百度智能云

OOS 文档

目录

目录 2
功能发布记录 4
产品描述 4
产品描述 4
使用限制 6
使用限制 6
运维编排OOS使用须知 6
操作指南 6
实例批量操作 6
概述 6
实例批量操作 6
模板管理 9
模板语法 9
条件分支 25
模板任务 29
公共模板 55
我的模板 56
执行管理 59
执行概述 59
参数传递 61
执行管理 62
定时运维 64
报警事件运维 67
报警事件运维 67
模板参数填写说明 70
事件详情过滤规则填写说明 72
多用户访问控制 73
最佳实践 75
实例开通自动续费 75
弹性调整EIP实例带宽 79
BSM-agent 83
云助手客户端概述 83
安装BSM-Agent 84
API参考 85
通用说明 85
公共请求与响应头 86
服务域名 86
模板任务（Operator）接口 87
运维模板（Template）接口 87
执行（Execution）接口 94

任务（Task）接口 98
附录 102
常见问题 105
常见问题 105
Python-SDK 107
概述 107
安装SDK 107
初始化 108
OOSClient 109
创建模板接口 110
初始化 113
OOSClient 113
模板任务（Operator）接口 114
运维模板（Template）接口 116
校验运维模板 118
更新运维模板 119
任务执行（Execution）接口 123
Java-SDK 127
概述 127
安装SDK工具包 127
初始化 128
OosClient 128
模板任务（Operator）接口 130
运维模板（Template）接口 133
任务执行（Execution）接口 142

功能发布记录

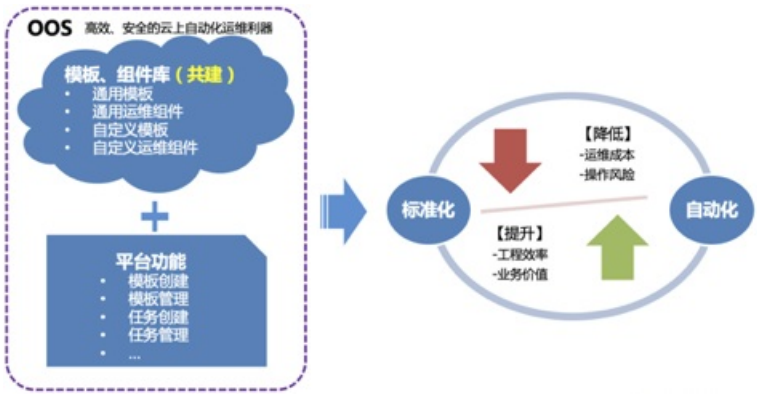
发布时间	功能概述
2024-06	• 创建模版可直接引用前序任务的输出参数
2024-03	• 任务执行失败后可以修改任务参数并重新执行，报警事件运维提供报警消息体示例
2023-10	• 支持报警事件运维能力
2023-08	• 实例管理能力拓展支持至BBC实例
2023-06	• 支持定时运维能力 • 优化信息布局和展示效果
2021-05	• 新增支持region：北京、保定、苏州、广州
2021-04	• 百度智能云提供运维编排服务OOS产品发布，协助上云用户快速建设自动化运维能力，提升运维效能。目前为白名单开放阶段，支持region：金融华中-武汉 • 能够自动化运行和可视化管理运维任务，如常见运维任务的自动化执行：批量执行脚本命令、更新镜像、带宽升级、上传文件至实例等 • 支持自定义业务需要的运维模板，实现复杂运维操作的编排

产品描述

产品描述

产品介绍

运维编排服务（Operation Orchestration Service，简称OOS） 是全面云上自动化运维平台的核心能力，协助用户快速建设自动化运维能力，并提供运维任务管理的服务，帮助用户解决复杂运维操作的处理效率问题。支撑了包括批量操作、周期/事件/时间触发、跨产品编排复杂运维任务等典型应用场景。



核心概念

- **任务**

任务是一个完整模板中的某个节点，定义了具体的操作动作，它可以是一个复杂运维操作的封装，也可以是其他云产品的一次接口调用，任务可以被撤销。

- **模板**

  一个YAML或JSON格式的文件，定义所需要编排的运维操作，一个模板通常包括一个或多个子任务。运维编排服务提供公共沉淀的运维任务模板，也具备用户根据其业务特点自定义模板能力。

- **执行**

  执行是使资源按照模板既定的“剧本”进行自动化操作的过程及结果，一次执行可以控制多台实例按照同一个模板排演，每一次执行都是一次真实的业务场景作业，使资源产生相应的变化。

- **执行模式**

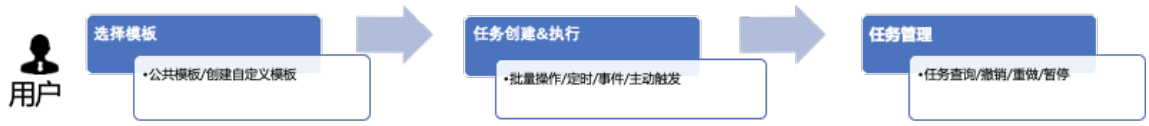
  不同的编排任务执行方式用于满足更多的运维场景需要，主要分为以下几类:

  - 自动执行：模板中定义的所有子任务依次自动执行。
  - 手动执行：模板中定义的子任务需要人工手动点击执行后再执行

  以下能力逐步开放中，敬请期待：

  - 全自动执行：模板中所有子任务依次自行执行，不管子任务返回状态。
  - 需要审批的执行：当执行到模版中定义的不可恢复性操作的子任务时（如删除、释放、停止类操作等），任务执行会切换为等待中状态，直到用户对该任务执行完成了的审批操作其才会进行下一步。
  - 失败暂停：任意子任务失败后暂停当前任务。
  - 单步执行：类似于调试（Debug）功能，当需要详细地了解每一个子任务的执行时，建议使用单步执行模式。

使用流程



应用场景

运维编排服务支持针对资源的多种运维场景，如：批量操作场景、定时运维场景、跨资源操作、根据事件触发的自动化场景等。

批量操作

当您需要针对多个云服务器BCC实例执行开机、关机、重启等常用操作，或者批量调整云服务器BCC实例周边资源（如磁盘、弹性公网IP等）的规格，均可以通过OOS的批量操作能力快速执行运维任务。

OOS为您内置了一些通用的批量操作模板，您无需开发，配置相关参数后便可直接使用。此外，您还可以通过创建私有的模版，把更契合您业务的运维场景固化为模版（如批量发送和执行远程命令、批量上传文件至指定实例、批量进行实例生命周期操作及属性修改等），便于后续快捷使用。欢迎您通过工单反馈更多运维场景，运维编排服务后续将逐步提供您需要的场景模版。

定时运维

当您需要周期地执行某项运维操作，可以通过定时运维功能把模版按照设置的周期定时自动执行。

报警事件运维

使用云监控BCM报警策略的监控和事件监听能力组件，自动跟踪监控数据的变化和监听事件，进行决策、触发处置动作，提升处理效率。例如：设置周期性的定时运维任务、联动云产品指标和事件告警触发运维模版等。

费用说明

运维编排服务本身不收费，但通过运维编排服务创建云服务器BCC实例、磁盘、弹性公网IP等资源，或者调整计费资源规格时，对应的资源会根据其原有收费标准进行收费，具体收费标准参见相关资源的定价。

使用限制

使用限制

运维编排服务中的任务、模板、执行均与区域密切相关，一个区域的模板只能用于编排同一区域下的云产品。

限制项目	默认配额	申请方式
单个模板文件大小	64 KB	无法申请
单个账号可以创建的模版的数量	200个	提交工单
单个账号可以同时执行的并发执行数	100个	提交工单
单个账号每天可以执行的总的组件数（任务数量）	5W个	提交工单
每个模板里面定义的任务数量	100个	无法申请
支持区域	华北-北京、华北-保定、华南-广州、华东-苏州、金融华中-武汉	无法申请

运维编排OOS使用须知

运维编排OOS具备自动执行运维任务的能力，开通和使用运维编排OOS代表您认可本服务的下述自动化行为：

1. 自动管理资源：运维编排OOS会调用相关产品的接口实现自动化操作，资源和操作范围包括且不限于：自动创建云服务器BCC实例、弹性公网IP、共享带宽EIP等更多资源，自动释放云服务器BCC实例、弹性公网IP、共享带宽EIP等更多资源，自动调整云服务器BCC实例的实例规格、弹性公网IP的带宽、共享带宽EIP的带宽等，任务以运维编排OOS控制台实际支持的任务范围为准。
2. 自动下发命令和上传文件：运维编排OOS支持在云服务器BCC实例和弹性裸金属服务器BBC中免登录执行操作系统命令和上传本地文件到操作系统中，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。

操作指南

实例批量操作

概述

当您需要批量对实例做常规运维操作时，可通过“快速新建任务”可视化地执行常用运维任务，主要支持以下几类场景：（部分能力持续开放中）

1. 发送远程命令：实例内部命令，如查看磁盘、IP地址等。
2. 批量上传文件：批量上传文件到指定实例。
3. （持续开放中）实例操作：如启动、停止、重启实例、初始化系统盘、更换系统盘、实例续费等。
4. （持续开放中）实例属性修改：批量修改实例属性、如修改实例名称、HostName、实例描述等。

实例批量操作

概述

当您需要批量发送远程命令或上传文件至指定实例或进行其他常用运维操作时，可以按照本文的方法，在“快速新建执行”功能下进行批量操作。

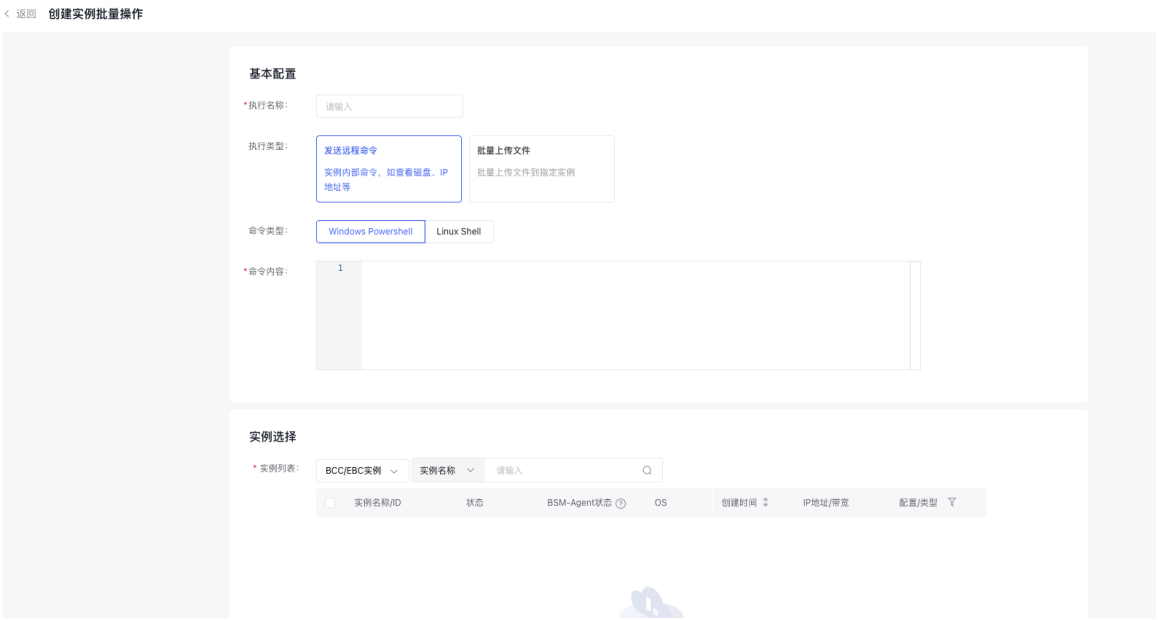
注意事项

- 关于命令类型：目前支持Linux Shell和Windows PowerShell两种命令类型，其他操作系统命令类型持续支持中，敬请期待。
- 目前运维编排服务支持的区域都可以进行实例批量操作，但是不同区域的命令不通用，需要在每个区域分别创建。

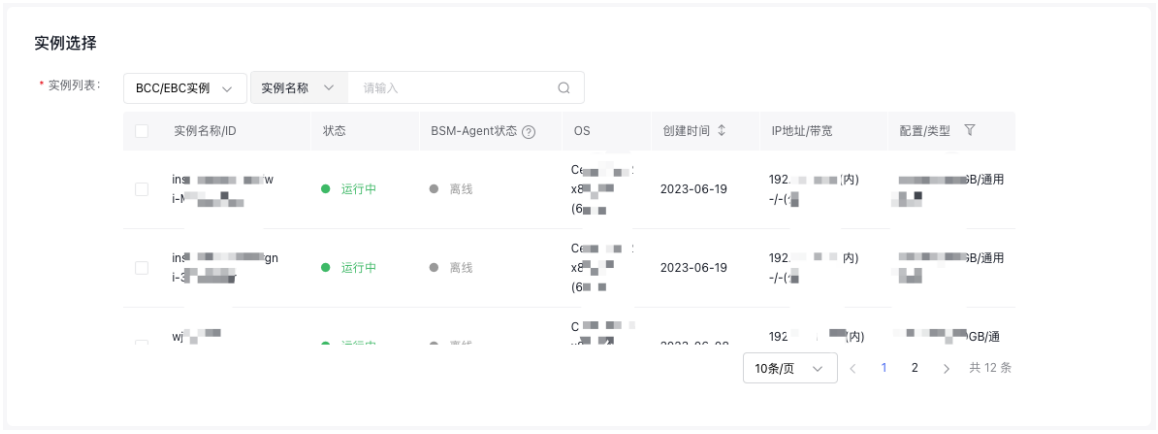
操作步骤

发送远程命令

1. 登录 运维编排服务控制台，选择“快速新建执行 > 实例批量操作 > 创建执行”；
2. 进行“基本配置”，其中根据需要选择执行类型，“发送远程命令”；
3. 选择“命令类型”后输入对应的“命令内容”



4. 根据需要选择需要生效的实例，支持批量选择；



5. 进行“高级选项”配置
   - **描述**：给当前执行增加备注信息
   - **标签**：给当前执行打标签进行标识
   - **执行模式**：
     - 自动执行：模板中定义的所有任务依次自动执行（默认模式，其他模式后续开放）
   - **速率控制类型**：
     - 并发控制：定义运维任务执行各个阶段同时操作的实例比例/个数
   - **最大错误次数** ：指定执行在停止前的最大错误次数，可以设置数值或者百分比，数值选项下默认值是0，表示最多错误个数为1

批量上传文件

1. 进行“基本配置”，其中根据需要选择任务类型，“批量上传文件”；

> 说明：批量上传文件需要开通BOS功能，未开通BOS时，会跳转到BOS开通页面。

2. 选择“源类型”，可选“BOS”或“HTTPS”，如选择BOS时

3. 根据需要选择BOS的Bucket下的文件上传至目标目录

4. 如选择HTTPS时,输入“源路径”

5. 输入需要将文件存储的目标目录，若目录不存在会直接新建。

模板管理

模板语法

模板结构

模板结构

模板是一段JSON或YAML格式的文本，使用UTF-8编码。模板定义了参数、任务、顺序等信息，您可以通过创建模板来定义需要的运维操作。

语法

```
{
  "name": "我的模板", // 模板名称，同一用户下必须唯一，必填
  "description": "模板描述", // 模板描述，选填

  // 模板的标签列表，选填
  "tags": [
    {
      "key": "tagKey",    // 标签健，必填
      "value": "tagValue"  // 标签值，必填
    }
  ],

  // 模板任务集合，必填
  "operators": [
    {
      // 任务名称，需要在当前模板内唯一，必填
      "name": "我的任务",

      // 任务描述，选填
      "description": "任务描述",

      // 任务Tags列表，选填
      "tags": [
        {
          "key": "opTagKey",    // 标签健，必填
          "value": "opTagValue"  // 标签值，必填
        }
      ],

      // 你可以通过'operator'或'template'指定当前任务的执行方式，这两个字段必须有且只有一个被设置
      // operator - 直接设置任务类型，参考文档：'任务类型'
      // template - 嵌套模板，允许在模板的任务中运行另一个模板
```

```
// template - 嵌套模板，允许在模板的任务中运行另一个模板
"operator": "BCE::Agent::ExcuteScript",
"template": {
    "ref": "我的模板_2", // 引用另一个模板
},

// 允许的重试次数，选填，0表示不进行重试
"retries": 0,

// 重试间隔，单位：毫秒，选填，默认值：5min
"retryInterval": 60000,

// 超时时长，单位：毫秒，选填，默认值：1小时
// 若任务执行时长超过该限制，会认为本次任务执行失败，进而触发后续的重试
"timeout": 3600000,

// 输入参数列表，选填
// 不同的任务类型有不同的输入参数，参考文档'任务类型'
// 你可以在设置输入参数时使用内置函数，参考文档'内置函数'
"properties": {
    "argName": "argValue"
},

// 可以使用一组参数，循环执行当前任务，选填
// 参考文档'任务循环'
"loops": [
    {
        "properties": {
            "argName": "argValue"
        }
    }
],

// 允许的并行比例，选填，默认值：0，表示串行执行。取值范围[0, 1]
// 该字段仅当loops字段存在时生效
// 计算得到的loops并发度 = max(1, len(loops) * parallelismRatio)
"parallelismRatio": 0.1,

// 允许的并行个数，选填，默认值：0，表示串行执行
// 只允许对parallelismRatio和parallelismCount之一进行设置
"parallelismCount": 0,

// 允许失败的loops比例，选填，默认值：0，表示不允许失败。取值范围[0, 1]
// 该字段仅当loops字段存在时生效，表示在循环中允许失败的比例。
"allowedFailureRatio": 0.1,

// 允许失败的loops个数，选填，默认值：0，表示不允许失败
"allowedFailureCount": 0,

// 是否需要手动触发，选填，默认值：false
// 若设置为true，OOS不会自动调度该任务，需要用户在控制台或通过API主动调度执行该任务
"manually": false,

// 初始调度延时，OOS将等待指定延时后才调度当前任务，单位：毫秒，选填
"scheduleDelayMilli": 30000,

// 若当前任务执行失败，则暂停整个执行，选填，默认值：false
"pauseOnFailure": false,

// 执行检查条件，其值是一个返回bool值的函数，选填
// 若返回false，当前operator将被跳过，下游节点将仍然被执行
// 参考文档'内置函数'
"condition": {"Fn::Equals": [{"Ref", "env"}, "production"]},
```

```
            // 暂停点，选填
            // 仅在设置了loops参数时生效
            // 例如，[1,3,6]表示在执行第1，3，6个子执行后，暂停loops执行
            // 暂停点优先于并发度：比如5并发，暂停点为1，那么会只执行一个子执行，然后暂停
            "breakpoints": [1, 3, 6],

            // 执行当前任务的触发条件，选填
            // 可选值：
            // ALL_SUCCESS - 全部上游节点执行成功，默认值
            // ONE_SUCCESS - 任一上游节点执行成功
            // ALL_FAILED  - 全部上游节点执行失败
            // ONE_FAILED  - 任一上游节点执行失败
            // ALWAYS      - 无论上游节点的状态，当前节点都会被执行
            "triggerRule": "ALL_SUCCESS",

            // Loops执行时，使用的窗口类型，选填
            // 可选值：
            // SLICING  - 滑动窗口，始终保持有'parallelismCount'个子执行在执行，默认值
            // TUMBLING - 滚动窗口，只有当上一个窗口中的子执行全部执行完成后，才会开始下一个窗口
            "loopWindowType": "SLICING",
        },
    ],

    // 是否线性执行，选填，默认值：false
    // 若设置为true，则认为operators线性连接，忽略links字段
    "linear": false,

    // 模板边集合，若linear=false，则该字段必填
    // src和dst字段，均填写任务的名称
    "links": [
        { "src": "我的任务", "dst": "我的任务_2" }
    ],

    // 模板的全局参数列表，选填
    // 你可以在模板任务的输入参数中通过内置函数引用全局参数，参考文档'内置函数'
    "properties": [
        {
            // 参数名，必填
            "name": "method",

            // 是否必填，选填，默认值：false
            "required": true,

            // 参数类型，必填
            // 可选值:
            // string  - 字符串
            // number  - 数值型
            // boolean - 布尔型
            // list    - 列表
            // object  - 复杂对象
            // bccInstance - bcc实例
            "type": "string",

            // 描述信息，选填
            "description": "参数描述",

            // 可选项列表，选填
            // 设置该参数后，用户可在console中使用下拉列表选择参数值
            "options": ["GET", "POST", "DELETE"],

            // 默认值，选填
```

```
            "defaultValue": "GET",
        },
    ],

    // 当前模板最多同时运行多少个执行，选填，默认值：0，表示无限制
    // 超出限制的执行将进行排队，等待被OOS调度执行
    "parallelism": 20,
}
```

