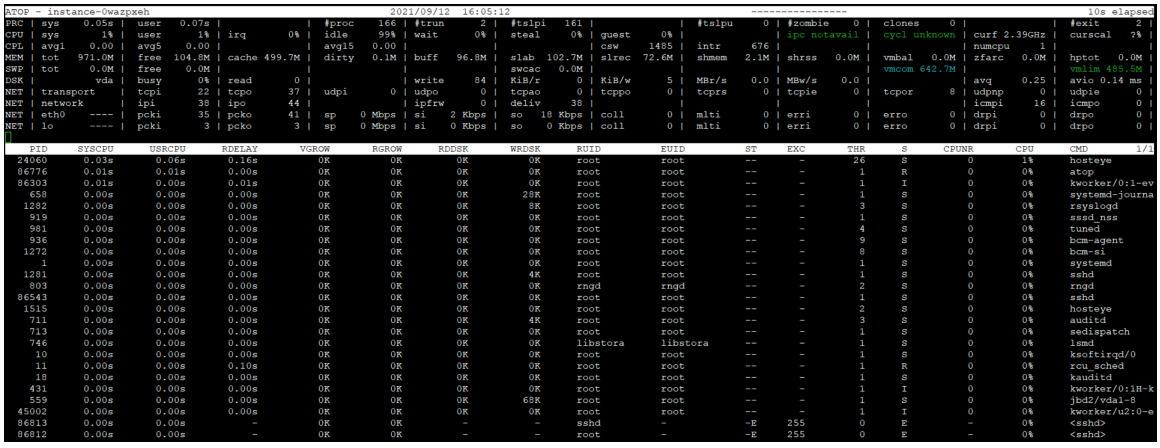
n：按照进程网络使用率进行降序筛选，需要额外安装内核模块才支持，默认不支持。

t：跳转到下一个监控采集点。

T：跳转到上一个监控采集点。

b：指定时间点。

系统资源监控字段说明



ATOP 行：主机名、采样日期和时间点。

PRC 行：进程整体运行情况。

sys 及 user：进程在内核态及用户态所占 CPU 的时间比例。

- proc：进程总数。
- zombie：僵死进程的数量。
- exit：atop 采样周期期间退出的进程数量。

CPU 行：CPU 整体（即多核 CPU 作为一个整体 CPU 资源）的使用情况。【各字段数值相 加结果为 N00%，N 为 CPU的核数】

sys 及 user：CPU 被用于处理进程时，进程在内核态、用户态所占 CPU 的时间比例。

irq：CPU 被用于处理中断的时间比例。

idle：CPU 处在完全空闲状态的时间比例。

wait：CPU 处在“进程等待磁盘 IO 导致 CPU 空闲”状态的时间比例。

CPL 行：CPU 负载情况。

avg1、avg5 和 avg15：过去1分钟、5分钟和15分钟内运行队列中的平均进程数量。

csw：指示上下文交换次数。

intr：指示中断发生次数。

MEM 行：内存的使用情况。

tot：物理内存总量。

cache ：用于页缓存的内存大小。

buff：用于文件缓存的内存大小。

slab：系统内核占用的内存大小。

SWP 行：交换空间的使用情况。

tot：交换区总量。

free：空闲交换空间大小。

PAG 行：虚拟内存分页情况

swin 及 swout：换入和换出内存页数。

DSK 行：磁盘使用情况，每一个磁盘设备对应一列。如果有 sdb 设备，那么增加一行 DSK 信息。

sda：磁盘设备标识。

busy：磁盘忙时比例。

read 及 write：读、写请求数量。

NET 行：多列 NET 展示了网络状况，包括传输层（TCP 和 UDP）、IP 层以及各活动的网口信息。

XXXi：各层或活动网口收包数目。

XXXo：各层或活动网口发包数目。

- kdump

kdump 是一种先进的基于 kexec 的内核崩溃转储机制。用来捕获kernel crash（内核崩溃）的时候产生的crash dump。当内核产生错误（系统崩溃、死锁或者死机）时，kdump会将内存导出为vmcore保存到磁盘。

配置kdump

以centos 7 / 8 为例：

安装kexec-tools

查看是否已安装kexec-tools

rpm -qa | grep kexec-tools 如果没有安装，执行下面命令安装kexec-tools。

yum install -y kexec-tools

开启kdump开机启动

systemctl enable kdump

设置craskkernel参数

首先查看该参数是否已经设置。

cat /proc/cmdline | grep crashkernel 如果有显示，则表示已经设置，如果没有显示，则需要重新设置。 编辑/etc/default/grub文件

```
 GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,115200 nopti nospectre_v2 nospec_store_bypass_disable"
GRUB_DISABLE_RECOVERY="true"
```

在GRUB_CMDLINE_LINUX一行添加crashkernel=auto。

更新grub

执行更新grub命令，使配置生效：

grub2-mkconfig -o /boot/grub2/grub.cfg

设置vmcore保存路径

vmcore默认保存在/var/crash目录下，如果要保存到其他目录，可编辑/etc/kdump.conf，修改path一行，将其修改成对应的目录。

```
 path  vmcore_directory

###### 请确保指定的路径有足够的空间保存vmcore，建议剩余空间不小于物理内存（RAM）的大小。
```

设置转存vmcore级别

查看/etc/kdump.conf文件，是否存在以下设置，如果存在则无需添加

`core_collector makedumpfile -d 31 -c`

-c：表示压缩vmcore文件，

-d：表示过滤掉部分无效的内存数据，可以根据需要调整，一般31即可，31是由如下的值与计算而成。

```
 zero pages   = 1
cache pages   = 2
cache private = 4
user  pages   = 8
free  pages   = 16
```

设置内核参数

编辑/etc/sysctl.conf，添加以下参数。

```
 kernel.hardlockup_panic=1
kernel.panic=5
kernel.panic_on_oops=1
kernel.softlockup_panic=1
kernel.unknown_nmi_panic=1
kernel.nmi_watchdog=1
---以下为可选参数---
kernel.panic_on_io_nmi=1
kernel.panic_on_warn=1
```

重启系统

reboot

参考：https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/kernel_administration_guide/kernel_crash_dump_guide?spm=a2c4g.11186623.0.0.16724472WdlEBD

Linux格式化与挂载实例挂载本地磁盘

背景

本文介绍如何将实例中的一块本地盘进行格式化并挂载文件目录。

- 适用镜像：百度智能云CentOS、Ubuntu系列公共镜像。
- 适用实例规格：支持本地盘的实例规格，例如bcc.l5系列。

操作步骤

本文将以在公共镜像 Linux Centos 8u4中格式化并挂载设备名为nvme0n1的本地盘进行演示：

1、登陆BCC控制台，远程连接云服务器操作系统，通过如下密令下载分区工具。

```
yum install -y parted
```

如您是Ubuntu系统，则通过如下命令下载。

```
apt-get install parted
```

2、通过如下命令获取数据盘的设备名称。

```
fdisk -l
```

示例结果表示当前BCC实例有4块本地数据盘：/dev/nvme0n1- /dev/nvme3n1，和系统盘CDS云磁盘/dev/vda1，/dev/vda2

```
[root@instance-brz8myc4 ~]# fdisk -l
Disk /dev/vda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: E93874C8-6799-4C8C-8E80-D6AB70DA4D7C

Device      Start      End  Sectors  Size Type
/dev/vda1    2048   411647   409600  200M EFI System
/dev/vda2  411648 41943006 41531359 19.8G Linux filesystem


Disk /dev/nvme0n1: 3.5 TiB, 3840755982336 bytes, 937684566 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/nvme2n1: 3.5 TiB, 3840755982336 bytes, 937684566 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/nvme1n1: 3.5 TiB, 3840755982336 bytes, 937684566 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/nvme3n1: 3.5 TiB, 3840755982336 bytes, 937684566 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

3、通过如下命令对数据盘进行分区。

```
parted /dev/nvme0n1
```

根据提示，依次操作如下。

4、通过如下命令切换为 GPT 分区格式。

```
mklabel gpt
```

5、通过如下命令划分一个主分区，并设置其开始和结束位置。

```
mkpart primary 1 100%
```

如果您打算将这块磁盘划分为多个分区，可以使用相关指令来指定每个分区的容量比例。比如，假设您想划分成两个分区，那么可以通过下方代码设定第一个分区占据30%的容量，第二个分区占据50%的容量。

```
mkpart primary 0% 30%
mkpart primary 30% 80%
```

6、通过如下命令来检查分区是否对齐，若返回1，则对齐。

```
align-check optimal 1
```

7、通过如下命令查看当前磁盘的分区表。

```
print
```

8、通过如下命令退出。

```
quit
```

```
[root@instance-brz8myc4 ~]# parted /dev/nvme0n1
GNU Parted 3.2
Using /dev/nvme0n1
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
(parted) mkpart primary 1 100%
(parted) align-check optimal 1
1 aligned
(parted) print
Model: NVMe Device (nvme)
Disk /dev/nvme0n1: 3841GB
Sector size (logical/physical): 4096B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  3841GB  3841GB               primary

(parted) quit
Information: You may need to update /etc/fstab.

[root@instance-brz8myc4 ~]#
```

9、通过如下命令使系统重读分区表。

```
partprobe
```

10、通过如下命令查看磁盘分区。

```
lsblk
```

```
[root@instance-brz8myc4 ~]# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda           8:0    0 447.1G  0 disk
vda         253:0    0    20G  0 disk
├─vda1      253:1    0   200M  0 part /boot/efi
└─vda2      253:2    0  19.8G  0 part /
nvme0n1     259:0    0   3.5T  0 disk
└─nvme0n1p1 259:4    0   3.5T  0 part
nvme2n1     259:1    0   3.5T  0 disk
nvme1n1     259:2    0   3.5T  0 disk
nvme3n1     259:3    0   3.5T  0 disk
```

11、通过如下命令创建文件系统，本文以 ext4 为例。

```
mkfs -t ext4 /dev/nvme0n1p1
```

```
[root@instance-brz8myc4 ~]# mkfs -t ext4 /dev/nvme0n1p1
mke2fs 1.45.6 (20-Mar-2020)
Discarding device blocks: done
Creating filesystem with 937684224 4k blocks and 234422272 inodes
Filesystem UUID: c9e8289c-1852-4a38-a07f-0de87953a4da
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
        102400000, 214990848, 512000000, 550731776, 644972544

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done
```

12、通过如下命令配置开机自动挂载。 如果您需要设置开机自动挂载云盘，需要在/etc/fstab中写入新分区信息，启动开机自动挂载分区。

创建挂载目录，以/mnt/data1为示例，可自行更改为所需的挂载点。

```
mkdir /mnt/data1
```

通过如下命令备份fstab文件。

```
cp /etc/fstab /etc/fstab.bak
```

通过如下命令修改fstab文件。

```
echo /dev/nvme0n1p1 /mnt/data1 ext4 defaults,nofail 0 0 >> /etc/fstab
```

其中nvme0n1p1为指定的磁盘分区，/mnt/data1为目标挂载点，ext4是分区的文件系统类型，可根据已创建的文件系统修改，defaults和nofail是文件系统的挂载参数。

```
[root@instance-cfazzam5 ~]# echo /dev/nvme0n1p1 /data1 ext4 defaults,nofail 0 0 >> /etc/fstab
[root@instance-cfazzam5 ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Tue Dec 28 08:08:49 2021
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=bb2751bb-45b9-4206-b7b0-b09e0bee17bb /                       ext4    defaults        1 1
UUID=60E2-EB01          /boot/efi               vfat    defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2

# When mounting the data disk, you need to add the nofail parameter
# Example
# /dev/vdb1 /mount_point ext4 defaults,nofail 0 0

/dev/nvme0n1p1 /data1 ext4 defaults,nofail 0 0
```

13、通过如下命令挂载/etc/fstab配置的文件系统。

```
mount -a
```

若无报错信息提示，表示您的/etc/fstab配置无误。

14、通过如下命令查看目前磁盘空间和使用情况。

```
df -h
```

```
[root@instance-brz8myc4 ~]# mkdir /data1
[root@instance-brz8myc4 ~]# mount /dev/nvme
nvme0       nvme0n1     nvme0n1p1   nvme1       nvme1n1     nvme2       nvme2n1     nvme3       nvme3n1
[root@instance-brz8myc4 ~]# mount /dev/nvme0n1p1 /data1
[root@instance-brz8myc4 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        252G     0  252G   0% /dev
tmpfs           252G     0  252G   0% /dev/shm
tmpfs           252G  2.5M  252G   1% /run
tmpfs           252G     0  252G   0% /sys/fs/cgroup
/dev/vda2        20G  3.9G   15G  21% /
/dev/vda1       200M  7.3M  193M   4% /boot/efi
tmpfs            51G     0   51G   0% /run/user/0
/dev/nvme0n1p1  3.5T   89M  3.3T   1% /data1
[root@instance-brz8myc4 ~]#
```

LinuxBCC实例提示空间不足问题解决

背景

在BCC Linux实例中会遇到：no space left on device 磁盘空间不足的情况。

问题原因

产生磁盘空间不足问题的原因主要有以下几种情况：

- block空间使用率达到100%。
- inode空间使用率达到100%。
- 已删除文件因句柄未释放导致相应空间未释放。
- block空间使用率达到100%。

使用df -h命令查看磁盘分区的使用情况。

df -h

```
[root@instance-b39e3equ ~]# df -h /dev/vdb1
Filesystem      Size  Used Avail Use% Mounted on
/dev/vdb1        93M   91M     0 100% /mnt
[root@instance-b39e3equ ~]#
```

如图所示说明block空间使用率100%。

解决方法：

1、清除非必要的文件。

2、磁盘进行扩容。

inode空间使用率达到100%

使用df -i命令查看inode使用情况。

```
[root@instance-b39e3equ mnt]# df -i /mnt/
Filesystem     Inodes IUsed IFree IUse% Mounted on
/dev/vdb1       25688 25688     0  100% /mnt
[root@instance-b39e3equ mnt]#
```

如图所示，/dev/vdb1的inode使用率已100%。

解决方法主要有以下几种：

1. 清除inode占用高的文件或者目录。

进入inode使用100%的目录。

运行：for i in ./*; do echo $i; find $i | wc -l; done，查找inode占用较高的目录。

逐层进入inode占用最高的目录，继续执行以上命令，定位占用过高空间的文件或目录，然后进行文件或目录的清理 。

2. 修改inode【需要提前备份数据】。

卸载磁盘分区或挂载目录：umount /mnt

重新格式化文件系统：mkfs.ext4 /dev/xvdb -N 51376 ，示例中指定51376个inode，请以实际情况为准。

查看修改后的inode节点数。

```
[root@instance-b39e3equ ~]# dumpe2fs -h /dev/vdb1 | grep -i "inode count"
dumpe2fs 1.42.9 (28-Dec-2013)
Inode count:              51376
[root@instance-b39e3equ ~]#
```

重新挂载：mount /dev/vdb1 /mnt

3. 扩容磁盘

已删除文件因句柄未释放

执行命令： du -sh 查看文件占用空间和磁盘可用空间相差较大，同时磁盘inode使用率较低。

执行以下命令，查看存在问题的目录否存在未被清除句柄的文件。

lsof | grep delete 如果存在进程占用已删除文件的句柄，使用kill -9命令结束进程释放磁盘空间。

kill -9 pid

服务器配置

Debian 9软件源配置

线上带ofed驱动的实例更换内核方案

概述

线上一些BCC/EBC镜像装有ofed驱动，用户创建带rdma网络功能的套餐的实例后即可直接使用rdma功能。但是ofed是在当前os内核版本下编译的，如果用户改变内核版本后，可能会无法识别具有rdma功能的网卡。这种情况下，就需要重新编译、安装ofed驱动。

本文档提供在不同os的实例上更换内核版本后重新安装ofed的方案。

操作步骤

CentOS 7

以升级内核到kernel-3.10.0-1160.90.1.el7.x86_64为例。

1. 安装内核、内核开发包。在实例中安装kernel-3.10.0-1160.90.1.el7.x86_64版本内核及相关内核开发包：

```
yum install -y kernel-3.10.0-1160.90.1.el7.x86_64 \
              kernel-devel-3.10.0-1160.90.1.el7.x86_64 \
              kernel-headers-3.10.0-1160.90.1.el7.x86_64 \
              kernel-tools-3.10.0-1160.90.1.el7.x86_64
```

2. 检查rdma无损配置包。需要检查实例里是否安装有rdma无损配置包rdma-userspace-config，如有，需要删除该包，等安装好ofed后再重新安装。执行命令如下：

```
##### 检查是否安装了rdma-userspace-config
rpm -qa |grep -i rdma-userspace-config
```

如果检查该无损配置包已经安装，则执行命令进行删除：

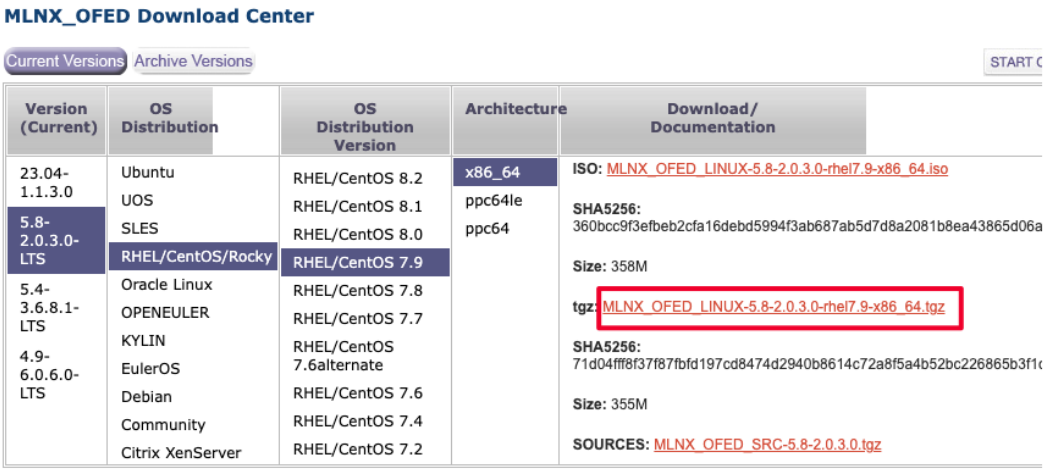
```
##### rpm -e rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载；

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-rhel7.9-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包



4. 编译安装ofed

- 执行以下脚本，编译、安装ofed驱动。
- 整个编译、安装过程要半个小时左右。

> 注：将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-rhel7.9-x86_64.tgz
kern_ver=3.10.0-1160.90.1.el7.x86_64

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi
if ! which gcc >& /dev/null; then
    yum install -y gcc
fi
if ! which make >& /dev/null; then
    yum install -y make
fi

##### Install Build Requirement
yum install -y elfutils-libelf-devel createrepo python-devel redhat-rpm-config rpm-build libtool
##### Install runtime requirement for all OFED components installation
yum install -y tcl gcc-gfortran fuse-libs tk libnl3-devel

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
cd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check \
    --package-install-options "--force" --distro rhel7.9 -q
if [ $? -ne 0 ]; then
    echo "MLNX OFED driver install ... Failed."
    exit 1
fi
cd ..

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /usr/lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /usr/lib/udev/rules.d/82-net-setup-link.rules /usr/lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut -f /boot/initramfs-${kern_ver}.img ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
 wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
rpm -ivh rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
 ##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# rpm -qa | grep mlnx | grep 3.10.0_1160
mlnx-ofa_kernel-modules-5.8-OFED.5.8.2.0.3.1.kver.3.10.0_1160.90.1.el7.x86_64.x86_64
mlnx-ofa_kernel-devel-5.8-OFED.5.8.2.0.3.1.kver.3.10.0_1160.90.1.el7.x86_64.x86_64
knem-modules-1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.3.10.0_1160.90.1.el7.x86_64.x86_64
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/3.10.0-1160.90.1.el7.x86_64/extra/mlnx-
ofa_kernel/drivers/net/ethernet/mellanox/mlx5/core/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
retpoline:      Y
rhelversion:    7.9
srcversion:     6A14E2ECBAE645B024A60B6
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX

##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# yum check dependencies
```

CentOS 8

以升级内核到kernel-4.18.0-348.7.1.el8_5.x86_64为例。

1. 安装内核、内核开发包。在实例中安装kernel-4.18.0-348.7.1.el8_5.x86_64版本内核及相关内核开发包：

```
yum install -y kernel-4.18.0-348.7.1.el8_5.x86_64 \
               kernel-devel-4.18.0-348.7.1.el8_5.x86_64 \
               kernel-headers-4.18.0-348.7.1.el8_5.x86_64 \
               kernel-tools-4.18.0-348.7.1.el8_5.x86_64
```

2. 检查rdma无损配置包。有些实例里安装有rdma无损配置包rdma-userspace-config。需要删除该包，等安装好ofed后重新安装：

```
 ##### 检查是否安装了rdma-userspace-config
rpm -qa |grep -i rdma-userspace-config
```

如果检查该无损配置包已经安装，则执行命令进行删除：

```
 如果存在就卸载该包
##### rpm -e rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载：

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-rhel8.4-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包

4. 编译安装ofed

执行以下脚本，编译、安装ofed驱动。整个编译、安装过程要半个小时左右。

> 注意，将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-rhel8.4-x86_64.tgz
kern_ver=4.18.0-348.7.1.el8_5.x86_64

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi
if ! which gcc >& /dev/null; then
    yum install -y gcc
fi
if ! which make >& /dev/null; then
    yum install -y make
fi

##### Install Build Requirement
yum install -y createrepo python36-devel libtool python36 kernel-rpm-macros gdb-headless rpm-build elfutils-libelf-devel
##### Install runtime requirement for all OFED components installation
yum install -y tk gcc-gfortran tcsh tcl libnl3-devel perl-Math-Complex cmake-filesystem

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
cd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check \
    --package-install-options "--force" --distro rhel8.4 -q
if [ $? -ne 0 ]; then
    echo "MLNX OFED driver install ... Failed."
    exit 1
fi
cd ..

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /usr/lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /usr/lib/udev/rules.d/82-net-setup-link.rules /usr/lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut -f /boot/initramfs-${kern_ver}.img ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
 wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
rpm -ivh --nodeps --force rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# rpm -qa | grep mlnx | grep 4.18.0_348
mlnx-ofa_kernel-devel-5.8-OFED.5.8.2.0.3.1.kver.4.18.0_348.7.1.el8_5.x86_64.x86_64
knem-modules-1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.4.18.0_348.7.1.el8_5.x86_64.x86_64
mlnx-ofa_kernel-modules-5.8-OFED.5.8.2.0.3.1.kver.4.18.0_348.7.1.el8_5.x86_64.x86_64
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/4.18.0-348.7.1.el8_5.x86_64/extra/mlnx-
ofa_kernel/drivers/net/ethernet/mellanox/mlx5/core/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
rhelversion:    8.5
srcversion:     AED21A09CA345D254692F69
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX
##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# yum check
```

Rocky Linux 8

以升级内核到kernel-4.18.0-477.13.1.el8_8.x86_64为例。

1. 安装内核、内核开发包。在实例中安装kernel-4.18.0-477.13.1.el8_8.x86_64版本内核及相关内核开发包：

```
yum install -y kernel-4.18.0-477.13.1.el8_8.x86_64 \
               kernel-devel-4.18.0-477.13.1.el8_8.x86_64 \
               kernel-headers-4.18.0-477.13.1.el8_8.x86_64 \
               kernel-tools-4.18.0-477.13.1.el8_8.x86_64
```

2. 检查rdma无损配置包。有些实例里安装有rdma无损配置包rdma-userspace-config。需要删除该包，等安装好ofed后重新安装：

```
##### 检查是否安装了rdma-userspace-config
rpm -qa |grep -i rdma-userspace-config
```

如果检查该无损配置包已经安装，则执行命令进行删除：

```
如果存在就卸载该包
##### rpm -e rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-rhel8.7-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包

4. 编译安装ofed

执行以下脚本，编译、安装ofed驱动。

注意，将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

整个编译、安装过程要半个小时左右。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-rhel8.7-x86_64.tgz
kern_ver=4.18.0-477.13.1.el8_8.x86_64

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi
if ! which gcc >& /dev/null; then
    yum install -y gcc
fi
if ! which make >& /dev/null; then
    yum install -y make
fi

##### Install Build Requirement
yum install -y createrepo python36-devel libtool python36 kernel-rpm-macros gdb-headless rpm-build elfutils-libelf-devel
##### Install runtime requirement for all OFED components installation
yum install -y tk gcc-gfortran tcsh tcl libnl3-devel perl-Math-Complex cmake-filesystem

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
cd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check \
    --package-install-options "--force" --distro rhel8.7 -q
if [ $? -ne 0 ]; then
    echo "MLNX OFED driver install ... Failed."
    exit 1
fi
cd ..

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /usr/lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /usr/lib/udev/rules.d/82-net-setup-link.rules /usr/lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut -f /boot/initramfs-${kern_ver}.img ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
rpm -ivh --nodeps --force rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# rpm -qa | grep mlnx | grep 4.18.0_477
mlnx-ofa_kernel-modules-5.8-OFED.5.8.2.0.3.1.kver.4.18.0_477.13.1.el8_8.x86_64.x86_64
mlnx-ofa_kernel-devel-5.8-OFED.5.8.2.0.3.1.kver.4.18.0_477.13.1.el8_8.x86_64.x86_64
knem-modules-1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.4.18.0_477.13.1.el8_8.x86_64.x86_64
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/4.18.0-477.13.1.el8_8.x86_64/extra/mlnx-
ofa_kernel/drivers/net/ethernet/mellanox/mlx5/core/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
rhelversion:    8.8
srcversion:     AED21A09CA345D254692F69
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX
##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# yum check
```

Rocky Linux 9

以升级内核到kernel-5.14.0-284.11.1.el9_2.x86_64为例。

1. 安装内核、内核开发包。在实例中安装kernel-5.14.0-284.11.1.el9_2.x86_64版本内核及相关内核开发包：

```
yum install -y kernel-5.14.0-284.11.1.el9_2.x86_64 \
              kernel-devel-5.14.0-284.11.1.el9_2.x86_64 \
              kernel-headers-5.14.0-284.11.1.el9_2.x86_64 \
              kernel-tools-5.14.0-284.11.1.el9_2.x86_64
```

2. 检查rdma无损配置包。有些实例里安装有rdma无损配置包rdma-userspace-config。需要删除该包，等安装好ofed后重新安装：

```
##### 检查是否安装了rdma-userspace-config
rpm -qa |grep -i rdma-userspace-config
```

如果检查该无损配置包已经安装，则执行命令进行删除：

```
如果存在就卸载该包
##### rpm -e rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-rhel9.1-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包

4. 编译安装ofed

执行以下脚本，编译、安装ofed驱动。

注意，将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

整个编译、安装过程要半个小时左右。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-rhel9.1-x86_64.tgz
kern_ver=5.14.0-284.11.1.el9_2.x86_64

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi
if ! which gcc >& /dev/null; then
    yum install -y gcc
fi
if ! which make >& /dev/null; then
    yum install -y make
fi

##### Install Build Requirement
yum install -y perl createrepo python3-devel libtool kernel-rpm-macros gdb-headless rpm-build elfutils-libelf-devel
##### Install runtime requirement for all OFED components installation
yum install -y tk gcc-gfortran tcsh tcl libnl3-devel cmake-filesystem

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
cd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check \
    --package-install-options "--force" --distro rhel9.1 -q
if [ $? -ne 0 ]; then
    echo "MLNX OFED driver install ... Failed."
    exit 1
fi
cd ..

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /usr/lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /usr/lib/udev/rules.d/82-net-setup-link.rules /usr/lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut -f /boot/initramfs-${kern_ver}.img ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
 wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
rpm -ivh --nodeps --force rdma-userspace-config-bbc-v1.0.3-1.el7.centos.x86_64.rpm
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# rpm -qa | grep mlnx | grep 5.14.0_284
mlnx-ofa_kernel-modules-5.8-OFED.5.8.2.0.3.1.kver.5.14.0_284.11.1.el9_2.x86_64.x86_64
mlnx-ofa_kernel-devel-5.8-OFED.5.8.2.0.3.1.kver.5.14.0_284.11.1.el9_2.x86_64.x86_64
knem-modules-1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.5.14.0_284.11.1.el9_2.x86_64.x86_64
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/5.14.0-284.11.1.el9_2.x86_64/extra/mlnx-
ofa_kernel/drivers/net/ethernet/mellanox/mlx5/core/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
rhelversion:    9.2
srcversion:     E4CF39F4680CAB323741675
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX
##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# yum check
```

Ubuntu 20.04

以升级内核到5.4.0-152-generic为例。

1. 安装内核、内核开发包。在实例中安装5.4.0-152-generic版本内核及其它相关组件：

```
apt-get install -y linux-image-5.4.0-152-generic \
                   linux-headers-5.4.0-152 \
                   linux-headers-5.4.0-152-generic \
                   linux-modules-5.4.0-152-generic \
                   linux-modules-extra-5.4.0-152-generic
```

2. 检查rdma无损配置包。有些实例里安装有rdma无损配置包rdma-userspace-config。需要删除该包，等安装好ofed后重新安装：

```
##### 检查是否安装了rdma-userspace-config
rpm -qa |grep -i rdma-userspace-config
```

如果检查该无损配置包已经安装，则执行命令进行删除：

```
如果存在就卸载该包
##### rpm -e rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-ubuntu20.04-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包

4. 编译安装ofed

执行以下脚本，编译、安装ofed驱动。整个编译、安装过程要半个小时左右。

> 注意，将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-ubuntu20.04-x86_64.tgz
kern_ver=5.4.0-152-generic

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi

apt-get update
if ! which gcc >& /dev/null; then
    apt-get -y install gcc
fi
if ! which make >& /dev/null; then
    apt-get -y install make
fi

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
pushd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check -q
if [ $? -ne 0 ]; then
    ECHO_ERROR "MLNX OFED driver install ... Failed."
    exit 1
fi
popd

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /lib/udev/rules.d/82-net-setup-link.rules /lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut  -f /boot/initrd.img-${kern_ver} ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc_1.0.4_x86.deb
dpkg -i rdma-userspace-config-bbc_1.0.4_x86.deb
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# dpkg -l | grep mlnx | grep 5.4.0-152
ii  knem-modules                    1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.5.4.0-152-generic amd64        kernel module for high-performance intra-node MPI communication for large messages
ii  mlnx-ofed-kernel-modules        5.8-OFED.5.8.2.0.3.1.kver.5.4.0-152-generic          amd64        mlnx-ofed kernel modules
ii  mlnx-ofed-kernel-utils          5.8-OFED.5.8.2.0.3.1.kver.5.4.0-152-generic          amd64        Userspace tools to restart and tune mlnx-ofed kernel modules
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/5.4.0-152-generic/updates/dkms/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
srcversion:     E9B015CBD3F471BDD31CB24
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX
##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# apt-get check
```

Ubuntu 22.04

以升级内核到5.15.0-75-generic为例。

1. 安装内核、内核开发包。在实例中安装5.15.0-75-generic版本内核及其它相关组件：

```
apt-get install -y linux-image-5.15.0-75-generic \
                   linux-headers-5.15.0-75 \
                   linux-headers-5.15.0-75-generic \
                   linux-modules-5.15.0-75-generic \
                   linux-modules-extra-5.15.0-75-generic
```

2. 有些实例里安装有rdma无损配置包rdma-userspace-config。需要删除该包，等安装好ofed后重新安装：

```
##### 检查是否安装了rdma-userspace-config
dpkg -l |grep -i rdma-userspace-config
如果存在就卸载该包
dpkg -r rdma-userspace-config
```

3. 下载ofed安装包。可以在百度软件源上下载，或者在ofed官网下载后传到实例内：

1. 百度软件源下载，以下载5.8-2.0.3版本ofed为例，在实例内执行下面命令下载

```
wget http://mirrors.baidubce.com/mlnx-ofed/5.8-2.0.3.0/MLNX_OFED_LINUX-5.8-2.0.3.0-ubuntu22.04-x86_64.tgz
```

2. 官网下载，在ofed官网下载对应版本的ofed安装包

4. 编译安装ofed

执行以下脚本，编译、安装ofed驱动。整个编译、安装过程要半个小时左右。

> 注意，将脚本和ofed安装包放到同一目录下，脚本中ofed版本和内核版本改为实际使用版本。

```
 ##### !/usr/bin/env bash
##### Update mlnx ofed drivers
##### -- Prepare mlnx ofed drivers
##### -- Extract and install

mlnx_ofed_version=5.8-2.0.3.0
mlnx_ofed_drv=MLNX_OFED_LINUX-${mlnx_ofed_version}-ubuntu22.04-x86_64.tgz
kern_ver=5.15.0-75-generic

if [ ! -d /lib/modules/${kern_ver}/build ]; then
    echo "There is no kernel build directory. Please check if kernel-devel is installed ..."
    exit 1
fi

apt-get update
if ! which gcc >& /dev/null; then
    apt-get -y install gcc
fi
if ! which make >& /dev/null; then
    apt-get -y install make
fi

##### Install required packages for installing MLNX_OFED_LINUX
apt-get install -y libnl-route-3-200 libnl-route-3-dev libnl-3-dev bison libfuse2 flex gfortran tk libnuma-dev libgfortran5

##### Install updated mlnx_ofed packages including kernel modules and userspace packages
mkdir update_drivers
tar xf ${mlnx_ofed_drv} --strip-components 2 -C update_drivers/
pushd update_drivers
./mlnxofedinstall --without-fw-update  --add-kernel-support -k ${kern_ver} --skip-distro-check -q
if [ $? -ne 0 ]; then
    ECHO_ERROR "MLNX OFED driver install ... Failed."
    exit 1
fi
popd

##### 82-net-setup-link.rules cause the name of NIC changed. Then disable this rule.
if [ -f /lib/udev/rules.d/82-net-setup-link.rules ]; then
    mv /lib/udev/rules.d/82-net-setup-link.rules /lib/udev/rules.d/82-net-setup-link.rules.orig
fi

if [ -f /usr/lib/udev/rules.d/83-mlnx-sf-name.rules ]; then
    mv /usr/lib/udev/rules.d/83-mlnx-sf-name.rules /usr/lib/udev/rules.d/83-mlnx-sf-name.rules.orig
fi

##### Disable rshim(tmfifo_net0)
systemctl disable rshim.service

rm -rf ${mlnx_ofed_drv} update_drivers
rm -rf /tmp/MLNX* /tmp/*.conf

##### update initramfs
dracut  -f /boot/initrd.img-${kern_ver} ${kern_ver}
```

- 如果第二步卸载了rdma-userspace-config，重新安装回来

```
 wget -q http://mirrors.baidubce.com/baidu/rdma_specs/rdma-userspace-config-bbc_1.0.4_x86.deb
dpkg -i rdma-userspace-config-bbc_1.0.4_x86.deb
service rdma start
```

5. 重启实例，进入新内核；

6. 检查安装。

```
 ##### 检查已安装5.8-2.0.3版本并在当前内核下编译出的ofed
[root@localhost ~]# dpkg -l | grep mlnx | grep 5.15.0-75
ii  knem-modules                         1.1.4.90mlnx1-OFED.5.8.0.4.7.1.kver.5.15.0-75-generic amd64       kernel module for high-performance intra-node MPI communication for large messages
ii  mlnx-ofed-kernel-modules             5.8-OFED.5.8.2.0.3.1.kver.5.15.0-75-generic           amd64       mlnx-ofed kernel modules
ii  mlnx-ofed-kernel-utils               5.8-OFED.5.8.2.0.3.1.kver.5.15.0-75-generic           amd64       Userspace tools to restart and tune mlnx-ofed kernel modules
[root@localhost ~]# modinfo mlx5_core
filename:       /lib/modules/5.15.0-75-generic/updates/dkms/mlx5_core.ko
alias:          auxiliary:mlx5_core.eth-rep
alias:          auxiliary:mlx5_core.eth
basedon:        Korg 5.17-rc4
version:        5.8-2.0.3
license:        Dual BSD/GPL
description:    Mellanox 5th generation network adapters (ConnectX series) core driver
author:         Eli Cohen <eli@mellanox.com>
srcversion:     E4CF39F4680CAB323741675
...
##### 检查网卡
[root@localhost ~]# ifconfig
##### 检查网卡驱动
[root@localhost ~]# ethtool -i ethX
##### 检查软件包依赖，如果缺少依赖包，或者软件包重复，或者其他问题，需要手动解决，安装缺失的依赖包，删除重复的软件包等；
[root@localhost ~]# apt-get check
```

linux pip配置百度源

概述

pip是通用的Python包管理工具，提供了对Python包的查找、下载、安装、卸载、更新的功能。本文档介绍基于使用CentOS、Ubuntu等Linux镜像的百度云线上BCC实例配置pip百度源的方案。

注意事项

请确保您已经安装与python版本对应的pip（pip 已内置于 Python 3.4 和 2.7 及以上版本，其他版本需另行安装）。

配置方法

在您的百度云实例中执行以下命令配置pip使用百度源：

```
 cat > /root/.pip/pip.conf << EOF
[global]
index-url=http://mirrors.baidubce.com/pypi/simple/
[install]
trusted-host=mirrors.baidubce.com
EOF

cat > /root/.pydistutils.cfg << EOF
[easy_install]
index-url=http://mirrors.baidubce.com/pypi/simple/
EOF
```

配置完后，即可使用百度pypi源安装您所需要的Python库。安装示例：

```
 pip3 install numpy
```

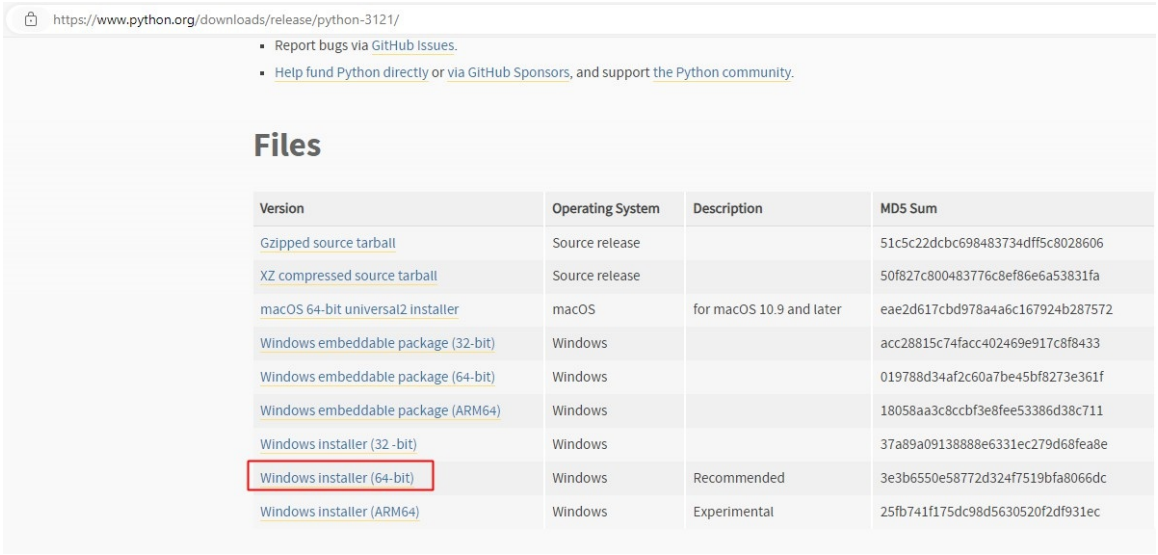
Windows pip配置百度源

概述

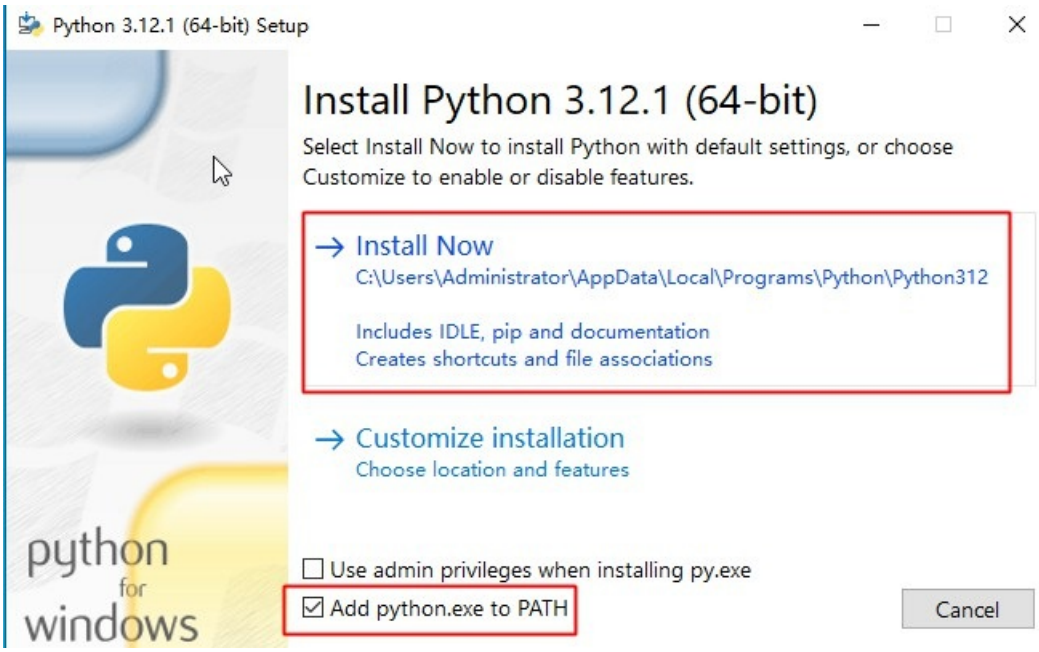
pip是通用的Python包管理工具，提供了对Python包的查找、下载、安装、卸载、更新的功能。本文档以Windows Server 2022为例介绍百度云线上Windows实例配置pip百度源的方法。

安装python pip

1. 您可以在python官网下载安装程序，参考链接：https://www.python.org/；



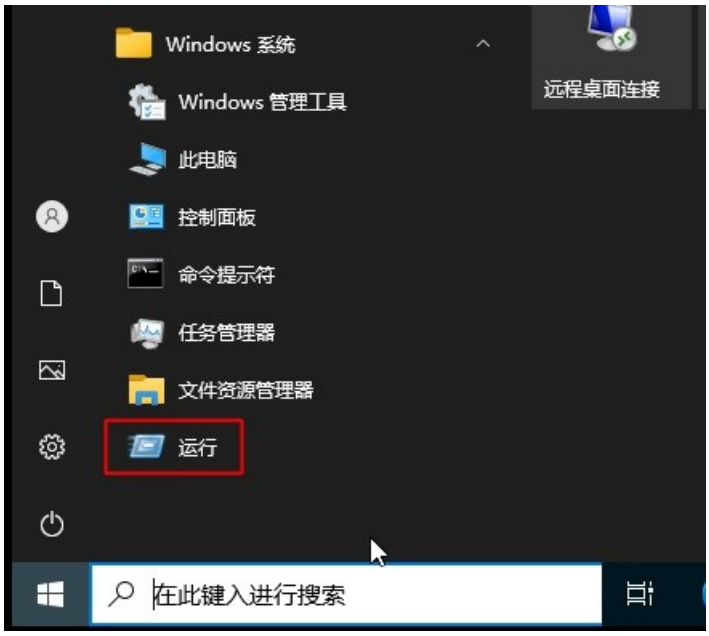
2. 下载installer之后，双击exe文件进行安装，选中“Add python.exe to PATH”，然后点击**Install Now**。

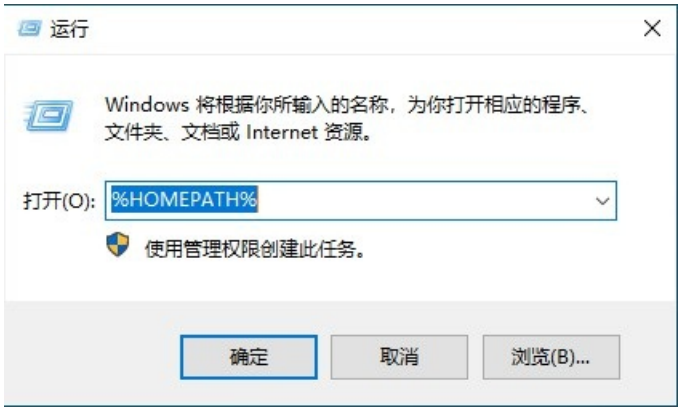


3. 打开cmd命令窗口，输入命令“`python --version`”和“`pip --version`”，显示python和pip的版本，表示安装成功。

配置python pip百度源

打开运行窗口，输入`%HOMEPATH%`，打开%HOMEPATH%目录

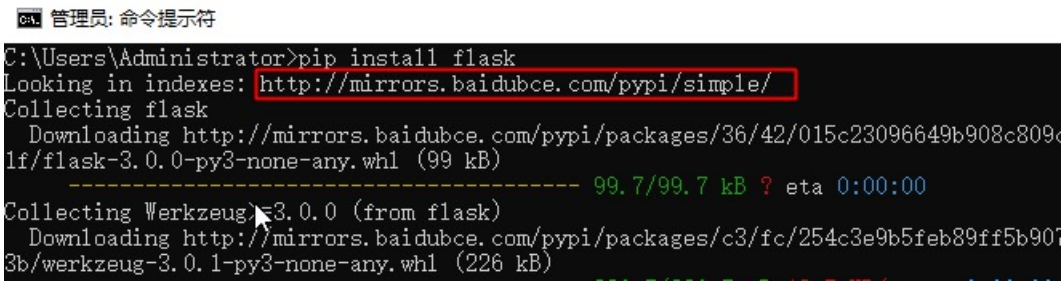




在此目录下创建 pip 文件夹，在 pip 目录下创建 pip.ini 文件, 内容如下：

```
 [global]
timeout = 6000
index-url = http://mirrors.baidubce.com/pypi/simple/
trusted-host = mirrors.baidubce.com
```

使用pip安装任意一个模块，可以看到使用的是百度源的地址。



常见问题

实例类

什么是BCC？

云服务器BCC（Baidu Cloud Compute）是处理能力可弹性伸缩的计算服务。管理方式比物理服务器更简单高效，可根据您的业务需要，随时创建、释放任意多台云服务器实例，提升运维效率。为用户快速部署应用构建稳定可靠的基础，降低网络规模计算的难度，使用户更专注于核心业务创新。且无需花费时间和金钱来购买及维护托管虚拟机的硬件，有效降低IT成本。

BCC可以动态调整CPU、内存大小吗？

目前已购买的BCC实例支持按需变配，通过变更实例规格调整CPU和内存的大小。

BCC是否可以支持VMware？

目前BCC暂不支持VMware等虚拟化软件和二次虚拟化。

BCC默认提供数据库吗？

BCC上本身没有预装数据库，您可自行安装所需数据库。同时建议您也可以单独购买百度智能云提供的关系型数据库RDS服务。

BCC是否支持Oracle数据库？

BCC不限制客户安装Oracle数据库，您可以查看相应的Oracle数据库安装教程进行安装。

BCC能否做负载均衡？

BCC可根据您的需求配置成负载均衡服务，但更建议您选择百度智能云提供的独立负载均衡产品BLB，该产品可进行四层/七层的负载均衡服务，并有针对性的做了优化，您可直接购买使用。

BCC能否可指定地域购买？

支持，如果您需要在多区域创建BCC实例，请参考区域选择说明。

如何更换操作系统/升级配置？

- 如果用户需要切换到其它的操作系统，可以针对单个实例进行重装操作系统。更换操作系统请参考重装操作系统。
- 升级配置请参考升级CPU和内存。

在线变配（热升级）的限制有哪些？

1. 仅CPU型号为6271（对应规格族为通用型g4、计算型c4、密集计算型ic4、内存型m4）的实例规格支持在线变配。
2. 在线变配的最大配置是42c176g，最小配置是1c1g。
3. CPU规格可热降到最低1c，内存不支持热降。
4. 所有32位的Windows/Linux实例不支持热变配，镜像为 Win2016/fedora/freebsd/opensuse 不支持热改配。

镜像	CPU	内存
Linux64（除fedora/freebsd/opensuse）	可热升、可热降	可热升，不可热降
Windows64（除Win2016）	可热升	可热升，不可热降

5. 实例连续热变配最多可支持8次。
6. GPU机型暂不支持在线变配。
7. CPU和内存必须同升同降，不能一升一降/一降一升。
8. 新建实例（该功能上线时间点之后创建的实例）默认具备热变配的能力。
9. 历史实例（该功能上线时间点之前创建的实例）在经过冷变配（即需要重启的改配方式）后才能进行热变配。

为什么 Ubuntu、Dedian 64bit 镜像的实例在线改配（热升级）后不生效？

需要在操作系统中执行以下操作，对于已经执行过在线改配的实例通过**手动方式**配置生效；配置**自动方式**后可对增量在线改配直接生效，无需再手动配置。

- 手动方式：

```
命令如下（cpunum为 新插入的cpu号）：
echo 1 >/sys/devices/system/cpu/cpunum/online
echo 1 >/sys/devices/system/cpu/cpu21/online
```

- 自动方式：

```
1）需要增加 udev 规则，增加文件/etc/udev/rules.d/40-cpu-mem-hotplug.rules 内容如下：
cat /etc/udev/rules.d/40-cpu-mem-hotplug.rules
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0", ATTR{online}="1"
SUBSYSTEM=="memory", ACTION=="add", ATTR{state}=="offline", ATTR{state}="online"
2）service udevd restart
```

改配后的CPU规格后可通过 lscpu 或 cat /proc/cpuinfo 确认。

改配后的内存规格可通过 free 或 cat /proc/meminfo 确认。

如果创建BCC时选定的配置有误，可以在创建完成后修改配置吗？

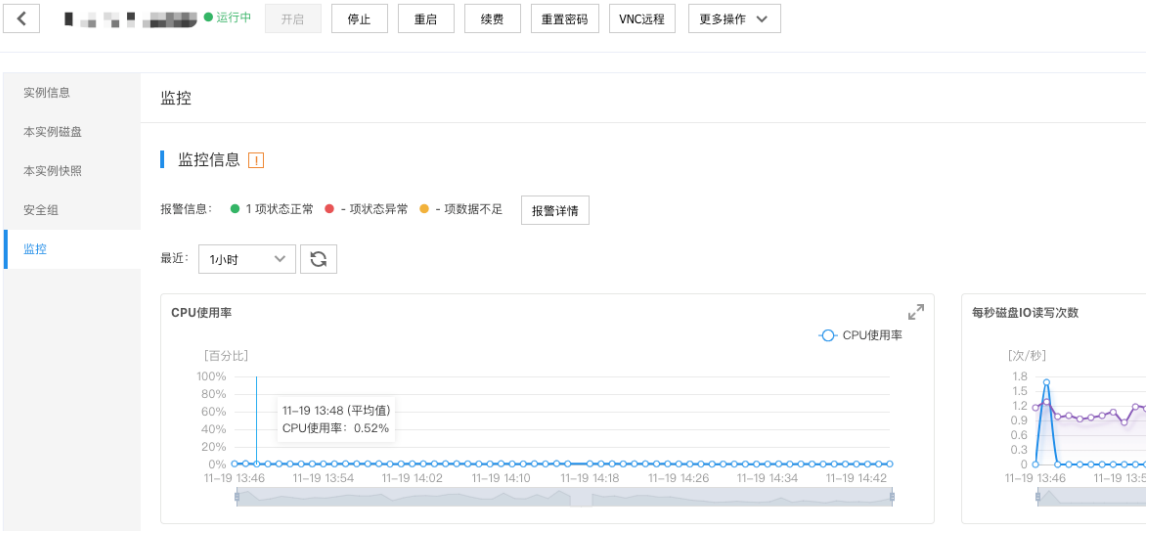
对于已购买的BCC实例，您可进行操作系统更换操作，您可重新选择合适的系统镜像进行重装，支持Windows和Linux操作系统类型的互换。同时，也支持升级配置。

控制台找不到某台BCC实例怎么办？

如果您购买的服务器在BCC界面没有查看到，建议通过实例页面左上角的区域切换功能查看其它区域。

怎么查看BCC资源使用情况？

您可以登录管理控制台，通过产品服务 / 云服务器BCC-实例列表 / 实例详情 / 监控进行查看，或进入系统内部进行查看。



BCC检查到CPU、磁盘运行异常能否发送短信或邮件提醒？

如果想要实现BCC检查到CPU、磁盘等运行异常后发送短信或邮件提醒的话，您可以在控制台-BCM监控内添加报警策略。

系统内看到的系统可用内存比标称值少

这种情况通常是由于硬件资源的预留和系统本身占用导致的正常差异。在云服务器中，一部分内存会被系统内核、管理程序（Hypervisor）以及其他系统服务预留使用，以保证系统的稳定运行和其他功能的正常提供。这意味着实际可用给用户操作系统的内存会少于标称的总内存大小。如您想在系统内部查标称内存值，可通过运行dmidecode -t memory命令进行查询。

网络类

BCC支持的MTU值是多少呢？

目前，内外网网卡的MTU(最大传输单元)值默认都是1500，您可以参考以下命令查询当前MTU设置值：

- Linux： 执行"netstat -i"查询。

```
[root@wang ~]# netstat -i
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500   0  1306254      0      0      0  1396652      0      0      0 BMRU
lo        65536   0   249113      0      0      0   249113      0      0      0 LRU
[root@wang ~]#
```

- Windows: 执行 cmd 命令 "netsh interface ip show interface"查询。

```
C:\Users\Administrator>netsh interface ipv4 show subinterfaces

   MTU  MediaSenseState   传入字节  传出字节      接口
------  ---------------  ---------  ---------  -------------
4294967295                1          0          0  Loopback Pseudo-Interface 1
  1500                1  1064902347  760795410  本地连接
```

如果大文件在传输过程中出现分片时，可能是由于MTU设置值较大，解决办法是将本机的MTU改为较小一些。以接口eth0为例，执行如下命令（xxxx为需要设置的MTU值）：