内置函数（BuiltinFunctions）

函数说明

- Fn::Base64Encode：返回输入字符串的Base64编码结果。
- Fn::Base64Decode：返回输入字符串的Base64解码结果。
- Fn::MergeMap：将列表中多个Map合并成一个Map。
- Fn::Join：将一组值连接起来，用特定分隔符隔开。
- Fn::Select：数据元容器通过查询索引返回的单个数据元。
- Fn::Split：通过指定分隔符对字符串进行切片，并返回所有切片组成的列表。
- Fn::Replace：将字符串中指定子字符串用新字符串替换。
- Fn::Min：返回列表中最小元素。
- Fn::Max：返回列表中最大元素。
- Fn::First：返回列表中第一个元素。
- Fn::Last：返回列表中最后一个元素。
- Fn::ListJoin：将多个参数转换成列表。
- Fn::Equals：比较两个值是否相等。
- Fn::If：根据判断条件真假返回不同的值。
- Fn::Not：对值取否操作。
- Fn::AddHour：对一个UTCT时间向后推移N小时。
- Fn::FormatUTCTime：按不同单位精度进行格式化UTC时间。
- Fn::Eval：执行字符串中的单行表达式，并返回执行结果。
- Fn::And：逻辑与。
- Fn::Or： 逻辑或。
- Fn::Intersection：取多个数组的交集。
- Fn::Union：取多个数组的并集。
- Fn::Difference：取多个数组的并集与交集的差集。
- Fn::MapJoin：将两个List组合成一个Map，组合时，会把第一个List内的所有值，作为Map中每个键值对的键，会把第二个List内的值，作为Map中每个键值对。
- Fn::Sub：将输入字符串中的变量替换为指定的值。

- Fn::ConvertMapToList：转化JSON对象为Key、Value格式的列表。
- Fn::GetAtt: 通常用于COS产品中，用于获取指定资源的属性。
- Fn::Exists: 返回指定的key是否存在

使用函数

内建函数的使用主要由内置函数名和待处理参数组成。调用函数的**声明**表达式即表示函数处理参数后返回的结果，且表达式作为函数返回的结果可以像其他参数一样被引用。

Fn::Base64Encode

返回输入字符串的Base64编码结果。

声明

`{"Fn::Base64Encode": "StringToBeBase64"}`

参数

StringToBeBase64：要进行Base64编码的String。

返回值

Base64方式表示的原始字符串。

示例

`{"Fn::Base64Encode": "hello"}`

示例返回

`"aGVsbG8="`

Fn::Base64Decode

返回Base64字符串的解码结果。

声明

`{"Fn::Base64Decode": "Base64ToBeString"}`

参数

Base64ToBeString：Base64编码方式表示的String。

返回值

Base64解码后的原始String。

示例

`{"Fn::Base64Decode": "aGVsbG8="}`

示例返回

`"hello"`

Fn::MergeMap

将列表中多个Map合并成一个Map。

声明

`{"Fn::MergeMap":"aListConatinsDicts"}`

参数

aListConatinsDicts：含有多个映射的List。

返回值

List中多个Map合并成的一个Map。

示例

`{"Fn::MergeMap":[{"key_1":"value_1"},{"key_2":"value_2"}]}`

示例返回

`{"key_1":"value_1","key_2":"value_2"}`

Fn::Join

将一组值连接起来，用特定分隔符隔开。

声明

`{"Fn::Join":["Connector","aListContainsString"]}`

参数

Connector：作为连接符String。

aListContainsString：包含字符串的List。

返回值

List内多个元素连接后的字符串。

示例

`{"Fn::Join":[",",["a","b","c"]]}`

示例返回

`"a,b,c"`

Fn::Select

数据元容器通过查询索引返回的单个数据元。

声明

`{"Fn::Select":["Index","Container"]}`

参数

Index：被查询Container的索引String或Number。

Container：被查询的List或Map。

返回值

查询指定索引返回的结果。

示例1

`{"Fn::Select":["IpAddress",{"IpAddress":["192.168.1.123","192.168.1.234"]}]}`

示例1返回

`["192.168.1.123","192.168.1.234"]`

示例2

`{"Fn::Select":["1",["192.168.1.123","192.168.1.234"]]}`

示例2返回

`"192.168.1.234"`

Fn::Split

通过指定分隔符对字符串进行切片，并返回所有切片组成的列表。

声明

`{"Fn::Split":["Separator","StringToSeprate"]}`

参数

Separator：作为分隔符的String。

StringToSeprate：将被按分隔符分成一个List的String。

返回值

StringToSeprate按分隔符分成的一个List。

示例

`{"Fn:: Split":[",","a,b,c"]}`

示例返回

`["a","b","c"]`

Fn::Replace

将字符串中指定子字符串用新字符串替换。

声明

`{ "Fn::Replace": [{"$varName": "StringToReplaceWith"}, "StringToReplace"]}`

参数

$varName：占位符，以$开始后面连接普通参数名即可，其中$后面参数名命名要求同模板参数名。

StringToReplaceWith：占位符将被替换成的值。

StringToReplace：将被替换的值（含占位符），如"$varName is foo"。

返回值

若参数StringToReplaceWith是"baz"，则返回为"baz is foo"。

示例

```
{  "Fn::Replace": [{"$varName": "baz"},   "$varName is foo"  ]}
```

示例返回

```
"baz is foo"
```

Fn::Min

返回列表中最小元素。

声明

```
{"Fn::Min":"aList"}
```

参数

aList：一个含有元素的List；List中元素类型可以为string,int，各元素必须为同一类型。

返回值

忽略值为None的元素，返回List中最小的元素，若集合元素全部为None则返回None。

示例

```
{"Fn::Min":["123","234"]}
```

示例返回

```
"123"
```

Fn::Max

返回列表中最大元素。

声明

```
{"Fn::Max":"aList"}
```

参数

aList：一个含有元素的List；List中元素类型可以为string、int，各元素必须为同一类型。

返回值

忽略值为None的元素，返回List中最大的元素，若集合元素全部为None则返回None。

示例

```
{"Fn::Max":["123","234"]}
```

示例返回

```
"234"
```

Fn::First

返回列表中第一个元素。

声明

```
{"Fn::First":"aList"}
```

参数

aList：一个含有元素的List；List中元素类型不限。

返回值

输出为集合中迭代返回的第一个元素；对loop来说，按照Items中迭代返回的Item生成的task execution id进行排序。

示例

```
{"Fn::First":["123","234"]}
```

示例返回

```
"123"
```

Fn::Last

返回列表中最后一个元素。

声明

`{"Fn::Last":"aList"}`

参数

aList：一个含有元素的List；List中元素类型不限。

返回值

输出为集合中迭代返回的最后一个元素；对loop来说，按照Items中迭代返回的Item生成的task execution id进行排序。

示例

`{"Fn::Last":["123","234"]}`

示例返回

`"234"`

Fn::ListJoin

将多个参数转换成列表。

声明

`{"Fn::ListJoin": "IterableContainer"}`

参数

IterableContainer是一个可迭代的集合，集合中元素类型不限。

返回值

将集合转换成List输出。

示例

`{"Fn::ListJoin": ["a",2, 3]}`

示例返回

`["a",2,3]`

Fn::Equals

比较两个值是否相等。如果两个值相等，则返回true；如果不相等，则返回false。

声明

`{"Fn::Equals": ["parameter1", "parameter2"]}`

参数

parameter1，parameter2为两个任意类型值。

返回值

如果两个值相等，则返回true；如果不相等，则返回false。

示例

`{"Fn::Equals": ["Running","Stopped"]}`

示例返回

`false`

Fn::If

如果指定的条件计算为true，则返回一个值；如果指定的条件计算为false，则返回另一个值。

声明

`{"Fn::If":["condition","value_if_true","value_if_false"]}`

参数

condition_name：待判断的条件参数。

value_if_true：当指定的条件计算为true时，返回此值。

value_if_false：当指定的条件计算为false时，返回此值。

返回值

当条件计算为true时，返回value_if_true。

当条件计算为false时，返回value_if_false。

示例

`{"Fn::If":[false,"Stopped","Running"]}`

示例返回

`"Running"`

Fn::Or

逻辑或操作。

声明

`{"Fn::Or": ["condition1","condition2"]}`

参数

condition1、condition2：将进行逻辑或的条件。

返回值

当condition1和condition2均为false时，返回false；否则，返回true。

示例

`{"Fn::Or":[true,false]}`

示例返回

`true`

Fn::And

逻辑或操作。

声明

`{"Fn::And": [ "condition1", "condition2"]}`

参数

condition1、condition2·：将要进行逻辑与的条件。

返回值

当condition1和condition2均为true时，返回true；否则，返回false。

示例

`{"Fn::And": [true,false]}`

示例返回

`false`

Fn::Not

对值取否操作。

声明

`{"Fn::Not": "condition"}`

参数

condition：将要取否的条件。

返回值

当condition为true时，返回false；当condition为false时，返回true。

示例

`{"Fn::Not": true}`

示例返回

`false`

Fn::AddHour

表示对一个UTCT时间向后推移N小时，使其成为一个新的UTC时间。

声明

`{"Fn::AddHour": ["utc_time", "hours_count" ]}`

参数

utc_time：一个要向后推移的UTC时间。

hours_count：要向后推移的小时数，限取整数。

返回值

utc_time被推移hours_count小时后的新UTC时间。

示例

`{"Fn::AddHour": ["2019-06-01T02:12:45Z", 6]}`

示例返回

`"2019-06-01T08:12:45Z"`

Fn::FormatUTCTime

表示对一个UTC时间按不同单位精度进行格式化，生成的新时间其精度只保留到需要的时间单位。

声明

`{"Fn::FormatUTCTime": ["utc_time", "to_be_utc_time_format" ]}`

参数

utc_time：待被格式化的UTC时间。

to_be_utc_time_format：将要把utc_time格式化成的新时间格式。

返回值

新格式的时间。

示例

`{"Fn::FormatUTCTime": ["2019-06-01T02:12:45Z","%Y-%m-%dT%H:%MZ"]}` **示例返回**

`"2019-06-01T02:12Z"`

Fn::Eval

表示执行一个字符串中的表达式（不支持表达式中含换行），并返回执行结果。

声明

`{"Fn::Eval":["expression"]}`

参数

expression：被执行的表达式（如运算式）。

返回值

表达式执行结果。

示例

`{"Fn::Eval":["1+3*2+3%2+1"]}`

示例返回

`9`

Fn::Intersection

对多个数组取交集，返回共同包含的值。

声明

`{"Fn::Intersection":[ list1, list2]}`

参数

list1、list2：要取交集的数组。

返回值

一个数组，其中的元素是多个数组共同包含的值。

示例

`{"Fn::Intersection":[["i-1","i-2"],["i-1","i-3"]]}`

示例返回

`["i-1"]`

Fn::Union

对多个数组中的元素去重后合并为一个数组，返回合并后的新数组。

声明

`{"Fn::Union":[ list1, list2]}`

参数

list1、list2：要合并的数组。

返回值

一个数组，其中的元素是多个数组的并集。

示例

```
{"Fn::Union":[["i-1","i-2"],["i-1","i-3"]]}
```

示例返回

```
["i-1","i-2","i-3"]
```

Fn::Difference

查看多个数组的并集与交集的差集，返回包含差集结果的新数组。

声明

```
{"Fn::Difference":[ list1, list2]}
```

参数

list1、list2：要筛选存在不同元素的数组。

返回值

一个数组，其元素是多个数组的并集与交集的差集。

示例

```
{"Fn::Difference":[["i-1","i-2"],["i-1"]]}
```

示例返回

```
["i-2"]
```

Fn::MapJoin

将两个List组合成一个Map，组合时，会把第一个List内的所有值，作为Map中每个键值对的键，会把第二个List内的值，作为Map中每个键值对的值。

声明

```
{'Fn::MapJoin': [List1, List2] }
```

参数

aListConatinsDicts：含有多个映射的List。

返回值

一个Map，其中每个键值对的键是List1的对应元素，每个键值对的值是List2的对应元素。

示例

```
{'Fn::MapJoin':[['i-1', 'i-2', 'i-3'],[1,2,3]]}
```

示例返回

```
{'i-1':1, 'i-2':2, 'i-3':3}
```

Fn::Sub

将输入字符串中的变量替换为指定的值。

声明

```
{'Fn::Sub':[TargetString, JSON]}
```

参数

TargetString、JSON

返回值

String

示例

```
{"Fn::Sub":["Var1: ${Var1}, Var2: ${Var2}",{"Var1":"Var1Value","Var2":"Var2Value"}]}
```

示例返回

```
"Var1: Var1Value, Var2: Var2Value"
```

Fn::ConvertMapToList

转化JSON对象为Key、Value格式的列表。

声明

`{'Fn::ConvertMapToList':[JSON/JSONString]}`

参数

JSON

返回值

List

示例1

`{"Fn::ConvertMapToList":[{"key1":"value1","key2":"value2"}]}`

示例1返回

`[{"Key":"key1","Value":"value1"},{"Key":"key2","Value":"value2"}]`

示例2

`{"Fn::ConvertMapToList":["{\"key1\": \"value1\", \"key2\": \"value2\"}"]}`

示例2返回

`[{"Key":"key1","Value":"value1"},{"Key":"key2","Value":"value2"}]`

任务循环（loops）

任务循环

OOS可以为任务设置loops属性，用来支持对单个任务的循环执行。

若设置了任务的loops属性，当前任务将包含多个子执行，每个子执行对应loops中的一个元素。

语法

假设我们有一个重启虚机的任务，如下所示

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "properties": { // 参数列表
        "instance": {
          "shortId": "i-Qnu499eJ", // 虚机id
        }
      }
    }
  ]
}
```

若我们想批量重启多个虚机，可使用任务的loops属性，对多个bcc实例循环执行该任务

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": [
        { // 第一组参数
          "instance": {
            "shortId": "i-Qnu499eJ",
          }
        },
        { // 第二组参数
          "instance": {
            "shortId": "i-Bt9Okyh8",
          }
        },
      ]
    }
  ]
}
```

除了上述形式，我们还可以使用更简化的形式来指定loops

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": {
        // 使用list指定instance参数的多个值
        "instance": [
          {
            "shortId": "i-Qnu499eJ",
          },
          {
            "shortId": "i-Bt9Okyh8",
          }
        ]
      }
    }
  ]
}
```

当然，你可以使用内置函数指定loops参数

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": {
        // 使用函数，引用当前执行的其他变量，主要该变量的类型，比如与参数类型匹配
        "instance": {"Ref": "inputInstances"},
      }
    }
  ]
}
```

并发控制

你可以批量执行循环中的子执行，通过以下属性控制循环执行的并发度和并发方式

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": [], // loops列表

      // 允许的并行比例，选填，默认值：0，表示串行执行。取值范围[0, 1]
      // 该字段仅当loops字段存在时生效
      // 计算得到的loops并发度 = max(1, len(loops) * parallelismRatio)
      "parallelismRatio": 0.1,

      // 允许的并行个数，选填，默认值：0，表示串行执行
      // 只允许对parallelismRatio和parallelismCount之一进行设置
      "parallelismCount": 0,

      // Loops执行时，使用的窗口类型，选填
      // 可选值：
      // SLICING - 滑动窗口，始终保持有'parallelismCount'个子执行在执行，默认值
      // TUMBLING - 滚动窗口，只有当上一个窗口中的子执行全部执行完成后，才会开始下一个窗口
      "loopWindowType": "SLICING",
    }
  ]
}
```

容错机制

你可以设置最多允许多少个子执行执行失败，若失败的子执行数小于等于设置的阈值，则仍然认为当前任务执行成功。

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": [], // loops列表

      // 允许失败的loops比例，选填，默认值：0，表示不允许失败。取值范围[0, 1]
      // 该字段仅当loops字段存在时生效，表示在循环中允许失败的比例。
      "allowedFailureRatio": 0.1,

      // 允许失败的loops个数，选填，默认值：0，表示不允许失败
      "allowedFailureCount": 0,
    }
  ]
}
```

设置断点

有时候，你希望在几个子执行执行完成后，可以暂停整个流程，以方便你进行一些观察之后再继续。OOS通过设置断点实现这一功能。

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务
      "loops": [], // loops列表

      // 断点（也称作暂停点），选填
      // 仅在设置了loops参数时生效
      // 例如，[1,3,6]表示在执行第1，3，6个子执行后，暂停loops执行
      // 暂停点优先于并发度：比如5并发，暂停点为1，那么会只执行一个子执行，然后暂停
      "breakpoints": [1, 3, 6],
    }
  ]
}
```

参数（properties）

参数

在创建和执行模板时，使用参数（Properties）可提高模板灵活性和可复用性，模版多个任务都可以引用同一个参数。

模板全局参数

你可以在创建模板时设置全局参数，在执行模板时动态指定这些模板的值，从而实现模板的复用。

```
{
  "name": "我的模板", // 模板名称

  // 模板全局参数，选填
  "properties": [
    {
      // 参数名，必填
      "name": "argName",

      // 是否必填，选填，默认值：false
      "required": true,

      // 参数类型，必填
      // 可选值:
      // string - 字符串
      // number - 数值型
      // boolean - 布尔型
      // list - 列表
      // object - 复杂对象
      // bccInstance - bcc实例
      "type": "string",

      // 描述信息，选填
      "description": "参数描述",

      // 可选项列表，选填
      // 设置该参数后，用户可在console中使用下拉列表选择参数值
      "options": ["value_1", "value_2", "value_3"],

      // 默认值，选填
      "defaultValue": "value_1",
    },
  ]
}
```

目前，几乎支持所有类型的参数，包括：

类型	描述	示例
string	字符串	`"text"`
number	数值型，包括整形和浮点型	`1`, `0.1`
boolean	布尔型	`true`, `false`
list	数组，填写json表达式	`[1, 2, 3]`, `["v1", "v2"]`
object	复杂对象（字典），填写json表达式	`{"key": "value"}`
bccInstance	bcc实例，按指定的字段填写	`[{"shortId": "i-2cb797e8"}]`

任务参数

和全局参数类似，每个任务也有其对应的参数列表。

在创建模板时，你需要为某一个任务的参数设置输入值。

```
{
  "operators": [
    {
      "name": "重启虚机", // 任务名称
      "operator": "BCE::Bcc::RestartInstance", // 重启虚机任务

      // 设置任务的参数
      "properties": {

        // 'instance'是一个bccInstance类型的参数
        "instance": {
          "shortId": "i-Qnu499eJ", // 虚机id
        }
      }
    }
  ]
}
```

引用参数

在设置任务的参数值时，你可以

- 引用全局参数
- 引用上游任务的输出参数

OOS支持两种引用方式

使用模板参数进行引用

OOS支持在字符串类型的参数中使用模板引用其他参数