```
ip link set dev eth0 mtu xxxx
```

如何创建具有公网IP的BCC？

您需为BCC绑定EIP并通过EIP访问公网。

BCC分配给用户的公网IP是否唯一？

购买BCC时可关联购买EIP，BCC会自动分配一个唯一的、独立的、固定的公网IP给用户。

外网IP是否固定？

是的，用户不能指定、选择或者增加购买云服务器公网IP地址。

BCC公网IP和私网IP的区别是什么？

购买BCC时可关联购买EIP，关联购买的EIP会自动绑定到BCC上，系统为EIP自动分配公网IP。

- 公网IP用于连接公网网络，如果服务器的应用需要从服务器外部公网访问，使用公网IP。
- 私网IP用于私网中多台云服务器之间的访问，如果您账号下有多台云服务器，使用私网IP地址进行不同服务器间数据通信传输，私网传输速度性能可参考实例规格的带宽、包速率等参数。

如果没有公网IP，如何从外网访问？

需要您至少有一台BCC可访问公网，通过该云服务器作中转，通过私有网络连通没有公网IP的BCC。

如何升级公网带宽？

对于每一个EIP实例，您可以根据实际情况进行带宽的调整。其中预付费的EIP实例，仅支持升级带宽，不支持降低带宽；后付费的EIP实例，可以随时在控制台中调整带宽大小。操作文档请参考升级EIP实例带宽。

能否释放公网IP？

用户可以自行释放空闲的EIP实例，节约租用成本，但以下三种情况不允许释放：

- 包年包月并且未到期的EIP实例
- 已经绑定BCC或BLB实例的EIP
- 因攻击导致暂停状态的EIP

如需释放包年包月且未到期的EIP实例，请提交财务工单申请退款，具体是否允许退款，请以工单处理结果为准

BCC如何绑定双公网IP？

BCC支持绑定弹性网卡，实现一个BCC上有多个公网和内网IP，具体参考弹性网卡文档。

地域网络线路是什么？

目前我们提供的是BGP多线路接入，同时接驳有联通、电信、移动、教育网等多家运营商线路。

带宽是否独享？

是独享的。

BCC创建之后，用户能否手动更改服务器IP地址？能否自行将服务器绑定到自己的公网 IP 地址？

BCC的公网 IP 地址是经过多重转换后，映射到服务器内部 IP 地址的。手动更改云服务器的网络设置，会造成云服务器无法访问等问题。

BCC支持按需变更内网IP，您可在控制台的实例列表页操作。

如果在BCC上部署一个互联网服务如Apache，MySQL等，监听端口如何绑定IP地址来发布服务？

如果需要在BCC上部署互联网服务，监听端口需要绑定内网地址或0.0.0.0来发布服务。BCC上服务的客户端使用公网IP访问服务。

例如某一BCC实例的公网IP是180.76.135.xxx，内网IP是192.168.xxx.xxx，在该BCC实例上部署Apache服务，配置文件的配置为：

Listen 0.0.0.0:80 Listen 0.0.0.0:8080

或者： Listen 192.168.xxx.xxx:80 Listen 192.168.xxx.xxx:8080

发布后，打开浏览器访问http：//180.76.135.xxx:80，即可访问BCC上发布的页面服务。

不同账号下的BCC如何实现内网互通？

默认情况下，同一个VPC下的云服务器默认内网互通，不同VPC的云服务器隔离。 如需互通，您可以开通对等连接服务，实现VPC之间的高速互联，请参考对等连接文档。

部分端口无法正常使用是怎么回事？

云服务器BCC对端口没有限制，需要您检查一下：
1.防火墙和安全组是否允许了该端口
2.服务是否正常监听了此端口
满足以上两点，端口才是正常的状态。

网络延迟变大/间歇性丢包是什么原因？

网络延迟变大/丢包的可能原因有很多，例如网络服务商物理线路问题、网络出口/入口波动、访问的资源节点不同等等。若果您发现网络上的问题，请您将ip地址通过工单反馈我们排查原因。

ping网站有丢包怎么办？

首先要确定 ping 所有外网是否都有延迟（比如 ping baidu.com）。

- 如果 ping 其它外网网址也有问题，应该是您本地网络或您宽带接入商的问题；
- 如果 ping 其它外网网址没有问题，请检查您所用云服务器的带宽是否被占满，如未占满，可以对虚机的 IP 地址执行 `trace route` 命令：在 Linux 环境下是`traceroute <IP>` 命令，在 Windows 下是 `tracert <IP>` 命令。

注：这里需要把`<IP>`替换成对应的 IP 地址。您可根据路由跟踪结果查看延时或超时主要出现在哪些IP节点，并确认这些IP的归属。或者您可提交工单联系技术人员，并将 `ping` 命令及 `traceroute/tracert` 命令的输出一起上传到工单里，由百度技术人员为您分析原因。

Windows云服务器为什么浏览网页异常？

由于BCC的Widnows系统是服务器版本，出于安全性考虑，默认启用了IE增强的安全配置，需要把相关站点添加到受信任站点列表才能正常访问。您也可以将“Internet Explorer增强的安全设置”修改为关闭状态。

- Windows 2008 Server版本：在“管理工具-服务器管理器->服务器摘要->安全信息”中修改“配置IE ESC”为关闭状态。
- Windows 2012 Server版本：在“管理工具->服务器管理器->本地服务器->属性”中修改“Internet Explorer增强的安全设置”为关闭状态。

存储类

如何在BCC上挂载并使用CDS？

在BCC上使用CDS需要先挂载CDS，然后登陆实例操作系统中对磁盘进行格式化。具体操作请参考挂载CDS磁盘以及格式化磁盘。

能否在BCC上挂载多个CDS？

可以，一个BCC上最多可挂载5个CDS百度智能云磁盘。若您希望挂载更多数量，请提交工单联系我们。

能否将CDS挂载到多个BCC上？

不可以。一个CDS只能挂载到一个BCC上。

在创建Windows BCC时选择了CDS云磁盘，但是为什么在登录到BCC后看不到此CDS磁盘？

Windows 服务器里的 CDS 磁盘默认情况下为脱机状态，需要手动联机才可使用。请进入“服务器管理器 >文件和存储服务>卷>磁盘”，将脱机的磁盘联机即可。

如何在不同的硬盘之间同步数据？

同账户下：

1. 如果是同步CDS系统盘数据，同区域下（如都是北京区域/广州区域等）的服务器，可以通过创建自定义镜像进行数据转移，在新服务器装系统时选择之前创建的自定义镜像即可。
2. 如果是同步CDS数据盘，同区域同可用区下（如北京可用区A），可以直接挂载到新的服务器上即可（或者可以通过磁盘快照创建新的CDS磁盘进行使用）；同区域不同可用区，可以通过老的CDS的快照创建其他可用区的磁盘，进行挂载到新的服务器上使用。
3. 暂不支持实例之间的本地盘数据同步，可通过线下数据拷贝进行自行迁移。

不同账户下：
可以通过共享镜像实现不同账户间同区域下BCC系统盘数据的快速迁移和环境部署。

其他如CDS数据跨账户/跨区（如北京->广州区域）迁移、临时数据盘数据迁移、系统盘数据跨区域（如北京->广州区域）迁移目前都只能通过线下数据拷贝进行自行迁移。

BCC实例可以挂载对象存储BOS吗？

BOS挂载到linux系统BCC服务器上，可以使用BOSFS实现（windos系统暂不支持挂载）。
BOSFS用于实现将远程bos服务挂载到本地文件系统中，实现类似于网络文件系统的功能，目的是便于用户使用系统命令或者标准的POSIX接口访问BOS上的文件，就像使用本地磁盘一样方便。具体参考BOSFS帮助文档
如果在配置过程中遇到任何问题，建议您提交存储和CDN/对象存储BOS工单，详细描述您遇到的具体问题，提供具体报错截图，我方工程师会协助您排查。

BCC新购买磁盘在控制台挂载后，在虚机内没有看到

BCC新购买磁盘在控制台挂载后，在虚机内没有看到，是由于新购并挂载的CDS磁盘windows或linux需要格式化后联机或挂载后才可以正常使用。具体参考格式化磁盘。

注意！！格式化会导致所有数据丢失，如CDS中有重要数据，请提前备份。

镜像类

使用自定义镜像创建BCC时，是否对套餐大小有所限制？

使用自定义镜像创建BCC时，对新服务器的各项参数（如CPU、内存）没有限制。

BCC目前是否支持上传用户自建镜像？

目前BCC已支持导入自定义镜像，具体参考导入自定义镜像概述。

通过自定义镜像创建的BCC支持修改系统吗？

支持，在BCC控制台对指定BCC操作重装操作系统即可。

快照类

为什么要使用快照功能？

当云磁盘数据被误操作删除时，利用之前创建的快照，可以快速而准确地回滚到任一快照时的数据状态。

创建快照时BCC会关闭吗？

创建快照时，BCC不需要关闭。

BCC释放后，自定义镜像和快照是否还存在？

- 自定义镜像：创建的自定义镜像只要不删除，将会一直保留。
- 快照：释放BCC时会同时释放对应的系统盘快照，可按需选择是否释放对应的CDS数据盘快照。

快照目前计费吗？

- 快照目前是计费服务，根据快照链进行按需付费。更多关于快照计费的介绍，请查看快照计费说明。

计费类

购买BCC如何付款？

BCC分为预付费（包年包月）、后付费（按量计费）和抢占实例三种付款模式。预付费形式下，您可在选购BCC之后对生成的订单进行支付即可；如果您选择的是后付费或抢占实例，下订单后即可开通，无需当时付款，其中后付费需要您账户余额大于100元。

如何索要发票？

您需要走线上流程，在“财务中心>发票管理”进行申请即可。

系统有到期提醒吗？

对于预付费（包年包月）形式的产品，在用户使用期限到期前七天，系统会每天以短信形式给用户发送即将到期的提醒通知。

BCC是后付费能改为预付费按月计算吗?

可以。通过“计费变更”功能，由按量计费（后付费）方式变更为包年包月计费（预付费）方式。

资源配置发生更改后，资费是如何变更的？

资源配置发生更改并完成支付后即可使用。

资源因为欠费被暂停使用了，充值之后系统会自动恢复吗？

- 在到期后七天内，BCC将自动关闭并保留数据，同时给用户发送续费通知。续费后，系统将自动恢复原服务的使用。
- 当欠费超过七天，BCC将会被删除且资源无法恢复，但用户的自定义镜像将会被保留。

实例释放后，自定义镜像是否还存在？

存在。

是否可以切换计费模式？

支持用户按量计费和包年包月计费模式之间转换，同时，对于包年包月的实例，您可选择当前包年包月到期后切换或者立即切换为按量计费模式。您可在 BCC 产品控制台直接操作变更。

性能类

服务器CPU跑满了怎么办？

首先要登录 BCC 查看是哪些进程/应用在消耗 CPU。

- 在 Linux 服务器上，可以使用 `nmon` 命令或者`top`命令查看。

- 在 Windows 上可以在任务管理器里（或者安装百度安全卫士）查看，分析后可强制结束相关进程。 如果您的进程属于正常负载，可考虑进行服务器配置升级，或选购额外的BCC分担负载。

服务器带宽跑满了怎么办？

如果是应用的流量已经达到了带宽的上限，需要重新调整带宽。如果是私有网络带宽已满，您可以通过实例变配，升级到带宽规格更大的实例规格，如果是公网带宽已满，您可以按需调整EIP的带宽上限。

安全类

密码丢失怎么办？

点击“实例详情”页面中的“重置密码”按提示即可完成密码的重置。

密码重置不成功，有密码规则吗？

目前重置密码处的密码规则为：8-32位字符，大写字母、小写字母、数字、特殊符号至少存在三种，符号仅限!@#$%^*()。

为什么我的 BCC 无法重置密码

出于安全方面的考虑，在控制台页面重置密码时对密码的规则要求比较复杂，必须是：

- 8-32位字符
- 大写字母、小写字母、数字、特殊符号至少存在三种
- 符号仅限!@#$%^*()

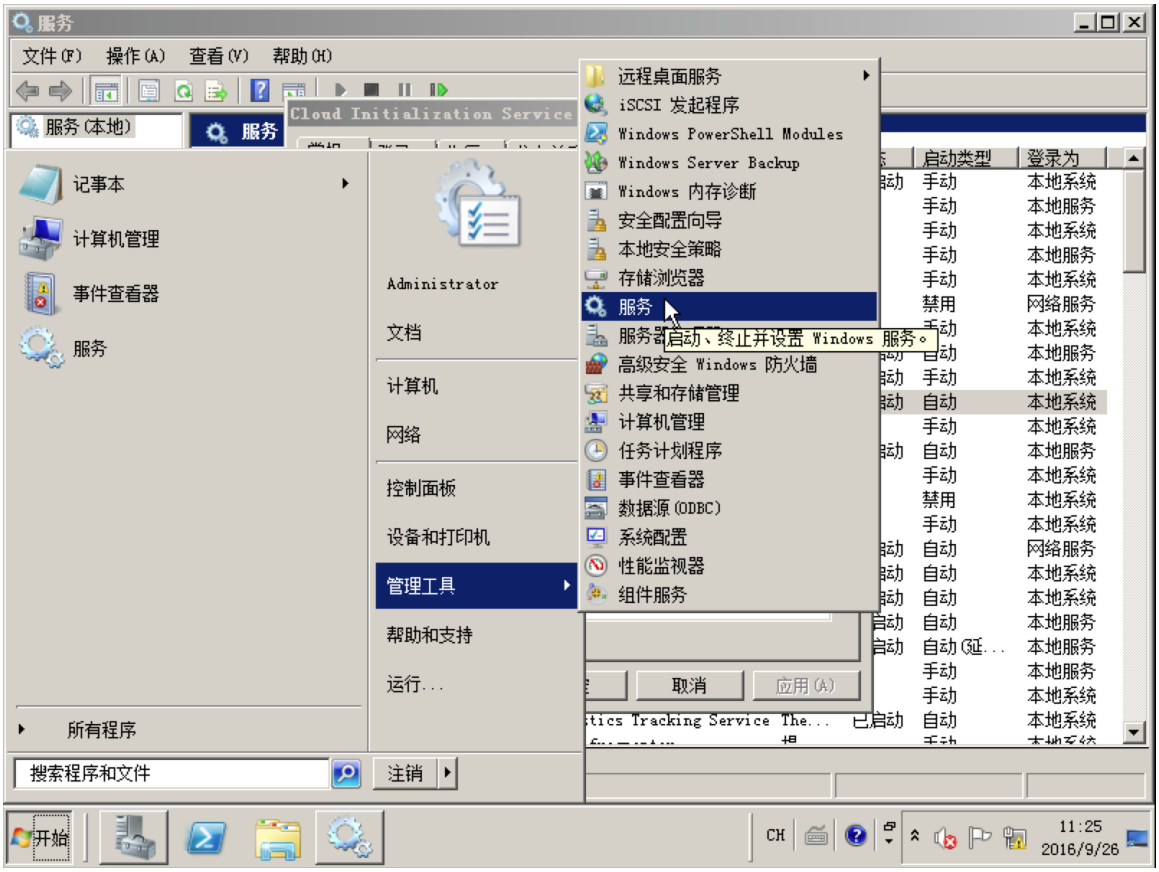
请您检查您指定的新密码是否符合此规则。特别提醒：特殊符号只能使用规则中列举出来的几个。

重置 Windows 虚机密码后新密码没有生效？

由于Windows 虚机重置密码操作依赖 cloudinit 服务，所以当 cloudinit 服务被禁，或者在虚机内安装安全管理类软件时限制了 cloudinit 服务运行，都可能导致重置密码失败。

为了确保重置密码可以生效，首先检查系统内的cloudinit服务是否可以正常运行：

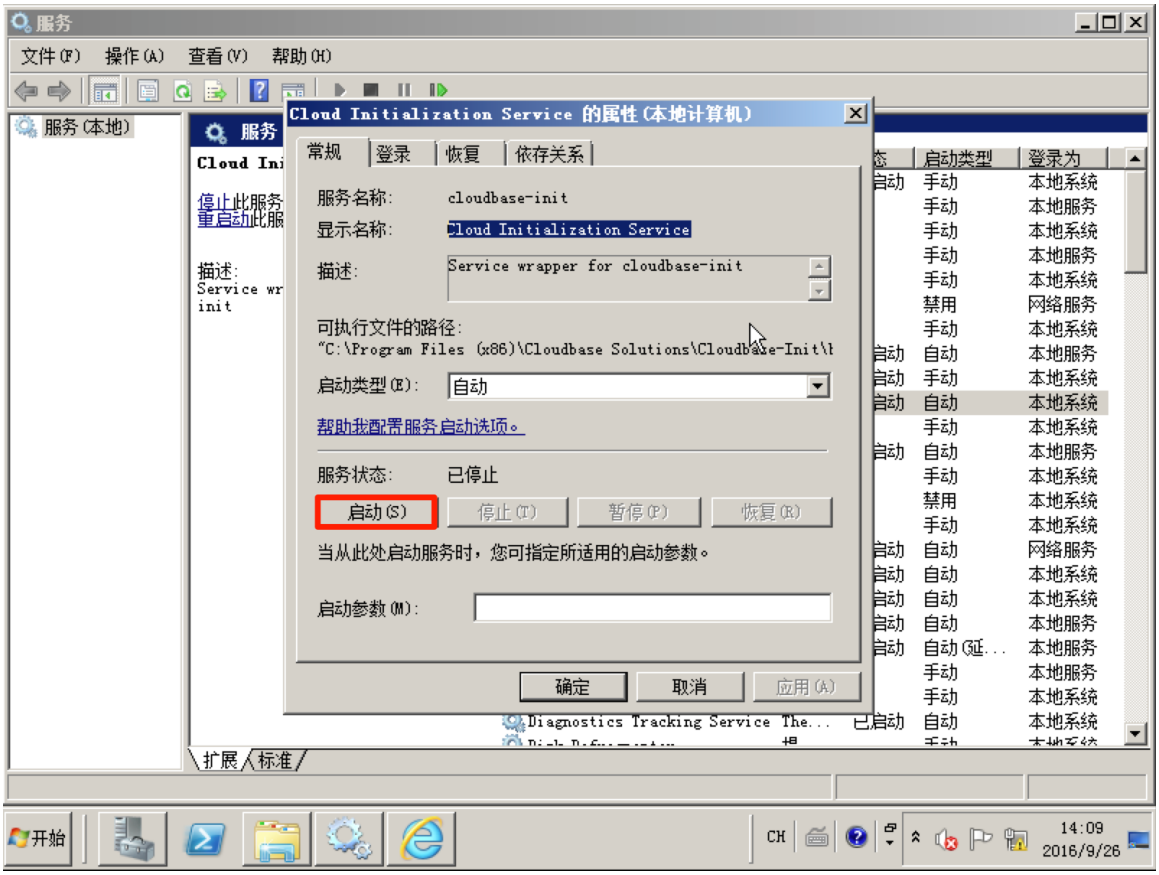
1.登录 Windows 虚机，点击“开始”->"管理工具"->"服务"。



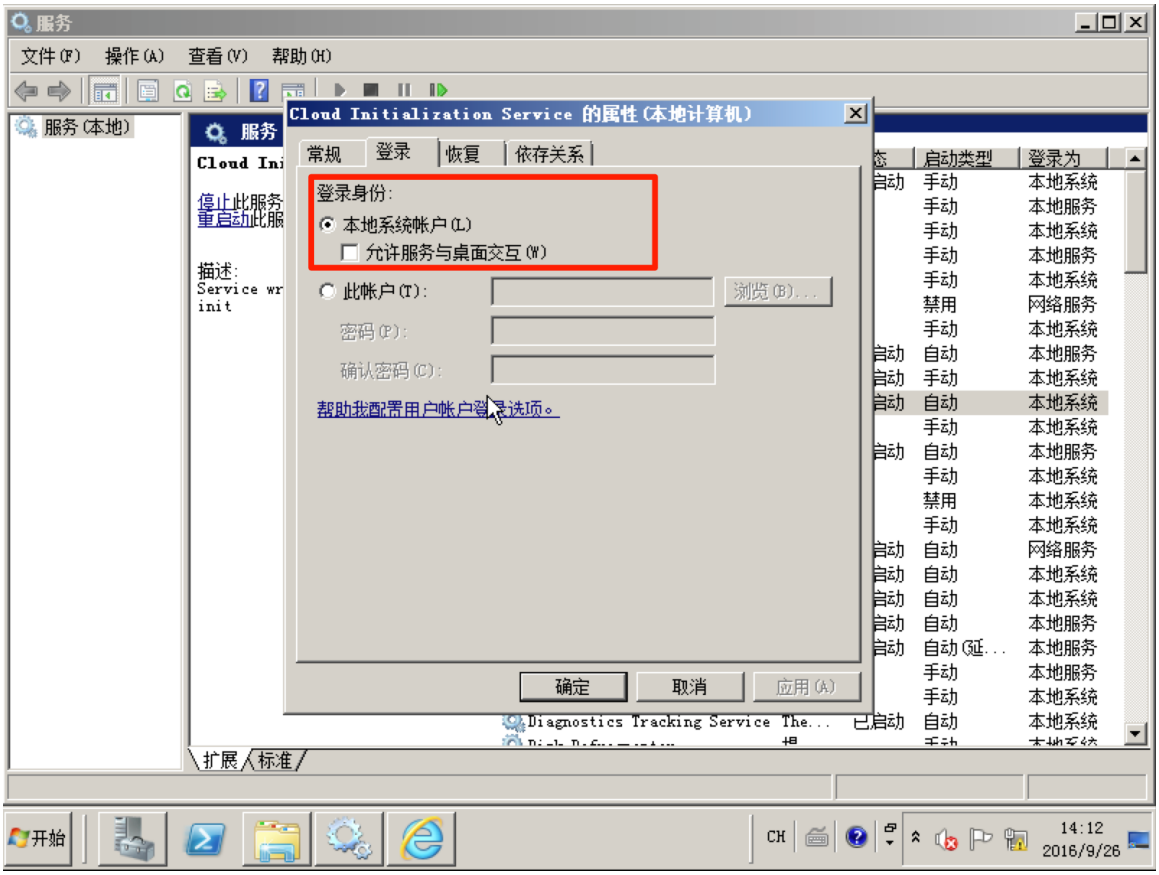
2.双击“Cloud Initialization Service”操作，在弹框中点击“启动”按键。

- 如果 cloudinit 正常启动，说明当前服务正常，可以继续返回BCC 控制台重置密码。

- 如果 cloudinit 启动时报错，说明当前服务出现错误，跳转到第3步骤。



3.切换“登录”页签，检查cloudinit服务的登录身份，保证当前身份为“本地系统账户”。否则可能由于权限问题导致cloudinit服务无法正常运行。



4.如果仍然无法使用新密码登录 windows 虚机，请填写工单说明详情。

> **注意：**
>
> 在安装安全类软件时，请在安装之后，主动“启动” 一次cloudinit服务（参考步骤2），这时如果安全软件会有一些弹窗提

> 示，提示cloudinit服务在调用系统的命令。请务必选择通过，否则cloudinit无法正常调用系统命令，并导致服务不可用。

服务器密码设置简单被入侵，杀毒后还是不行怎么办？

可以在百度智能云控制台上重设密码，进入 BCC 的实例详情页面后，可以看到“重置密码”的操作。

如果系统中已经有病毒或者木马，可以选择“重装操作系统”来清理系统，注意这会导致系统盘数据的丢失！

同时建议检测程序中的漏洞及后门，以减少被入侵的可能。

如何对BCC资源状况进行定期的全面检查？

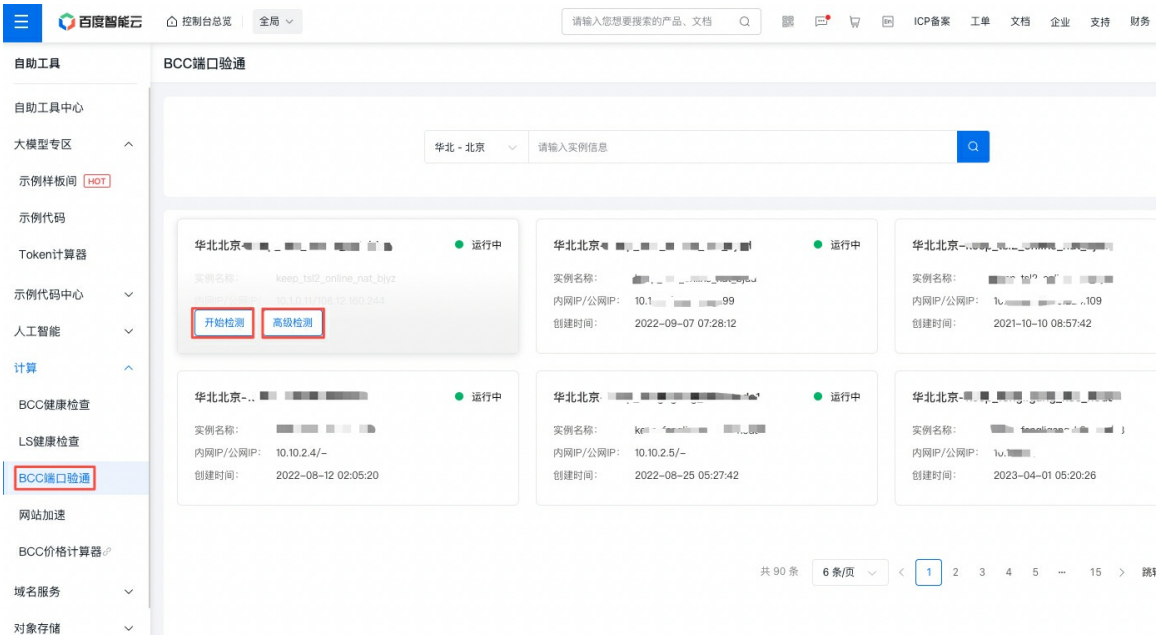
您可开通“云顾问 Cloud Advisor”服务，定期获得云上资源在安全、可用性、性能、成本方面的检查报告。

报告中包含若干BCC相关检查项目，如：BCC-主机安全检测结果、BCC-低使用率、BCC-共享镜像、BCC-可用区分布、BCC-安全检测服务SRD检测结果、BCC-安装主机安全客户端、BCC-实例停用、BCC-操作系统版本、BCC-高使用率等。

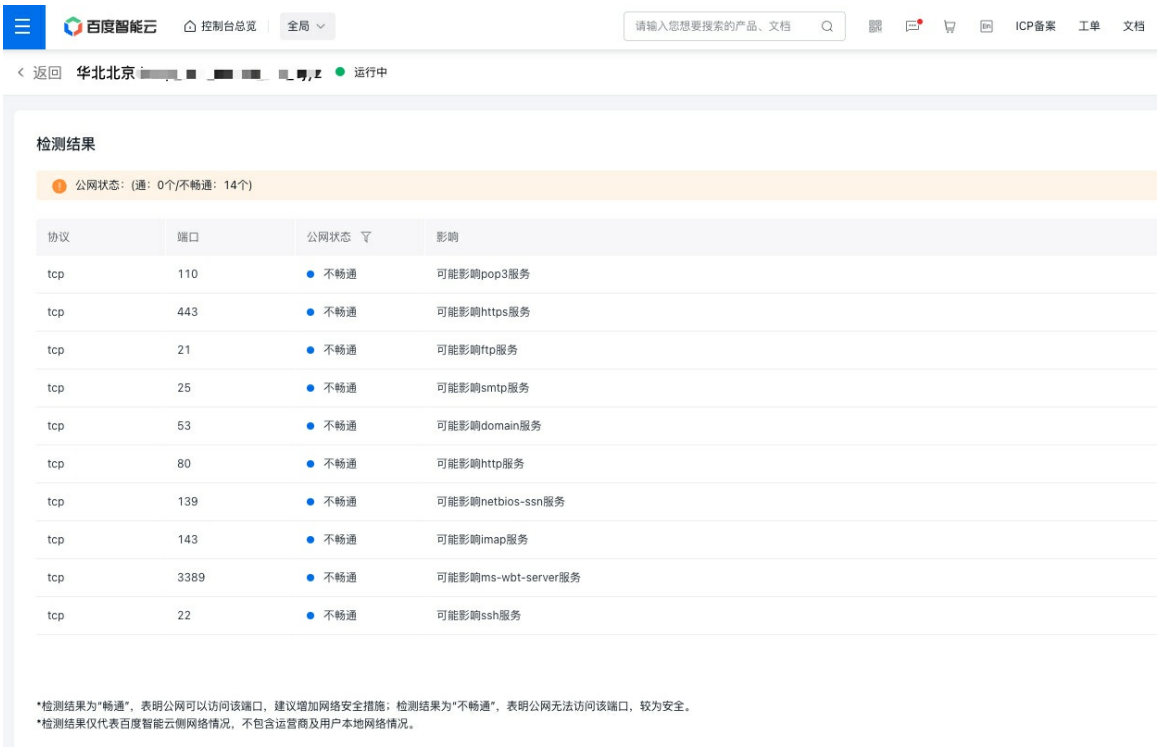
了解或开通云顾问服务，请前往 云顾问主页

怎么快速查看BCC开放了哪些端口，端口是否畅通？

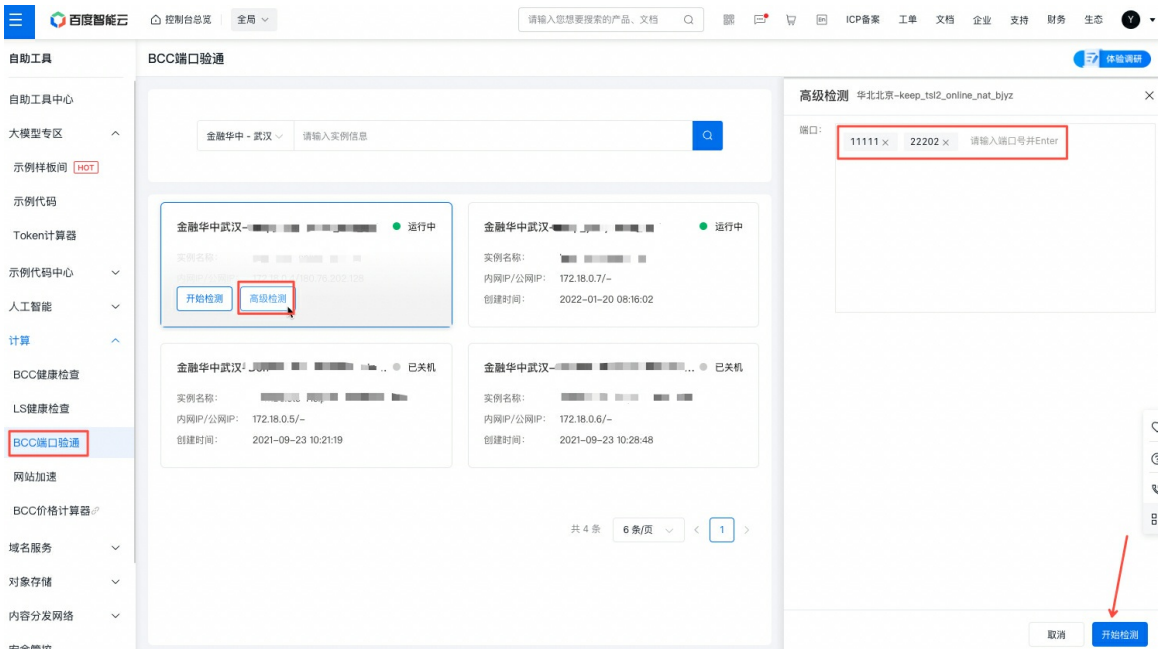
要快速查看当前BCC服务器开放了哪些端口，可以使用BCC端口验通工具进行快速检测



- 普通检测：点击开始检测即可检测一些常用的端口是否畅通



- 高级检测：点击高级检测可以自定义想要检测的端口，可以选择多个端口。



怎么添加IP登录白名单?

百度智能云对于在短时间内多次输入错误密码的IP会进行拦截处理，如果您认为是自己的IP被拦截了，想要解封，可以到控制台搜索找到主机安全产品，选择配置管理-登录管理设置，点击添加白名单，将自己的本地公网IP添加进去即可解除拦截。

一般10分钟内生效，如果10分钟后依然无法远程连接，那么说明并非我方拦截导致，建议您提交计算与网络/云服务器BCC工单，详细描述您的问题，提供实例ID，实例IP，远程端口，我方会有工程师为您排查解决。

登录 Windows BCC需要“网络级别身份验证”的功能？

方法一：出于安全方面的考虑，Windows系统的云服务器上远程桌面默认设置开启了“仅允许运行使用网络级别身份验证的远程桌面计算机连接”。

本地的 Windows 服务器必须支持“网络级别身份验证”功能，否则无法登录 Windows 云服务器 。检查的方法如下： a.启动远程桌面连接 b.鼠标右击窗口标题栏，然后点击“关于” c.在弹出窗中，需要看到有“支持网络级别的身份验证”的字样 如果在上述操作中没有看到“支持网络级别的身份验证”的字样，就需要安装相应的补丁包以支持此功能。

方法二：禁用“网络级别身份验证”功能。 首先需要用户使用VNC远程登录Windows BCC服务器: a.运行中输入 gpedit.msc 后进入本地组策略设置； b.将 计算机配置--管理模版--Windows组件--远程桌面服务--远程桌面会话主机--安全--要求使用网络级别的身份验证对远程连接的用户进行身份验证--设置禁用； c.打开 计算机--属性--高级系统设置--远程 ，远程桌面中选择“允许运行任意版本远程桌面的计算机连接（较不安全）”。

新建了一个安全组，把原来一个实例给关联了，但是在默认安全组里还有。该如何处理？

1. 当所有BCC实例取消对安全组管理时才可以删除安全组，安全组列表的关联实例数量一栏显示当前有 多少BCC实例关联该安全组。
2. 云服务器是可以关联多个安全组的，举个简单的例子：比如安全组A只允许访问22端口，安全组B允许ping，这样叠加安全组A和B就既可以访问22端口也可以ping。当您关联了新的安全组后可以在云服务器实例详情页面的安全组页里手动取消不想关联的安全组

> **注意：**云服务器至少要关联一个安全组。

如何防止 DDOS 攻击 和 CC 攻击？

百度智能云提供免费的BSS(Baidu Security Service)产品，为BCC提供CC和DDoS的攻击防护，用户可以结合自己的业务特点设置合适的流量清洗阈值。

被DDOS攻击的IP该如何处理？

BCC服务器的DDOS防护阈值目前默认为5Gbps（内地区域BCC5G，香港区域BCC为1Gbps），触发黑洞后IP将被运营商封禁24

小时，无法解封。

如果您想要更好的防护DDOS攻击，建议您使用我方的DDOS高防服务，详情请您参考DDoS高防IP

能否使几台BCC之间不需要输入密码也能通过SSH相互访问？

您可以通过密钥对的方式来进行实现，具体可以参考密钥对文档

如何预防服务器和网站受到黑客攻击？

服务器安全方面：

1.建议业务不需要的端口在防火墙或安全组设置拦截，不对外开放。请参考安全组文档

2.定期更改密码，增加密码复杂度

3.在操作系统内安装安全杀毒软件

4.使用自动快照策略定期创建快照。以便在出现问题时，可以及时回滚数据。请参考自动快照

其他安全方面更多的建议可以百度搜索关键字“安全加固”即可找到相应的教程，可供您参考

如何修复Dirty COW (CVE-2016-5195) Linux漏洞?

Linux内核的内存子系统在处理写入复制（copy-on-write, COW）时产生了竞争条件（race condition）。恶意用户可利用此漏洞以欺骗系统修改可读的用户空间代码然后执行，一个低权限的本地用户能够利用此漏洞获取其他只读内存映射的写权限。

- **漏洞编号**： CVE-2016-5195
- **漏洞危害**：黑客可以通过远程入侵获取低权限用户后，在服务器本次利用该漏洞在全版本Linux系统上实现本地提权，从而获取到服务器root权限。
- **影响范围**：该漏洞在全版本Linux系统（Linux kernel >= 2.6.22）均可以实现提权：

```
 CentOS 6.5  32位/64位
CentOS 7.1  32位/64位
Debian 7.5  64位
Debian 8.1  64位
Ubuntu Server 12.04.4 LTS 32位/64位
Ubuntu Server 14.04.1 LTS 32位/64位
```

- **漏洞修复方案**：

1.下载百度智能云提供的fix-tool-allinone.tar压缩包到服务器。

2.解压压缩包

```
root@linux ~]# tar xf fix-tool-allinone.tar.gz
```

3.进入`fix-tool-allinone`目录

```
 [root@linux ~]# cd fix-tool-allinone
[root@linux fix-tool-allinone]# ls
fix-tool-32-bit.tar.gz
fix-tool-64-bit.tar.gz
run.sh
```

4.运行run.sh

```
 [root@linux fix-tool-allinone]# ./run.sh
DirtyCow check and fix needs quite some time, please be patient.

pokeball
  (___)
  (o o)_____/
   @@ `     \
    \ ____, /miltank
    //    //
   ^^    ^^
mmap 7f804ec7f000

madvise 0

ptrace 0

Your kernel is 3.10.0 which IS vulnerable.
Updating linux kernel to fix DirtyCow vulnerablity!
Please confirm whether to continue the operation or not.[y/n]
```

该工具集成了 pokemon 程序，利用 DirtyCow 漏洞完成普通用户对具有只读权限的文件的修改。工具首先创建一个普通用户，并将测试程序以及程序使用的测试文件拷贝到普通用户的家目录；然后运行pokemon程序进行漏洞检测。

如果检测到存在漏洞，则会输出上述提示，用户输入'y'则会**升级内核**以修复该漏洞，输入'n'则直接退出。

5.升级内核后，用户需要自己手动重启服务器。如果用户有业务不方便立即升级内核，可以在检测完成后输入'n'，等服务器空闲时再进行升级。

虚拟机运维类

如何关闭windowsServer2008IE增强机制？

参考windowsServer2008关闭IE增强机制教程

如何查看Linux下默认的DNS？

先查看 `/etc/resolv.conf`，如果您的BCC有公网访问权限，可以执行`nslookup baidu.com` 命令来查看，关注输出结果中以`Server:`开头的内容。

Windows 2012 如何安装.net framework 3.5？

如果需要安装.net framework 3.5，请您暂时修改windows update的源，不使用百度智能云的源。

> **注意：**请您记录下原先的值，安装完毕后再修改回来。

1. 打开命令提示行cmd，输入“gpedit.msc” 以打开“组策略”。
2. 修改windows更新源，打开“计算机配置>管理模板>windows 组件>windows 更新”，双击“指定 Intranet Microsoft 更新服务位置”，复制保存一下“设置检测更新的Intranet更新服务”和“设置Intranet统计服务器”中的值。
3. 选择“未配置”，然后点击“确定”。
4. 命令提示行cmd输入下面命令更新组策略：“gpupdate”。
5. 打开服务器管理器重新安装.net framework 3.5组件，安装过程需要从网络下载安装文件，可能比较缓慢，请您耐心等待。
6. 安装完毕后请您再按照上述步骤2，3把windows update的源修改为百度智能云的源。

cloudbase-init 账户是做什么用的，可以删除吗，删除后有什么影响等?

此账户为云平台相关的账户，如删除会影响云平台的管理功能，建议您不要修改/删除此账号，该账户在 Linux 系统中不会存在。

如何同步设置虚拟机系统时钟？

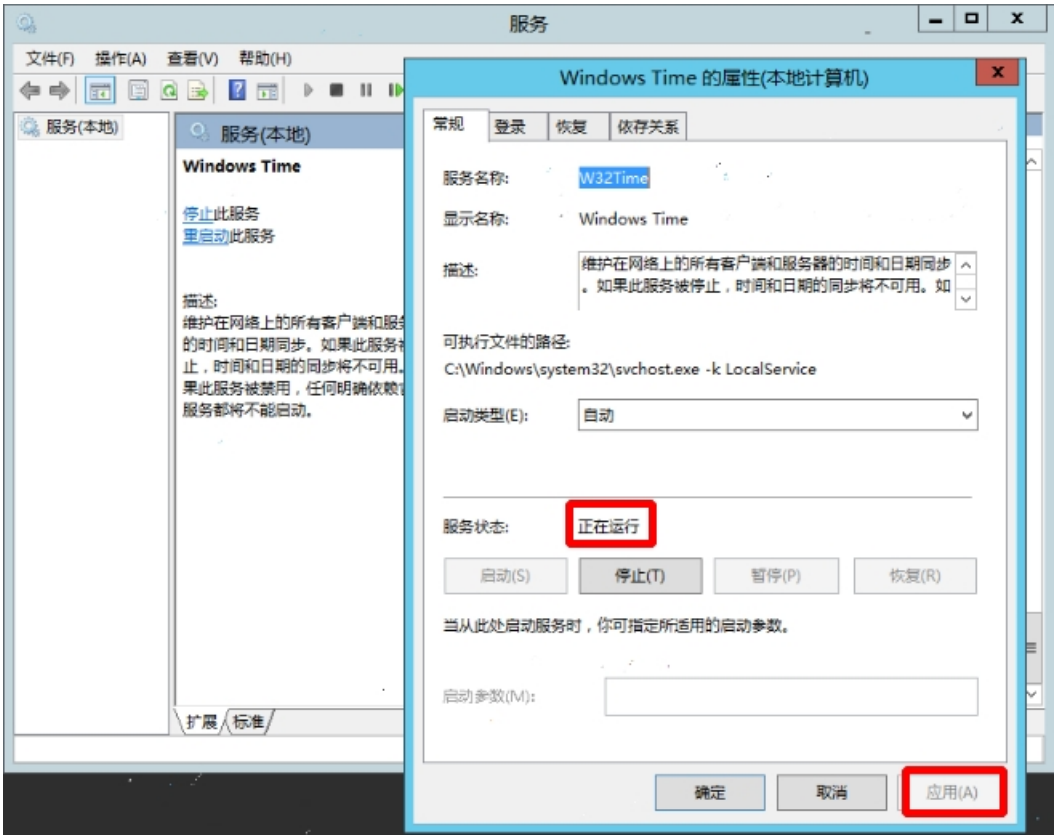
公有云虚拟机采用网络时间协议（Network Time Protocol，NTP）来同步网络中各个虚拟机的时间。一些对系统时间比较敏感的行业，要求一致的系统时间，避免因为时间误差引起的问题。

下面以公有云Windows Server 2012 R2系统为例，来展示NTP服务的设置。

时间服务

Windows操作系统默认开始Windows Time服务。 可以通过以下方式来确认时间服务是否开启

1. 在 “开始” 按钮单击右键， 选择 运行，输入 Services.msc, 会弹出服务面板。
2. 在 “服务” 窗口中选择 “Windows Time” 服务， 右键 >属性 。



3. 在属性窗口, 状态应该是 “正在运行”， 如果不是，点击 “启动” ,然后点击 “应用” 即可。

设置NTP时间服务器地址

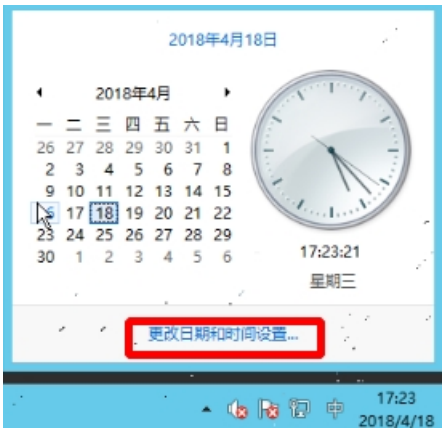
Windows操作系统默认使用的是微软的NTP时间服务器 time.windows.com，在百度的公有云虚拟机中使用了公有云搭设的NTP服务器 ntpsr.baidubce.com。

NTP服务器的设置方法如下:

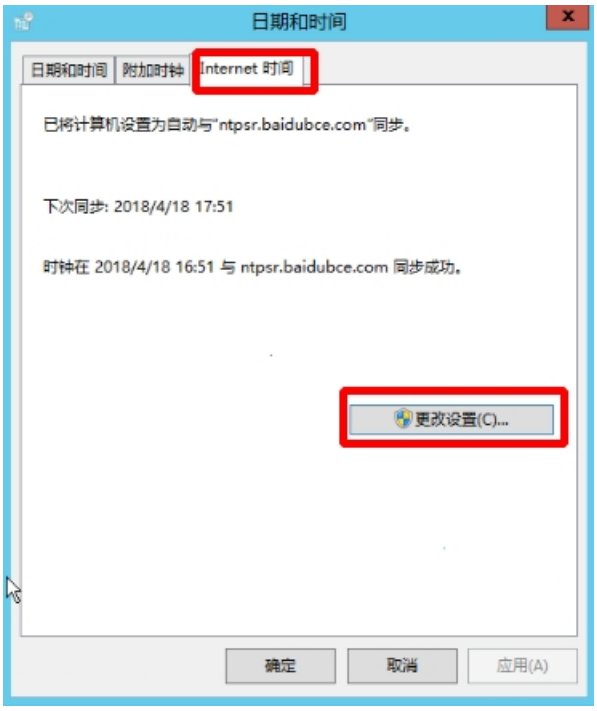
1. 在 开始 按钮单击右键， 选择 运行，输入 gpedit.msc;
2. 在策略管理器中选择 管理模板-->系统–>windows时间服务–>时间提供程序;
3. 双击“配置 windows NTP 客户端”，选择“已启用”，类型选择为NTP，NtpServer填写ntpsr.baidubce.com;
4. 双击“启用 windows NTP 客户端”，选择“已启用”;
5. 双击“配置 windows NTP 服务器”，选择“已禁用”;

通过界面设置NTP时间服务器地址

1. 打开系统时间， 选择 更改日期和时间设置;



2. 在 日期和时间 窗口，选择 Internet时间 tab页， 选择 更改设置;



3. 输入百度智能云的ntp server地址：`ntpsr.baidubce.com`即可;

Linux NTP服务配置

Linux的时钟同步采用ntp或chrony。

1. 如果使用ntp进行时钟同步，配置文件位于/etc/ntp.conf：
   - 修改server配置配置即可，默认配置为：server ntpsr.baidubce.com;
   - 同时修改/etc/ntp/step-tickers，默认配置为：ntpsr.baidubce.com;
2. 如果使用chrony镜像时钟同步，配置位于/etc/chrony.conf:
   - 同样修改server配置即可：默认配置为：server ntpsr.baidubce.com iburst

云服务器 BCC Centos6 系统下iptables 服务的使用规则？

Centos6 默认使用iptables配置防火墙，本文简要介绍 iptables 防火墙常见命令:

设置iptables 开机启动

目前**Centos6.x**系列的镜像都预装了iptables服务，默认关闭状态；如果需要开机自动启动，需要执行如下命令：

```
chkconfig iptables on
```

iptables 常用命令

- 查看filter表

```
iptables -L -n --line-number
```

说明：如果不加-t的话，默认就是filter表；操作其他表，需要添加 -t 命令。

- 查看nat表 iptables -t nat -vnL POSTROUTING --line-number
- 修改规则 iptables -R INPUT 3 -j DROP //将规则3改成DROP
- 删除iptables规则

```
 iptables -D INPUT 3      //删除input的第3条规则
iptables -t nat -D POSTROUTING 1   //删除nat表中postrouting的第一条规则
iptables -F INPUT        //清空 filter表INPUT所有规则
iptables -F          //清空所有规则
iptables -t nat -F POSTROUTING    //清空nat表POSTROUTING所有规则
```

- 设置默认规则 iptables -P INPUT DROP //设置filter表INPUT默认规则是 DROP
  service iptables save

云服务器BCC Centos7 系统下 firewalld 服务的使用规则？

Centos7 默认预装了 firewalld 服务，采用 firewall-cmd 命令或者 firewall-config 图形界面，可以动态管理 kernel netfilter 的临时或永久接口规则，规则更新实时生效，不需要重启 firewalld 服务。

开启 firewalld 服务

```
systemctl start firewalld
```

设置firewalld 开机启动

```
systemctl enable firewalld
```

查看服务状态

```
systemctl status firewalld
```

常用命令

- 查看运行状态

```
firewall-cmd --state
```

- 查看激活的zone

```
firewall-cmd --get-active-zones
```

- 查看指定zone的所有信息 firewall-cmd --zone=public --list-all

- 查看永久放行的服务

```
firewall-cmd --get-service —permanent
```

- 拒绝所有包 # firewall-cmd --panic-ond
- 取消拒绝状态

```
# firewall-cmd --panic-off
```

- 查看是否拒绝

```
$ firewall-cmd --query-panic
```

- 更新防火墙规则，不需要重启服务

```
# firewall-cmd --reload
```

- 更新防火墙规则，并且重启服务

```
# firewall-cmd --complete-reload
```

- 添加接口到指定zone，默认都在public zone

```
# firewall-cmd --zone=public --add-interface=eth0
```

如果不加--permanent参数，则该规则仅仅是运行时规则，firewall-cmd --reload 后该规则消失。

- 设置默认zone，如果不设置，默认为public，本命令立即生效无需重启 # firewall-cmd --set-default-zone=myzone
- 查看给定zone所有打开的端口： # firewall-cmd --zone=myzone --list-ports
- 添加一个端口到区域

```
# firewall-cmd --zone=myzone --add-port=8080/tcp
```

如果不加--permanent参数，则该规则仅仅是运行时规则，firewall-cmd --reload 后该规则消失。

- 打开一个服务 # firewall-cmd --zone=work --add-service=ssh
- 移除服务

```
# firewall-cmd --zone=work --remove-service=ssh
```

您还可以访问 firewalld 官方文档查询相关命令：http://www.firewalld.org/documentation/

云服务器BCC ubuntu/debian系统下Ufw 服务概述

Ubuntu/Debian 默认使用Ufw服务，本文简要介绍 Ufw 防火墙常见命令:

Ufw开机启动