```
{
  "operators": [
    {
      "name": "执行脚本", // 任务名称
      "operator": "BCE::Agent::ExecuteScript", // 执行脚本任务

      // 设置任务的参数
      "properties": {
        "content": "echo hello {{.paramFirstName}} {{.paramFamilyName}}"
      }
    }
  ]
}
```

oos内部使用的是go模板引擎text/template，支持完整的go模板语法。

> 注意，使用模板渲染的输出类型是字符串
> 如果需要引用一个非字符串类型的参数，就需要使用函数进行引用，如下节所示。

使用函数进行引用

你可以使用内置函数来引用参数，这是一种更加灵活和强大的方式，因为你不止可以简单的引用，你还可以对应用的参数进行任意的函数操作。

```
{
  "operators": [
    {
      "name": "执行脚本", // 任务名称
      "operator": "BCE::Agent::ExecuteScript", // 执行脚本任务

      // 设置任务的参数
      "properties": {
        "content": {"Fn::Join": [" ", ["echo", {"Ref": "paramName"}]]}, // 等效于 echo {{.paramName}}
        "instance": {"Ref": "paramInstance"}
      }
    }
  ]
}
```

条件分支

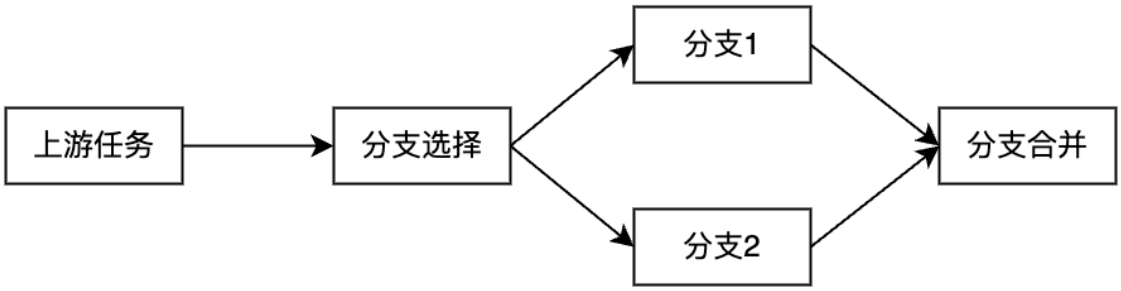
条件分支

模板是由多个任务构成的有向无环图（directed acyclic graph，简称DAG），你可以在DAG中构造分支，并通过上游任务的执行结果，选择执行哪一条分支。

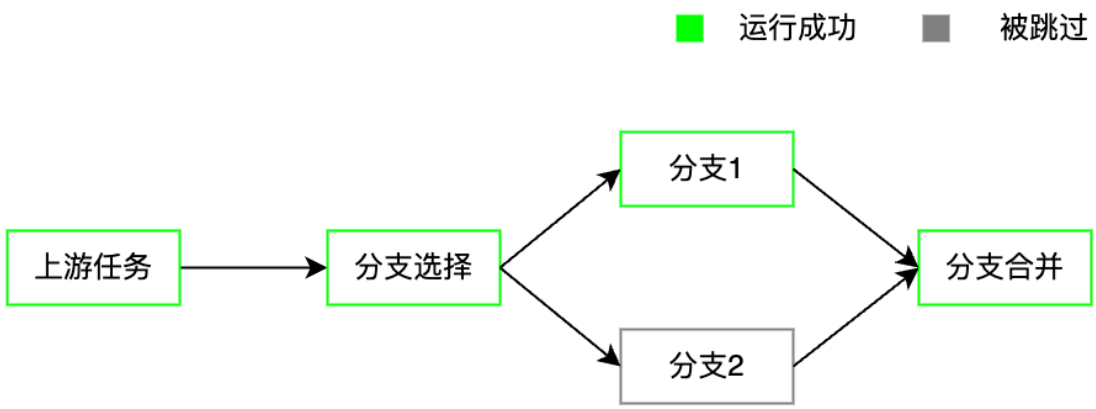
一个示例模板如下所示：

```
{
  "operators": [
    {
      // 根据该任务的输出决定分支
      "name": "上游任务",
      "operator": "BCE::CallAPI",
    },
    {
      // 通过内置任务'BCE::SwitchTo'定义条件分支
      "name": "分支选择",
      "operator": "BCE::SwitchTo",
      "properties": {
        "switch": [
          {
            "case": {"Ref": "success"},
            "next": "分支1",
          },
          {
            "case": {"Fn::Not": {"Ref": "success"}},
            "next": "分支2",
          },
        ],
        "default": "分支1",
      }
    },
    {
      // 下游分支1
      "name": "分支1",
    },
    {
      // 下游分支2
      "name": "分支2",
    },
    {
      // 分支合并后的任务
      "name": "分支合并",
    },
  ],
  "links": [
    {"src": "上游任务", "dst": "分支选择"},
    {"src": "分支选择", "dst": "分支1"},
    {"src": "分支选择", "dst": "分支2"},
    {"src": "分支1", "dst": "分支合并"},
    {"src": "分支2", "dst": "分支合并"},
  ]
}
```

构建的模板如下所示：



若"上游任务"输出success=true，则会运行分支1，分支2会被跳过（state=SKIPPED），执行结束后的状态如下图所示：



触发规则（TriggerRule）

触发规则

默认情况下，当一个任务的所有上游任务都成功运行后，运维编排服务才会开始调度并运行当前任务。除此之外，运维编排服务也支持用户为每个任务设置不同的触发规则。

```
{
  "operators": [
    {
      "name": "我的任务", // 任务名称

      // 执行当前任务的触发条件，选填
      // 可选值：
      // ALL_SUCCESS - 全部上游节点执行成功，默认值
      // ONE_SUCCESS - 任一上游节点执行成功
      // ALL_FAILED  - 全部上游节点执行失败
      // ONE_FAILED  - 任一上游节点执行失败
      // ALWAYS      - 无论上游节点的状态，当前节点都会被执行
      "triggerRule": "ALL_SUCCESS",
    }
  ]
}
```

各触发规则的说明如下。我们先定义如下几个变量：

- **num_total** = 所有上游任务数量
- **num_success** = 成功的上游任务数量（state = SUCCESS | IGNORED）
- **num_failed** = 失败的上游任务数量（state = FAILED | UPSTREAM_FAILED）
- **num_skipped** = 被跳过的上游任务数量（state = SKIPPED）

注意，num_total = num_success + num_failed + num_skipped

TriggerRule	触发规则	触发方式
ALL_SUCCESS	全部上游节点执行成功	1. 若num_success == num_total，则运行当前节点（state=RUNNING） 2. 若num_skipped == num_total，则跳过当前节点（state=SKIPPED） 3. 否则（num_failed > 0），令当前节点失败（state=UPSTREAM_FAILED）
ONE_SUCCESS	任一上游节点执行成功	1. 若num_success > 0，则运行当前节点（state=RUNNING） 2. 若num_skipped == num_total，则跳过当前节点（state=SKIPPED） 3. 否则（num_success == 0），令当前节点失败（state=UPSTREAM_FAILED）
ALL_FAILED	全部上游节点执行失败	1. 若num_failed == num_total，则运行当前节点（state=RUNNING） 2. 否则（num_failed != num_total），跳过当前节点（state=SKIPPED）
ONE_FAILED	任一上游节点执行失败	1. 若num_failed > 0，则运行当前节点（state=RUNNING） 2. 否则（num_failed == 0），跳过当前节点（state=SKIPPED）
ALWAYS	无论上游节点的状态，当前节点都会被执行	总是执行当前节点（state=RUNNING）

任务重试

任务重试

您可以为每个任务设置重试次数和重试间隔。

```
{
  "operators": [
    {
      "name": "我的任务", // 任务名称

      // 允许的重试次数，选填，0表示不进行重试
      "retries": 0,

      // 重试间隔，单位：毫秒，选填，默认值：5min
      "retryInterval": 60000,
    }
  ]
}
```

若任务的失败次数未超过重试次数，在重试间隔期间，任务会进入等待重试状态（state=UP_FOR_RETRY）。

轮询类任务

我们在现实中经常遇到以下场景：

1. 发起一个任务

2. 轮询这个任务的状态，直至成功或超时

我们可以利用任务的重试机制，完成这一任务。

```
{
  "operators": [
    {
      "name": "发起任务",
      "operator": "BCE::Agent::CallAPI", // 调用发起任务的API
    },
    {
      "name": "轮询任务",
      "operator": "BCE::Agent::CallAPI", // 调用查询任务的API

      // 重试10次，每30秒重试一次，即允许这个任务执行5分钟，若超时则认为轮询失败
      "retries": 10,
      "retryInterval": 30000,
    },
  ]
}
```

嵌套模板

嵌套模板

运维编排服务支持在模板的任务中，嵌套执行另一个模板。

```
{
  "operators": [
    {
      "name": "我的任务",

      // 在任务中，嵌套执行另一个模板
      "template": {
        "ref": "我的模板_2", // 引用另一个模板
      },

      // 由properties指定的参数，会用作嵌套模板的全局参数
      "properties": {
        "arg1": "value1",
        "arg2": "value2",
      },

      // 您仍然可以设置嵌套模板的重试次数和重试间隔
      "retries": 3,
      "retryInterval": 60000,

      // 您也可以循环执行嵌套模板
      "loops": [],
    }
  ]
}
```

被嵌套执行的模板，会作为当前任务的子执行被oos调度和执行，你可以在任务详情中查看子执行的执行详情。

递归嵌套模板

您可以在嵌套的模板任务中，继续嵌套其他模板，以实现递归嵌套。

目前允许的最大递归深度是10，即允许递归嵌套10个子模板。

模板任务

API相关

执行API

用途 此Operator可以用来调用某个云产品（如云服务器BCC）的OpenAPI。

支持的API操作 BCC：创建/删除/开机/关机/重启

EIP：创建/删除/绑定BCC

ENI：创建/删除/绑定BCC/绑定EIP

CDS：创建/删除/绑定BCC

参数

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

名称	类型	描述
serviceName	String	必填，云产品名称，当前支持BCC、EIP、ENI、CDS四种云产品
apiName	String	必填，云产品的API名称，当前仅支持创建/删除，部分云产品支持绑定操作，详见【用途-支持的API操作】【API帮助】
parameters	Object	选填,调用OpenAPI所需要的参数，具体参考【API帮助】中的参考API文档

示例

```
{
"name":"executeAPI"
  "operator": "BCE::ExecuteAPI",
"properties": {
 "serviceName": "BCC",
 "apiName": "CreateInstanceBySpec",
 "parameters": {
  "imageId": "m-8S2nu8s3",
  "spec": "bcc.g3.c1m1",
  "name": "test_YYY1_bcc",
  "rootDiskSizeInGb": 40,
  "purchaseCount": 1,
  "billing": {
   "paymentTiming": "Postpaid",
   "reservation": {
    "reservationLength": 1,
    "reservationTimeUnit": "Month"
   }
  }
 }
}
}
```

API帮助

BCC

API名称	参考API文档
CreateInstanceBySpec	https://cloud.baidu.com/doc/BCC/s/2k3rau7n4
DeleteInstanceBySpec	https://cloud.baidu.com/doc/BCC/s/2jwvyoclc

EIP

API名称	参考API文档
Create	https://cloud.baidu.com/doc/EIP/s/Wjwvz30fv
Delete	https://cloud.baidu.com/doc/EIP/s/Rjwvz32ig
Bind	https://cloud.baidu.com/doc/EIP/s/9jwvz31gn

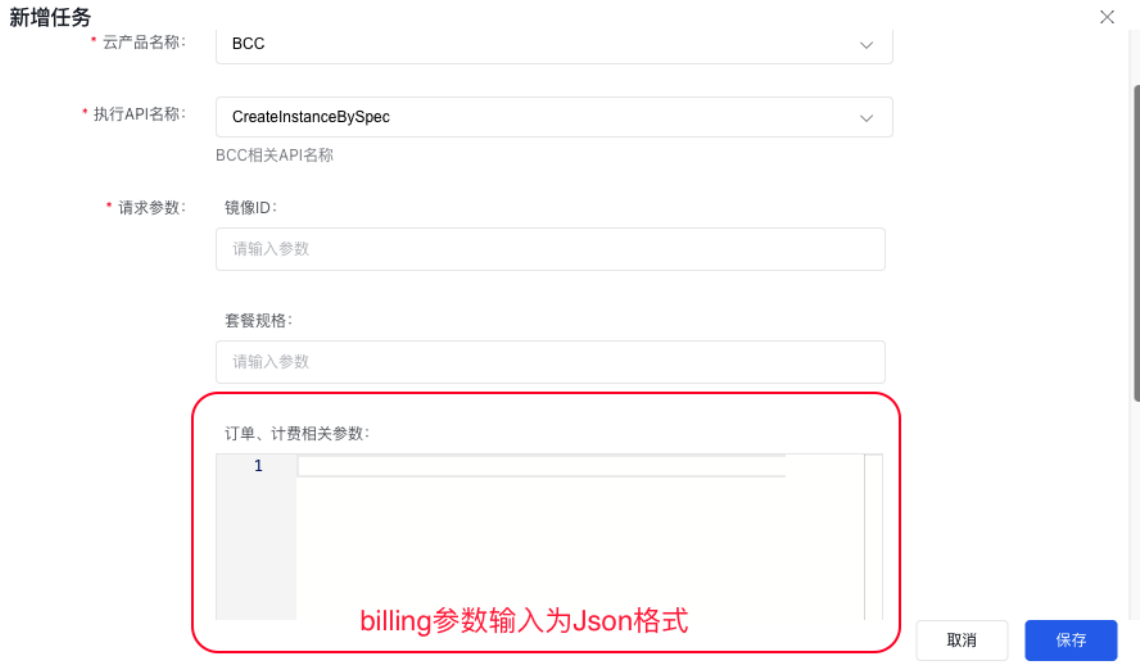
ENI

API名称	参考API文档
Create	https://cloud.baidu.com/doc/VPC/s/pkknfara0
Delete	https://cloud.baidu.com/doc/VPC/s/ikknfinx7
BindEip	https://cloud.baidu.com/doc/VPC/s/vkknfr857
AttachBcc	https://cloud.baidu.com/doc/VPC/s/qkknfp44w

CDS

API名称	参考API文档
Create	https://cloud.baidu.com/doc/BCC/s/Ujwvyo1ta
Delete	https://cloud.baidu.com/doc/BCC/s/1jwvyo4xu
AttachBcc	https://cloud.baidu.com/doc/BCC/s/Yjwvyo3xg

Console使用说明 创建BCC/创建CDS 通过执行API创建BCC/CDS除【订单、计费相关参数】以外其他参数输入和直接执行创建BCC/CDS的操作符一致



【订单、计费相关参数】输入说明

订单信息定义

Billing

参数名称	类型	描述
paymentTiming	String	付费方式，包括预支付（Prepaid）和后支付（Postpaid）
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

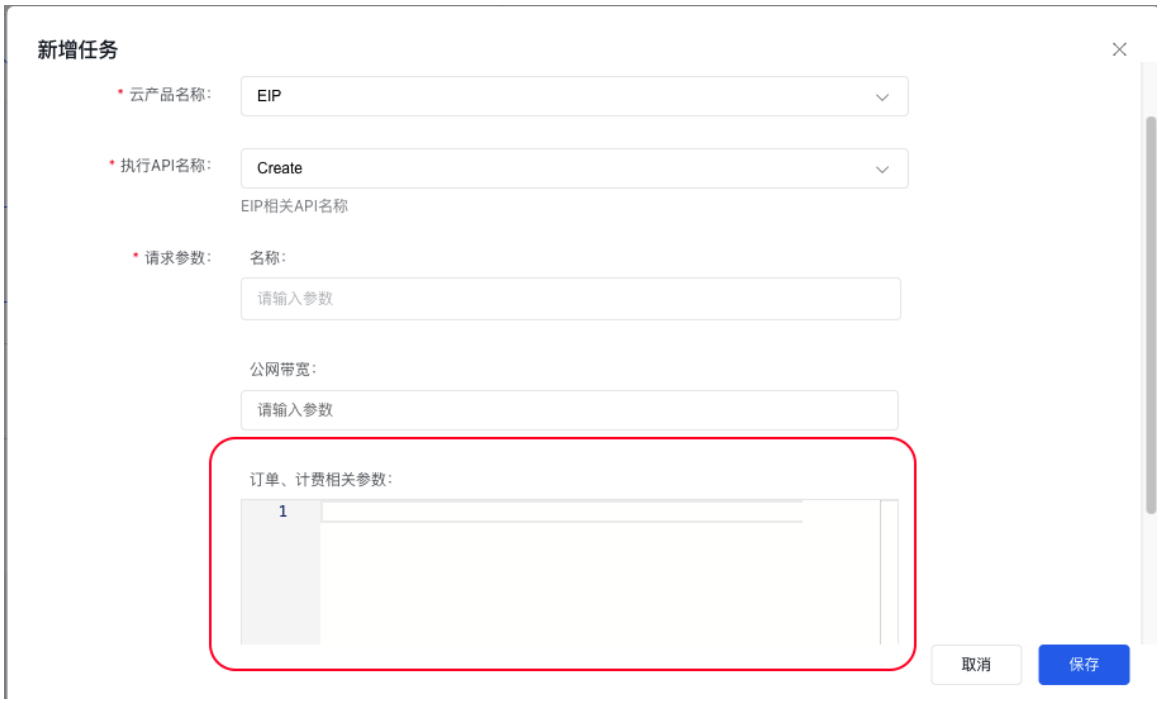
Reservation

参数名称	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

输入示例

```
{
  "paymentTiming": "Prepaid",
  "reservation": {
     "reservationLength": 1,
     "reservationTimeUnit": "Month"
  }
}
```

创建EIP 通过执行API创建EIP除【订单、计费相关参数】以外其他参数输入和直接执行创建EIP的操作符一致



订单信息定义

Billing

参数名称	类型	描述
paymentTiming	String	付费方式，包括预支付（Prepaid）和后支付（Postpaid）
billingMethod	String	计费方式，按流量（ByTraffic）、按带宽（ByBandwidth）、按增强95（ByPeak95）（只有共享带宽后付费支持），后支付时必须设置
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

参数名称	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

输入示例

```
{
  "paymentTiming": "Prepaid",
  "billingMethod": "ByTraffic",
  "reservation": {
     "reservationLength": 1,
     "reservationTimeUnit": "Month"
  }
}
```

创建ENI 通过执行API创建ENI除【内网IPv4】【内网IPv6】以外其他参数输入和直接执行创建EIP的操作符一致

内网IPv4/内网IPv6

Billing

参数名称	类型	描述
publicIpAddress	String	弹性网卡的公网Ip地址，即eip地址
primary	boolean	是否是主Ip
privateIpAddress	String	弹性网卡的内网Ip地址

输入示例

```
{
  "primary": true,
  "privateIpAddress": "2400:da00:e003:0:1d2:100:0:12"
}
```

检查API

用途 该动作用来检查云产品资源是否符合预期状态，如是预期状态，则任务执行成功。否则，任务执行失败。通常用来做前置检查。例如：要停止某BCC实例，则可以先检查该BCC实例的状态是运行中状态（Running），如果不是Running，则不执行停止操作。当前支持BCC/EIP/ENI/CDS的实例检查操作。 **参数**

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数 | 名称 | 类型 | 描述 | | --- | --- | --- | |serviceName| String |必填，需要检查云产品名称当前仅支持BCC/EIP/ENI/CDS | |apiName|String| 必填，执行检查API的名称，默认为CheckFor | | parameters | Object | 选填，执行检查API的请求参数，具体格式见【API帮助】| | desiredValues | List | 必填，期望值，需要等待的具体的API返回值要匹配到以下值之一，匹配则表示成功，否则会任务失败| | propertySelector | String | 必填，需要等待的具体的API返回值，jsonPath形式，具体返回体格式见【API帮助】|

示例 检查云服务器BCC的返回值是否符合预期值，调用检查云服务器BCC状态的API入参和出参如下所示：

请求结构