```
ufw enable
```

查看服务状态

```
ufw status
```

Ufw常用命令

```
 sudo ufw allow ssh        //允许ssh服务，即所有的外部IP可以访问本机的22/tcp (ssh)端口
sudo ufw delete allow      //ssh 删除上面的ssh访问规则
sudo ufw allow 22      //允许外部访问22端口(tcp/udp)
sudo ufw allow from 192.168.1.12    //允许此IP访问所有的本机端口
sudo ufw deny ssh      //禁止外部访问ssh服务
sudo ufw allow 2290:2300/tcp   //打开 2290到2300 的端口以供 tcp 协议
sudo ufw reset       //删除/重置所有的规则
sudo ufw status numbered     //列出所有的规则以及规则对应的序号
sudo ufw delete 1      //删除序号为1的规则
```

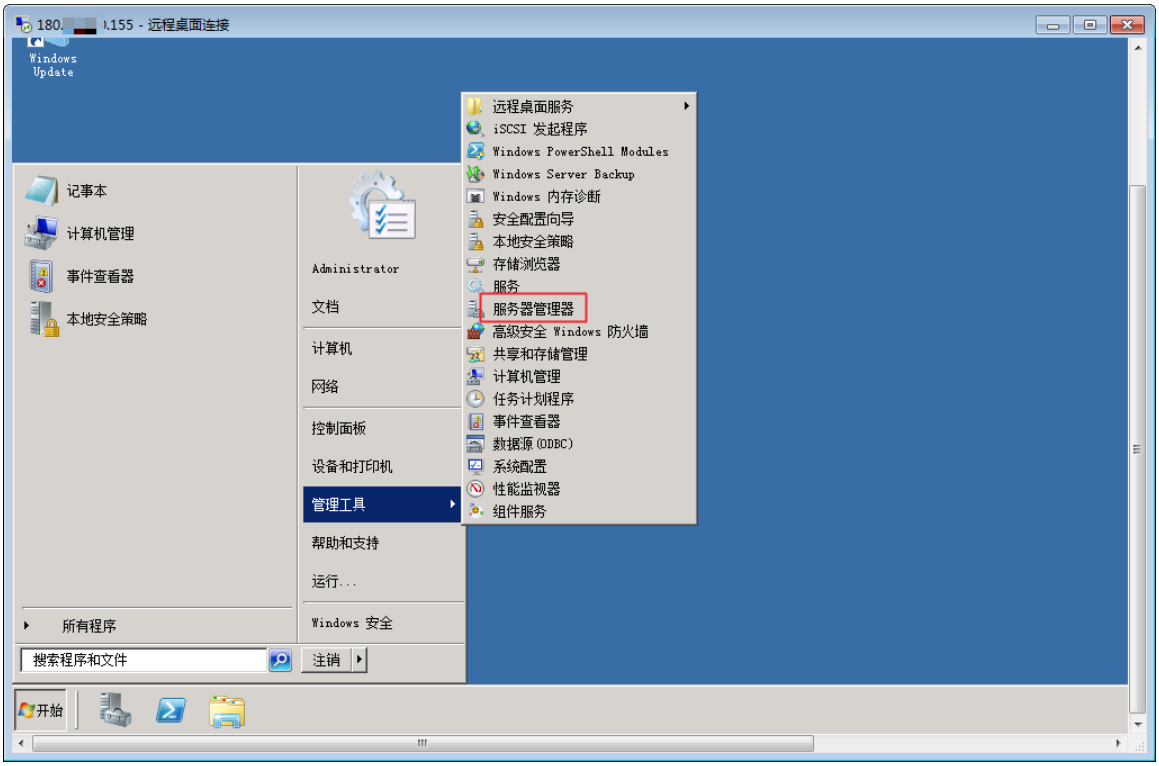
您还可以访问 ufw社区帮助文档：https://help.ubuntu.com/community/UFW

BCC Windows Server 如何配置IIS和FTP？

在Windows Server 2008 R2以及Windows 2012 系统上安装使用IIS和FTP，步骤如下：

Windows server 2008

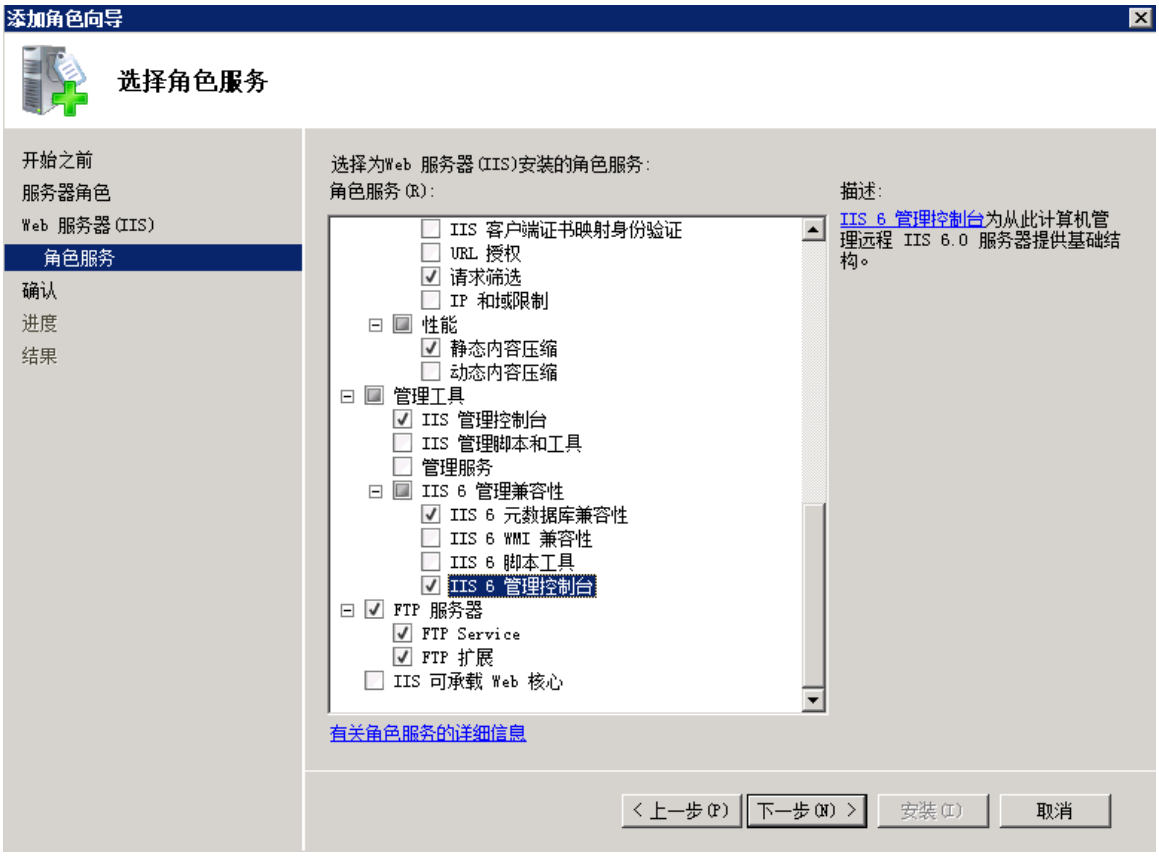
1. 远程登录 BCC 服务器，点击开始-管理工具—服务器管理器。



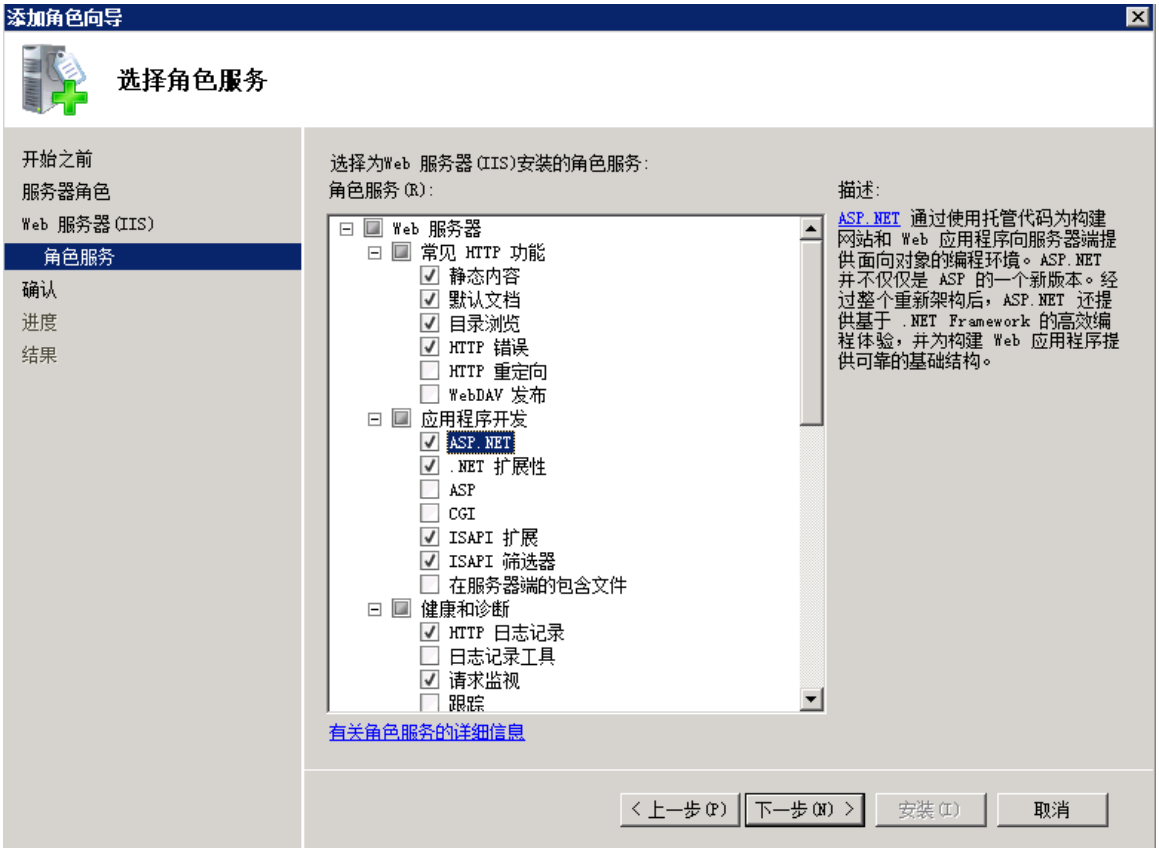
2. 打开服务器管理器，找到添加角色，然后点击，弹出添加角色对话框，选择下一步。

3. 在“添加角色向导”中下一步，配置服务器角色，选择web 服务器。

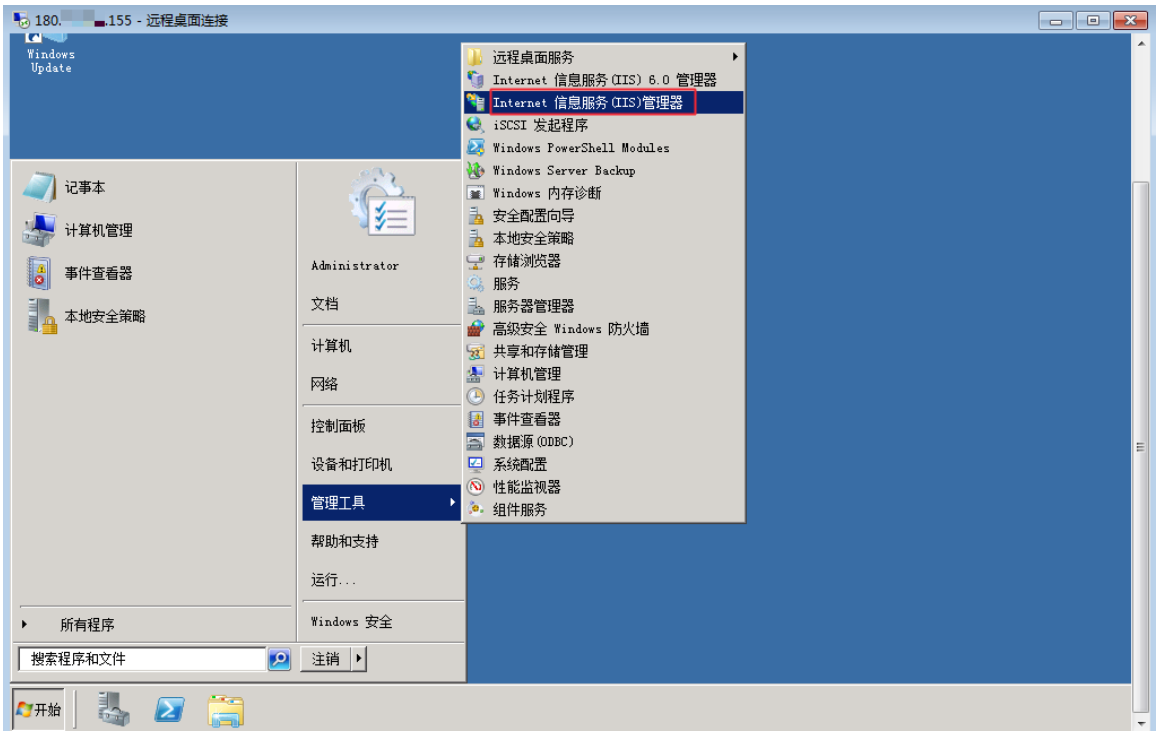
4. 点击“下一步”，选择FTP服务器和FTP管理控制台”以及应用程序环境，应用程序开发一般都是ASP和.NET。



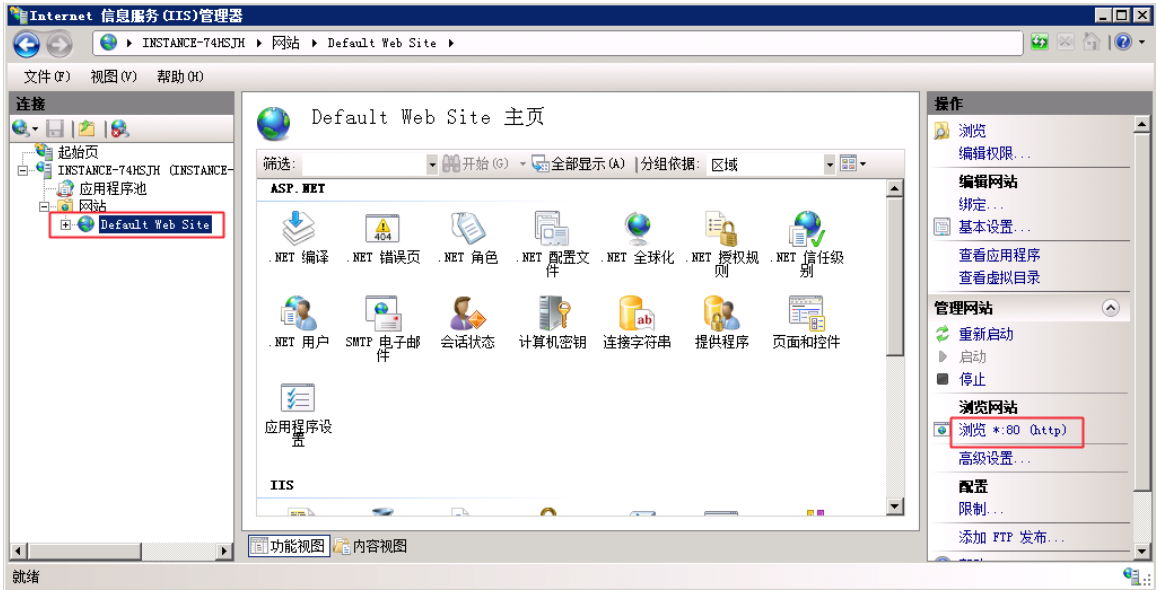
5. 确认角色和功能等配置，点击“安装”。



6. 安装成功后，选择开始-管理工具-IIS管理器。

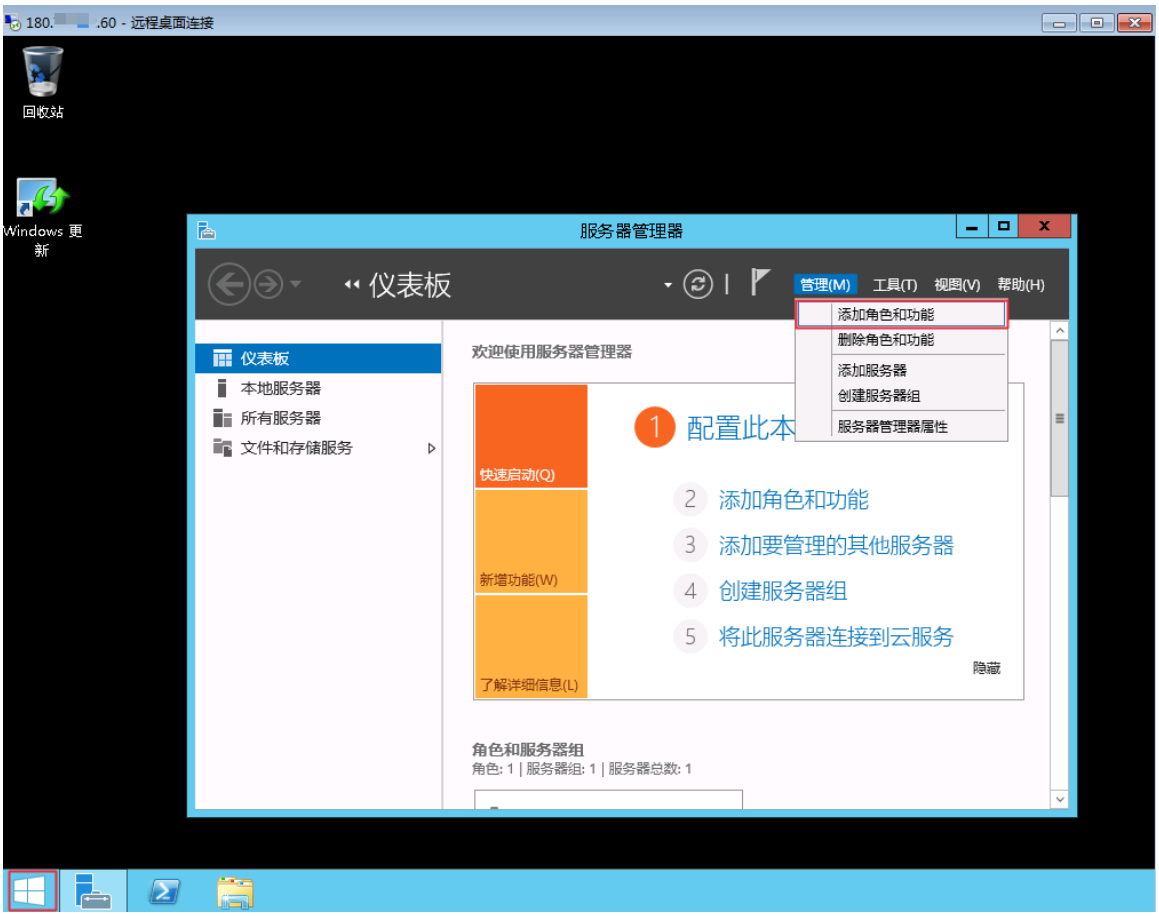


7. 在IIS管理中，可以看到IIS的默认站点，点击浏览网站，可以在浏览器中打开IIS的默认页面，IIS安装完成。

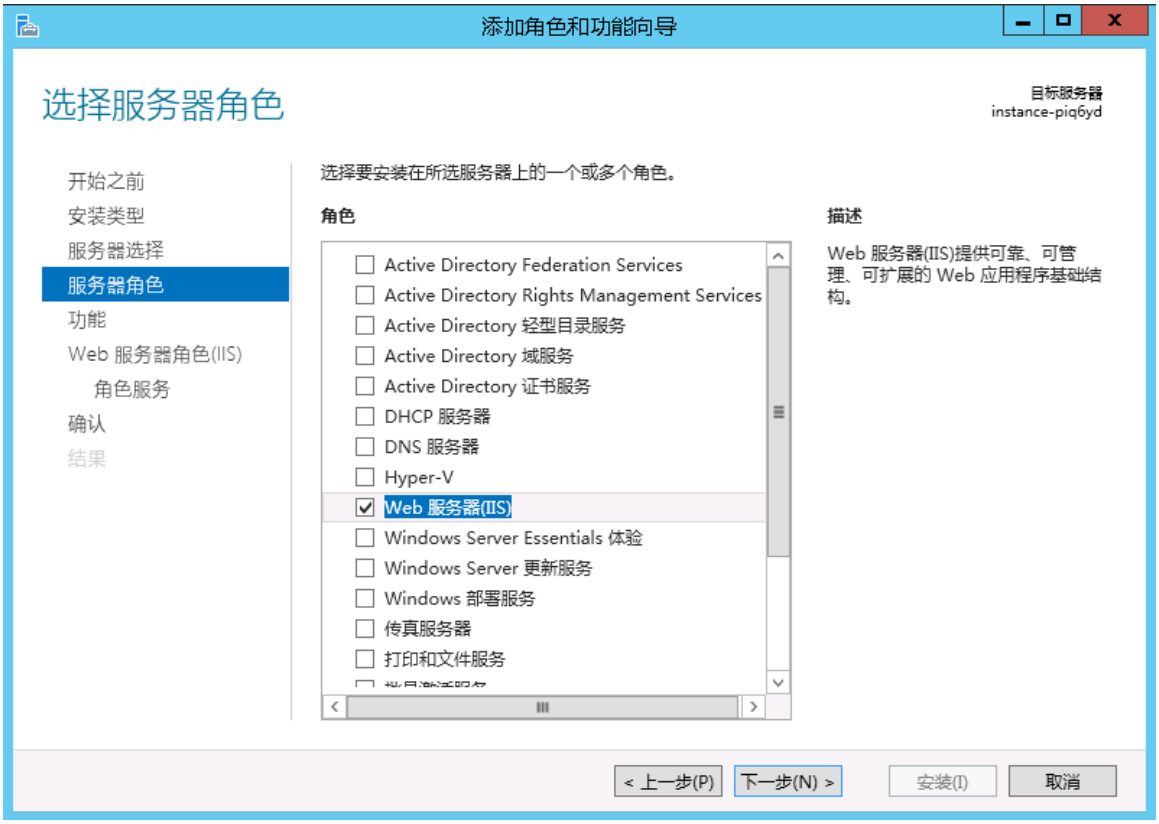


Windows server 2012

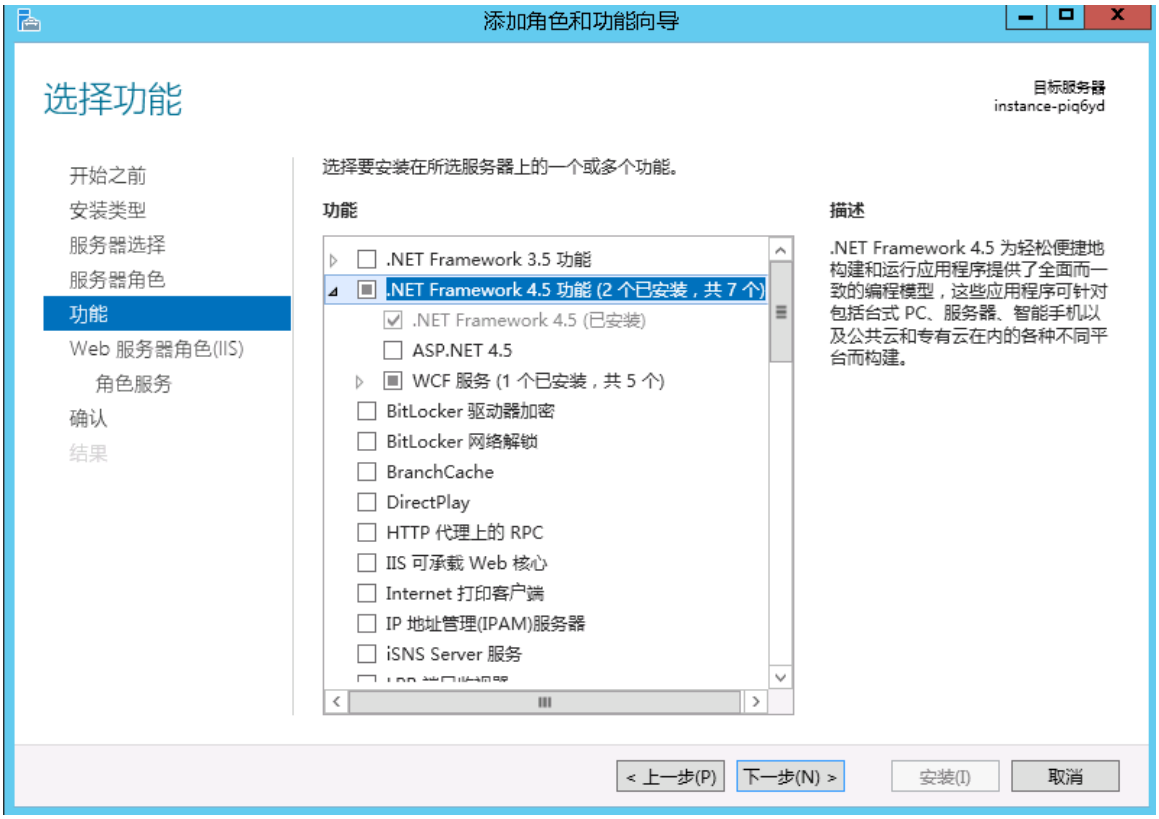
1. 远程登录 BCC 服务器，点击开始-服务器管理器，选择管理-添加角色和功能。



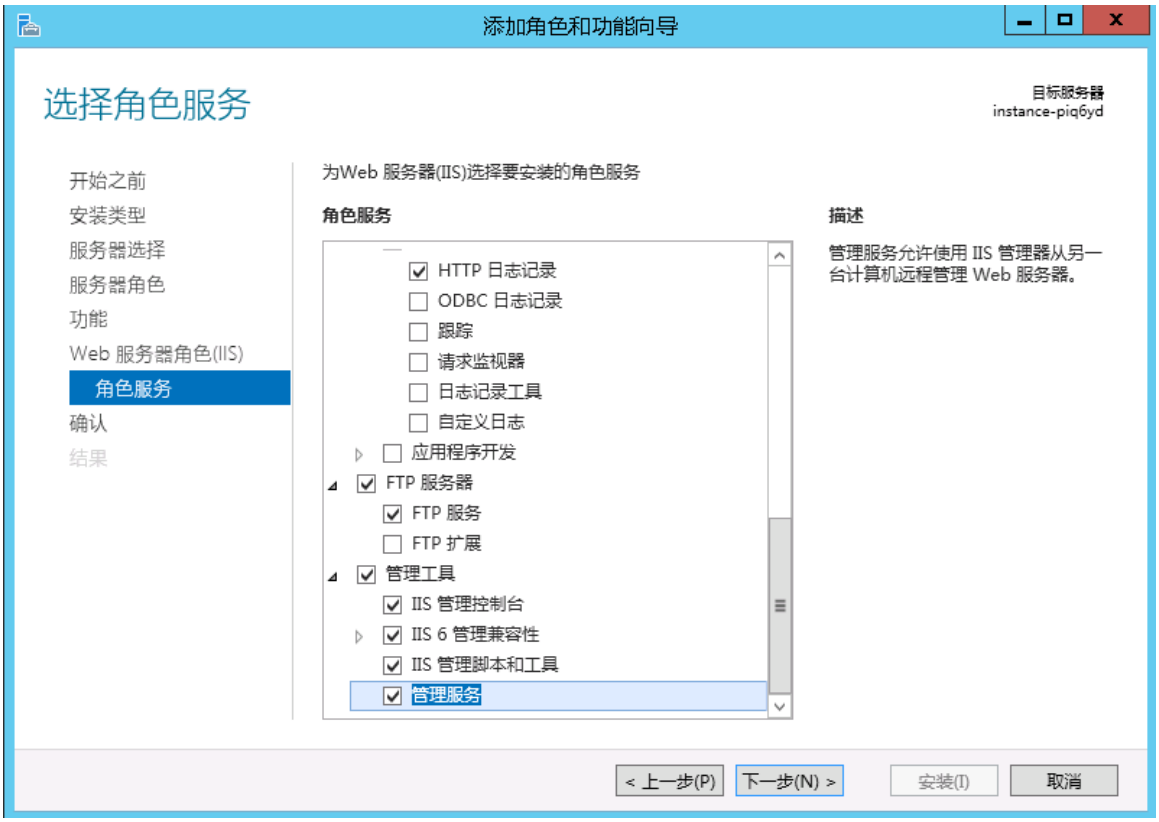
2. 进入添加角色和功能向导，选择服务器角色-Web服务器(IIS)。



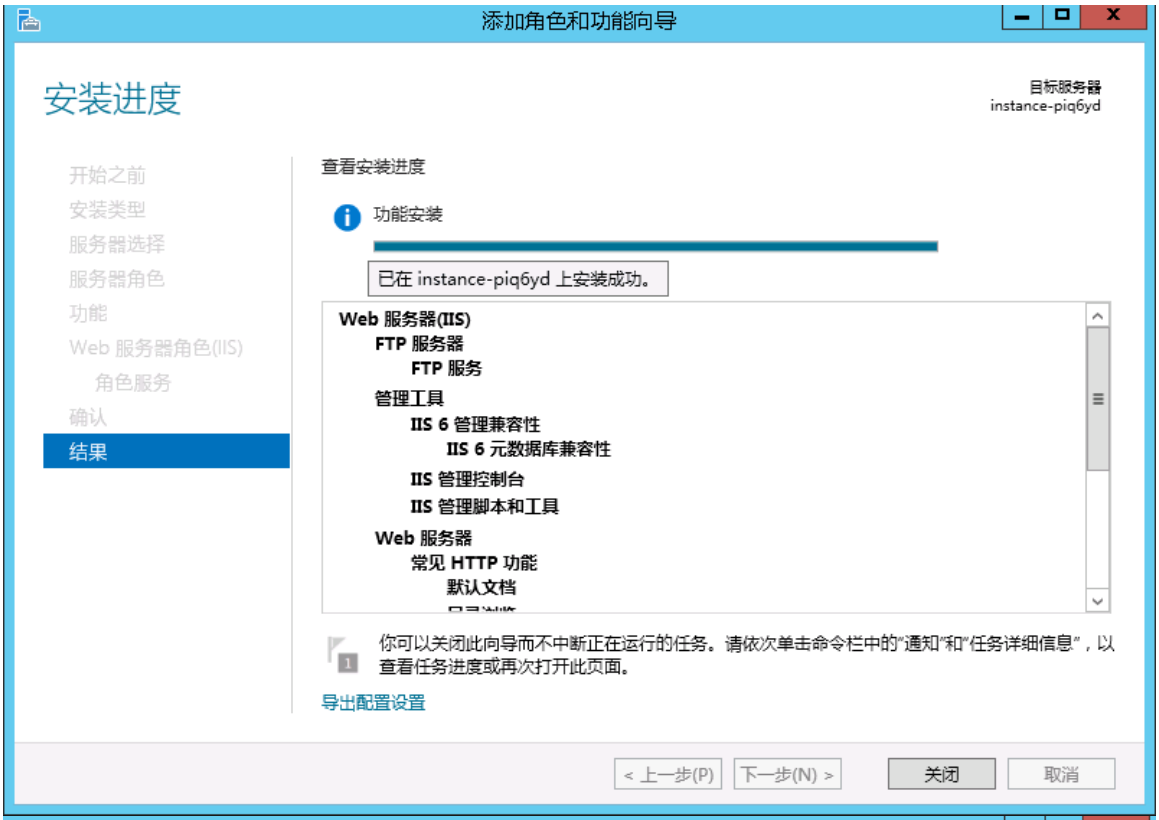
3. 根据需要选择安装对应功能，点击下一步。

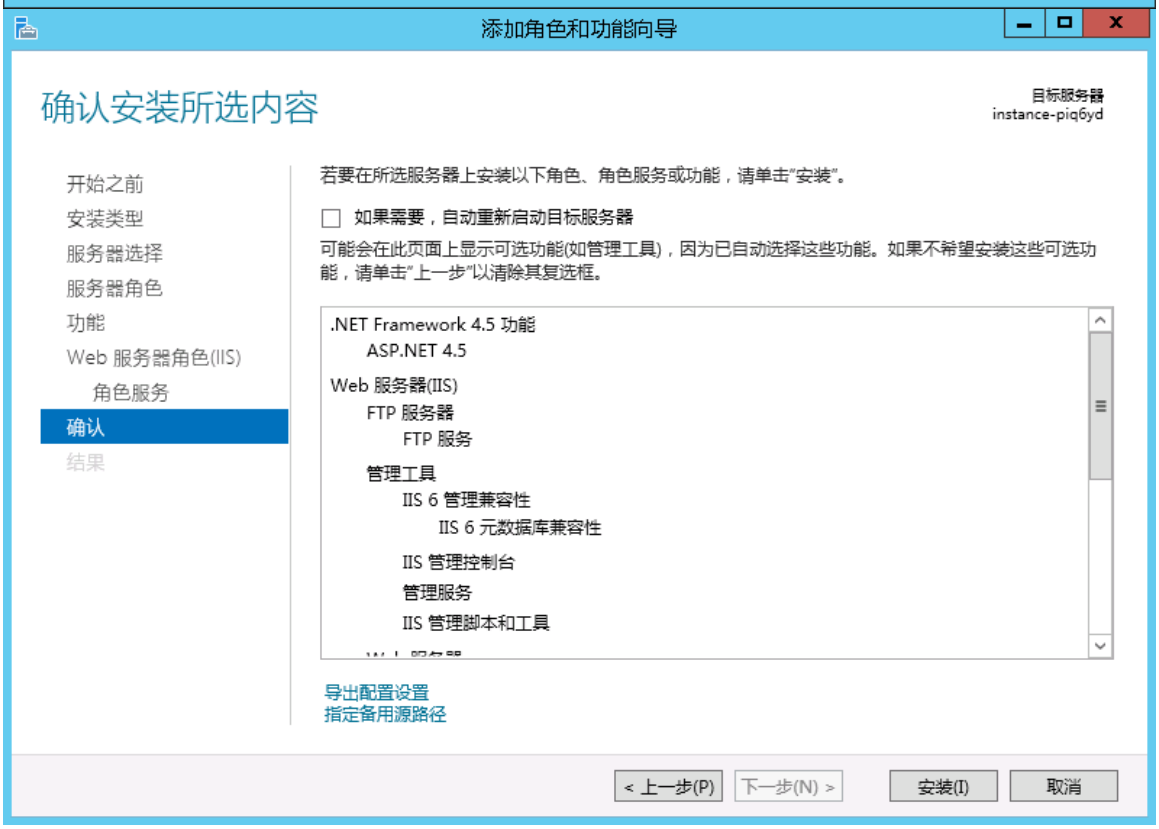


4. 安装角色服务，选择对应的安全性选项，其中FTP 服务器和管理工具项需要必选，点击下一步。



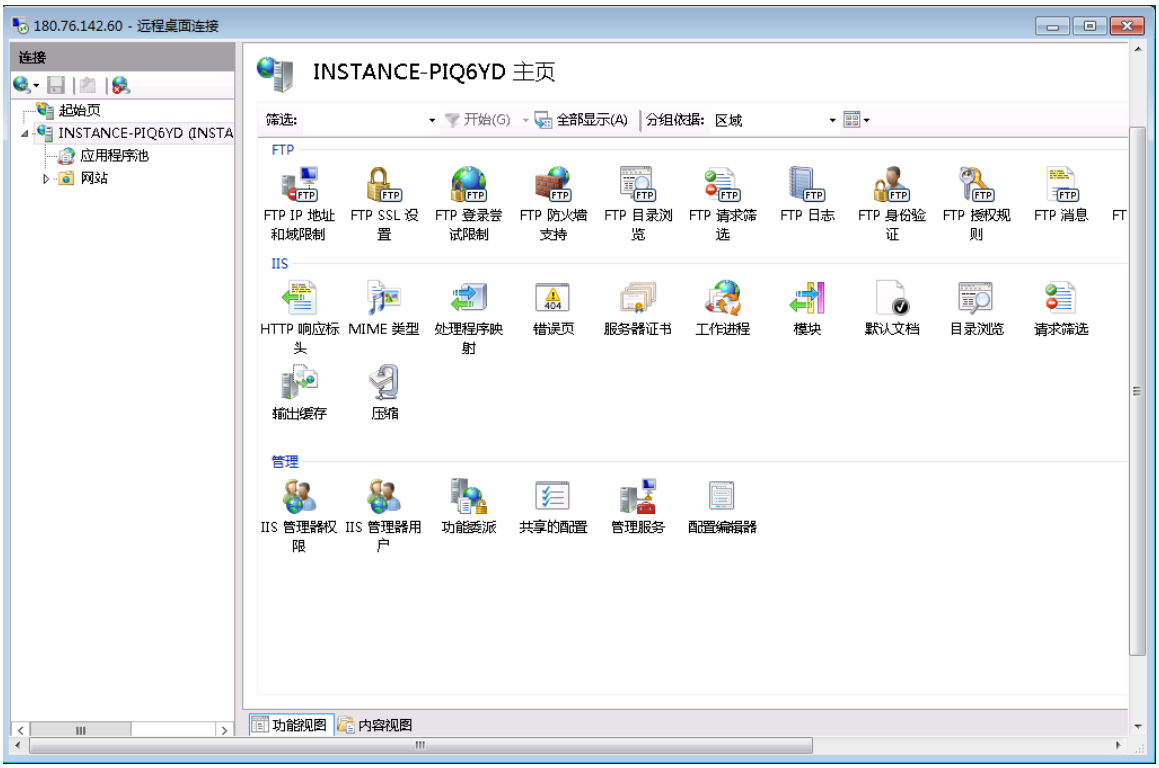
5. 点击安装，等待安装完成。





6. 安装完毕，点击工具-IIS管理器。

7. 打开了IIS管理器，看到应用程序池，网站，即在 Windows2012 安装IIS成功。



8. 测试以上建立的ftp服务器是否成功，可以在浏览器上输入公网地址ftp://180.xx.xxx.xx，支持登录访问则说明建立成功。

FTP上传经常中断怎么办？

首先要确认您使用的是主动模式（Port）还是被动模式（PASV）。

- 如果为主动模式，需要确认您在服务器上开放了高端端口，并且在FTP Server中指定了这些端口；
- 如果使用的是被动模式，则需要您本地可开放相关端口用于数据交互，但如果您的网络结构受限（如防火墙、路由限制），可能造成无法正常使用，出现频繁连接中断；
- 如果排除以上问题，请确认您本地的网络环境是否稳定，同时您的服务器带宽是否被占满；
- 如果排除以上问题，请确认在 ftp server 端（也就是 BCC 服务器上）的“数据传输超时”的设置足够大：比如在 Linux 上使

用 vsftp 时，需要修改 `/etc/vsftpd/vsftpd.conf` 文件里的 `data_connection_timeout` 参数；在 Windows 系统里操作比较复杂，请参考这个链接：
http://www.iis.net/configreference/system.applicationhost/sites/sitedefaults/ftpserver/connections 。

如果超时的设置已经足够大，可以先按上述关于“ping网站有丢包怎么办？”里介绍的步骤测试本地网络的情况。如果排除了本地网络的问题，需要提交工单联系技术人员，并将测试本地网络的各种输出和结果一起上传到工单里。

预留实例券类

是否支持退订预留实例券？

预留实例券一经购买，将为您保留实例券对应资源，因此不支持退券退费。

是否能为已经创建过的BCC实例购买预留实例券？

可以，购买相同实例规格的预留实例券可以匹配已有的同规格后付费BCC实例

券实例和BCC实例的关系是怎样的关系？

预留实例券实例与BCC实例的生命周期独立存在，仅在计费时券会抵扣BCC实例出账，用户无需给被抵扣的BCC实例付费。

实例券到期了，是否会影响BCC实例？

实例券仅会抵扣同规格后付费实例出账，不会影响BCC实例资源的状态，您仍可以继续登陆、使用BCC实例，但是之前抵扣的BCC实例会按照后付费价格出账。

购买实例券了还需要给BCC实例付款吗？

在出账时，预留实例券会抵扣同券数量的同规格后付费BCC实例计费，这些后付费BCC实例在券生效期间不会产生费用，注意券仅抵扣BCC实例，不会抵扣BCC实例包含的CDS系统盘数据盘、EIP等资源。

故障处理

Windows服务器ping不通

故障现象

Windows云服务器ping不通

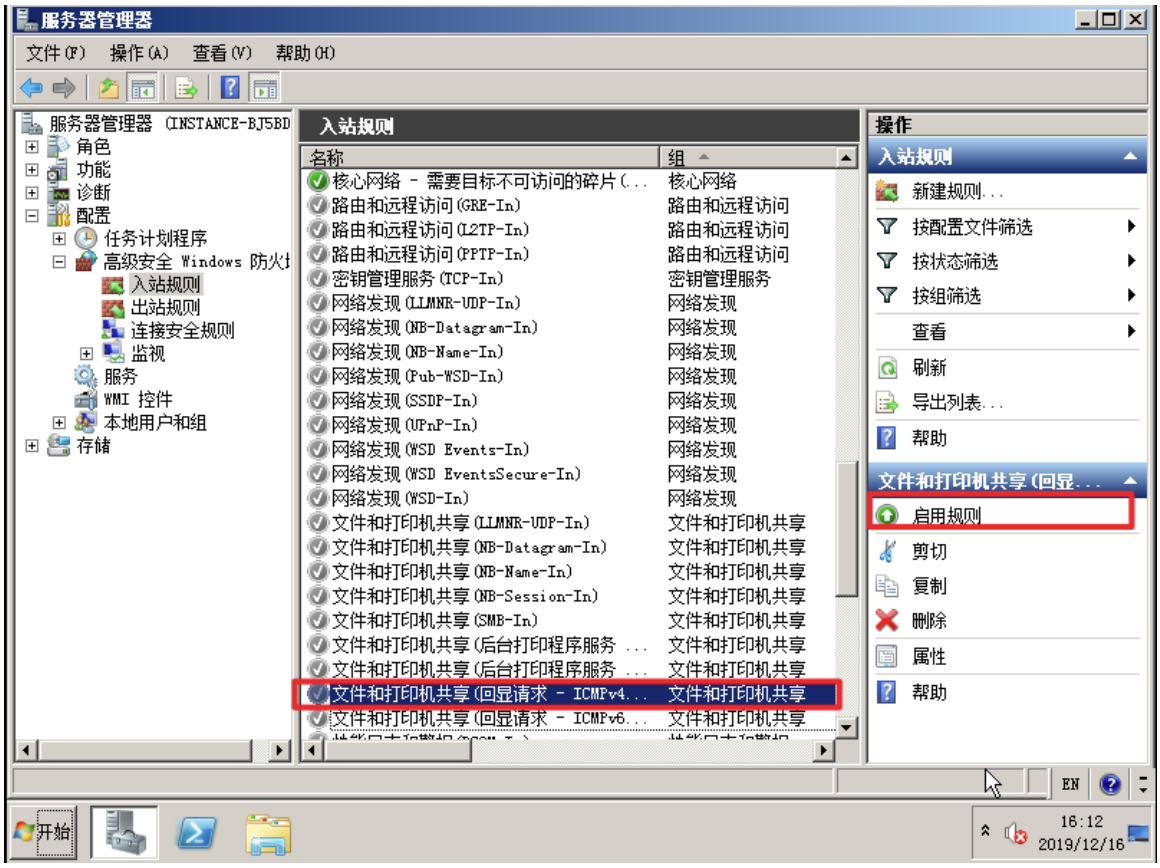
故障原因

由于Windows server默认禁止ping，请您打开高级安全防火墙入站的“文件和打印机共享(回显请求 - ICMPv4-In)”规则。

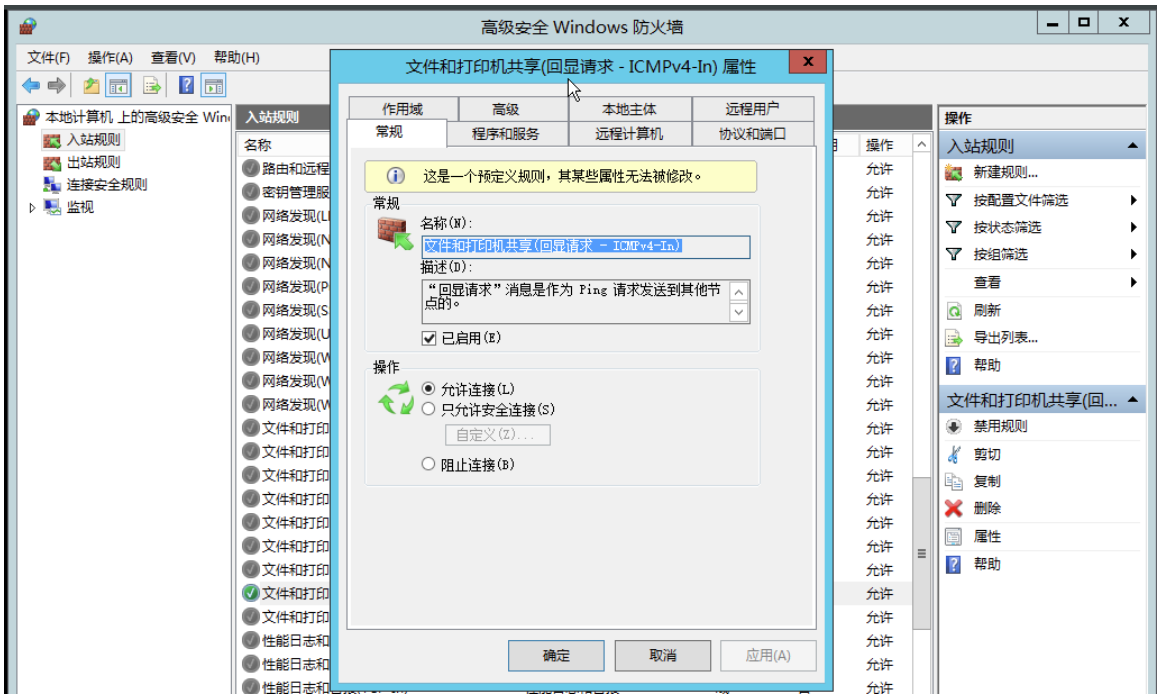
故障处理

您可参考如下位置开启windows实例的高级安全防火墙的入站规则。

- Windows Server 2008 防火墙位置： 服务器管理器--配置--高级安全Windows防火墙--入站规则



- Windows Server 2012 防火墙位置： 服务器管理器--工具--高级安全windows防火墙--入站规则



无法远程连接Windows实例

故障现象

用户无法远程连接Windows云服务器

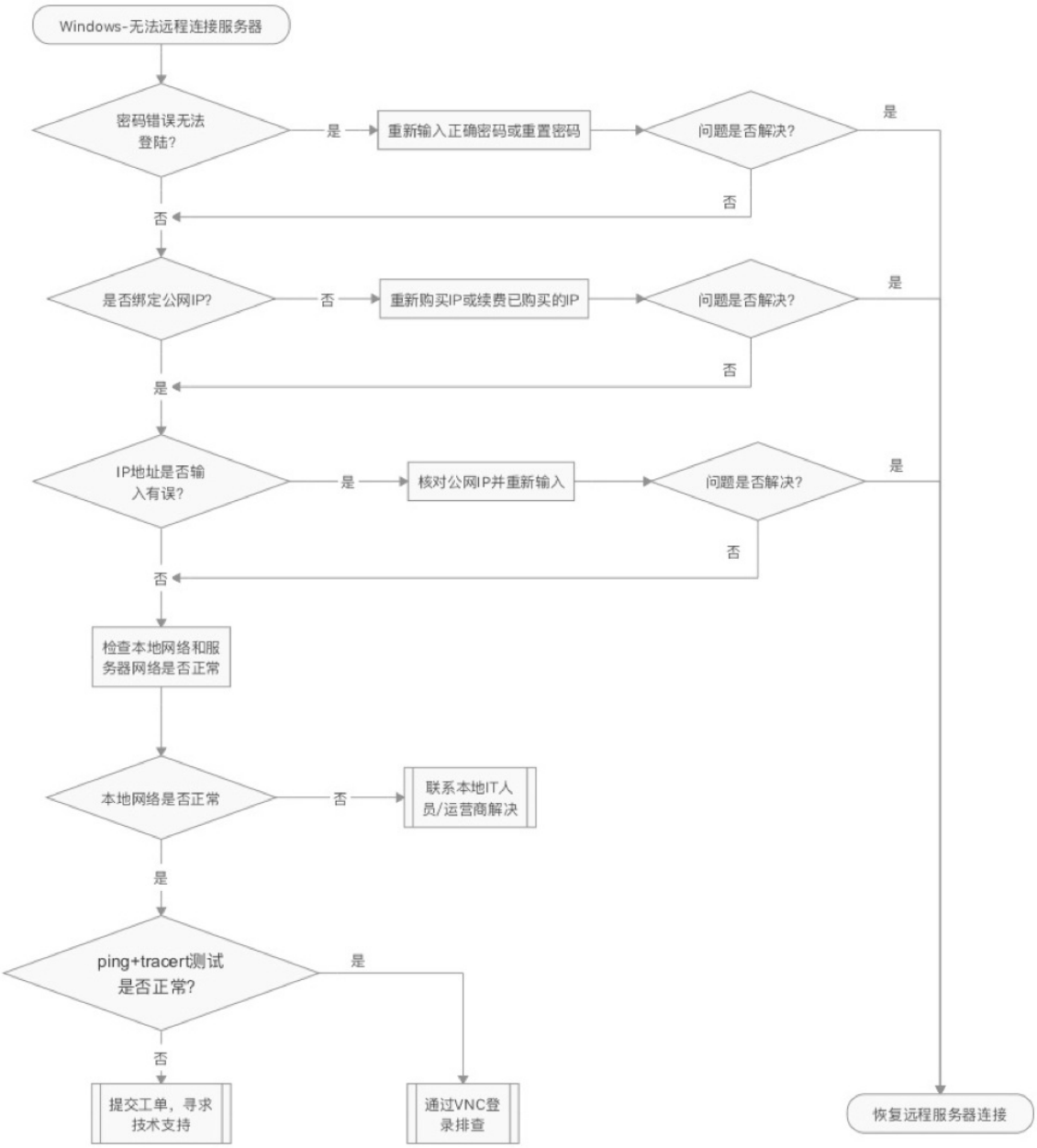
故障原因

- 登录密码输入有误
- 未绑定公网IP或者输入IP有误
- 网络异常
- 系统故障
- 服务器CPU、内存使用异常
- 未设置允许远程连接
- 端口未放行
- 安全组规则设置错误
- 主机安全封禁本地IP

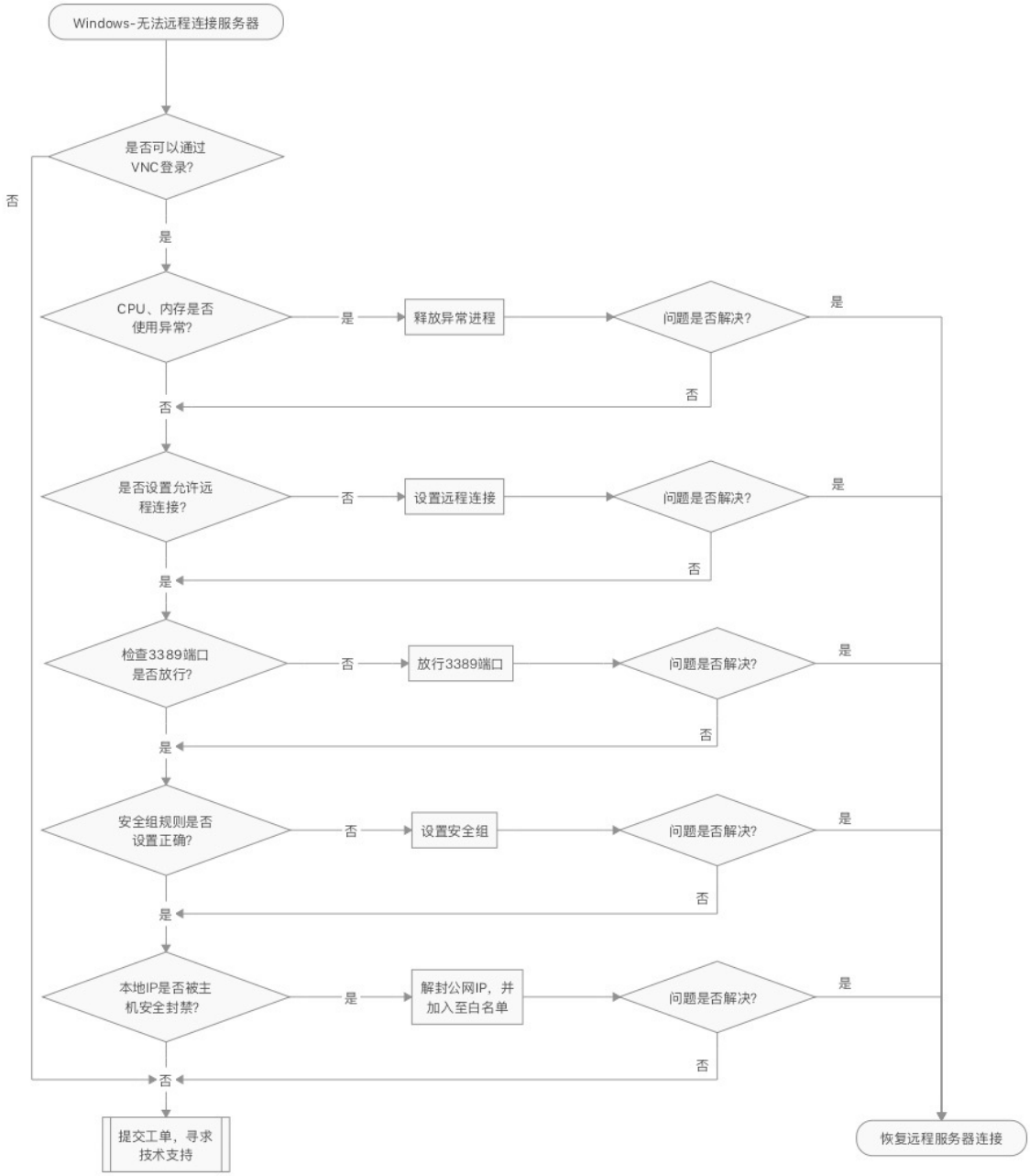
处理流程

当您无法通过远程连接软件登录Windows实例时，推荐您先进行本地检测确认本地环境正常。若未解决问题，可通过VNC登录的方式登录云服务器逐步排查故障。

本地检测流程



通过VNC登录定位故障流程



故障处理

登录密码输入有误

确认密码输入是否正确，如果忘记密码，可以按照参考文档重置密码：

1. 登录百度智能云控制台，选择产品-计算-云服务器BCC
2. 找到实例，点击右侧的“更多”，选择实例设置-重置密码，需要注意的是：执行重置密码操作会强制重启服务器，建议在业务低峰期执行

如果确认输入的是正确的密码，但是报错"密码错误"，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

未绑定公网IP或者输入IP有误

1. 可以在控制台查看EIP绑定情况，具体可以点击BCC实例名称，然后在基本信息中查看是否已绑定EIP，如果未绑定，或者绑定的EIP已到期，可以直接重新购买或者续费，如果已绑定EIP，可按照如下步骤进行排查。

2. 检查远程连接时输入的IP地址是否与控制台中显示的一致，如果输入不一致或者直接输入的内网IP，也是无法远程连接的，确认IP输入无误后如果依然无法连接，可以继续按照下面的步骤进行排查。

网络异常

需要检查本地网络和服务器网络是否正常，排查步骤如下：

1. 在本地电脑访问其他网站，确认自己本地的网络是正常的
2. 使用当地其他电脑进行远程连接，如果是本地网络问题或运营商问题，请联系本地IT人员或运营商解决，排除本地网络故障后进行下一步检查
3. 在本地电脑对公网IP进行ping+tracert测试，Windows默认是禁ping的，在测试前，需要先开启ping。打开高级安全防火墙入站的“文件和打印机共享(回显请求 - ICMPv4-In)”规则，参考Windows服务器ping不通。

> 注意：
>
> 2008 防火墙位置： 服务器管理器--配置--高级安全Windows防火墙--入站规则

> 2012 防火墙位置： 服务器管理器--工具--高级安全windows防火墙–入站规则

测试后如果ping+tracert是正常的，但是依然无法远程，可以继续按照如下步骤排查，如果ping+tracert异常，可以在工单系统提供测试的完整截图，由百度智能云的技术人员协助排查

系统故障

如果无法远程连接，可以使用VNC尝试登录服务器，以判断服务器系统是否是正常的，后续的排查也可通过VNC登录进行定位，具体步骤可参考使用VNC远程登录云服务器，具体内容如下：

1. 登录百度智能云控制台，选择产品服务-计算-云服务器BCC。
2. 找到无法远程连接的实例，点击右侧的远程登录并选择通过VNC登录。

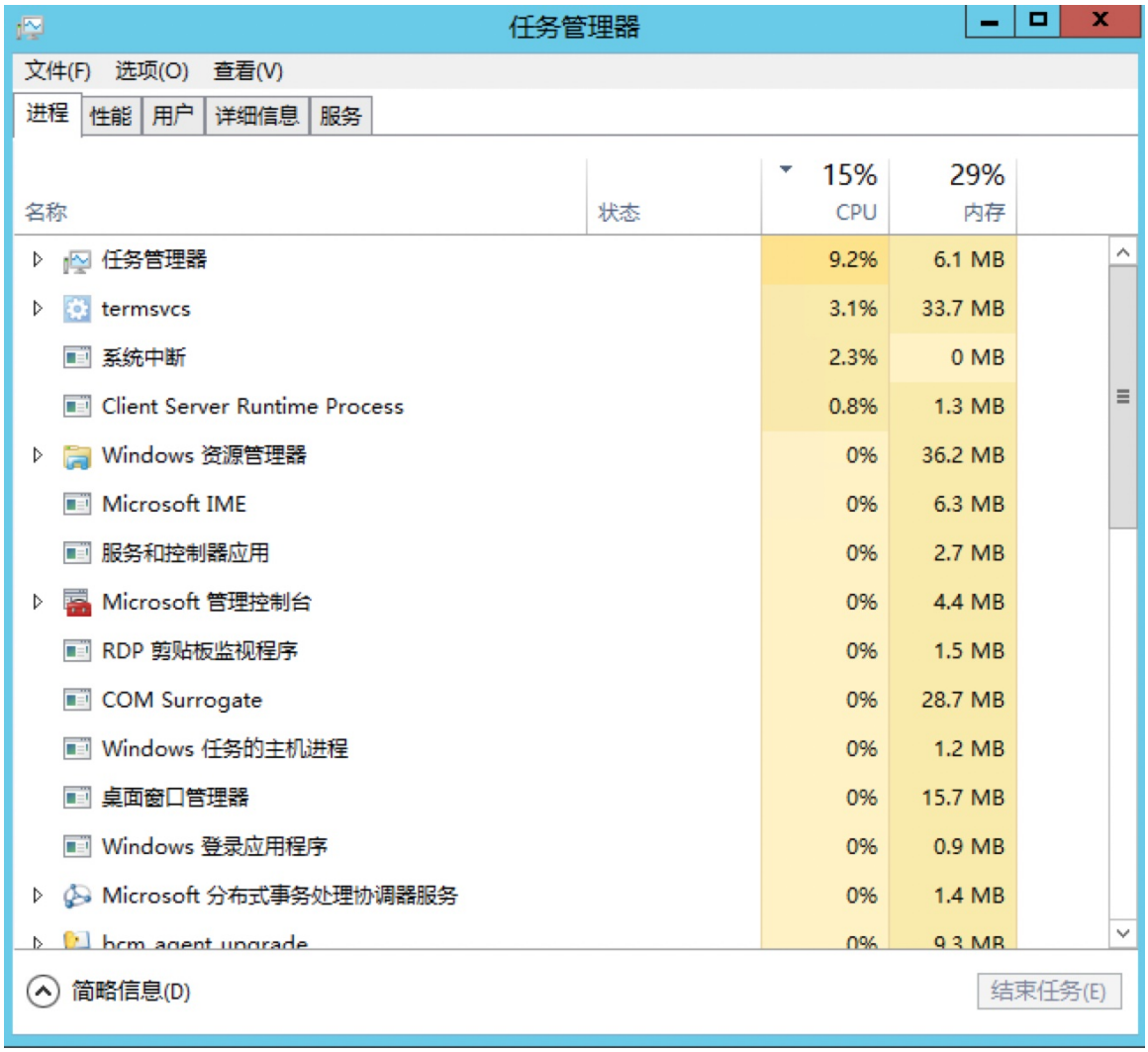
如果可以通过VNC登录服务器，说明系统正常，可以继续按步骤排查，如果通过VNC无法登录，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

服务器CPU、内存使用异常

如果通过VNC登录可以服务器， 需要检查服务器CPU、内存使用情况，具体步骤如下：

1. 鼠标右键点击任务栏，点击"任务管理器"。

2. 查看当前CPU、内存的使用情况，如果某进程占用大量资源，可以在做好快照的前提下，先结束该进程，然后再观察是否可以远程连接，如果无法远程连接，可以继续按照如下步骤排查。

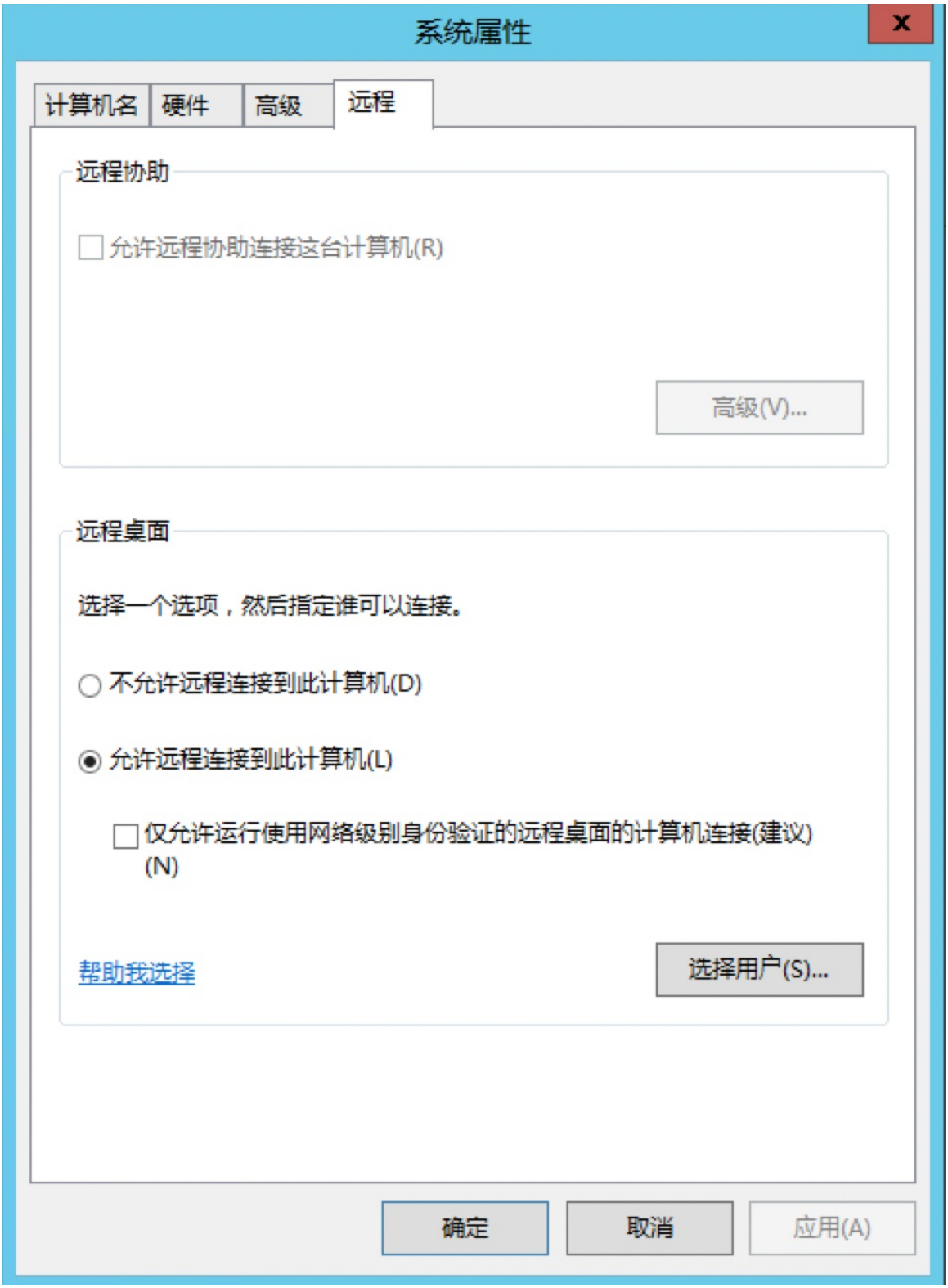


未设置允许远程连接

确认服务器是否允许远程连接服务器，以Windows2012为例，使用VNC登录设置，然后按照如下步骤排查：

1. 点击控制面板-系统。

2. 点击远程设置，选择允许远程连接到此服务器->选择用户，添加Administrator，或是可以是其他非管理员用户，最后确定保存。

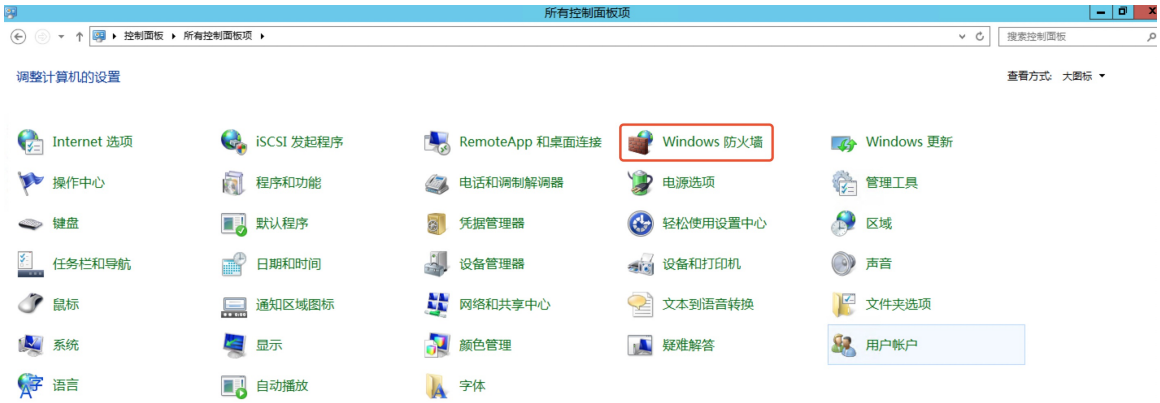


如果其他版本服务器遇到类似问题，可以自行百度搜索相关版本如何开启远程连接，确认开启后继续按照如下步骤操作。

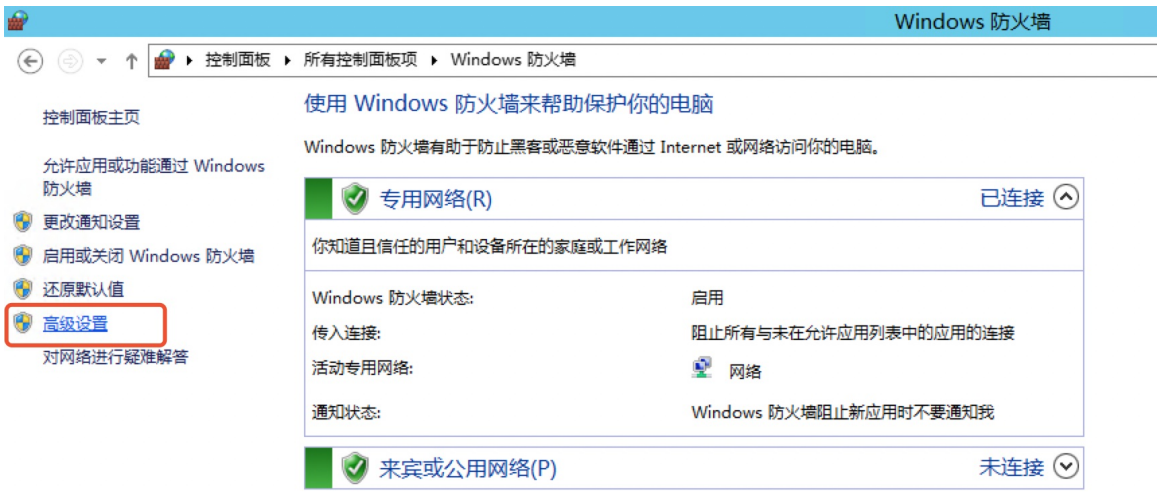
端口未放行

如果未更改过远程连接默认端口，需要检查服务器的3389端口是否已开通，如果未开通，需要添加防火墙出入站规则，放行3389端口，以Windows2012为例，可按照如下步骤排查：

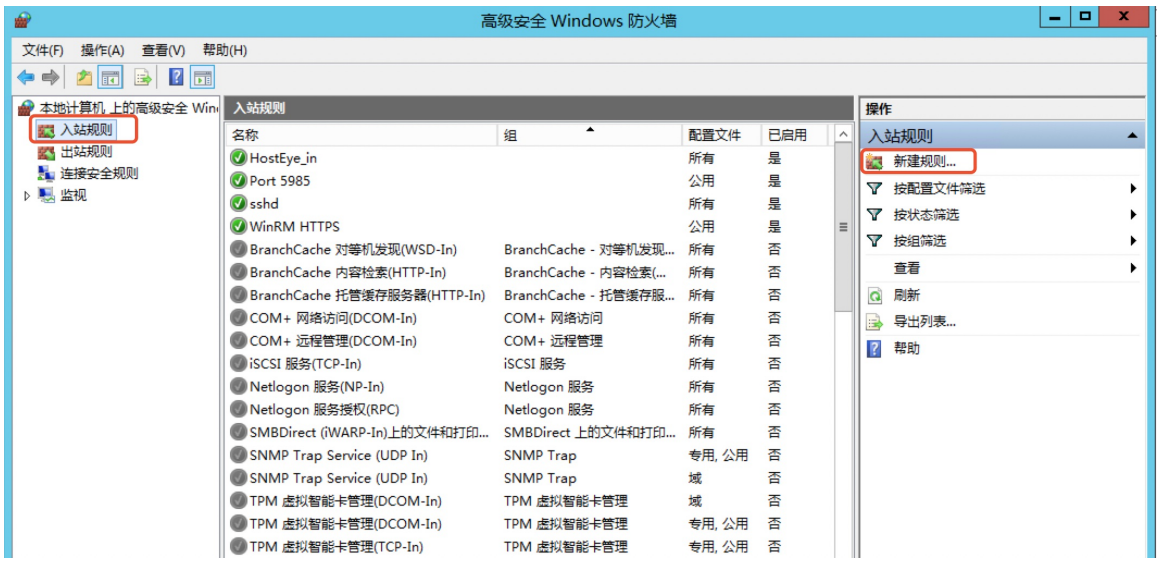
1. 打开控制面板，点击"Windows防火墙"。



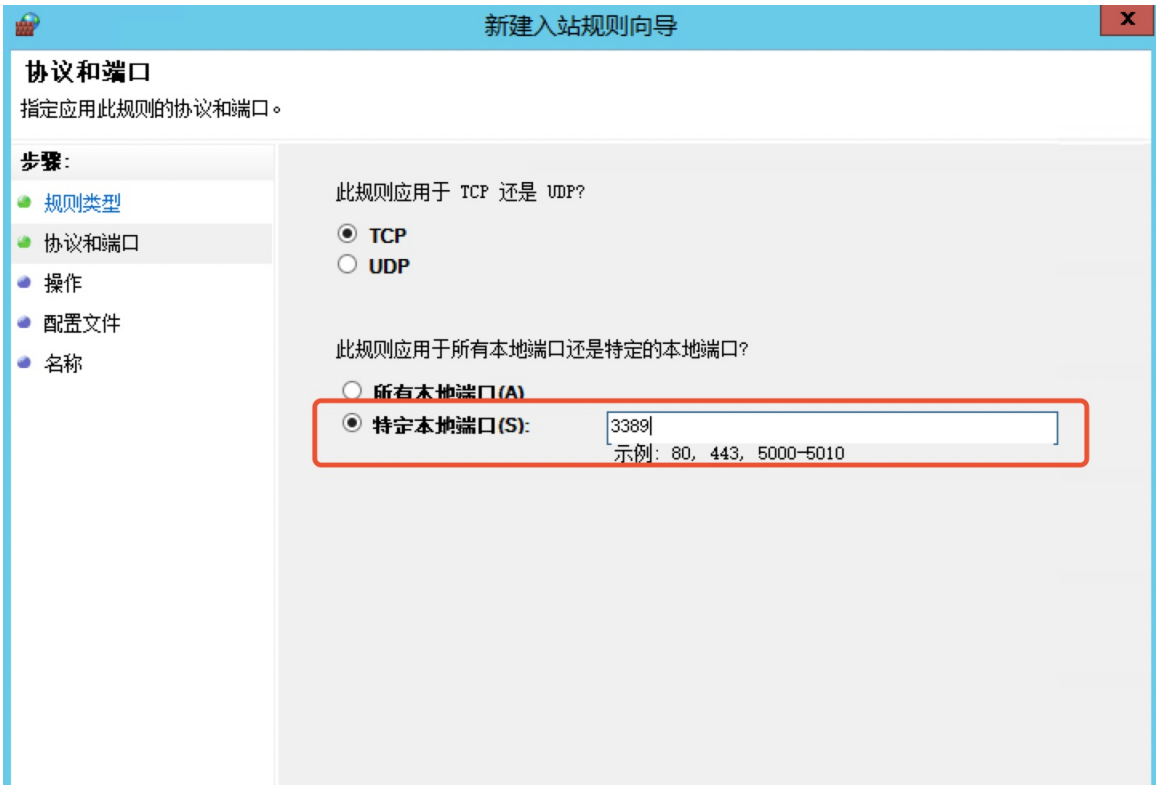
2. 点击高级设置。



3. 点击入站规则->新建规则。



4. 选择端口，添加3389



5. 出站新建规则类似上述操作

如果其他版本服务器遇到类似问题，可以自行百度搜索下相关版本如何放行3389端口，如果已放行3389，可继续按照如下步骤进行排查，如果在开启3389端口过程中遇到问题，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

安全组规则设置错误

需要检查安全组是否放行了3389端口，是否设置了固定的源IP，可先创建默认安全组，然后再进行测试：

如果安全组设置正常，但是还是无法远程连接，可以继续按照如下步骤进行排查

主机安全封禁本地IP

更换其他电脑看是否可以连接到服务器，如果更换电脑后可以连接，但是在本机无法连接，需要排查一下本地的IP是否被主机安全封禁了，排查步骤可参照登录管理，具体如下：

1. 在百度智能云控制台找到产品服务-安全和管理-主机安全。
2. 点击无法登录的BCC，点击右侧的 ”查看详情”。

3. 查看登录事件，看下本地IP是否被封禁。

4. 若您的公网IP被拦截，进行解封即可，然后把您的公网IP加入到白名单。

5. 添加白名单后，再次远程连接试一下，如果依然无法远程连接，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

无法远程连接Linux实例

故障现象

用户无法远程连接BCC云服务器

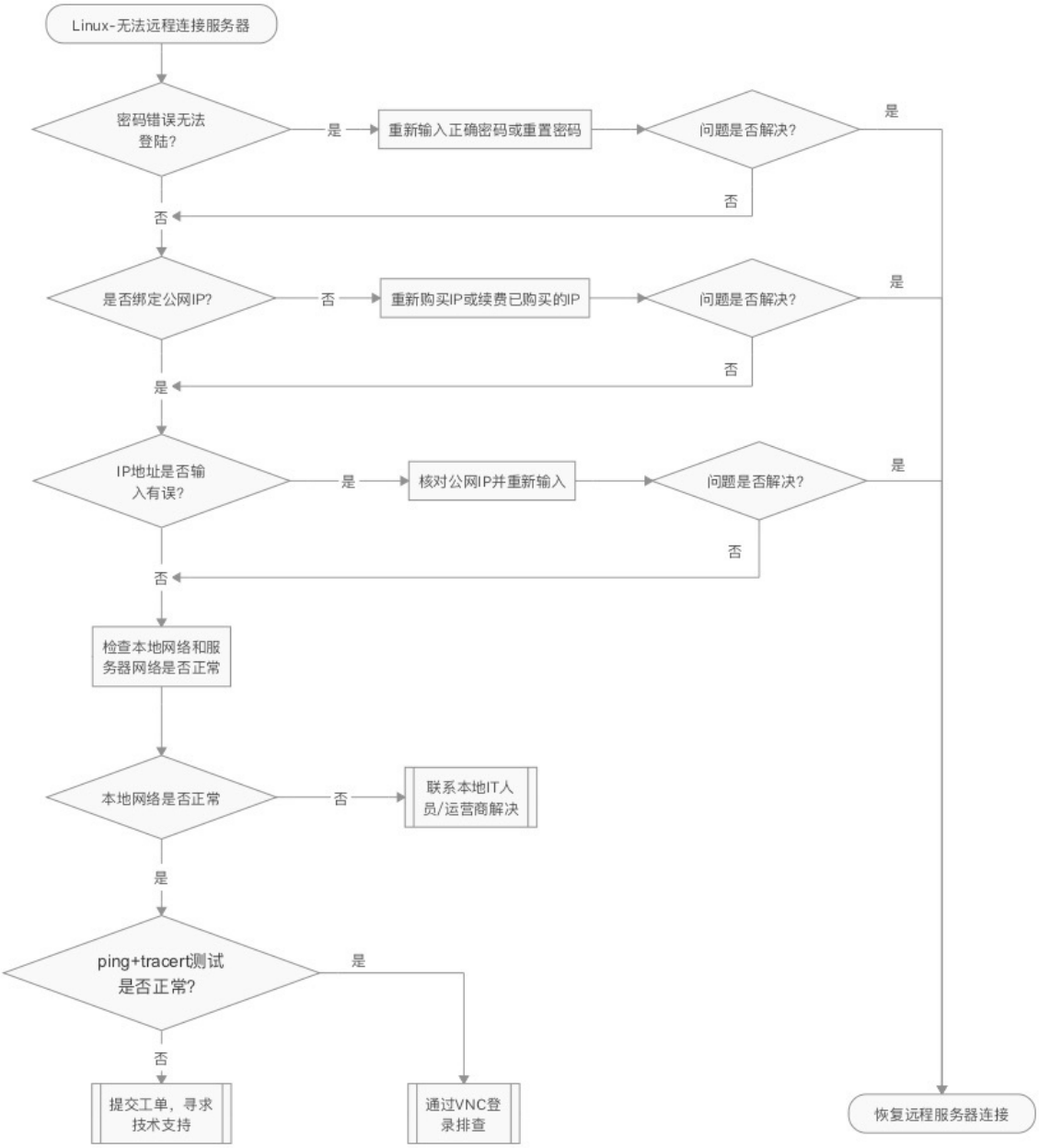
故障原因

- 登录密码输入有误
- 未绑定公网IP或者输入IP有误
- 网络异常
- 系统故障
- 服务器CPU、内存使用异常
- 未设置允许远程连接
- 设置了禁止root用户登录
- 安全组规则设置错误
- 主机安全封禁本地IP

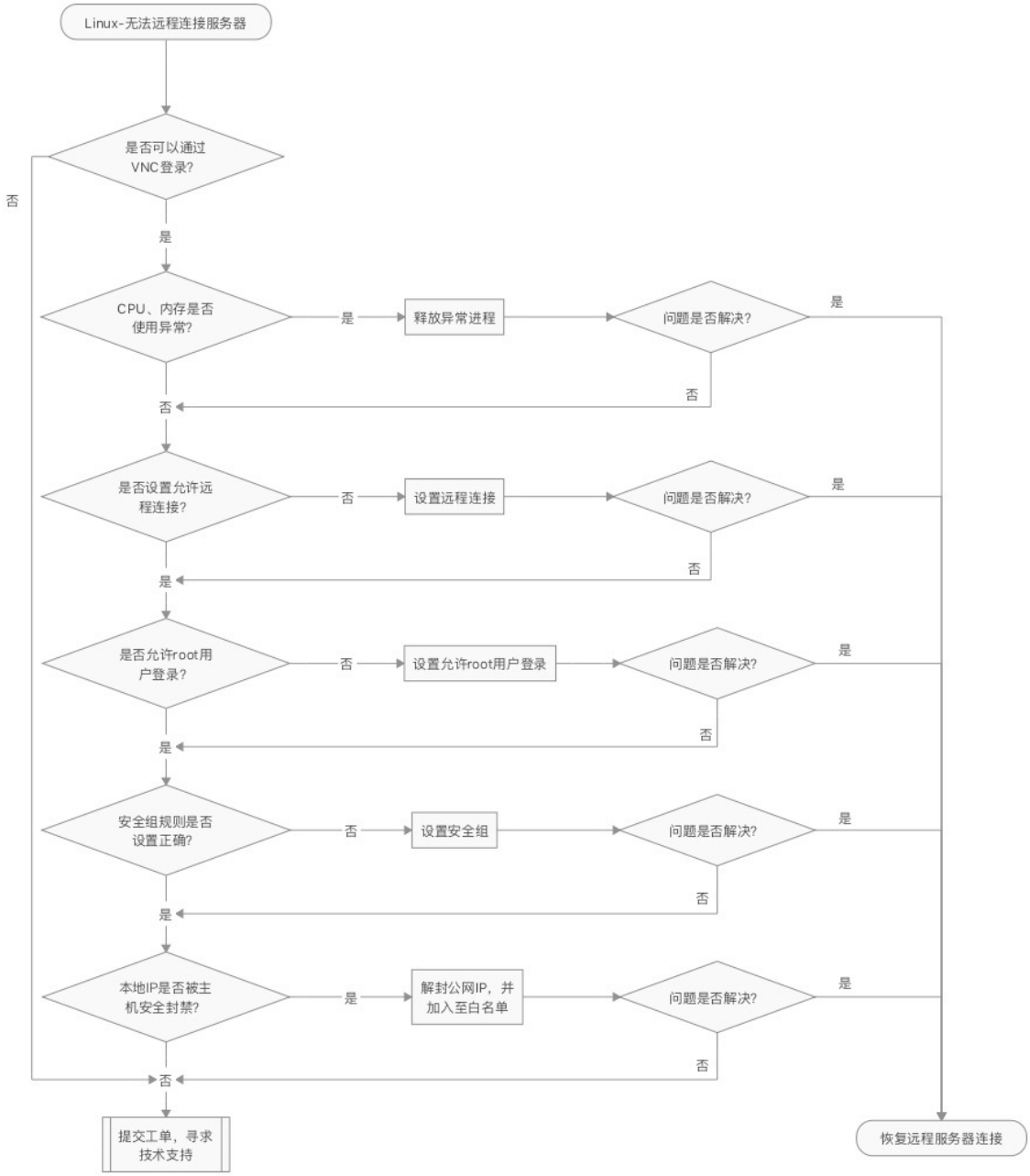
处理流程

当您无法通过远程连接软件登录Linux实例时，推荐您先进行本地检测确认本地环境正常。若未解决问题，可通过VNC登录的方式登录云服务器逐步排查故障。

本地检测流程



通过VNC登录定位故障流程



故障处理

登录密码输入有误

确认密码输入是否正确，如果忘记密码，可以按照参考文档重置密码：

1. 登录百度智能云控制台，选择产品-计算-云服务器BCC。
2. 找到实例，点击右侧的“更多”，选择实例设置-重置密码，需要注意的是：执行重置密码操作会强制重启服务器，建议在业务低峰期执行。

如果确认输入的是正确的密码，但是报错"密码错误"，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

未绑定公网IP或者输入IP有误

1. 可以在控制台查看EIP绑定情况，具体可以点击BCC实例名称，然后在基本信息中查看是否已绑定EIP，如果未绑定，或者绑定的EIP已到期，可以直接重新购买或者续费，如果已绑定EIP，可按照如下步骤进行排查。

2. 检查远程连接时输入的IP地址是否与控制台中显示的一致，如果输入不一致或者直接输入的内网IP，也是无法远程连接的，确认IP输入无误后如果依然无法连接，可以继续按照下面的步骤进行排查。

网络异常

需要检查本地网络和服务器网络是否正常，排查步骤如下：

1. 在本地电脑访问其他网站，确认自己本地的网络是正常的
2. 使用当地其他电脑进行远程连接，如果是本地网络问题或运营商问题，请联系本地IT人员或运营商解决，排除本地网络故障后进行下一步检查
3. 在本地电脑对公网IP进行ping+tracert测试，如果服务器无法ping，以centos6 为例，可按照如下步骤开启ping，其他系统可自行搜索相关版本的操作步骤：

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

测试后如果ping+tracert是正常的，但是依然无法远程，可以在工单系统提供测试的完整截图，由百度智能云的技术人员协助排查。

系统故障

如果无法远程连接，可以使用VNC尝试登录服务器，以判断服务器系统是否是正常的，后续的排查也可通过VNC登录进行定位，具体步骤可参考使用vnc远程登录云服务器，具体内容如下：

1. 登录百度智能云控制台，选择产品服务-计算-云服务器BCC
2. 找到无法远程连接的实例，点击右侧的VNC远程

如果可以通过VNC登录服务器，说明系统正常，可以继续按步骤排查，如果通过VNC无法登录，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

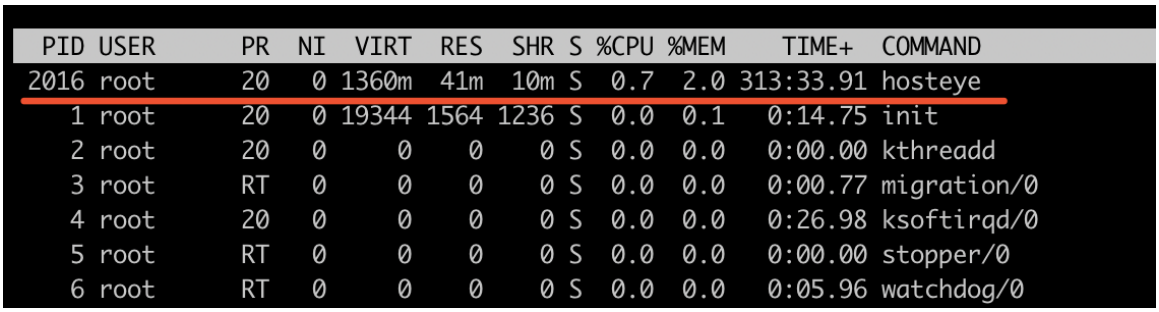
服务器CPU、内存使用异常

如果通过VNC登录可以服务器， 需要检查服务器CPU、内存使用情况，具体步骤如下：

1. 执行"top"查看当前CPU、内存资源使用情况

```
[redacted]]# top
top - 22:30:07 up 29 days, 10:27,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 151 total,   1 running, 142 sleeping,   0 stopped,   8 zombie
Cpu(s):  0.2%us,  0.3%sy,  0.0%ni, 99.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2054032k total,  1751088k used,   302944k free,   163824k buffers
Swap:  1049596k total,        0k used,  1049596k free,  1238520k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2016 root      20   0 1360m  41m  10m S  0.7  2.0 313:33.47 hosteye
    1 root      20   0 19344 1564 1236 S  0.0  0.1   0:14.75 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.77 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:26.98 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:05.96 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:03.30 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/1
    9 root      20   0     0    0    0 S  0.0  0.0   0:25.09 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:05.01 watchdog/1
   11 root      20   0     0    0    0 S  0.0  0.0  11:03.38 events/0
   12 root      20   0     0    0    0 S  0.0  0.0   3:15.62 events/1
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events/0
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events/1
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_long/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_long/1
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_power_ef
   18 root      20   0     0    0    0 S  0.0  0.0   0:00.00 events_power_ef
```

2. 如果某进程占用大量资源，可以在做好快照的前提下，先结束该进程，然后再观察是否可以远程连接，具体步骤如下：
    1. 执行top后显示的第一列为进程ID，以第一行的hosteye进程为例，进程ID为2016。



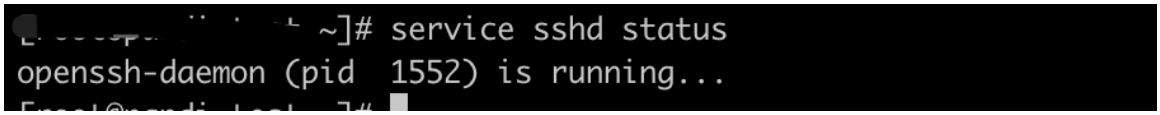
2. 执行`ps -ef | grep 2016`，查看进程的具体信息，如果确认占用大量资源的进程是异常的，不是自己启动的，可以执行`kill -9 ID`杀掉进程。

终止进程后，观察是否可以远程连接，如果还是无法远程连接，可以继续按照如下步骤排查。

未设置允许远程连接

如果无法远程连接Linux服务器，需要先排查服务器是否已启动了sshd服务，sshd是否监听了22端口，使用VNC登录设备，然后按照如下步骤排查：

1. 执行`service sshd status`，查看ssh是否是已启动的状态



2. 如果显示未启动，可以先启动sshd服务或者重启：`service sshd start/restart`
3. ssh设置完毕后，设置防火墙放行22端口，以centos6为例，可执行如下命令：

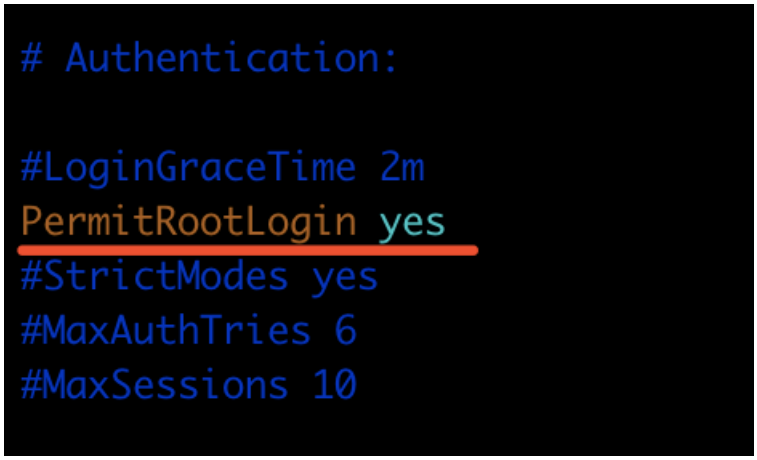
```
 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
/etc/rc.d/init.d/iptables save
/etc/rc.d/init.d/iptables restart
/etc/init.d/iptables status
```

如果其他版本服务器遇到类似问题，可以自行百度搜索下相关版本如何放行22端口，如果设置完毕后依旧无法远程连接，可继续按照如下步骤进行排查。

设置了禁止root用户登录

需要排查sshd配置文件是否允许root用户登录，以centos6为例，使用VNC登录设备，然后按照如下步骤排查：

进入sshd的配置文件：`vim /etc/ssh/sshd_config`，将PermitRootLogin改为yes



如果设置完毕后依旧无法远程连接，可继续按照如下步骤进行排查

安全组规则设置错误

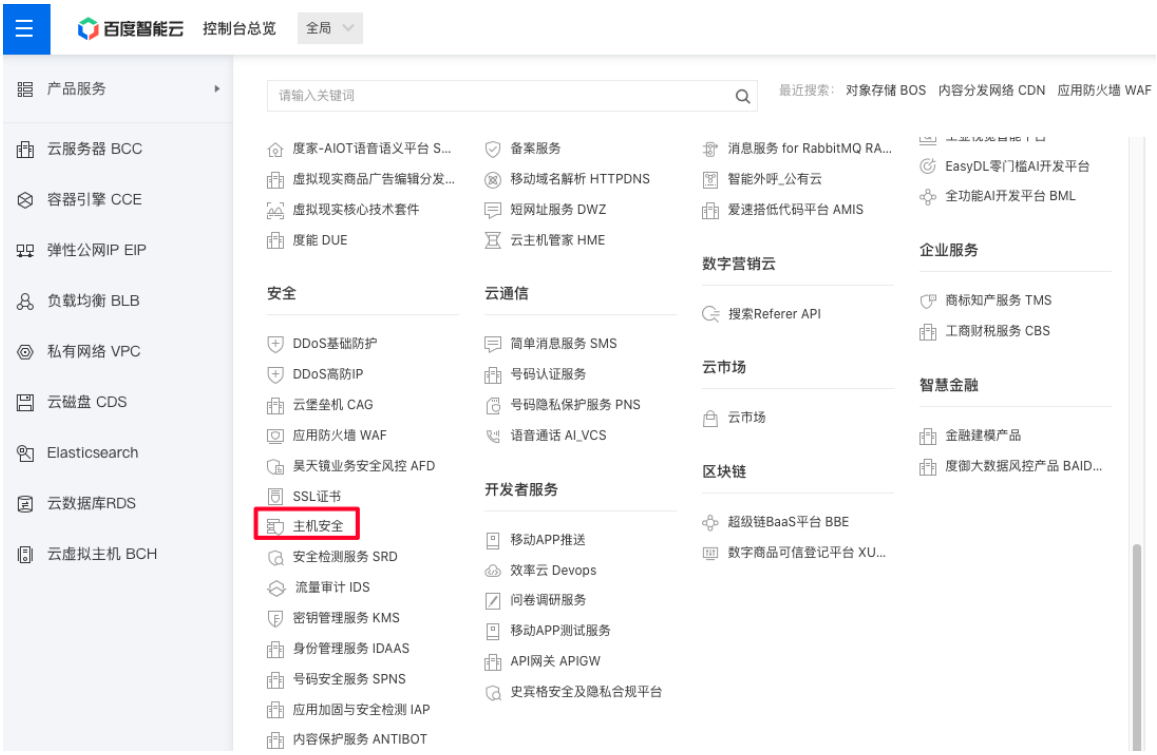
需要检查安全是否放行了22端口，是否设置了固定的源IP，需要先创建默认安全组，然后再进行测试：

如果安全组设置正常，但是还是无法远程连接，可以继续按照如下步骤进行排查。

主机安全封禁本地IP

更换其他电脑看是否可以连接到服务器，如果更换电脑后可以连接，但是在本机无法连接，需要排查以下本地的IP是否被主机安全封禁了，排查步骤可参照登录管理：

1. 在百度智能云控制台找到产品服务-安全和管理-主机安全。



2. 点击无法登录的BCC，点击右侧的”查看详情”。

3. 查看登录事件，看下本地IP是否被封禁

4. 若您的公网IP被拦截，进行解封即可，然后把您的公网IP加入到白名单

5. 添加白名单后，再次远程连接试以下，如果依然无法远程连接，可以在工单系统反馈问题现象，由百度智能云的技术人员协助排查。

Linux云服务器重置密码后无法登录实例

背景

当您在BCC控制台中操作重置密码功能未生效时，可以登录VNC，进入单用户模式重置密码。

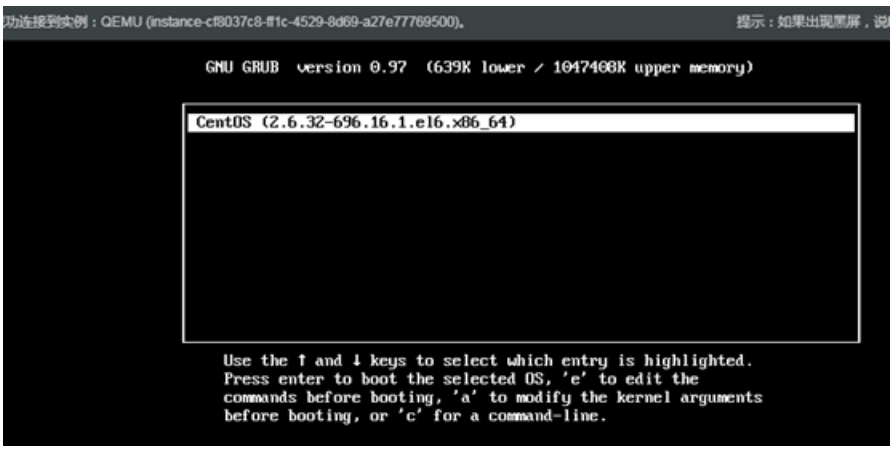
风险须知

进入单用户重置密码需要重启服务器，请确认风险做好数据备份后操作。

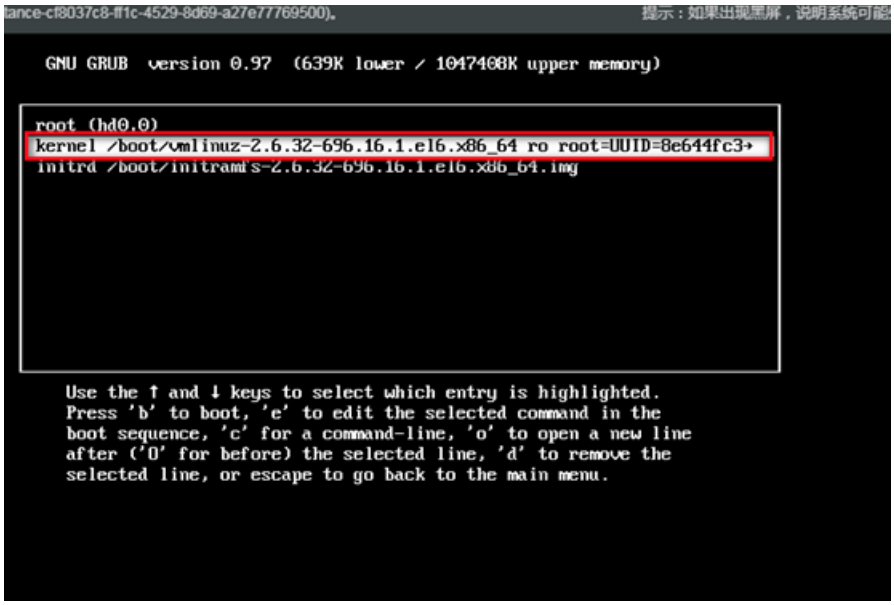
Centos 6进入单用户重置密码

1. VNC连接服务器后，在VNC右上方点击 **发送Ctrl+Alt+Del** 进行重启服务器，进入重启状态，此时及时按Esc键，进入下一界面：

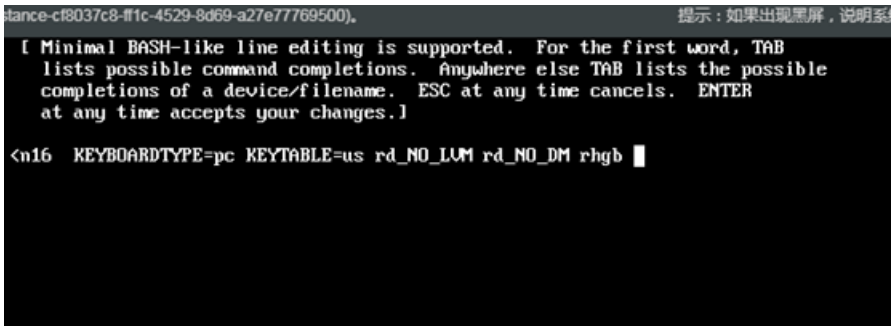
2. 按Esc键后，进入Grub界面，此时按e键：



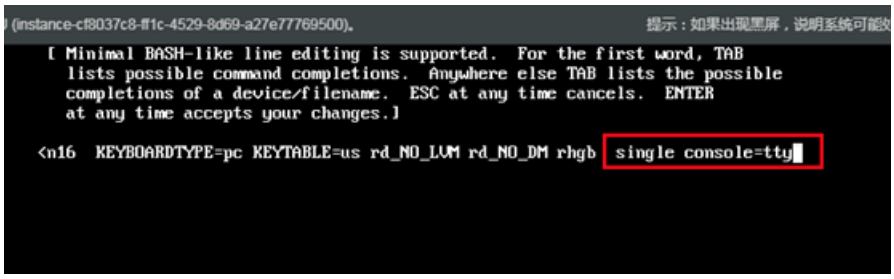
3. 之后进入如下界面，上下键选中如下红色标记的Kernel，按e键：



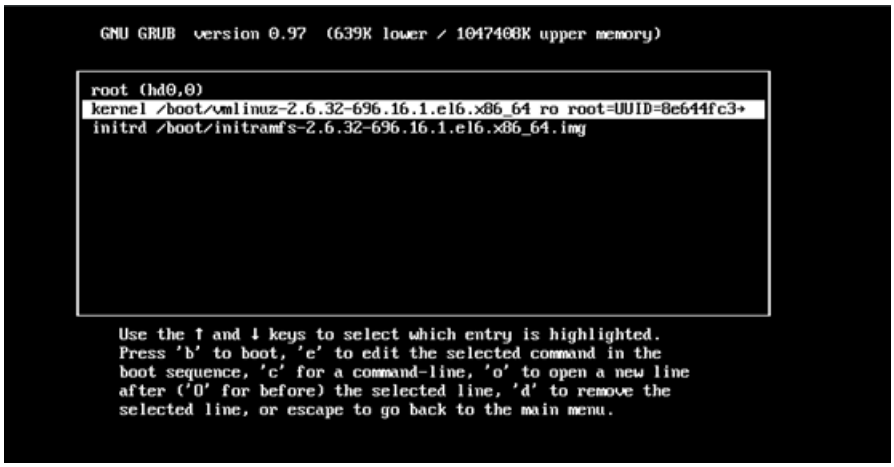
4. 之后进入如下界面：



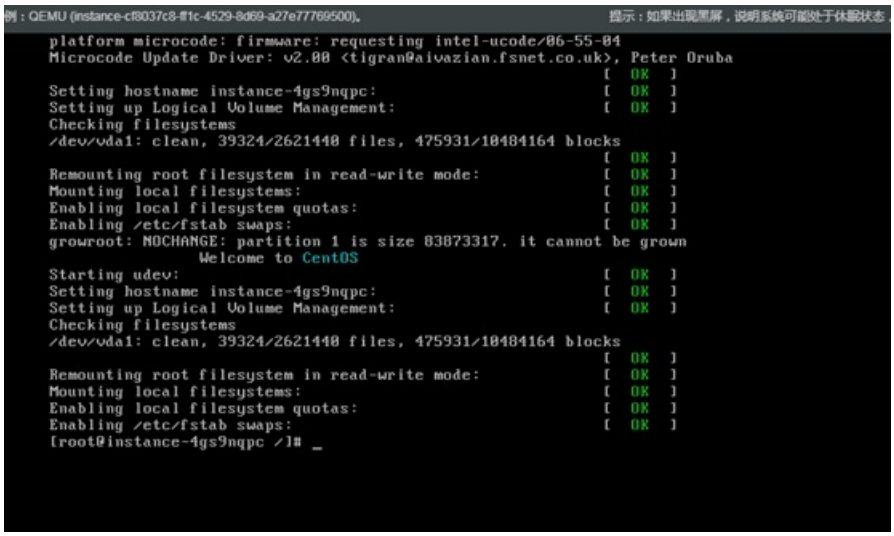
5. 在末尾增加：`single console=tty`， 修改后如下所示：



6. 按 **Enter** 键 ，会返回之前界面如下，再按b键进行启动：



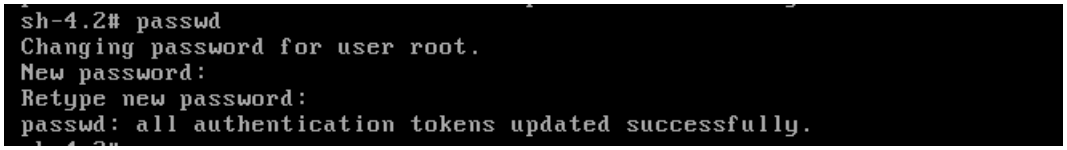
7. 之后会启动系统，进入单用户模式，不需要密码进入，再此界面可修改密码等操作。



8. 重置密码。

重置root密码可以直接使用命令passwd，如果对其他用户重置密码，需要按照passwd user的格式执行。

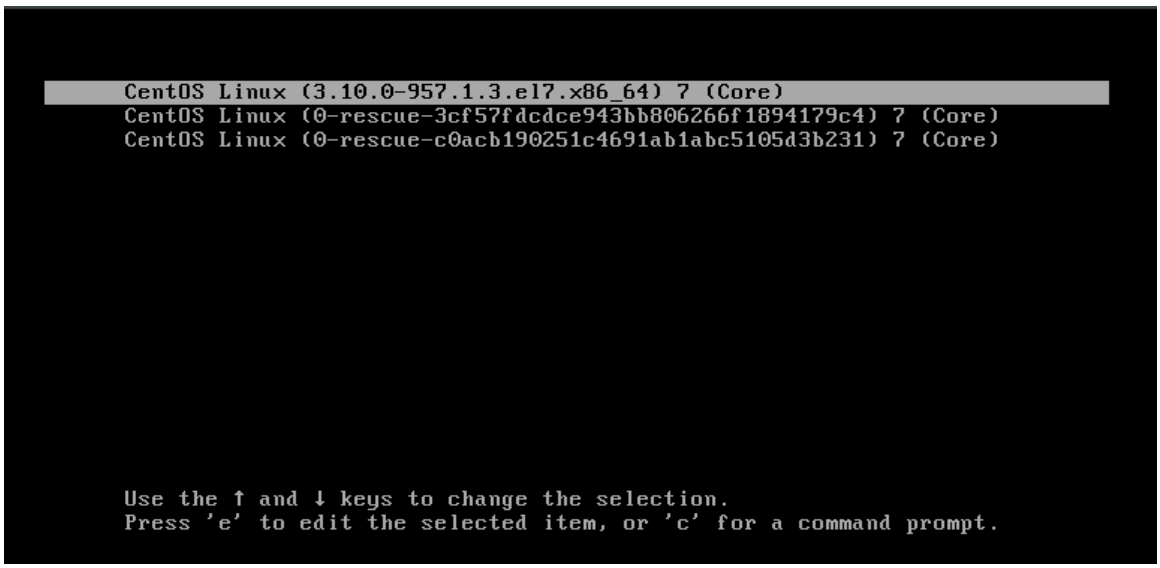
```
$ passwd
```



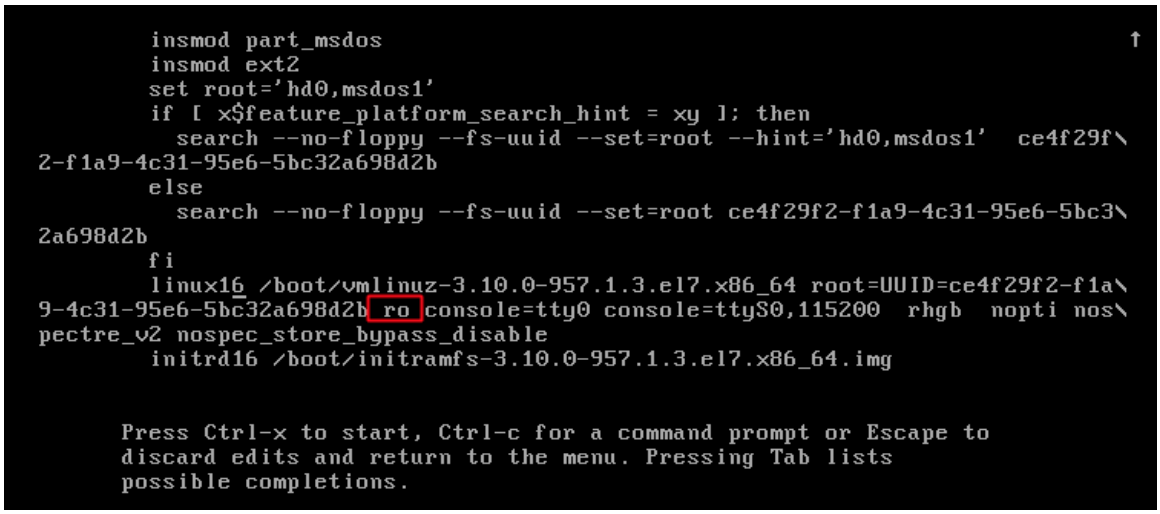
显示 **all authentication tokens updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

Centos 7进入单用户重置密码

1. 点击界面右上角的按钮 **发送Ctrl+Alt+Del** 进行重启服务器，进入重启状态，显示如下页面快速按e：

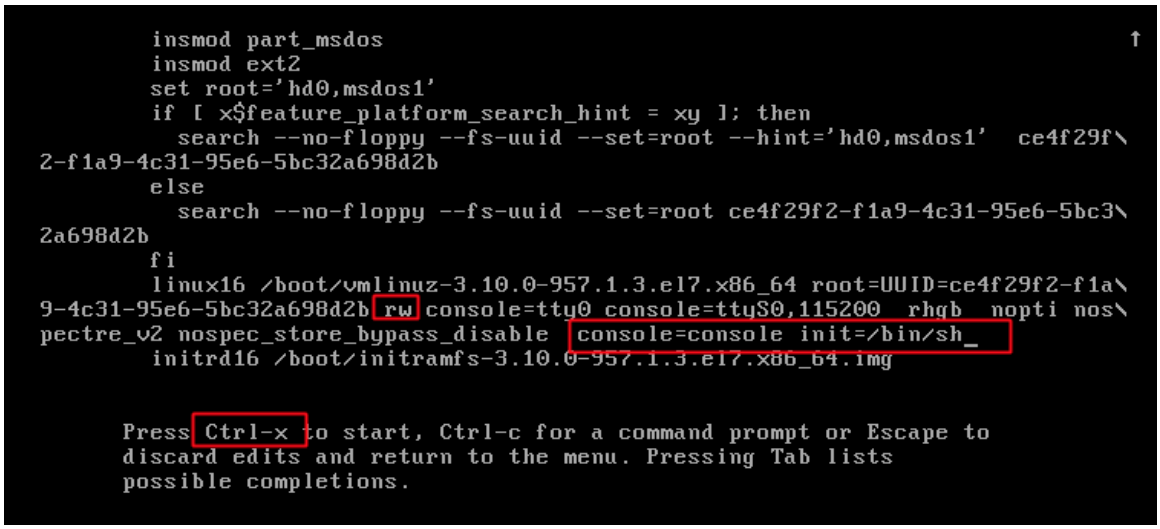


显示此页面：



2. 找到第一个linux16部分，修改ro为rw，在末尾后面添加 `console=console init=/bin/sh`。

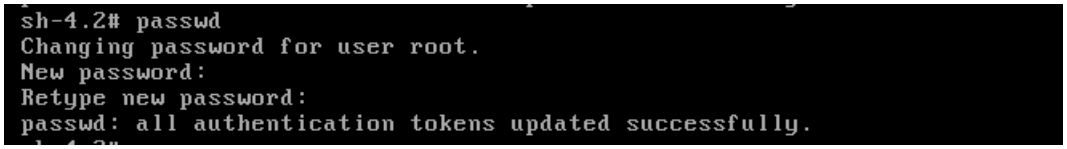
按 **Ctrl+X** 重启进入单用户模式。



3. 重置密码。

重置root密码可以直接使用命令passwd，如果对其他用户重置密码，需要按照passwd user的格式执行。

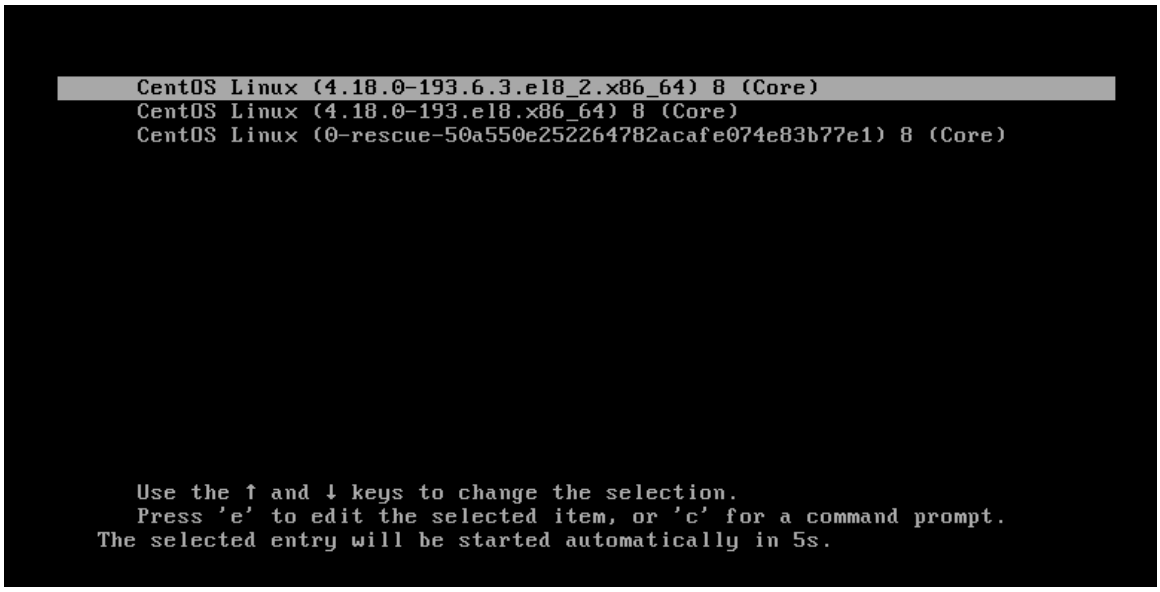
```
$ passwd
```



显示 **all authentication tokens updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

Centos 8进入单用户重置密码

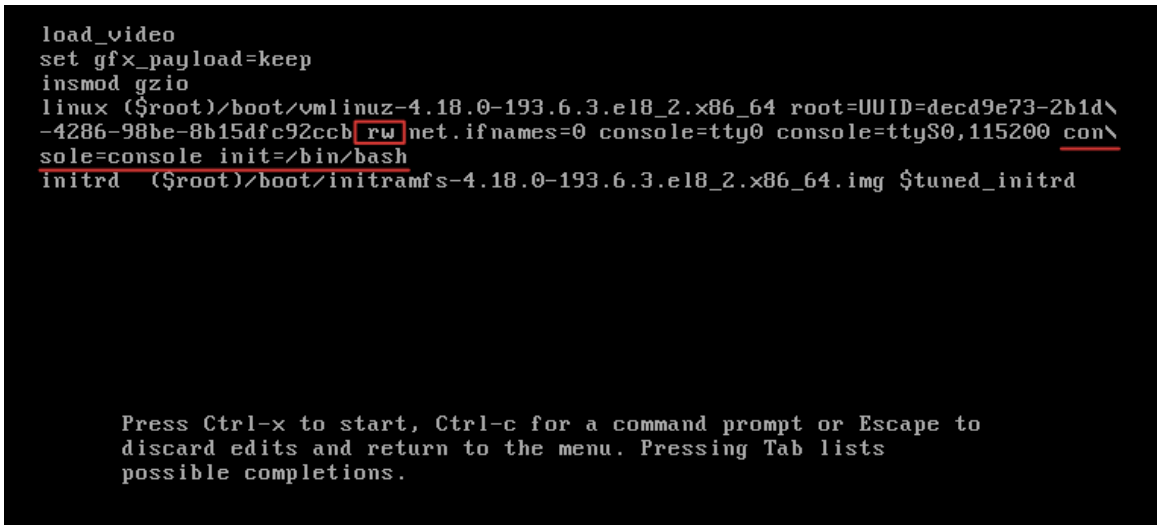
1. 点击界面右上角的按钮 **发送Ctrl+Alt+Del** 进行重启服务器，进入重启状态，显示如下页面快速按e：



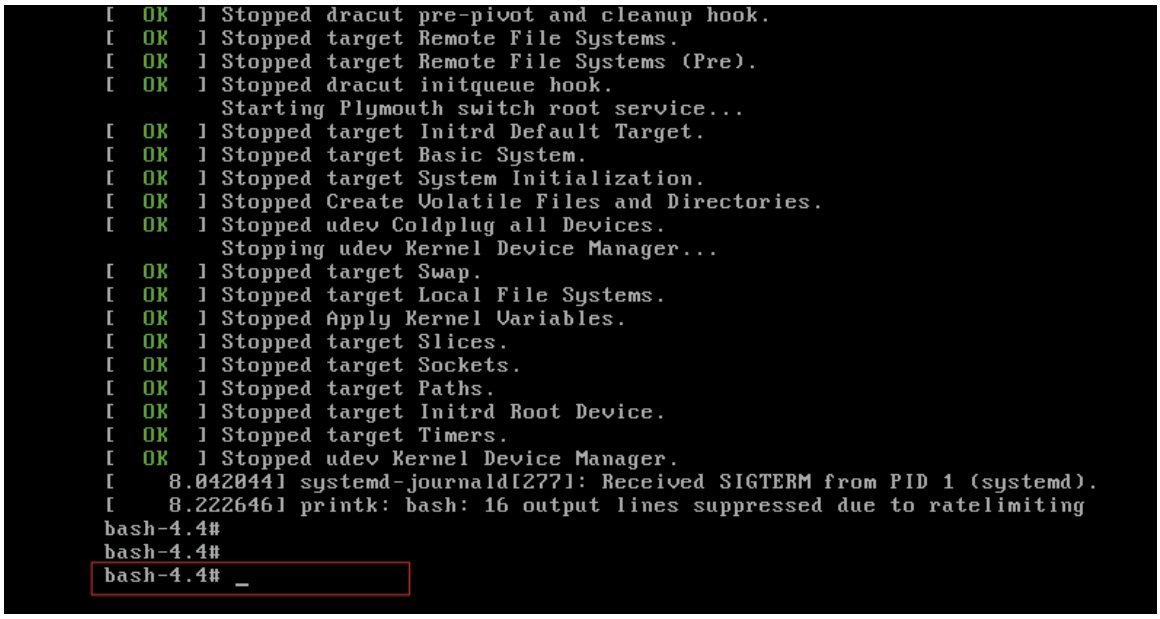
显示此页面：

2. 找到第一个linux部分，修改ro为rw，在末尾后面添加 `console=console init=/bin/sh`。

按 Ctrl+X 重启进入单用户模式。



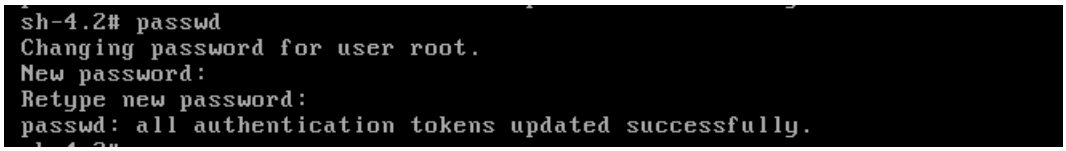
3. 显示如下界面，即正常进入了单用户模式。



4. 重置密码。

重置root密码可以直接使用命令passwd，如果对其他用户重置密码，需要按照passwd user的格式执行。