```
 POST /v{version}/instance/listByInstanceId HTTP/1.1
Host: bcc.bj.baidubce.com
Authorization: authorization string
{
    "instanceIds": ["i-y5nWLU4r"]
}
```

返回参数

```
 {
    "instances": [
        {
            "id": "i-y5nWLU4r",
            "status": "Running"
        }
    ]
}
```

检查API输入示例

```
 {
    "name": "checkForBcc",
    "operator": "BCE::CheckFor",
    "properties": {
        "parameters": {
            "instanceIds": [
                "i-y5nWLU4r"
            ]
        },
        "desiredValues": [
            "Running"
        ],
        "propertySelector": "$.instances[*].status"
    }
}
```

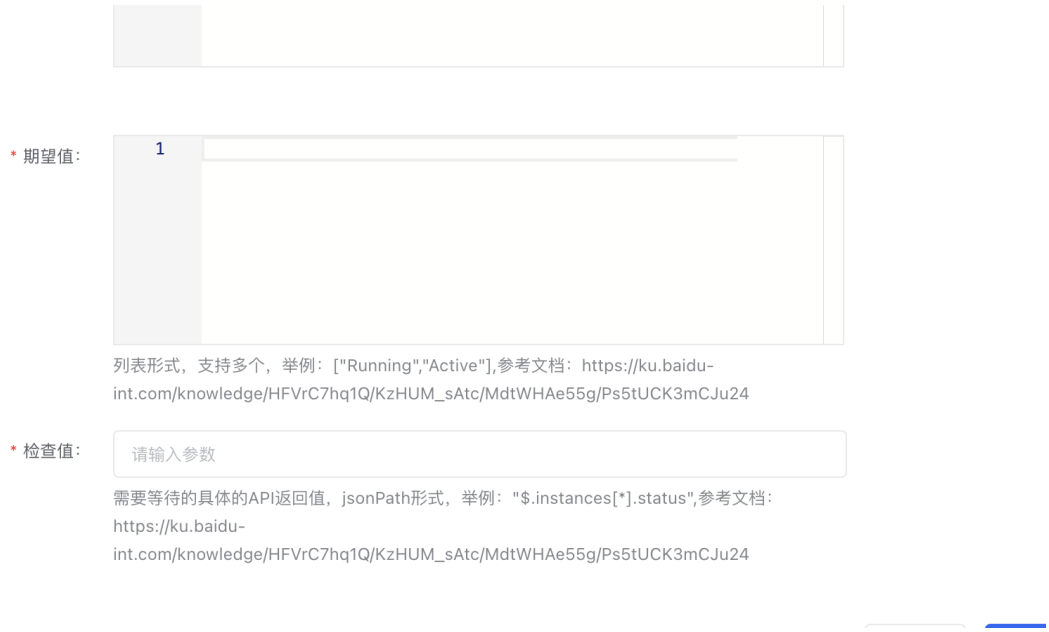
API帮助 检查API调用的接口：

BCC：https://cloud.baidu.com/doc/BCC/s/0kstrzyz9

EIP：https://cloud.baidu.com/doc/EIP/s/Pjwvz30qy

ENI：https://cloud.baidu.com/doc/VPC/s/0kknfjt6o

CDS：https://cloud.baidu.com/doc/BCC/s/Sjwvyo5be **Console使用说明 期望值及检查值输入说明** 输入界面：



举例 需要检查BCC状态是否为Runing状态.

填写方式：

通过BCC的API文档可以看到返回的数据结构如下所示

```
{
  "instances": [
    {
      "id": "i-y5nWLU4r",
      "status": "Running"
    }
  ]
}
```

需要检查的字段为status，期望的返回值为Runing。

则填写方式为：

期望值：["Running"]

检查值：$.instances[*].status

* 期望值：

```
1 ["Running"]
```

列表形式，支持多个，举例：["Running","Active"],参考文档：https://ku.baidu-int.com/knowledge/HFVrC7hq1Q/KzHUM_sAtc/MdtWHAe55g/Ps5tUCK3mCJu24

* 检查值：

$.instances[*].status

需要等待的具体的API返回值，jsonPath形式，举例："$.instances[*].status",参考文档：https://ku.baidu-int.com/knowledge/HFVrC7hq1Q/KzHUM_sAtc/MdtWHAe55g/Ps5tUCK3mCJu24

消息推送

用途 通过回调地址进行消息推送。

参数

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

名称	类型	描述
callBackUrl	String	必填，回调地址
content	String	必填，推送文案内容
type	String	必填，推送文案类型，当前仅支持文本类型

示例

```
{
  "name": "notify",
  "operator": "BCE::Notify",
  "properties": {
     "callBackUrl": "http://apiin.im.baidu.com/api/msg/groupmsgsend?access_token=d41f256ca8930904aa9a48c3a4d4cb615",
     "content":"push message test",
     "type":"TEXT"
  }
}
```

BCC相关

创建BCC实例

用途 用于创建一台云服务器BCC实例

请求参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | imageId | String | 镜像ID，必填 | | spec | String | 套餐规格，必填 | | paymentTiming | String | 付费方式，必填，Prepaid：预付费，Postpaid：后付费 | | reservationLength | int | 购买时长，预付费情况下必填，单位：月 | | rootDiskSizeInGb | int |系统盘大小，选填，默认值：40，单位：GB | | name | String | 虚机名称，选填 |

输出参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | instances | BccInstance | BCC实例，包含实例短ID信息 |

请求参数示例

```
{
  "name": "CreateBccInstance",
  "operator": "BCE::BCC::CreateInstanceBySpec",
  "properties": {
     "imageId": "m-8S2nu8s3",
     "spec": "bcc.g5.c2m8",
     "paymentTiming": "Prepaid",
     "reservationLength": 1,
     "rootDiskSizeInGb": 40,
     "name": "test_hostname"
  }
}
```

输出参数示例

```
{
  "instances": {
     "instanceId": "i-pqJV5Ptu"
  }
}
```

重启BCC实例

用途 用于批量重启云服务器BCC实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | instances | List <BccInstance> / BccInstance | BCC实例，必填，支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "restartInstance",
  "operator": "BCE::BCC::RestartInstance",
  "properties": {
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC实例短ID
        },
        {
           "instanceId":"i-pqJV5Ptb"
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "restartInstance",
  "operator": "BCE::BCC::RestartInstance",
  "properties": {
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

停止BCC实例

用途 用于批量关机云服务器BCC实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | instances | List<BccInstance>/BccInstance | BCC实例，必填，支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "stopBccInstance",
  "operator": "BCE::BCC::StopInstance",
  "properties": {
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        },
        {
           "instanceId":"i-pqJV5Ptb"
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "stopBccInstance",
  "operator": "BCE::BCC::StopInstance",
  "properties": {
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

启动BCC实例

用途 用于批量开机云服务器BCC实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | instances | List<BccInstance>/BccInstance | 实例ID，必填，支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "startBccInstance",
  "operator": "BCE::BCC::StartInstance",
  "properties": {
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        },
        {
           "instanceId":"i-11111111"
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "startBccInstance",
  "operator": "BCE::BCC::StartInstance",
  "properties": {
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

删除BCC实例

用途 用于批量释放云服务器BCC实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | instances | List<BccInstance>/BccInstance | 实例ID，必填，支持多选 | | relatedReleaseFlag | boolean | 是否关联释放当前时刻实例挂载的EIP和数据盘，必填，默认值：false | | deleteCdsSnapshotFlag | boolean | 是否释放云磁盘快照，必填，默认值：false | | bccRecycleFlag | boolean | 实例释放时是否进入回收站，必填，默认值：true | | deleteRelatedEnisFlag | boolean | 是否删除关联的ENI，必填，默认值：false |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "deleteBccInstance",
  "operator": "BCE::BCC::DeleteInstanceBySpec",
  "properties": {
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu"
        },
        {
           "instanceId":"i-11111111"
        }
     ],
     "relatedReleaseFlag": false,
     "deleteCdsSnapshotFlag": false,
     "bccRecycleFlag": true,
     "deleteRelatedEnisFlag": false
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "deleteBccInstance",
  "operator": "BCE::BCC::DeleteInstanceBySpec",
  "properties": {
     "instances": {
        "Ref":"globalInstanceList"
     },
     "relatedReleaseFlag": false,
     "deleteCdsSnapshotFlag": false,
     "bccRecycleFlag": true,
     "deleteRelatedEnisFlag": false
  }
}
```

虚机执行脚本

用途 用于在多台云服务器BCC实例内执行自定义脚本

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | __workerSelectors__ | List/Object | 实例ID，必填，支持多选 | | content | String | 脚本内容，必填 | | user | String | 执行用户，必填 | | workDir | String | 执行目录，必填 |

示例 __workerSelectors__参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "executeShell",
  "operator": "BCE::Agent::ExecuteShell",
  "properties": {
     "content": "ls",
     "user": "root",
     "workDir": "/",
     "__workerSelectors__": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        },
        {
           "instanceId":"i-rXV43yGW" // BCC短ID
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "executeShell",
  "operator": "BCE::Agent::ExecuteShell",
  "properties": {
     "content": "ls",
     "user": "root",
     "workDir": "/",
     "__workerSelectors__": {
        "Ref":"globalInstanceList"
     }
  }
}
```

虚机下载文件

虚机从BOS下载文件 用途 用于在多台云服务器BCC内部从BOS下载文件

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | __workerSelectors__ | List/Object | 实例ID，必填，支持多选 | | sourceType | String | 下载源，必填，取值：bos | | bosBucketName | String | bos桶名称，必填 | | bosFilePath | String | bos文件名称全路径，必填 | | fileDir | String | 保存文件路径，必填 | | filename | String | 保存文件名称，必填 |

示例 __workerSelectors__参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "fileDownload",
  "operator": "BCE::Agent::FileDownload",
  "properties": {
    "sourceType": "bos",
    "bosBucketName": "testBucketName",
    "bosFilePath": "hello/test.log",
    "fileDir": "/tmp",
    "filename": "new.log",
    "__workerSelectors__": [
      {
        "instanceId":"i-pqJV5Ptu" // BCC短ID
      },
      {
        "instanceId":"i-rXV43yGW" // BCC短ID
      }
    ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "fileDownload",
  "operator": "BCE::Agent::FileDownload",
  "properties": {
    "sourceType": "bos",
    "bosBucketName": "testBucketName",
    "bosFilePath": "hello/test.log",
    "fileDir": "/tmp",
    "filename": "new.log",
    "__workerSelectors__": {
      "Ref":"globalInstanceList"
    }
  }
}
```

虚机从HTTP下载文件

用途 用于在多台BCC内部从HTTP下载文件

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | __workerSelectors__ | List/Object | 实例ID，必填，支持多选 | | sourceType | String | 下载源，必填，取值：http | | source | String | url地址，必填 | | fileDir | String | 保存文件路径，必填 | | filename | String | 保存文件名称，必填 |

示例 __workerSelectors__参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "fileDownload",
  "operator": "BCE::Agent::FileDownload",
  "properties": {
     "sourceType": "http",
     "source": "http://localhost:8837/hello.log",
     "fileDir": "/tmp",
     "filename": "new.log",
     "__workerSelectors__": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        },
        {
           "instanceId":"i-rXV43yGW" // BCC短ID
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "fileDownload",
  "operator": "BCE::Agent::FileDownload",
  "properties": {
     "sourceType": "http",
     "source": "http://localhost:8837/hello.log",
     "fileDir": "/tmp",
     "filename": "new.log",
     "__workerSelectors__": {
        "Ref":"globalInstanceList"
     }
  }
}
```

筛选BCC实例

用途 用于通过指定条件筛选符合条件的云服务器BCC实例列表

注意：当前仅支持通过是否自动续费，云服务器BCC实例状态两个条件来进行筛选，每个条件仅支持单选，多个条件同时选择为逻辑“与”的关系。 **参数**

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

名称	类型	描述
autoRenew	String	选填，是否自动续费，true为筛选出已经开通自动续费的实例，false为筛选出未开通自动续费的实例。
status	String	选填，实例状态，Recycled-已回收 / Running-运行中 / Stopped-已关机 / Stopping-关机中 / Starting-启动中 / Expired-已到期。

示例

```
{
  "name": "filterInstance",
  "operator": "BCE::BCC::FilterInstance",
  "properties": {
    "autoRenew": "true"
    "status": "Expired"
  }
}
```

自动续费BCC实例

用途 用于开通云服务器BCC自动续费并创建自动续费规则。

- 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。
- 续费后会合并EIP/CDS

参数

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

名称	类型	描述
instances	List<BccInstance>/BccInstance	实例ID，必填，支持多选
autoRenewTime	int	必填，自动续费时长，单位：月，范围[1,12]

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下：

直接指定

```
{
  "name":"autoRenewInstance"
  "operator": "BCE::BCC::AutoRenewInstance",
  "properties": {
     "instances": [
        {
           "instanceId": "i-pqJV5Ptu"
        },
        {
           "instanceId": "i-11111111"
        }
     ],
     "autoRenewTime": 1
  }
}
```

引用全局参数

```
{
  "name":"autoRenewInstance"
  "operator": "BCE::BCC::AutoRenewInstance",
  "properties": {
     "instances": {
        "Ref": "globalInstanceList"
     }
     "autoRenewTime": 1
  }
}
```

虚机执行Http请求

用途 用于在多台云服务器BCC实例内执行Http请求

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | __workerSelectors__ | List/Object | 实例ID，必填，支持多选 | | method | String | 请求方法类型，必填 | | url | String | 请求地址，必填 | | headers | Map<String, Object> | 请求头，选填 | | body | Object | 请求体，选填 | | timeoutMill | Integer | 请求连接超时时间，选填，默认10000ms |

示例 __workerSelectors__参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "executeHttp",
  "operator": "BCE::Agent::ExecuteHttp",
  "properties": {
     "method": "GET",
     "url": "http://localhost:8080/api/getUser",
     "body": {
     },
     "headers": {
        "Content-Type": "application/json"
     },
     "timeoutMill": 5000,
     "__workerSelectors__": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        },
        {
           "instanceId":"i-rXV43yGW" // BCC短ID
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "executeHttp",
  "operator": "BCE::Agent::ExecuteHttp",
  "properties": {
     "method": "GET",
     "url": "http://localhost:8080/api/user/111",
     "body": {
     },
     "headers": {
        "Content-Type": "application/json"
     },
     "timeoutMill": 5000,
     "__workerSelectors__": {
        "Ref":"globalInstanceList"
     }
  }
}
```

CDS相关

创建CDS磁盘

用途 用于创建一个云磁盘CDS。 **请求参数**

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

名称	类型	描述
storageType	String	必填，CDS磁盘存储类型，包括SSD_Enhanced（增强型SSD），hp1 (高性能云磁盘)和hdd (通用型HDD) 3种类型，默认 hp1。
zoneName	String	选填，指定可用区信息，默认为空，由系统自动选择。命名规范为“国家-region-可用区序列，小写，例如北京可用区A为"cn-bj-a"。
cdsSizeInGB	int	选填，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。
paymentTiming	String	必填，付费方式，必填，Prepaid：预付费，Postpaid：后付费。
reservationLength	int	购买时长，预付费情况下必填，单位：月。
name	String	选填，磁盘名称。

输出参数 | 名称 | 类型 | 描述 | | --- | --- | --- | |volumeId| String | CDS磁盘ID |

请求参数示例

```
{
  "name":"createCds"
  "operator": "BCE::CDS::Create",
  "properties": {
        "storageType": "SSD_Enhanced",
        "zoneName": "cn-bj-a",
        "paymentTiming": "Prepaid",
        "reservationLength": 1,
        "cdsSizeInGB": 50,
        "name": "test_hostname"
  }
}
```

输出参数示例

```
{
  "volumeId":"v-fFxXyjvr"
}
```

删除CDS磁盘

用途 用于释放未挂载的云磁盘CDS。 **参数**

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

参数	类型	描述
volumeId	String	必填，待释放的磁盘ID。

示例

```
{
  "name": "deleteCDS",
  "operator": "BCE::CDS::Delete",
  "properties":
  {
    "volumeId": "v-ldisKo35"
  }
}
```

CDS挂载BCC

用途 用于将指定云磁盘CDS挂载在指定云服务器BCC实例下。

注意： 云磁盘CDS需要挂载在与其处于相同可用区下的指定云服务器BCC实例上。 **参数**

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

参数	类型	描述
volumeId	String	必填，磁盘ID。示例：v-3zmCcxbR
instances	BccInstance	必填，BCC实例，暂不支持多选

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "attachBcc",
  "operator": "BCE::CDS::AttachBcc",
  "properties": {
     "volumeId": "v-3zmCcxbR",
     "instances": [
        {
           "instanceId": "i-pqJV5Ptu"
        }
     ]
  }
}
```

引用全局参数

引用模板中的globalInstanceList全局参数

```
{
  "name": "attachBcc",
  "operator": "BCE::CDS::AttachBcc",
  "properties": {
     "volumeId": "v-3zmCcxbR",
     "instances": {
        "Ref": "globalInstanceList"
     }
  }
}
```

CDS解绑BCC

用途

用于将指定的云磁盘CDS从云服务器BCC实例中卸载。

> 注意：只有实例状态为 Running 或 Stopped 时，磁盘才可以执行此操作。
>
> 如果 volumeId 的磁盘不挂载在 instanceId 的实例上，该操作失败。

参数

名称	类型	描述
name	String	模板任务名称，必填
operator	String	模板任务类型，必填
properties	Map<String, Object>	任务执行所需参数，必填，详细内容见下表

properties参数

参数	类型	描述
volumeId	String	必填，待卸载的磁盘对应的ID。示例：v-3zmCcxbR
instances	BccInstance	所挂载的BCC实例，必填，暂不支持多选

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "detachBcc",
  "operator": "BCE::CDS::DetachBcc",
  "properties": {
     "volumeId": "v-3zmCcxbR",
     "instances": [
        {
           "instanceId": "i-pqJV5Ptu"
        }
     ]
  }
}
```

引用全局参数

引用模板中的globalInstanceList全局参数

```
{
  "name": "detachBcc",
  "operator": "BCE::CDS::DetachBcc",
  "properties": {
     "volumeId": "v-3zmCcxbR",
     "instances": {
        "Ref": "globalInstanceList"
     }
  }
}
```

EIP相关

创建EIP实例

用途 用于创建一个弹性公网IP EIP实例

请求参数 | 名称 | 类型 | 描述 | | -------------------- | -------------- | -- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | -------------------- | -------------- | -- | | name | String | EIP实例名称，必填 | | bandwidthInMbps | int | 公网带宽，必填，单位：Mbps | | paymentTiming | String | 付费方式，必填，Prepaid：预付费，Postpaid：后付费 | | billingMethod | String | 计费方式，后付费时必填，ByTraffic：按流量付费，ByBandwidth：按带宽付费 | | reservationLength | int | 购买时长，预付费时必填，单位：月 |

输出参数 | 名称 | 类型 | 描述 | | -------------------- | -------------- | -- | | eip | String | EIP实例的公网IP地址 |

请求参数示例

```
 // 后付费
{
   "name": "createEipInstance",
   "operator": "BCE::EIP::Create",
   "properties": {
      "name": "my_test_eip",
      "bandwidthInMbps": 1,
      "paymentTiming": "Postpaid",
      "billingMethod": "ByTraffic"
   }
}
// 预付费
{
   "name": "createEipInstance",
   "operator": "BCE::EIP::Create",
   "properties": {
      "name": "my_test_eip",
      "bandwidthInMbps": 1,
      "paymentTiming": "Prepaid",
      "reservationLength": 1
   }
}
```

输出参数示例

```
 {
   "eip": "100.88.0.19"
}
```

EIP绑定BCC

用途 用于绑定指定的弹性公网IP EIP实例到指定的云服务器BCC实例上

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eip | String | 绑定实例的公网IP地址，必填，示例：100.88.0.19 | | instances | BccInstance | BCC实例，必填，暂不支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
 {
   "name": "eipBindBcc",
   "operator": "BCE::EIP::Bind",
   "properties": {
      "eip": "100.88.0.19",
      "instances": [
         {
            "instanceId":"i-pqJV5Ptu" // BCC短ID
         }
      ]
   }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "eipBindBcc",
  "operator": "BCE::EIP::Bind",
  "properties": {
     "eip": "100.88.0.19",
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

EIP解绑BCC

用途 用于解绑指定的弹性公网IP EIP实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eip | String | 绑定实例的公网IP地址，必填，示例：100.88.0.19 |

示例

```
{
  "name": "eipUnbindBcc",
  "operator": "BCE::EIP::Unbind",
  "properties": {
     "eip": "100.88.0.19"
  }
}
```

删除EIP实例

用途 用于释放指定的弹性公网IP EIP实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eip | String | 绑定实例的公网IP地址，必填，示例：100.88.0.19 |

示例

```
{
  "name": "deleteEipInstance",
  "operator": "BCE::EIP::Delete",
  "properties": {
     "eip": "100.88.0.19"
  }
}
```

ENI相关

创建ENI实例

用途 用于创建一个弹性网卡ENI实例

请求参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数

名称	类型	描述
name	String	网卡名称，必填，大小写字母、数字、中文以及-_/.特殊字符，必须以字母或者中文开头，长度1-65
subnetId	String	所在子网，必填
securityGroupIds	List<String>	安全组，必填，支持多选，示例：["group1", "group2"]

输出参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eniId | String | ENI实例ID | | privateIpAddress | String | ENI内网IP地址 |

请求参数示例

```
{
  "name": "createEniInstance",
  "operator": "BCE::ENI::Create",
  "properties": {
     "name": "my_test_eni",
     "subnetId": "sbn-6jrjkmdx06d9",
     "securityGroupIds": ["9b713e60-cc90-4bbd-a741-4b8a638f819b", "beba4e9d-06f6-4c25-8ed4-d0d50ea90925"]
  }
}
```

输出参数示例

```
{
  "eniId": "eni-5ahcgh7yynsr",
  "privateIpAddress": "192.168.0.19"
}
```

删除ENI实例

用途 用于删除一个弹性网卡ENI实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数

名称	类型	描述
eniId	String	ENI实例ID，必填

示例

```
{
  "name": "deleteEniInstance",
  "operator": "BCE::ENI::Delete",
  "properties": {
     "eniId": "eni-5rp4ii9fjt0k"
  }
}
```

ENI绑定BCC

用途 用于绑定指定的弹性网卡ENI到指定的云服务器BCC上

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eniId | String | ENI实例ID，必填 | | instances | BccInstance | BCC实例，必填，暂不支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "eniAttachBcc",
  "operator": "BCE::ENI::AttachBcc",
  "properties": {
     "eniId": "eni-5rp4ii9fjt0k",
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "eniAttachBcc",
  "operator": "BCE::ENI::AttachBcc",
  "properties": {
     "eniId": "eni-5rp4ii9fjt0k",
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

ENI解绑BCC

用途 用于从指定的云服务器BCC实例上解绑指定的弹性网卡ENI实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | eniId | String | ENI实例ID，必填 | | instances | BccInstance | BCC实例，必填，暂不支持多选 |

示例 instances参数可以直接指定也可以引用模板中的全局参数，示例分别如下： **直接指定**

```
{
  "name": "eniDetachBcc",
  "operator": "BCE::ENI::DetachBcc",
  "properties": {
     "eniId": "eni-5rp4ii9fjt0k",
     "instances": [
        {
           "instanceId":"i-pqJV5Ptu" // BCC短ID
        }
     ]
  }
}
```

引用全局参数 引用模板中的globalInstanceList全局参数

```
{
  "name": "eniDetachBcc",
  "operator": "BCE::ENI::DetachBcc",
  "properties": {
     "eniId": "eni-5rp4ii9fjt0k",
     "instances": {
        "Ref":"globalInstanceList"
     }
  }
}
```

ENI绑定EIP

用途 用于绑定指定的弹性网卡ENI到指定的弹性公网IP EIP实例上。

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | privateIpAddress | String | ENI内网IP地址，必填 | | publicIpAddress | String | EIP公网IP地址，必填 | | eniId | String | ENI实例ID，必填 |

示例

```
{
  "name": "eniBindEip",
  "operator": "BCE::ENI::BindEip",
  "properties": {
     "privateIpAddress": "192.168.0.19",
     "publicIpAddress": "100.88.9.200",
     "eniId": "eni-5ahcgh7yynsr"
  }
}
```

ENI解绑EIP

用途 用于从指定的弹性公网IP EIP实例上解绑指定的弹性网卡ENI实例

参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | name | String | 模板任务名称，必填 | | operator | String | 模板任务类型，必填 | | properties | Map<String, Object> | 任务执行所需参数，必填，详细内容见下表 |

properties参数 | 名称 | 类型 | 描述 | | ---------- | ---------- | ---------- | | publicIpAddress | String | EIP公网IP地址，必填 | | eniId | String | ENI实例ID，必填 |

示例

```
{
  "name": "eniUnbindEip",
  "operator": "BCE::ENI::UnbindEip",
  "properties": {
     "publicIpAddress": "100.88.9.200",
     "eniId": "eni-5ahcgh7yynsr"
  }
}
```

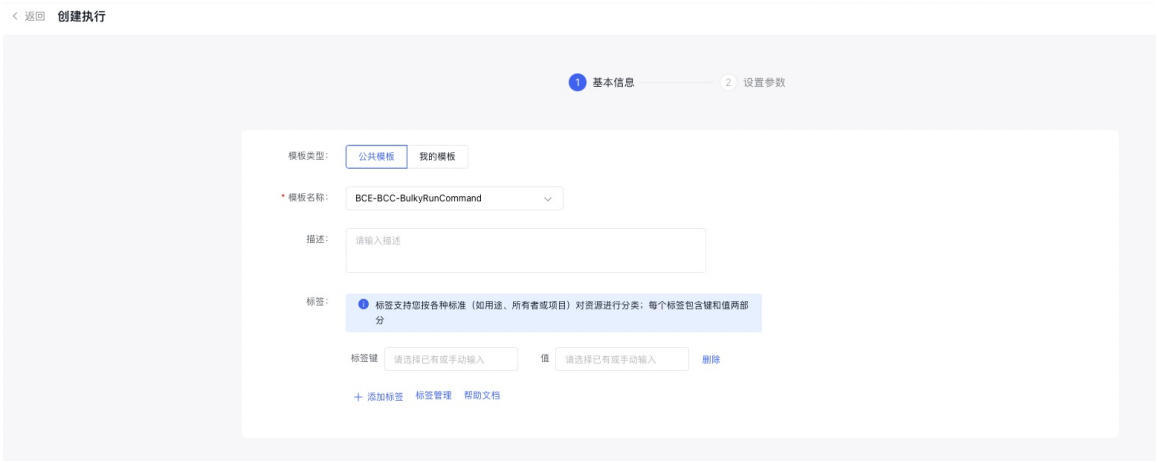
公共模板

运维编排服务的模板主要用于定义您编排的运维任务，模板分为公共模板和我的模板。公共模板沉淀了多种场景的模版供您选择，包括发送远程命令、上传下载文件、快速续费云服务器等，我的模板支持您根据自己的个性化业务场景自定义创建模版。

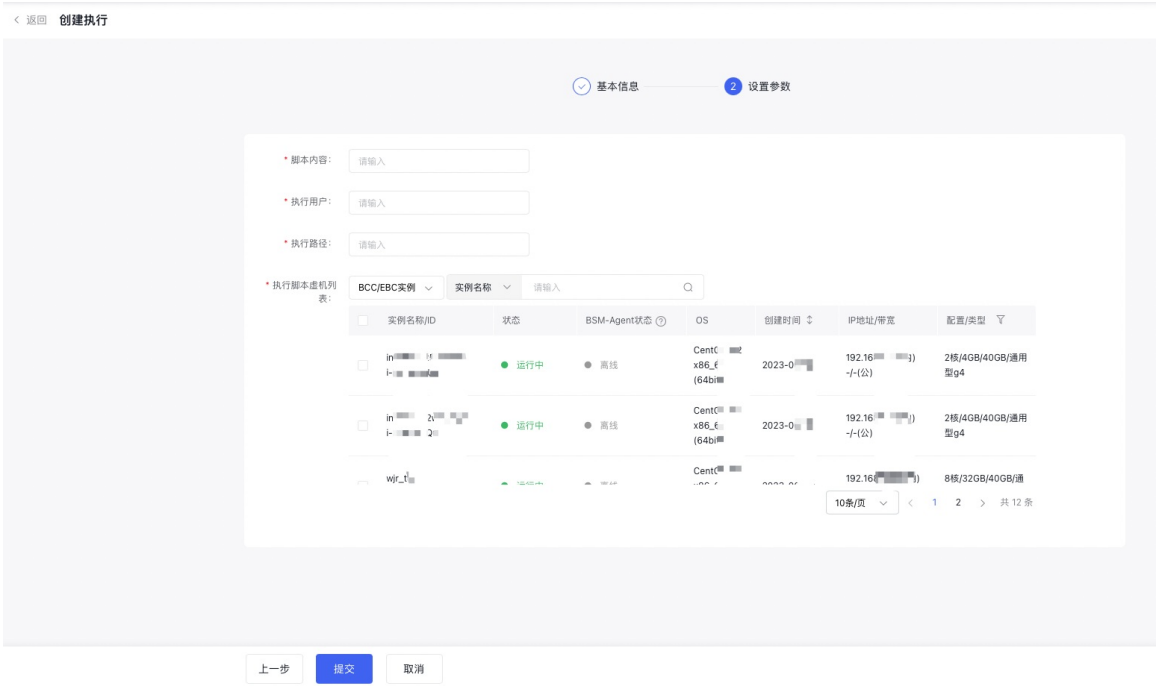
公共模板的使用步骤

1. 登录 运维编排服务控制台，选择“模板管理 > 公共模板”：

2. 根据实际的业务场景选择对应的公共模板进行“创建任务”，如：批量在BCC实例上运行命令：



3. 下一步设置参数中，进行如下配置：



- **参数设置** ：选用不同的模板需要设置的参数信息不同，本文中需要您输入要执行的脚本内容、执行时使用的用户名称、脚本执行的路径、需要执行的实例。

我的模板

概述

如果平台提供的公共模板无法满足您的操作需求，您可以通过我的模板创建更符合业务场景需求的模板。创建模板支持通过可视化流程配置、YAML格式配置和JSON格式配置。

注意事项

- 线性流程模板支持通过可视化流程、YAML和JSON三种方式进行配置，非线性流程的模板只能通过JSON和YAML格式进行配置。
- 在配置模版时只能选用运维编排服务提供的任务类型，更多任务类型和自定义任务功能持续支持中，您可以通过工单提交更多任务需求。

操作步骤

创建模板

1. 登录 运维编排服务控制台，选择**模板管理** > **我的模板** > **创建模板**。
2. 在**基本信息**填入**模板名称**，按需可选配置标签和描述信息。

> 说明： 模板名称支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母或中文开头，长度1-200

3. 在**模板配置**按您的使用习惯选择**流程配置**、**YAML**、**JSON**中的一种方式配置

4. 完成模板配置中的参数信息，点击保存模板即可完成创建。

通过流程配置创建模板

1. 在**创建模板**时，在**模板配置**选择**流程配置**，

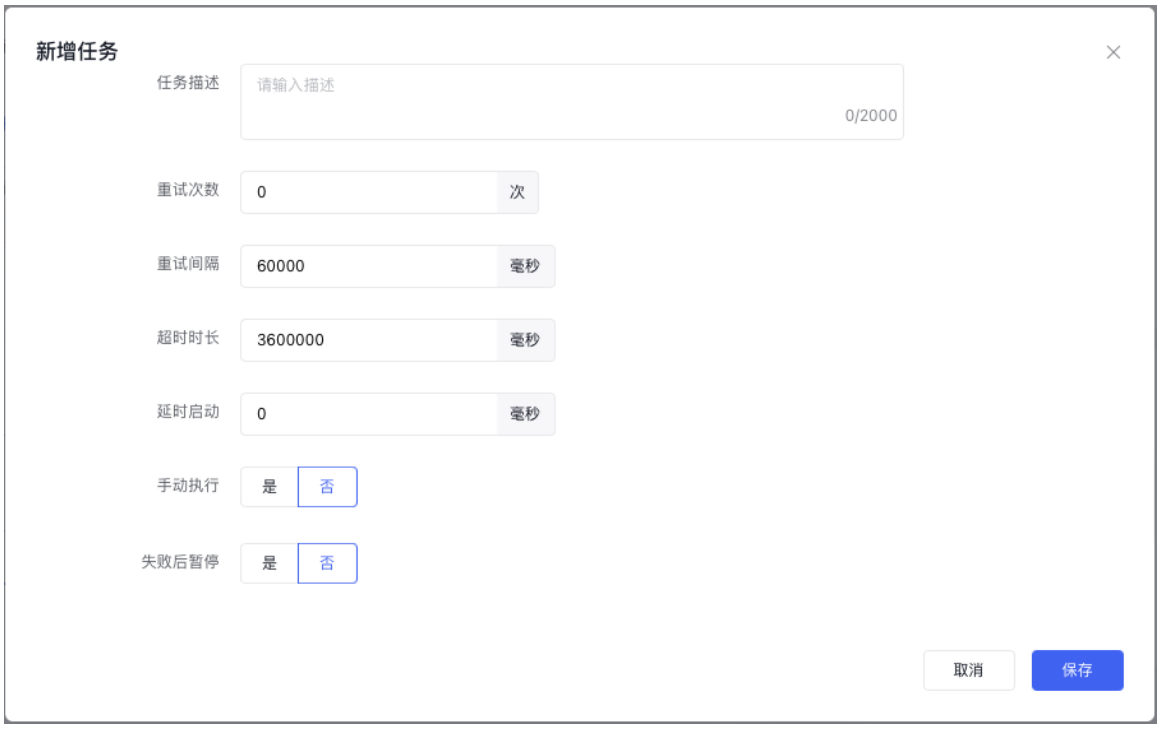
2. 在**全局参数**点击**添加参数**，按需添加步骤间共享的参数，此项非必填。

> 说明： 在任务步骤间共享的参数可以在全局参数中定义，步骤参数中可通过${var_name}的方式引用。

3. 在**任务步骤**点击**添加任务**，选择**任务类型**我们提供了针对4类资源的20+项常用任务供您选用。

4. 设置任务执行参数，参数如下：

   - 重试次数：任务失败后重试的次数，默认失败后不重试。
   - 重试间隔：重试的间隔时间，默认为60000毫秒，即60秒。
   - 超时时长：任务执行超过设定的时长后仍旧没有返回结果，会判定这一任务超时和失败。
   - 延时启动：在执行任务前，延迟的时间。
   - 手动执行：通过手动方式执行。
   - 失败后暂停：开启后，如果当前任务执行失败，后续其他任务不再继续执行。



5. 设置任务**参数**，不同任务类型需要配置的参数不同，您可以按照参数要求进行配置。

6. 完成上述配置后，点击保存即可完成一个任务步骤的配置，重复以上步骤，完成线性任务流程模板的配置。

7. 点击**校验模板**，平台会自动检查模板的必要信息是否齐全，并通过界面提示您修改。

8. 通过校验后点击**保存模板**，即可完成创建。

通过YAML创建模板

1. 在**创建模板**时，在**模板配置**选择**YAML**，

2. 在代码输入框填入YAML格式的模板内容。

> 说明： 为便于使用，输入模板过程中，平台会针对代码内容给予提示。您也可以直接复制平台外的模板内容直接粘贴到代码输入框后再进行调整。

3. 点击**校验模板**，平台会自动检查模板必要信息是否齐全，并通过界面提示您修改。

4. 点击**保存模板**，即可完成创建。

通过JSON创建模板

1. 在**创建模板**时，在**模板配置**选择**JSON**，

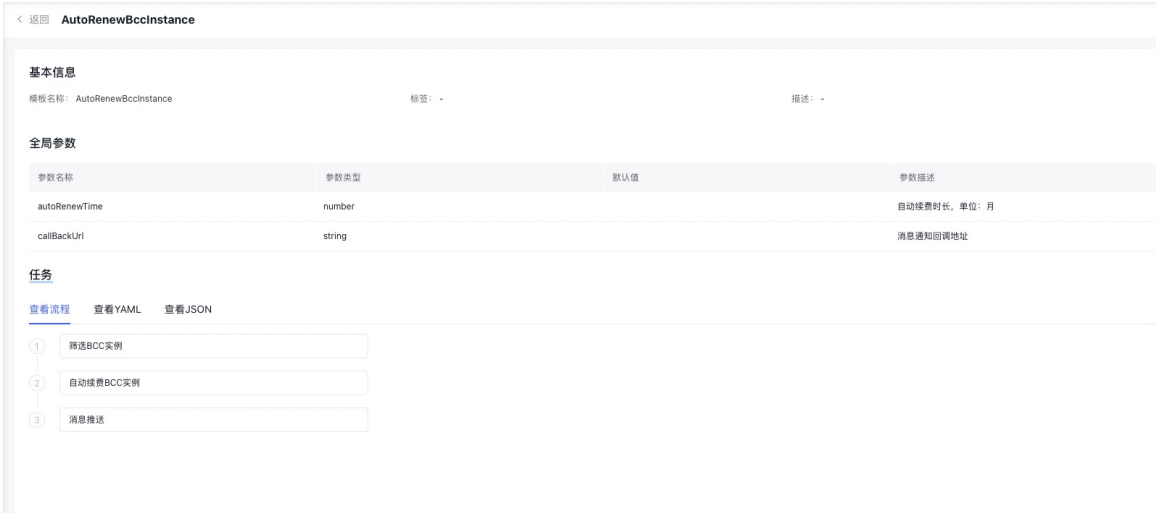
2. 在代码输入框填入JSON格式的模板内容。

> 说明： 为便于使用，输入模板过程中，平台会针对代码内容给予提示。您也可以直接复制平台外的模板内容直接粘贴到代码输入框中再进行调整。

3. 点击**校验模板**，平台会自动检查模板必要信息是否齐全，并通过界面提示您修改。

4. 点击**保存模板**，即可完成创建。

模板详情 在我的模板列表中点击**模板名称**即可进入模版详情，在此可以查看模版的详细参数和内容信息。



修改模板

1. 在我的模板列表中点击操作列中的**修改**。
2. 按需修改模板中除了模板名称之外的参数信息。

> 说明： 线性流程模板支持通过可视化流程、YAML和JSON三种方式进行编辑，非线性流程的模板需要通过JSON和YAML格式进行编辑。
>
> 如果把一个线性流程使用YAML方式修改为非线性流程，则在切换到流程配置视图时，将无法查看到这一非线性的可视化流程。