```
$ passwd
```

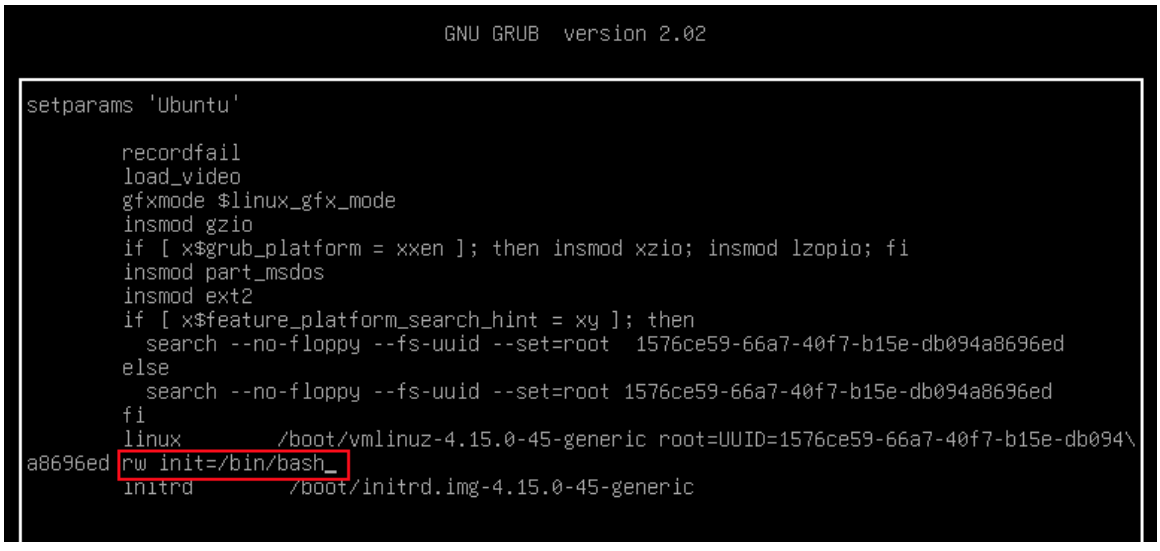


显示 **all authentication tokens updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

Ubuntu 18.04进入单用户重置密码

1. 点击界面右上角的按钮“发送Ctrl+Alt+Del”进行重启服务器，进入重启状态，显示如下页面快速按e：

2. 找到linux行末尾，删除不需要加载的参数至ro参数，修改ro为rw（以读写方式挂载根分区），并添加 `init=/bin/bash`。



3. 按 **Ctrl+X** 重启，进入单用户模式

3. 按 Ctrl+X 重启，进入单用户模式。

4. 重置密码。

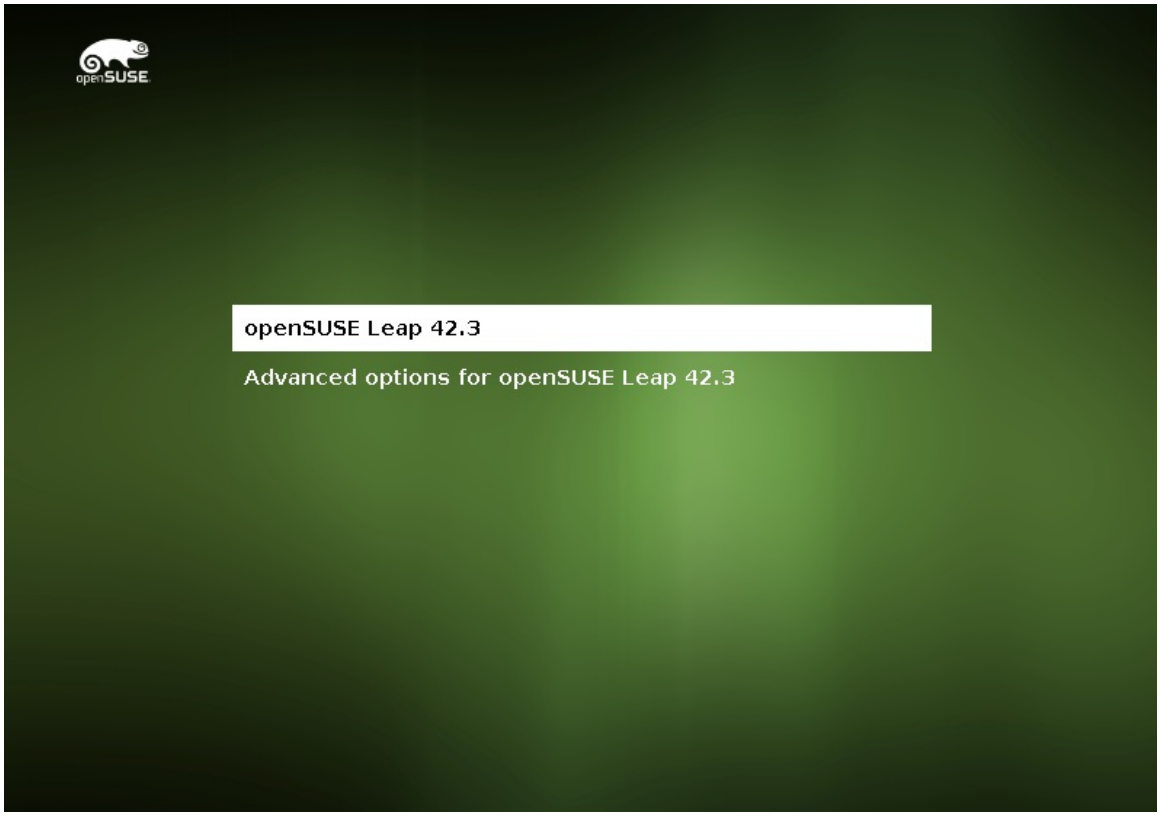
重置root密码可以直接使用命令passwd，如果对其他用户重置密码，需要按照passwd user的格式执行。