创建执行 模版管理中的创建执行功能与执行管理中创建执行功能一致，可参见对应章节的介绍。

执行管理

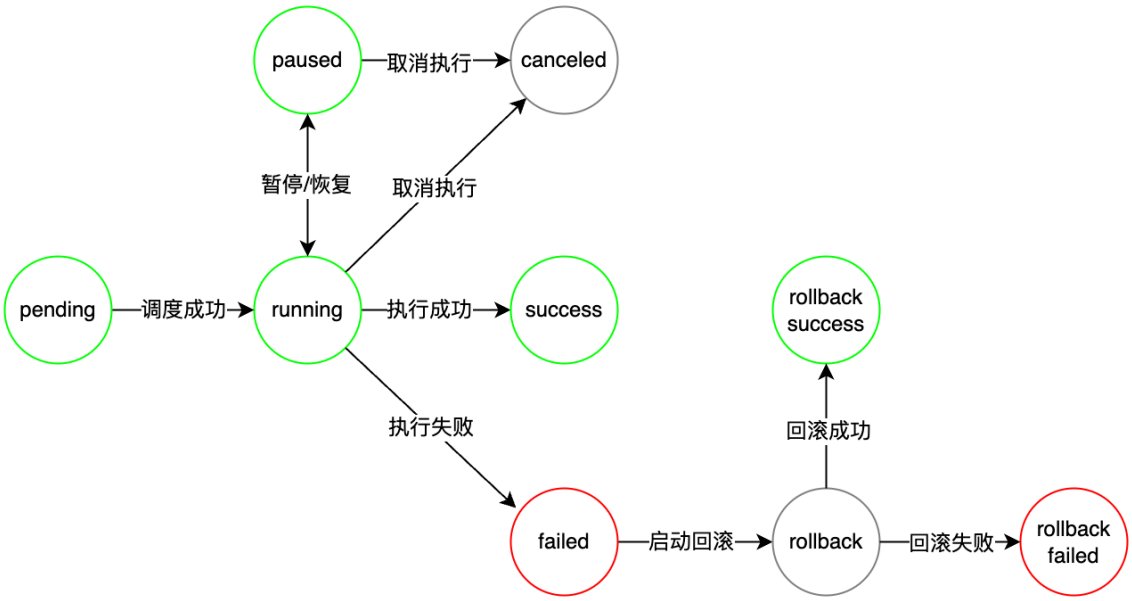
执行概述

执行概述

成功创建一个模板之后，就可以基于这个模板创建一次执行。您可以将一次执行看做是模板的一个运行实例。

执行状态

一个执行被创建后，可能进行多种状态转换，以下为状态转换图。



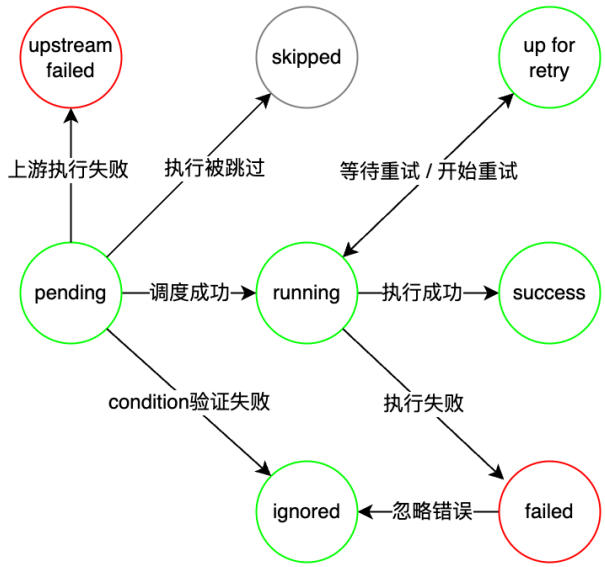
状态说明如下：

State	状态	说明
PENDING	等待中	表示执行正在等待被调度。例如，若模板开启了最大执行数限制，超出限制的执行，会处于等待状态
RUNNING	运行中	表示执行正在运行中
PAUSED	暂停中	表示执行当前已暂停运行。例如被用户手动暂停，或模板开启了失败后暂停等选项
SUCCESS	成功	表示执行已运行成功
FAILED	失败	表示执行已运行失败
CANCELED	已取消	表示执行已被取消。例如用户可以手动取消执行
ROLLBACK	回滚中	表示执行正在回滚中。若模板配置了回滚任务，用户可在执行失败时手动或自动启动回滚
ROLLBACK_SUCCESS	回滚成功	表示执行回滚成功
ROLLBACK_FAILED	回滚失败	表示执行回滚失败

任务状态

一个执行是由多个任务（Task）构成的有向无环图（directed acyclic graph，简称DAG），每个任务是DAG中的一个节点，这些节点的状态共同决定了执行的最终状态。

任务的状态转换图如下所示：



状态说明如下：

State	状态	说明
PENDING	等待中	表示任务正在等待被调度
RUNNING	运行中	表示任务正在运行中
SUCCESS	成功	表示任务已运行成功
FAILED	失败	表示任务已运行失败
UP_FOR_RETRY	等待重试中	表示任务正在等待被重试。 若任务设置了重试次数和重试间隔，且失败次数没有超过重试次数，那么在等待重试间隔期间，任务处于该状态
UPSTREAM_FAILED	上游任务失败	表示任务由于上游任务失败而也被置于失败状态。
SKIPPED	被跳过	表示任务被跳过而未运行。 1. 若启用了分支功能，未被执行的分支的任务会处于SKIPPED状态 2. 若设置了非默认的触发方式（TriggerRule），不符合触发条件的任务会被跳过而处于SKIPPED状态
IGNORED	已忽略	表示任务已被忽略。 1. 用户可以手动忽略失败的任务 2. 若任务配置了Condition检查条件，检查失败的任务会被oos自动忽略 注意，被忽略的任务的下游任务仍会被执行

参数传递

参数传递

一个执行在运行的过程中，会有三种参数的来源，如下所示：

参数来源	说明
全局输入参数	在创建模板时，你可以为模板添加全局输入参数 在创建模板执行时，你需要为模板的全局参数赋值，这些参数值构成了执行的全局输入参数
任务输入参数	不同的任务类型，有不同的输入参数，在创建模板时，你需要为每个任务的输入参数赋值。 在赋值时，你可以填写常量，也可以使用内置函数来引用全局参数或上游任务的输出参数。这些参数构成了任务的输入参数
任务输出参数	每个任务在运行后，会产生输出参数，输出参数可以作为后续任务的输入参数被引用

这些参数会对所有在运行中的任务可见。当发生参数同名的情况时，这些参数会互相覆盖，了解这些参数的优先级和覆盖规则，对正确运行模板就尤为重要。当然，你也可以利用这一覆盖规则，在执行过程中，动态的修改下游任务可见的参数值。

参数覆盖规则

参数覆盖规则如下：

> **当前任务输入参数** > **上游任务输出参数** > **全局输入参数**

这一参数覆盖规则，和一般编程语言中的变量覆盖规则是一致的，容易理解且符合直觉。

以下是一个示例：

```
{
  // 全局参数
  "properties": {
     "arg": "v1",
  },
  "operators": [
     {
        "name": "我的任务_1" ,

        // 任务输入参数
        "properties": {
           "arg": "v2",  // 与全局参数同名，会覆盖全局参数
        }
     },
     {
        "name": "我的任务_2" ,

        // 任务输入参数
        "properties": {
           // 使用函数引用参数'arg'
           // 若上游任务'我的任务_1'输出参数包含arg=v3，将覆盖全局参数，读取到arg=v3
           "local": {"Ref": "arg"},
        }
     },
  ]
}
```

1. 全局参数设置了`arg=v1`
2. `我的任务_1`执行时，任务的输入参数将覆盖全局参数，读取到`arg=v2`
3. 假设，`我的任务_1`执行成功后，输出`arg=v3`
4. `我的任务_2`执行时，通过函数引用参数`arg`，此时，上游任务的输出参数优先级高于全局参数，因此读取到`arg=v3`

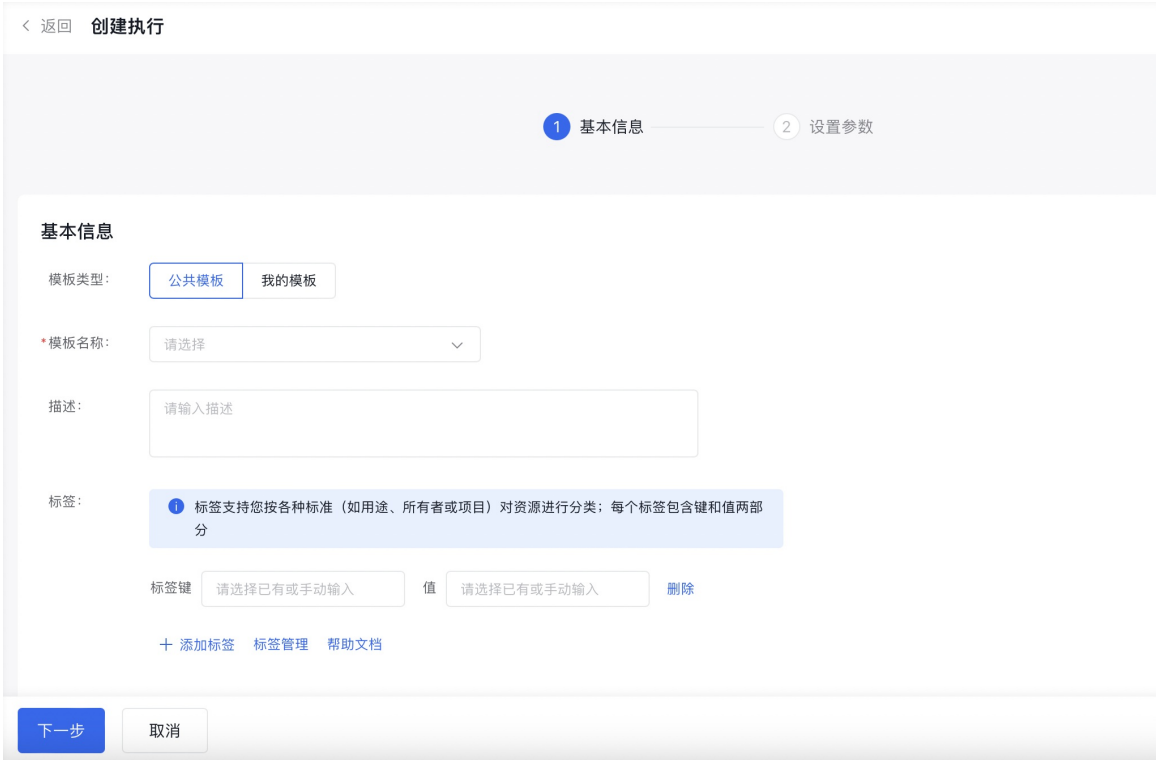
执行管理

概述

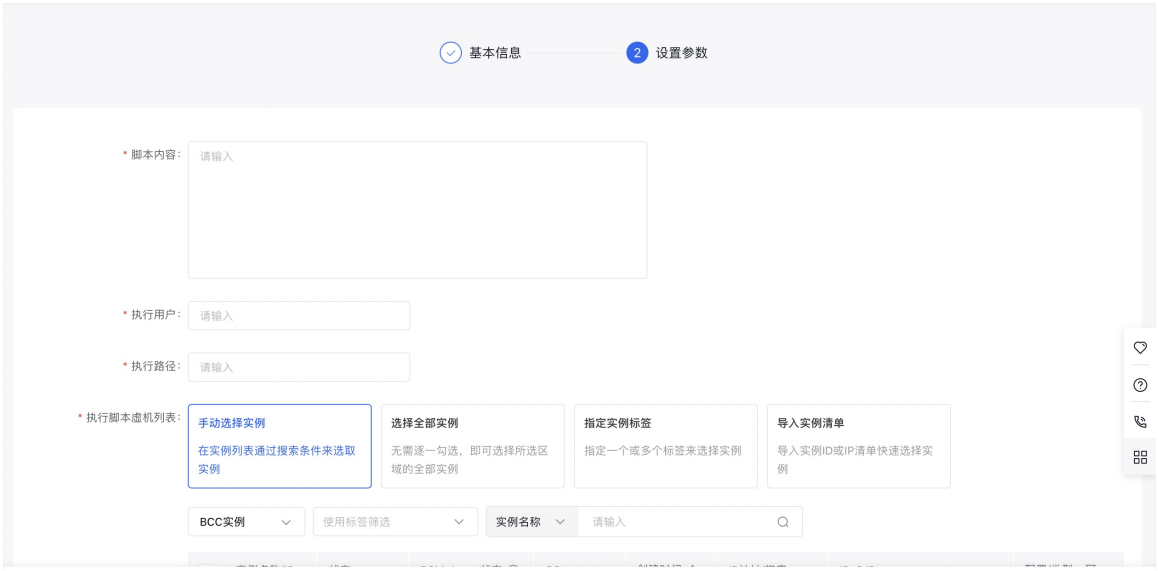
执行管理展示了模版执行后产生的执行记录，您可以通过时间和模版名称检索所关注的执行记录查看执行中所有任务运行的全生命周期过程及其结果。平台最长保存30天内的历史记录。您也可以在执行管理页面创建新的执行。

创建执行

1. 登录 运维编排OOS控制台，选择“执行管理”。
2. 点击“创建执行”，在弹出的页面完善参数信息。



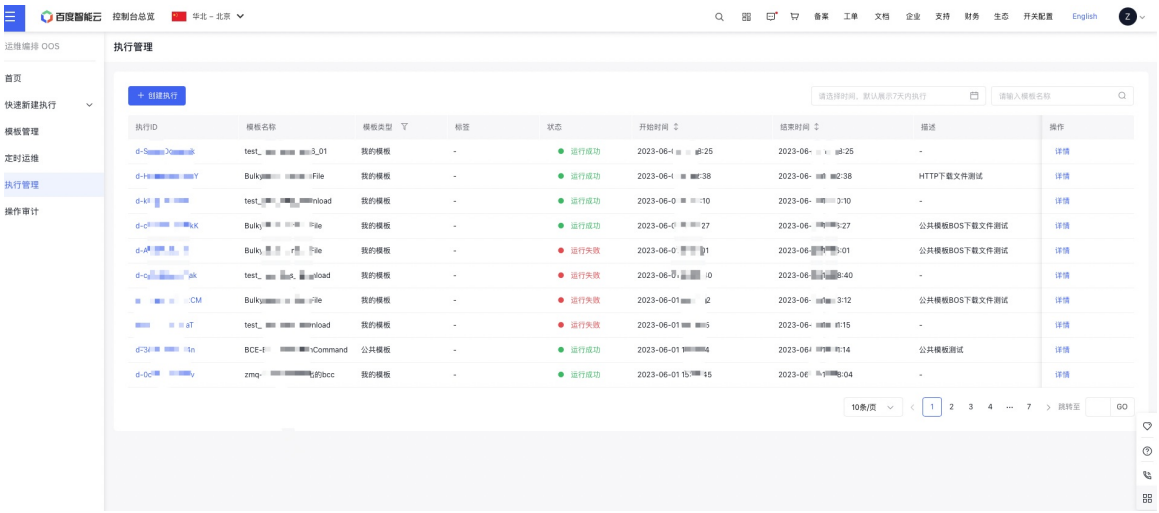
3. 选择要执行的模版，您可以通过切换公共模版/我的模版，查看对应的模版。
4. 完成基本信息中的其他参数。
5. 根据您选择的模板，在第二步完成模板声明的需要定义的参数信息。例如针对实例执行命令的模版，除了需要选定针对的具体实例和设定命令内容，还需要设置执行速率控制。



6. 点击**提交**即可完成创建。

执行详情

执行详情展示了执行的基本信息、执行结果、任务信息、任务的输入输出和日志等信息，点击执行管理列表操作列中的“详情”，即可打开执行详情页面。



基本信息 展示了执行ID、模板名称、执行状态、标签、描述、执行开始时间、执行结束时间，通过点击模版名称可以跳转到对应模版的详情页面。

执行结果 支持以图形化展示线性流程的执行状态和结果，可以通过点击各节点查看对应节点的任务执行信息。

- 任务信息：展示下图的信息。

- 输入：展示当前执行运维模板的输入参数，键&值
- 输出：展示当前执行运维模板的输出参数，键&值
- 日志：展示当前任务节点执行的日志

定时运维

概览

运维编排服务OOS的定时运维可以解决日常运维中常见的简单而繁琐的操作问题，通过设定执行周期，免除了人工执行的不便。

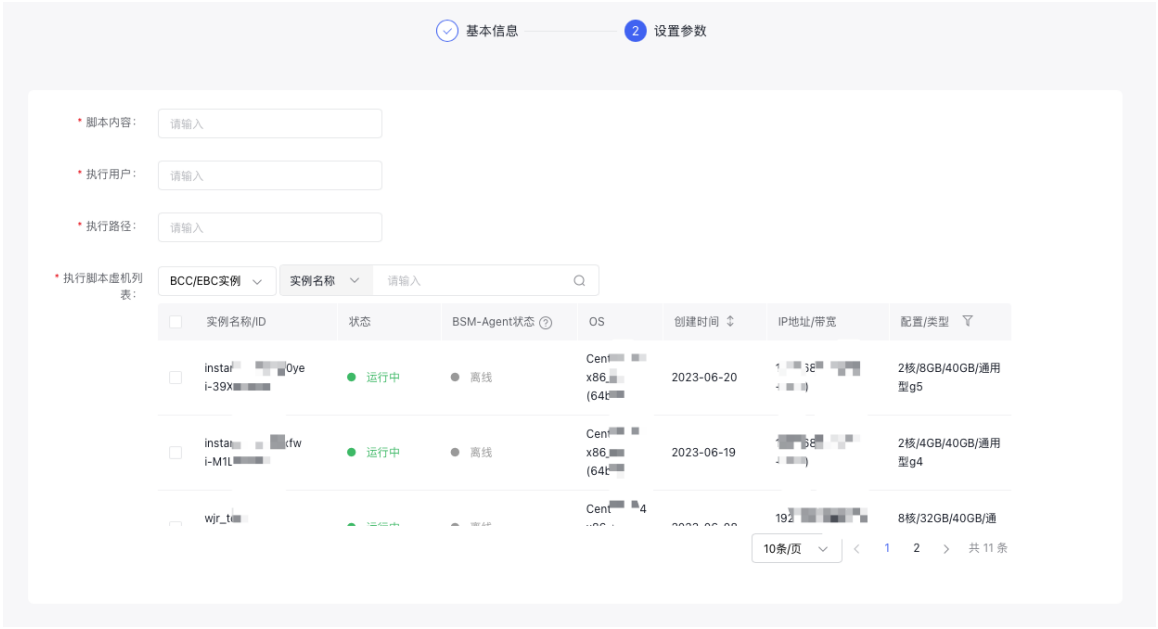
创建定时运维

1. 登录 运维编排OOS控制台
2. 点击**定时运维**菜单，点击列表页中的**创建定时运维**。
3. 设置重复周期的**Cron表达式**，之后您可以在下方执行时间预览中查看到接下来5次执行时间。

4. 按需选择要执行的模板类型及模版。
5. 填入**定时运维名称**和其他参数，其他参数可按需填写。

> 说明： 定时运维名称只能输入200个字符以内的中英文字符、数字和- _，且只能以中文汉字或英文字母开头。

6. 根据您选择的模板，完成模板声明的需要定义的参数信息。例如下图针对实例执行脚本的模版，需要选定针对的具体实例和设定脚本内容等信息。



> 说明: 一个执行最多可以针对100台实例。

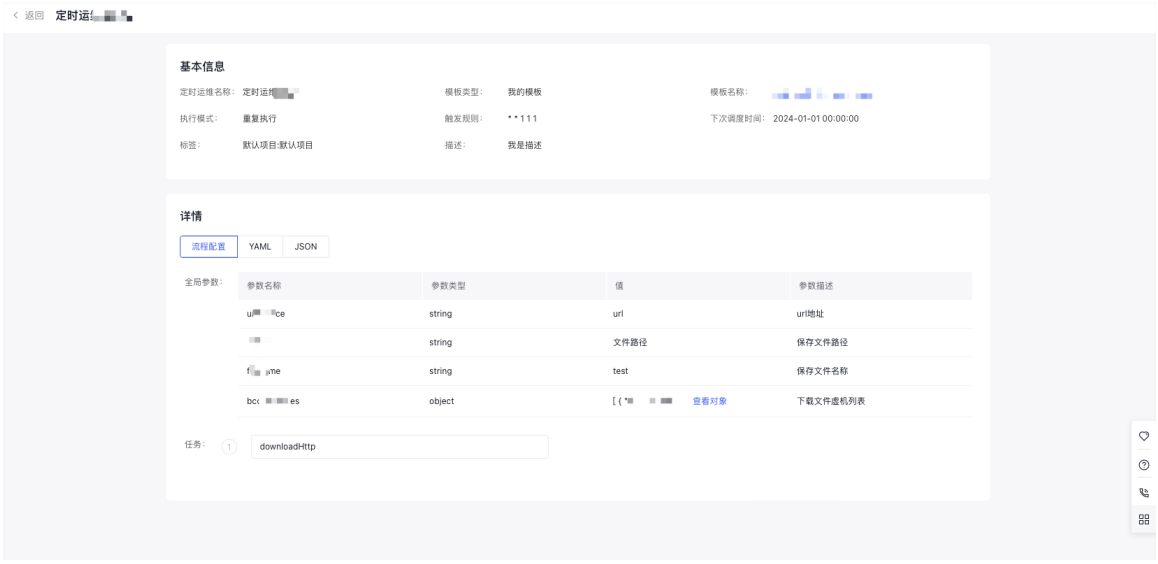
7. 点击**提交**即可完成创建，随后定时运维将在设定的时间执行，您可以通过点击操作列中的**执行记录**在执行管理中查看执行详情。

修改定时运维

您可以修改除了定时运维的名称之外的参数。

定时运维详情

您可以点击定时运维列表的名称打开定时运维的详情，定时运维详情展示了创建定时运维时设置的参数信息。



基本信息

展示创建定时运维时设置的基本参数，包含定时运维名称、模版类型、模板名称、执行模式、触发规则、下次调度时间、标签、描述等信息。

详情

展示模版配置的流程和任务信息，支持流程配置、YAML、JSON三种展示形式。

> 说明: 仅线性流程同时支持三种展示形式，非线性流程仅能以YAML、JSON形式展示。

报警事件运维

报警事件运维

概览

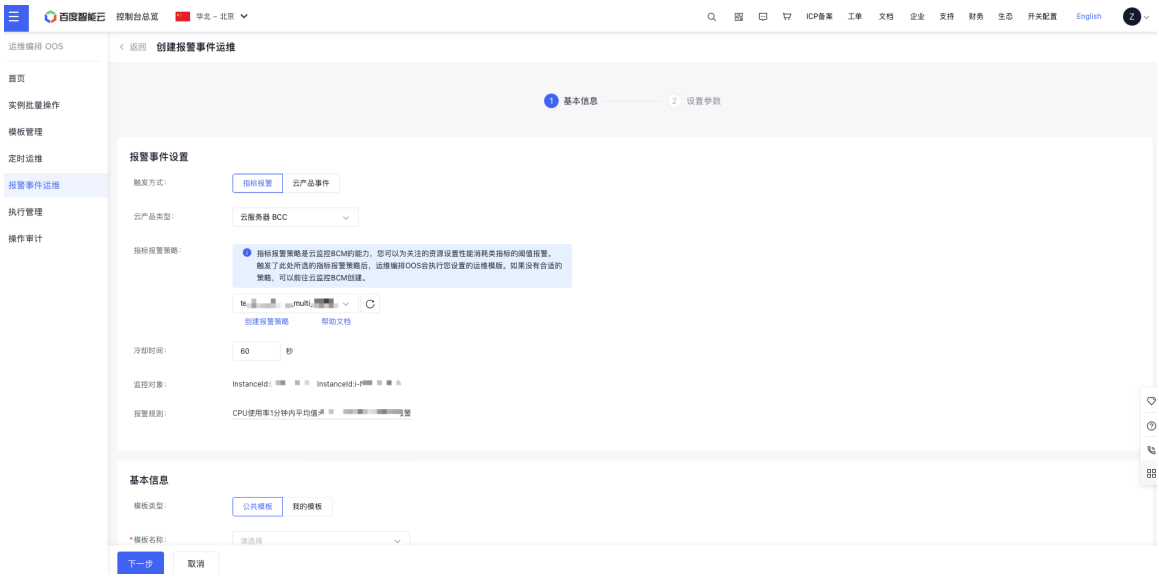
运维编排服务OOS会实时监测报警和事件信息，当云产品的监控指标达到预先设定的阈值触发报警，或者系统产生了云产品事件时，运维编排OOS会按照预先设置的策略执行对应的运维模板。报警事件运维与BCM云监控能力互通，您无需再次配置触发策略，运维编排OOS可以直接引用BCM监控报警和事件，打通了运维问题的解决链路，实现了基于预定策略的自动化运维，提升您的运维管理效率。

创建报警事件运维编排

1. 登录 运维编排OOS控制台

2. 点击**报警事件运维**菜单，点击列表页中的**创建报警事件运维**。

监控报警触发的运维编排 适用于需要根据监控报警触发运维动作的场景，如当EIP实例带宽使用率超过80%时自动调整带宽。



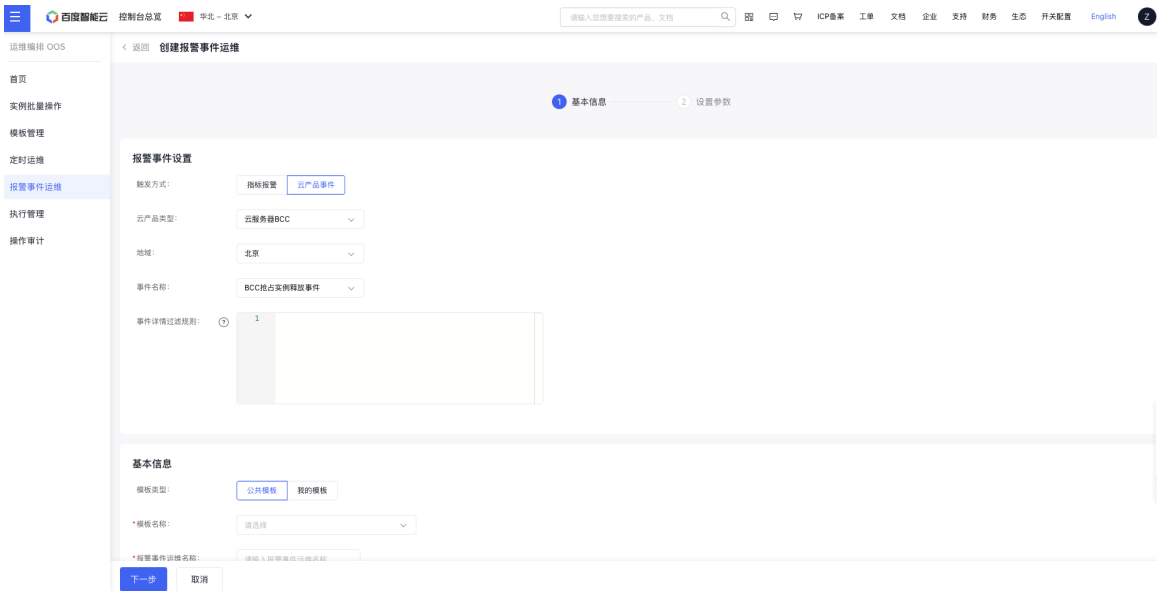
- 触发方式选择指标报警并按需选择云产品类型，例如您需要根据BCC的监控数据为触发条件，则此处选择云服务器BCC。
- 按需选择监控报警策略，报警策略已经根据您选择的云产品类型进行了筛选。

> 说明： 指标报警策略来自于云监控BCM，如果下拉列表中没有您需要的指标报警策略，请点击**创建报警策略**前往BCM控制台创建。

- 冷却时间用于防止短时间内重复产生的报警数据导致多次执行运维动作，在冷却时间内，针对同一报警策略产生的报警数据会被运维编排OOS忽略不触发运维动作，您可以按需调整冷却时间。
- 监控对象与报警规则是您在云监控BCM控制台创建监控报警策略时定义的，因此此处仅作信息展示，无需调整。如果此处展示的对象和报警规则与您需求不一致，可以通过**创建报警策略**重新创建新的策略。

> 说明： 指标报警策略支持采用“与”和“或”的多条规则触发，更多创建报警策略的操作指导可参考BCM文档

事件触发的运维编排 适用于需要根据云产品事件触发运维动作的场景，如在抢占式实例释放前执行备份数据操作。



- 触发方式选择云产品事件，随后选择具体的云产品类型，例如您需要根据释放抢占式BCC实例的事件作为触发条件，则此处云产品选择云服务器BCC。
- 选择事件发生的地域，仅对应地域的事件会触发运维动作，目前仅支持单选。
- 按需选择具体的事件名称，此处展示的事件已经根据您选择的云产品类型进行了筛选，例如您需要根据释放抢占式BCC实例的事件作为触发条件，则事件名称选择BCC抢占式实例释放事件。

> 说明： 云产品事件来自于云监控BCM，了解更详情的信息请参考云产品事件

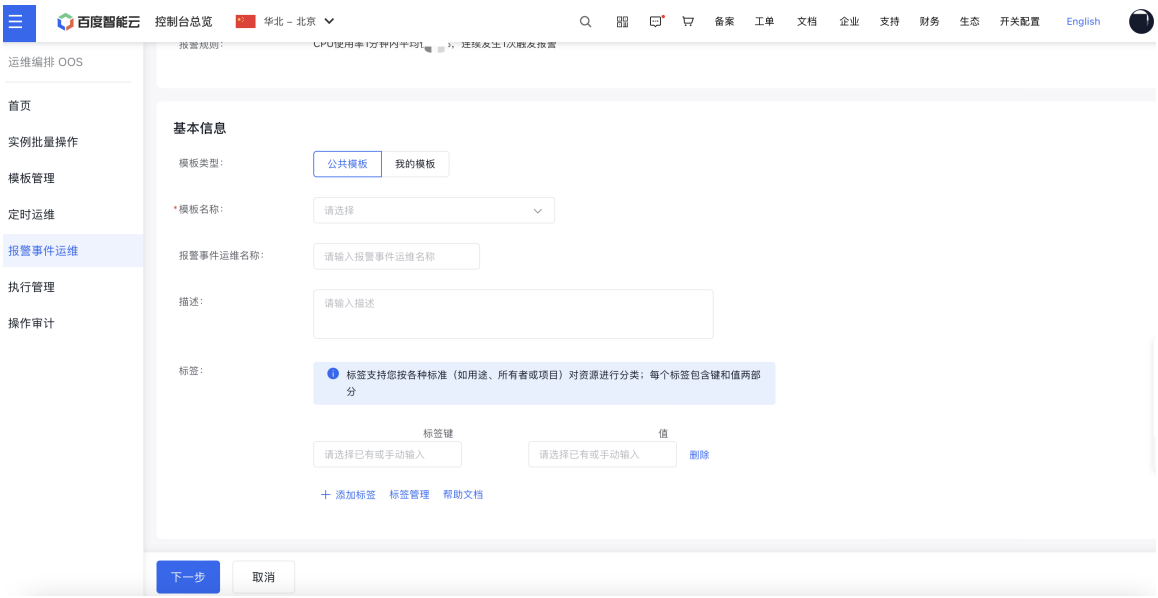
- 事件详情过滤规则用于在事件的详细信息中按照特定参数及其特定取值进行筛选，仅命中关键词和值的事件才会触发您设置的运维动作。此项非必填，如果您不需要过滤特定事件，保持此处为空即可。

> 说明： 您可以根据详情中的字段对云产品事件进一步过滤，过滤规则格式为[{"key":"xx","op":"xx","Value":"xx"}] 过滤规则中key即过滤的条件，op定义包含关系（一个值用=，多个值用in，不包含用not in） ， Value是要筛选出的一个或多个值，多个值用英文,间隔。 例如您仅需要过滤事件等级为NOTICE类型并且事件类型是GPU温度过高或BCC状态变化的通知，您可以在此处填入：

```
[
  {"key": "resource.eventLevel", "op": "=", "value": "NOTICE"},
  {"key": "resource.eventType", "op": "in", "value": ["GpuHighTemperature", "InstanceStateChange"]}
]
```

更多详细介绍请参考事件详情过滤规则填写说明。

3. 按需选择要执行的模板类型及模版。
4. 填入**报警事件运维名称**和其他参数，其他参数可按需填写。



> 说明： 报警事件运维名称只能输入200个字符以内的中英文字符、数字和- _，且只能以中文汉字或英文字母开头。

5. 根据您选择的模板，完成模板声明的需要定义的参数信息。例如针对实例执行脚本的模版，需要选定针对的具体实例和设定脚本内容等信息。

> 说明： 模版参数的取值支持直接输入/选择固定值和引用全局参数，此外还支持引用报警事件消息体中的字段。模版参数的设置方式简述如下：

- 选择固定值：通常情况下，该参数设置方式为下拉单选框，您可以直接选择其中的一项作为参数值。
- 输入固定值：通常情况下，该参数设置方式数值输入框或文本输入框，您可以直接输入具体的参数内容。
- 引用全局参数(输入函数)：您可以直接在文本输入框内输入{{"Ref":""}}，在“”中输入全局参数名的参数引用全局参数的值。
- - 引用报警事件消息体中的字段(输入函数)：您可以直接在文本输入框内输入{{.消息体名.参数名}}，来引提取消息字段，例如需要从报警详情（metric）中提取BCC实例id，参数值应该使用表达式 {{ .metric.shortInstanceId }} 部分常用消息体及其可引用的参数参见模板参数填写说明，查看更多云产品指标信息，查看更多云产品事件信息。

6. 点击**提交**即可完成创建，随后报警事件运维将在按照您设置的策略持续监控并在产生相应的报警和事件时触发执行，您可以通过点击操作列中的**执行记录**在执行管理中查看执行详情。

启用/禁用报警事件运维

报警事件运维后，默认为启用状态，如果你需要暂时停用报警事件运维，可以禁用对应的报警事件运维即可。

修改报警事件运维

您可以修改除了报警事件运维的名称之外的参数。

克隆报警事件运维

当您需要创建一个与现存运维编排类似的报警事件运维时，可以通过克隆功能快速创建。

报警事件运维详情

您可以点击报警事件运维列表中的名称打开详情，详情展示了创建报警事件运维时设置的参数信息。

基本信息 展示报警事件运维的基本参数，包含报警事件运维名称、模版类型、模板名称、状态、标签、描述等信息。

触发信息 展示报警事件运维的触发相关信息，包含触发方式、产品类型、触发的具体策略等

详情 展示模版配置的流程和任务信息，支持流程配置、YAML、JSON三种展示形式。

模板参数填写说明

模板参数除了可以填入固定值，也可以按照go template格式从消息体中提取字段:
针对指标报警触发的报警事件运维，不同类型云产品消息体示例如下：

```
 // 云服务器BCC、裸金属BBC
{
 "metric": {
  "resourceId": "6***6",            // 实例长ID
  "resourceType": "Instance",       // 实例类型
  "tagKey": "tagValue",             // 绑定标签
  "os": "linux",                    // 实例操作系统
  "shortInstanceId": "i-5Rugahea",  // 实例短ID
  "InstanceId": "i-5Rugahea",       // 实例ID
  "name": "CPUUsagePercent",        // 监控项名称
  "value": 13.1357357               // 监控项值
 }
}

// 弹性公网IP
{
   "metric": {
      "resourceId": "ip-486c6c40",      // 实例ID
      "resourceType": "Instance",       // 实例类型
      "EipValue": "180.76.149.149",     // EIP地址
      "name": "InPacketLossPercent",    // 监控项名称
      "value": 53.846,                  // 监控项值
      "tagKey": "tagValue"              // 绑定标签
   }
}

// NAT网关
{
   "metric": {
      "resourceId": "nat-fjtdv32tpttk",     // 实例ID
      "resourceType": "Instance",           // 实例类型
      "name": "InBandwith",                 // 监控项名称
      "value": 21011,                       // 监控项值
      "tagKey": "tagValue"                  // 绑定标签
   }
}

// 弹性网卡ENI
{
   "metric": {
      "resourceId": "eni-stb22yf41gu4",     // 实例ID
      "resourceType": "Instance",           // 实例类型
      "name": "VNicInBPS",                  // 监控项名称
      "value": 12342,                       // 监控项值
      "tagKey": "tagValue"                  // 绑定标签
   }
}

// 云磁盘CDS
{
   "metric": {
      "resourceId": "v-YQB0MbD1",           // 实例ID
      "resourceType": "Instance",           // 实例类型
      "name": "ReadBytes",                  // 监控项名称
      "value": 34532,                       // 监控项值
      "tagKey": "tagValue"                  // 绑定标签
   }
}
```

以从消息体（metric）中提取BCC实例id为例，参数值应该使用表达式 {{ .metric.shortInstanceId }}，针对且规则的报警策略，消息体中可能包含多份监控数据信息，以云服务器BCC为例，消息体格式如下：

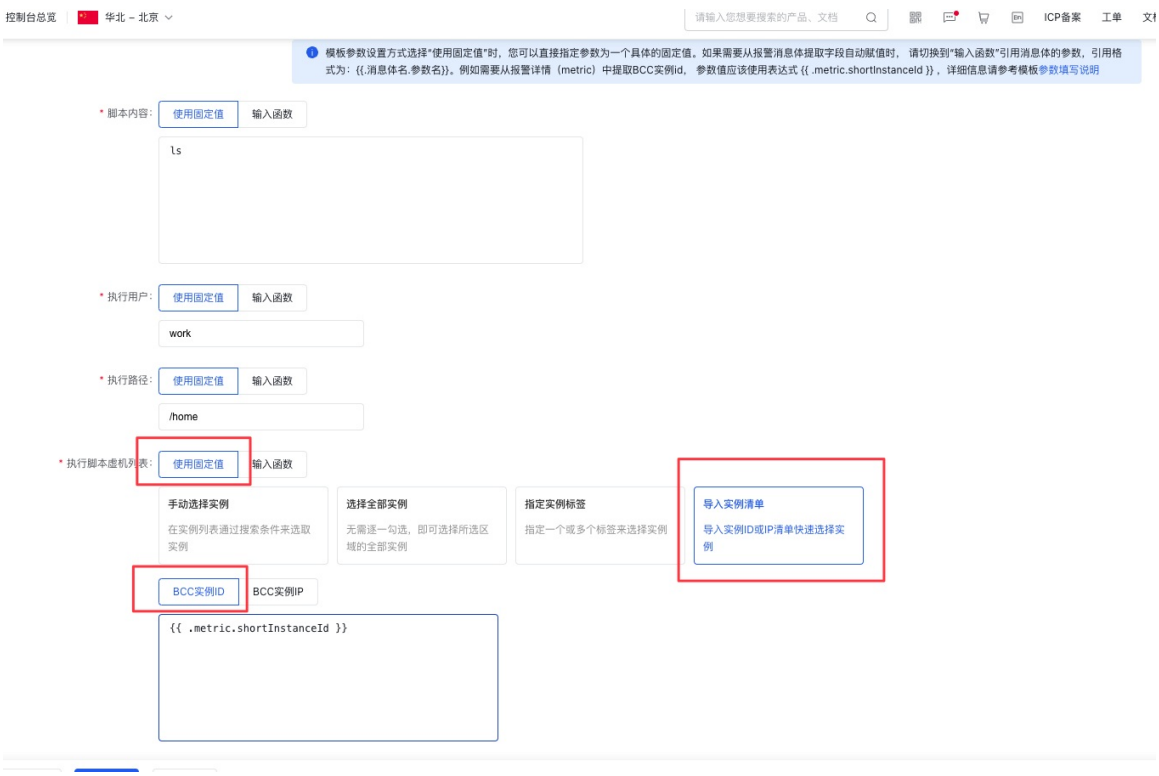
```
{
  "metrics": [
    {
      "shortInstanceId": "i-fYuDvBsy",    // 实例ID
      "resourceType": "Instance",         // 实例类型
      "os": "linux",                      // 虚机系统类型
      "name": "CPUUsagePercent",          // 监控项名称
      "value": 80.355,                    // 监控项值
      "tagKey": "tagValue"                // 虚机绑定标签
    },
    {
      "shortInstanceId": "i-fYuDvBsy",    // 实例ID
      "resourceType": "Instance",         // 实例类型
      "os": "linux",                      // 虚机系统类型
      "name": "CpuSystemPercent",         // 监控项名称
      "value": 80.355,                    // 监控项值
      "tagKey": "tagValue"                // 虚机绑定标签
    }
  ]
}
```