```
$ passwd
```

显示 **all authentication tokens updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

opensuse 42.3进入单用户重置密码

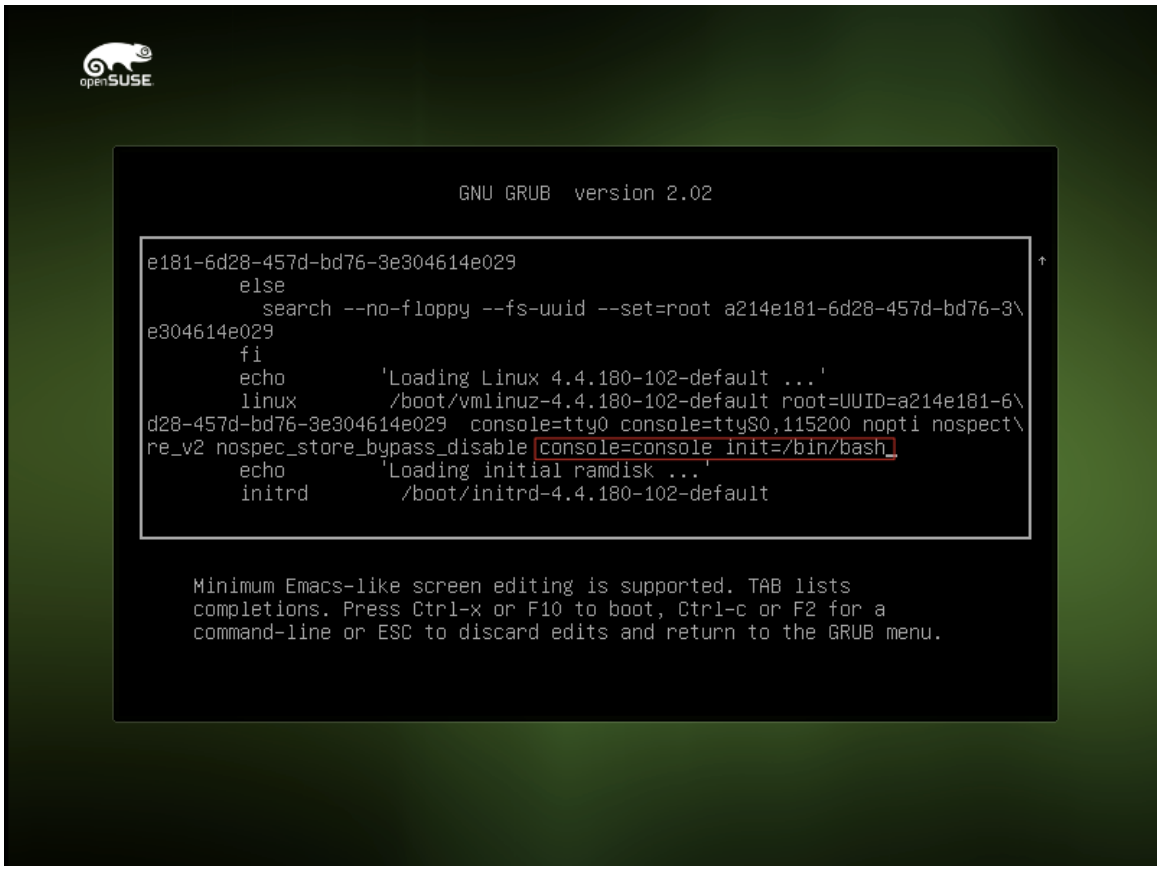
1. 点击界面右上角的按钮发送 **Ctrl+Alt+Del** 进行重启服务器，进入重启状态，显示如下页面快速按e:



显示此页面：

2. 找到第一个linux部分，在末尾后面添加 `console=console init=/bin/sh`。

按 **Ctrl+X** 重启，进入单用户模式。



3. 显示如下界面，即正常进入了单用户模式。

4. 重置密码。

重置root密码可以直接使用命令passwd，如果对其他用户重置密码，需要按照passwd user的格式执行。

```
$ passwd
```

显示 **all authentication tokens updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

Linux云服务器进入单用户重置密码报错：Authentication token lock busy

问题描述

单用户重置密码时遇到 **Authentication token lock busy**报错。

原因分析

由于磁盘仅设置只读权限导致，需要配置可写权限。

处理方式

1. 执行以下命令重新挂载磁盘。

```
$ mount -o remount,rw /
```

2. 再次重置密码。

```
$ passwd
```

显示 **password updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

Linux云服务器进入单用户重置密码报错：Authentication token manipulation error

问题描述

单用户重置密码时遇到 **passwd:Authentication token manipulation error** 报错。

```
[root@instance-v1j9um04 ~]# passwd  root
Changing password for user root.
New password:
Retype new password:
passwd: Authentication token manipulation error
[root@instance-v1j9um04 ~]# _
```

原因分析

- 由于密码文件（/etc/passwd和/etc/shadow）权限异常导致
- 磁盘空间已满导致。

处理方式

1. 执行 `df -h` 查看磁盘空间，如果空间已满，需要清理空间，且磁盘空间正常，则执行以下命令查看文件属性。

```
$ lsattr /etc/passwd /etc/shadow
```

```
[root@instance-v1j9um04 ~]# lsattr /etc/passwd /etc/shadow
----------e- /etc/passwd
-----a-------e- /etc/shadow
[root@instance-v1j9um04 ~]#
```

2. 根据 `lsattr` 命令查看，/etc/shadow目录有a或者i的权限，需要把a以及i的权限取消，执行如下命令取消权限。

```
$ chattr -a /etc/shadow
$ chattr -i /etc/shadow
```

3. 执行以下命令查看文件属性，只有"e"权限即为正常。

```
[root@instance-v1j9um04 ~]# chattr -a /etc/shadow
[root@instance-v1j9um04 ~]# lsattr /etc/shadow
----------e- /etc/shadow
[root@instance-v1j9um04 ~]# _
```

4. 再次重置密码。

```
$ passwd
```

显示 **password updated successfully** 即为重置密码成功，输入 **exit** 后可以从单用户退出到正常模式。

GPU实例宕机故障排查

故障现象

使用CentOS等Linux系统的GPU实例且部署的业务会频繁进行内存分配时，实例发生宕机。

故障原因

CentOS等Linux操作系统默认开启透明大页，若您GPU实例上部署的业务中有频繁的内存分配场景，就可能触发透明大页频繁进行内存规整、内存迁移等操作来聚合透明大页，这些操作可能会通过发送ipi让某个CPU执行flush tlb操作，当这些flush tlb操作堆积起来，CPU没办法调度其他进程运行，可能造成soft lockup，最终导致系统宕机。

> 透明大页THP（Transparent Huge Pages）是Linux内核中的一个通用特性，它可以自动将小页面合并成大页面，减小TLB miss，提高内存访问的效率。但是，透明大页的存在有可能会造成系统的卡顿，更严重的可能造成宕机。所以需要您根据自己的业务场景来决定打开或者关闭。
>
> 一般而言，如果用户的业务有频繁的短生命周期的内存分配的场景，比如AI训练、PFS等场景，建议关闭透明大页功能。

处理步骤

验证是否由于透明大页引起

步骤一 登录服务器

可以通过ssh远程连接服务器，如果无法远程连接，可以使用VNC登录。

步骤二 查看系统日志

1.使用cd进入/var/crash目录或者/home/coresave目录下查看，如果有类似有 127.0.0.1-2024-07-23-15:00:45 这样的目录，说明kdump服务将宕机发生时的信息记录了下来。进入这个目录，可以看到存在一个vmcore-dmesg.txt文件：

```
[root@bddx-bcc-online-com0-800.bddx.baidu.com 127.0.0.1-2024-07-23-15:00:45]# pwd
/home/coresave/127.0.0.1-2024-07-23-15:00:45
[root@bddx-bcc-online-com0-800.bddx.baidu.com 127.0.0.1-2024-07-23-15:00:45]# ls
vmcore  vmcore-dmesg.txt
```

2.打开vmcore-dmesg.txt文件，如果发现异常栈中有如下函数调用，比如do_huge_pmd_anonymous_page，说明可能是因为透明大页引起的异常。

```
[1056864.772511]  flush_tlb_mm_range+0xe7/0x120
[1056864.777545]  ? page_vma_mapped_walk+0x39f/0x7a0
[1056864.783004]  ptep_clear_flush+0x54/0x60
[1056864.787733]  try_to_unmap_one+0x257/0xab0
[1056864.792627]  ? __mod_memcg_lruvec_state+0x21/0x100
[1056864.798282]  rmap_walk_file+0xf7/0x260
[1056864.802875]  try_to_unmap+0x9c/0xf0
[1056864.807192]  ? page_remove_rmap+0x280/0x280
[1056864.812176]  ? page_not_mapped+0x20/0x20
[1056864.816880]  ? page_get_anon_vma+0x80/0x80
[1056864.821743]  migrate_pages+0x857/0xb00
[1056864.826252]  ? isolate_freepages_block+0x390/0x390
[1056864.831780]  ? __bpf_trace_mm_compaction_kcompactd_sleep+0x10/0x10
[1056864.838693]  compact_zone+0x722/0xd60
[1056864.843064]  compact_zone_order+0xbe/0x100
[1056864.847878]  ? set_next_entity+0x99/0x1b0
[1056864.852589]  try_to_compact_pages+0xb8/0x260
[1056864.857567]  __alloc_pages_direct_compact+0x87/0x160
[1056864.863234]  __alloc_pages_slowpath+0x437/0xd50
[1056864.868469]  __alloc_pages_nodemask+0x283/0x2c0
[1056864.873697]  do_huge_pmd_anonymous_page+0x12e/0x7a0
[1056864.879249]  ? do_anonymous_page+0x213/0x380
[1056864.884197]  __handle_mm_fault+0x9aa/0xca0
[1056864.888969]  handle_mm_fault+0xc2/0x1d0
[1056864.893488]  __do_page_fault+0x1ed/0x4c0
[1056864.898079]  do_page_fault+0x37/0x130
[1056864.902414]  ? page_fault+0x8/0x30
[1056864.906490]  page_fault+0x1e/0x30
[1056864.910508] RIP: 0033:0x7fe795ad8b24
```

3.使用如下命令查看当前机器上是否开启透明大页：

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

如果发现如下显示，说明透明大页是开启状态。

```
[root@wfs-test ~]# cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
[root@wfs-test ~]#
```

关闭透明大页

您可以通过执行以下命令来持久化关闭透明大页，但该命令需要重启系统才能生效。

```
sudo grubby --args="transparent_hugepage=never" --update-kernel="/boot/vmlinuz-$(uname -r)"
```

若您当前不想重启系统，可以也执行以下命令临时关闭透明大页：

```
sudo sh -c 'echo "never" > /sys/kernel/mm/transparent_hugepage/enabled'
```

如此不用重启系统，透明大页已关闭，若后续系统重启，持久化关闭透明大页命令生效，透明大页仍处于关闭状态。

关闭大页后，您可以尝试运行业务，观测一段时间，看是否还会宕机。若仍会宕机，可提交工单联系技术人员协助排查。

Linux云服务器执行 yum update 报错存在依赖冲突

问题描述：

以 bcc rocky 9.3 为例，现象如下:

执行 dnf update / yum update 命令时遇到一些软件包冲突。

```
[root@localhost ~]# dnf update
Last metadata expiration check: 0:02:26 ago on Mon 13 Jan 2025 05:09:23 PM CST.
Error:
 Problem 1: cannot install both openmpi-2:4.1.1-7.el9.x86_64 from appstream and openmpi-4.1.5a1-1.58511.x86_64 from @System
  - package clusterkit-1.8.428-1.58511.x86_64 from @System requires libmpi.so.40()(64bit), but none of the providers can be installed
  - cannot install the best update candidate for package openmpi-4.1.5a1-1.58511.x86_64
  - problem with installed package clusterkit-1.8.428-1.58511.x86_64
 Problem 2: package perftest-23.07.0.0.27-1.el9.x86_64 from baseos requires libefa.so.1()(64bit), but none of the providers can be installed
  - package perftest-23.07.0.0.27-1.el9.x86_64 from baseos requires libefa.so.1(EFA_1.1)(64bit), but none of the providers can be installed
  - cannot install both libibverbs-51.0-1.el9.x86_64 from baseos and libibverbs-58mlnx43-1.58511.x86_64 from @System
  - cannot install the best update candidate for package perftest-4.5-0.18.gfcddfe0.58511.x86_64
  - problem with installed package libibverbs-58mlnx43-1.58511.x86_64
 Problem 3: cannot install both ucx-1.15.0-2.el9.x86_64 from appstream and ucx-1.14.0-1.58511.x86_64 from @System
  - package ucx-knem-1.14.0-1.58511.x86_64 from @System requires ucx(x86-64) = 1.14.0-1.58511, but none of the providers can be installed
  - cannot install the best update candidate for package ucx-1.14.0-1.58511.x86_64
  - problem with installed package ucx-knem-1.14.0-1.58511.x86_64
 Problem 4: cannot install both ucx-1.15.0-2.el9.x86_64 from appstream and ucx-1.14.0-1.58511.x86_64 from @System
  - package ucx-cma-1.15.0-2.el9.x86_64 from appstream requires ucx(x86-64) = 1.15.0-2.el9, but none of the providers can be installed
  - package ucx-xpmem-1.14.0-1.58511.x86_64 from @System requires ucx(x86-64) = 1.14.0-1.58511, but none of the providers can be installed
  - cannot install the best update candidate for package ucx-cma-1.14.0-1.58511.x86_64
  - problem with installed package ucx-xpmem-1.14.0-1.58511.x86_64
(try to add '--allowerasing' to command line to replace conflicting packages or '--skip-broken' to skip uninstallable packages or '--nobest' t
o use not only best candidate packages)
[root@localhost ~]#
```

原因分析：

上述问题镜像已经预装 **NVIDIA MLNX_OFED 驱动**，用户选购智能卡服务时提供高性能网络支持。由于 ofed 驱动相关软件包（如： openmpi / perftest / ucx）比系统软件源同名的软件包版本低，update 阶段系统默认使用软件源提供软件包覆盖本地与**NVIDIA MLNX_OFED 驱动**相关软件包，但系统无法解决软件包依赖问题，导致安装阶段出现依赖冲突，update 失败。

处理方案：

使用 **dnf update / yum update** 升级系统软件包时，明确上述依赖冲突问题由 **NVIDIA MLNX_OFED 驱动**导致，可以使用 **dnf update --nobest** 进行升级。 通过查看软件包是否包含 **MLNX_OFED 驱动**版本信息判断是否为 **MLNX_OFED 驱动**相关软件包，如：

```
[root@localhost ~]# rpm -qi openmpi
Name        : openmpi
Version     : 4.1.5a1
Release     : 1.58511
Architecture: x86_64
Install Date: Mon 18 Nov 2024 08:23:26 PM CST
Group       : Development/Libraries
Size        : 20968058
License     : BSD
Signature   : RSA/SHA256, Mon 10 Jun 2024 11:05:39 PM CST, Key ID a024f6f0e6d6a281
Source RPM  : openmpi-4.1.5a1-1.58511.src.rpm
Build Date  : Mon 10 Jun 2024 10:42:43 PM CST
Build Host  : c-237-149-20-023.mtl.labs.mlnx
Relocations : /usr/mpi/gcc/openmpi-4.1.5a1
Packager    : Mellanox Technologies
Vendor      : Mellanox Technologies # 包含当前 MLNX_OFED 驱动信息

# 服务等级协议SLA

## BCC服务等级协议SLA(V3.0)

**协议生效时间：2024年09月25日**
```

本服务等级协议（Service Level Agreement，以下简称 "SLA"）规定了百度智能云向客户提供的云服务器BCC（Baidu Cloud Compute，简称"BCC"）的服务可用性等级指标及赔偿方案。

1.定义

服务周期： 一个服务周期为一个自然月。

单实例服务周期总分钟数： 按照单实例服务周期内的总天数 * 24（小时） * 60（分钟）计算。

实例不可用： 因百度智能云故障导致了，一台设置了出入允许规则的BCC实例以TCP或者UDP协议与任一IP地址的双向（出/入）都无法连通，且该状态持续一分钟以上，视为该分钟内BCC实例不可用。

单实例服务不可用分钟数： 在一个服务周期内单BCC实例不可用分钟数之和。

单地域多可用区服务不可用： 如用户BCC实例在同一地域部署于至少2个可用区（以下简称：单地域多可用区），若该地域任一可用区发生该用户的全部BCC实例不可用，且该用户在该地域其他可用区的BCC实例亦同时发生实例不可用（以下简称：同地域其他可用区不可用BCC实例），则此同地域其他可用区不可用BCC实例被视为单地域多可用区服务不可用。

单实例单地域多可用区服务不可用分钟数： 在一个服务周期内，单BCC实例的单地域多可用区服务不可用的分钟数之和。

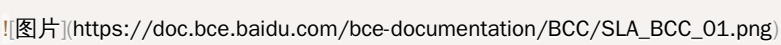
月度服务费： 为用户在一个服务周期（即自然月）内单BCC实例所支付的服务费用总额，如用户一次性支付多个月的服务费，则将按照所购买的月份分摊计算月度服务费。

2. 服务可用性

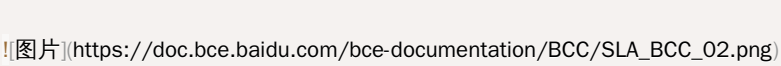
2.1 服务可用性计算方式

BCC的服务可用性将根据服务周期，按如下两种维度分别统计每台BCC实例的可用性：

(1) 单实例维度：



(2) 单地域多可用区维度：



2.2 服务可用性承诺

1.对于单实例维度，百度智能云承诺一个服务周期内BCC的服务可用性不低于99.975%；

2.对于单地域多可用区维度，百度智能云承诺一个服务周期内BCC的服务可用性不低于99.995%；

如BCC服务未达到上述服务可用性承诺，客户可以根据本协议第三条约定获得赔偿。赔偿范围不包括以下原因所导致的不可用：

(1) 百度智能云预先通知用户后进行系统维护所引起的，包括割接、维修、升级和模拟故障演练；

(2) 任何百度智能云所属设备以外的网络、设备故障或配置调整引起的；

(3) 用户的应用程序或数据信息受到黑客攻击而引起的；

(4) 用户维护不当或保密不当致使数据、口令、密码等丢失或泄漏所引起的；

(5) 用户的疏忽或由用户授权的操作所引起的；

(6) 不可抗力以及意外事件引起的；

(7) 本地盘导致的不可用（使用本地盘存储的实例的本地磁盘中数据有丢失的风险，比如本地磁盘损坏）；

(8) 弹性裸金属实例、使用本地盘存储的实例、带有异构卡（GPU/NPU/FPGA等）的实例因硬件等故障等待客户授权及维修

(8) 弹性裸金属实例、使用本地盘存储的实例、带有异构卡（GPU/NPU/FPGA等）的实例因硬件等故障等待客户授权及维修期间造成的服务不可用；

(9) 对于弹性裸金属实例，由于用户未安装最新版本的监控组件导致百度智能云不能及时感知硬件故障而产生的服务不可用；

(10) 用户的抢占实例因为出价低于市场价格或者库存不足而被释放；由于账户欠费/实例到期导致BCC实例被暂停服务或被释放等；

(11) 用户未遵循百度智能云产品使用文档或使用建议引起的，如用户在控制台、API 或者SDK等控制方式对BCC实例进行停机、重启、重装系统、卸载云盘等操作引起的不可用；

(12) 由于客户所安装软件或者其他非百度智能云直接运营的第三方软件或者配置引起的实例出现错误；

(13) BCC服务只赔付BCC自身，不赔付BCC实例所关联的其他云服务；

(14) 其他非百度智能云原因所造成的BCC服务无法正常使用。

3. 服务赔偿条款

3.1 赔偿标准

（1）对于单BCC实例，如服务可用性低于99.975%，可按照下表中的标准获得赔偿，赔偿方式仅限于用于购买BCC产品的代金券，且赔偿总额不超过未达到服务可用性承诺当月客户就该BCC实例支付的单实例月度服务费。

服务可用性	赔付代金券金额
低于99.975%但是等于或高于99%	月度服务费的10%
低于99%但等于或高于95%	月度服务费的25%
低于95%	月度服务费的100%

（2）对于以单地域多可用区部署的BCC，如服务可用性低于99.995%，可按照下表中的标准获得赔偿，赔偿方式仅限于用于购买BCC产品的代金券，且赔偿总额不超过未达到服务可用性承诺当月，用户就该BCC实例支付的月度服务费。

服务可用性	赔付代金券金额
低于99.995%但是等于或高于99%	月度服务费的10%
低于99%但等于或高于95%	月度服务费的25%
低于95%	月度服务费的100%

（3）如果某BCC实例同时满足赔偿方案（1）和（2）的赔偿标准，则以其中赔偿额较高的为准进行计算赔偿金额。

3.2 赔偿申请时限

用户可以在每月的第五（5）个工作日后对上个月发生故障的BCC实例提出赔偿申请，赔偿申请必须限于在BCC故障的相关月份结束后的两（2）个月内提出。超出申请时限的赔偿申请将不被受理。

4. 其他说明

(1) 在法律法规允许的范围内，百度智能云负责对本协议进行解释说明。
(2) 本协议一经公布立即生效，百度智能云有权对本SLA条款作出修改。如本SLA条款有任何修改，百度智能云将以网站公示或发送邮件的方式通知您。如您不同意百度智能云对SLA所做的修改，您有权停止使用BCC服务，如您继续使用BCC服务，则视为您接受修改后的SLA。
(3) 本协议项下百度智能云对于用户所有的通知均可通过网页公告、站内信、电子邮件、手机短信或其他形式等方式进行；该等通知于发送之日视为已送达收件人。因用户未及时获知百度智能云的服务变更或终止条款遭受损失的，百度智能云不承担任何责任。
(4) 本协议的订立、执行和解释及争议的解决均应适用中国法律并受中国法院管辖。如双方就本协议内容或其执行发生任何争议，双方应尽量友好协商解决；协商不成时，任何一方均可向北京市海淀区人民法院提起诉讼。
(5) 本协议构成双方对本协议之约定事项及其他有关事宜的完整协议，除本协议规定的之外，未赋予本协议各方其他权利。
(6) 如本协议中的任何协议无论因何种原因完全或部分无效或不具有执行力，本协议的其余协议仍应有效并且有约束力。
(7) 关于用户约束条款，详见[《用户服务协议》](Agreements/百度智能云用户服务协议/用户的权利与义务.md)中的"用户的权利与义务"相关条款内容。
(8) 关于服务商免责条款，详见[《用户服务协议》](Agreements/百度智能云用户服务协议/免责声明.md)中的"免责声明"相

关条款内容。