此时如果从消息体中提取第一份监控数据中的BCC实例id，应该使用表达式 {{ (index .metrics 0).shortInstanceId }}，如果从消息体中提取第二份监控数据中的监控项名称，应该使用表达式 {{ (index .metrics 1).name }}

特殊说明

若为模板参数是BCC/BBC实例类型，其引用参数填写位置**不能**写在“**输入函数**”下的输入框，需要填写在“**使用固定值-导入实例清单-BCC/BBC实例ID**”下面的输入框内，示例如下所示：

正确示例：



事件详情过滤规则填写说明

针对云产品事件触发的报警事件运维，运维编排服务OOS支持丰富且灵活的过滤规则配置，多个过滤规则之间是**且关系**，满足所有过滤规则筛选的事件才会触发运维动作，其格式如下：

```
[
  {"key": targetKey1, "op": op1, "value": targetValue1},
  {"key": targetKey2, "op": op2, "value": targetValue2}
]
```

targetKey可以从云产品事件结构体中获取，op目前支持的可选项及含义如下：

op取值	含义	示例	说明
=	等于	{"key": "resource.resourceId", "op": "=", "value": "i-oFtYX3xx"}	发送事件结构体中的resourceId等于i-oFtYX3xx则满足过滤规则
=	等于	{"key": "resource.region", "op": "!=", "value": "bj"}	发送事件结构体中的region不等于bj则满足过滤规则
in	被包含	{"key": "resource.eventType", "op": "in", "value": ["GpuHighTemperature", "InstanceStateChange"]}	发送事件结构体中的eventType等于GpuHighTemperature或InstanceStateChange则满足过滤规则
notin	不被包含	{"key": "resource.eventLevel", "op": "notin", "value": ["NOTICE", "WARN"]}	发送事件结构体中的eventLevel不等于NOTICE且不等于WARN则满足过滤规则
exists	存在该字段	{"key": "resource.resourceType", "op": "exists", "value": ""}	发送事件结构体中存在resourceType字段则满足过滤规则
notexists	不存在该字段	{"key": "resource.timestamp", "op": "notexists", "value": ""}	发送事件结构体中不存在timestamp字段则满足过滤规则
contains	字符串包含	{"key": "resource.eventId", "op": "contains", "value": "event___"}	发送事件结构体中的eventId字段包含event___则满足过滤规则

以云服务器BCC事件为例，如果想配置事件等级为NOTICE类型，并且事件类型是GPU温度过高或BCC状态变化通知的过滤规则，则配置示例如下：

```
[
  {"key": "resource.eventLevel", "op": "=", "value": "NOTICE"},
  {"key": "resource.eventType", "op": "in", "value": ["GpuHighTemperature", "InstanceStateChange"]}
]

// 云服务器BCC推送报警事件示例如下
{
    "eventId": "event-bcc-123",
    "eventLevel": "NOTICE",
    "timestamp": "2022-12-08T14:47:00Z",
    "resourceType": "INSTANCE",
    "resourceId": "i-6PcfS1xx",
    "longInstanceId": "90828af5-f212-43a0-96b8-d9bee26e0bcd",
    "region": "bj",
    "eventType": "InstanceStateChange",
    "eventAlias": "BCC状态变化通知",
    "content": "{\"info\":\"BCC实例状态变化\",\"advice\":\"您的BCC实例i-6PcfS1xx状态已变更为运行中\"}"
}
```

多用户访问控制

介绍

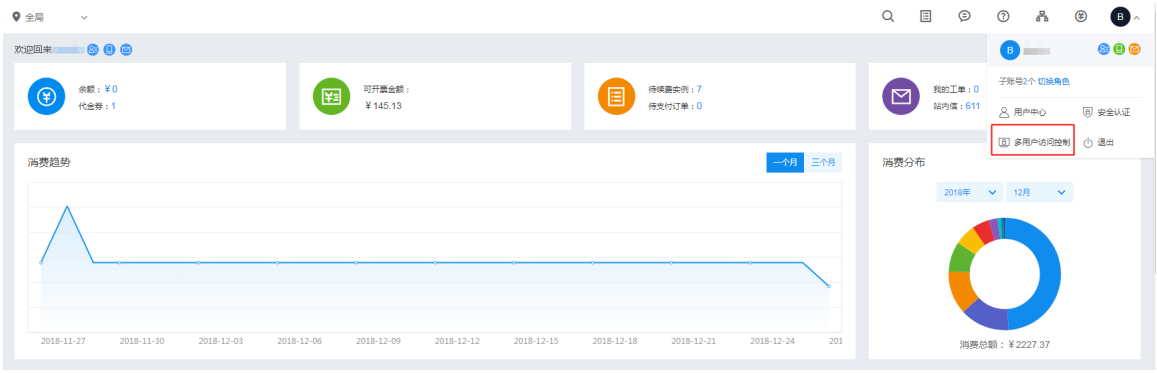
多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限，适用于企业内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，当您的企业存在多用户协同操作资源的场景时，推荐您使用多用户访问控制。

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

OOS支持系统策略，实现OOS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
OOSFullControlAccessPolicy	完全控制管理运维编排(OOS)的权限	创建模版、查询模板列表、查询模版详情、更新模板、删除模版、校验模版、查询任务列表、查询任务详情、查询任务参数、创建执行、查询执行列表、查询执行详情、更新执行、单步执行、创建定时运维、查询定时运维列表、查询定时运维详情、更新定时运维、删除定时运维、查询实例任务详情、更新实例任务、查询执行日志、查询操作审计、查询Agent状态、创建实例批量操作、查看实例批量操作详情
OOSReadAccessPolicy	只读访问运维编排(OOS)的权限	查询模板列表、查询模版详情、查询任务列表、查询任务详情、查询任务参数、查询执行列表、查询执行详情、查询定时运维列表、查询定时运维详情、查询实例任务详情、更新实例任务、查询执行日志、查询操作审计、查询Agent状态、查看实例批量操作详情

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。

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。

最佳实践

实例开通自动续费

概览

运维编排服务OOS可以解决日常运维中常见的简单而繁琐的操作问题，本实践通过自动识别待续费BCC实例并开通自动续费，实现降本增效目的。

需求场景描述

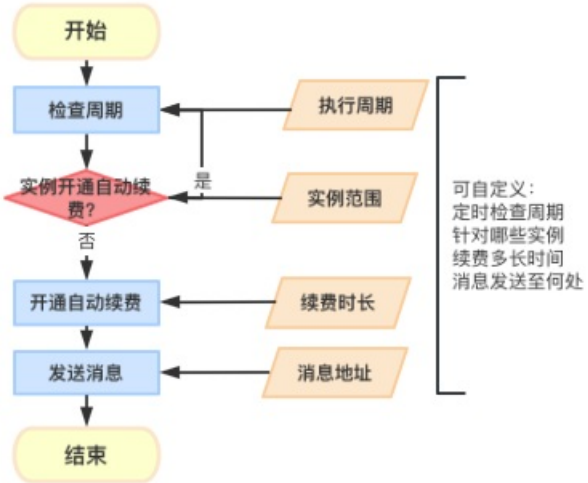
企业在预付费资源到期后需要继续使用该资源，由于资源到期的时间不一，就要求人工定期查看资源到期情况并续费，另一方面在有大量存量资源的场景下，手动操作不仅效率低耗费大量精力，而且容易出错。如果您也遇到类似的问题，可以跟随本文的方法，借助运维编排OOS能力，提升运维效率，降低人力成本。

前提条件

1. 开通了运维编排OOS服务。
2. 授予运维编排OOS服务对应云产品管理权限策略。
3. 实例已经安装了BSM-Agent，如实例的BSM-Agent离线，需要手动安装BSM-Agent，可参考云助手客户端安装章节。

方案概述

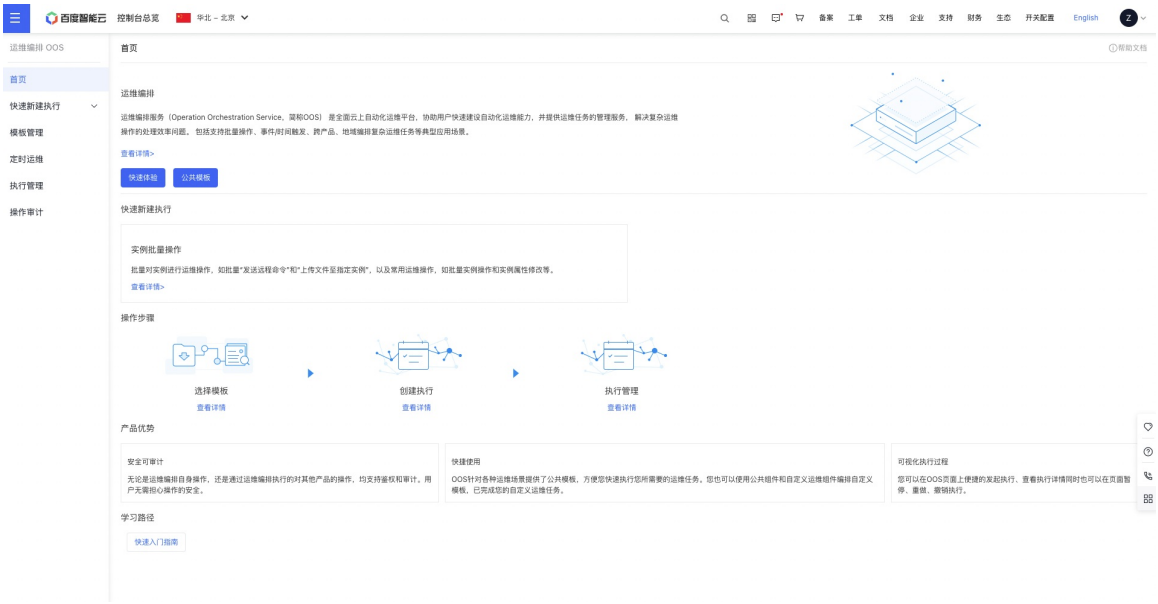
百度智能云运维编排OOS提供了自动续费模版，您可以使用这一模版创建一个定时运维，按需设置检查可开通自动续费的周期、设置续费时长，完成设置后运维编排OOS将按照您设置的周期检查您全部实例是否存在待开启自动续费的实例，并把筛选出来的实例开通自动续费，操作结果也会发送至您指定的地址。



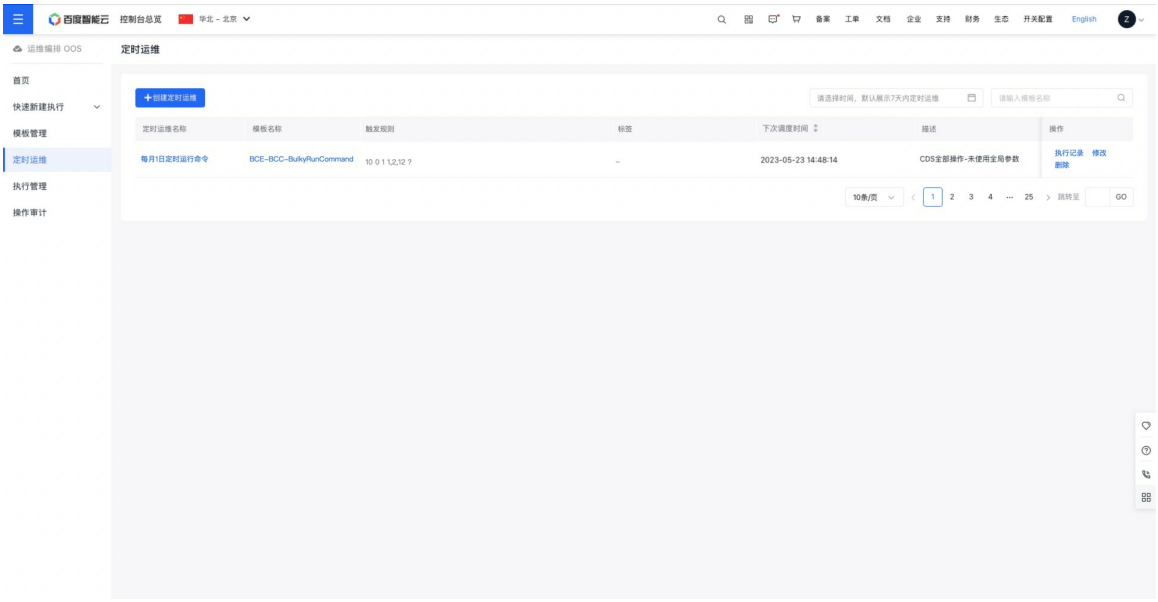
配置步骤

1. 登录百度智能云。

2. 通过产品服务点击“运维编排OOS”进入运维编排页面。



3. 点击**定时运维**菜单，点击列表页中的**创建定时运维**。

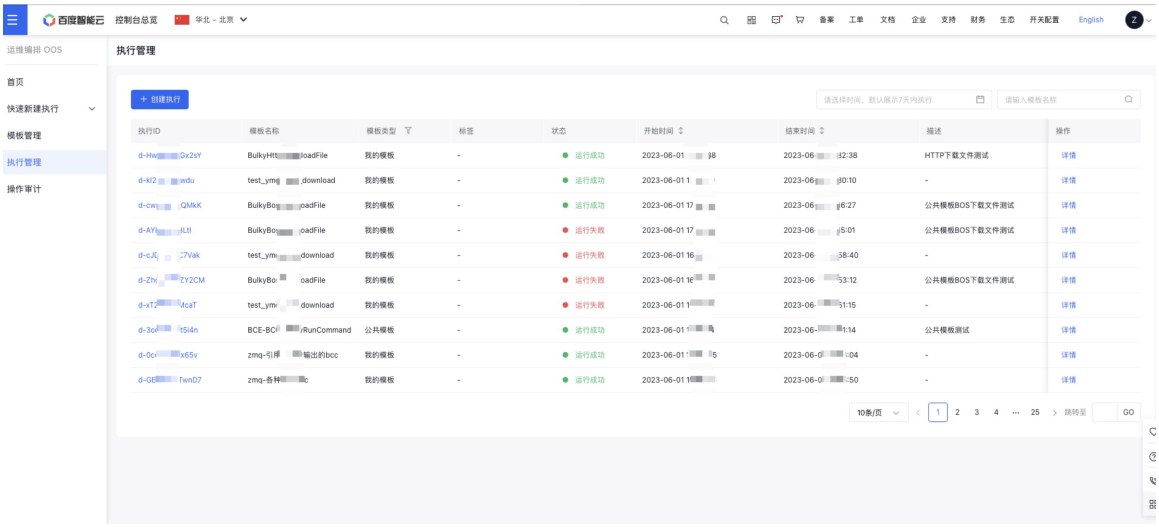


4. 设置重复周期的**Cron表达式**，之后您可以在下方执行时间预览中查看到接下来5次执行时间。

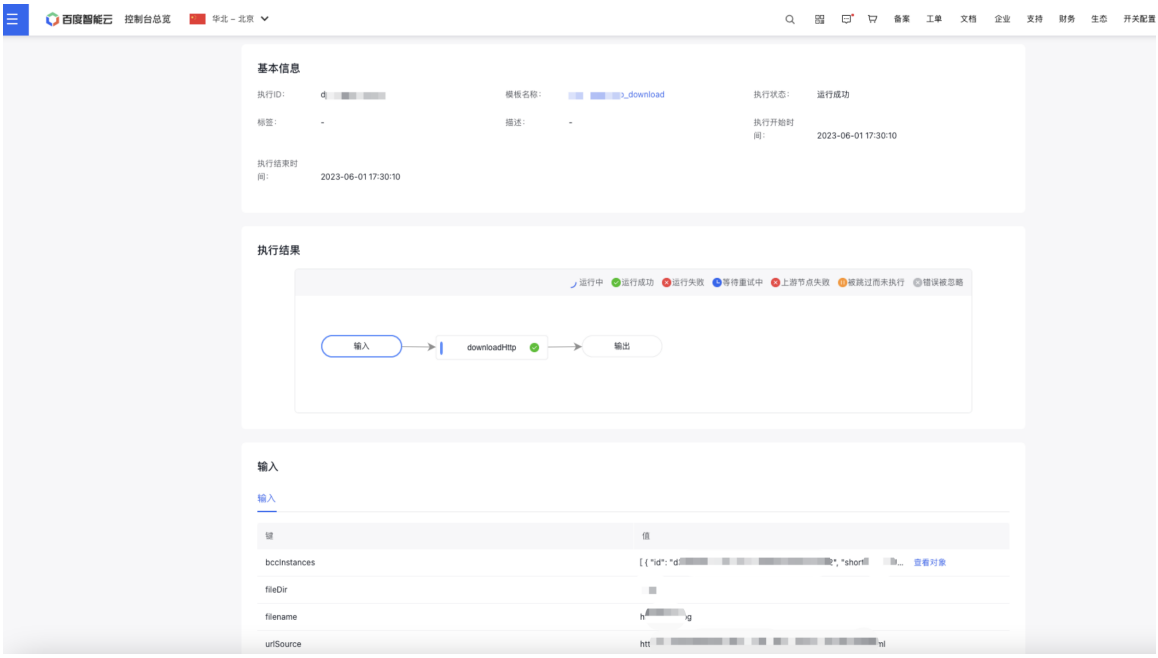
5. 模板类型选择**公共模板**，选择**AutoRenewBccInstance**模板。
6. 填入**定时运维名称**和其他参数，其他参数可按需填写。

> 说明： 执行名称只能输入200个字符以内的中英文字符、数字和- _，且只能以中文汉字或英文字母开头。

7. 单击下一步，选择自动续费时的续费时长，如6个月。
8. 按需设置消息通知回调地址，续费后相关信息会发至您设置的地址。
9. 点击提交，返回到定时运维列表即可查看到已经创建的定时运维。
10. 当定时运维执行后，可以通过点击操作列中的**执行记录**，跳转到**执行管理**页面，并查看到该定时运维所用模板的全部执行记录。



13. 点击**详情**，可以查看到的执行结果和输入参数等信息。



Cron表达式说明

Cron表达式采用中国时区UTC+8时间，可对应输入系统本地时间；支持5域的cron表达式，包括分、时、日、月、星期。

表达式支持的多种字符

（1）：表示匹配该域的任意值。假如在Minutes域使用, 即表示每分钟都会触发事件。

（2）?：只能用在DayofMonth和DayofWeek两个域。它也匹配域的任意值，但实际不会。因为DayofMonth和DayofWeek会相互影响。例如想在每月的20日触发调度，不管20日到底是星期几，则只能使用如下写法： 13 13 15 20 ?, 其中最后一位只能用？，而不能使用，如果使用*表示不管星期几都会触发，实际上并不是这样。

（3）-：表示范围。例如在Minutes域使用5-20，表示从5分到20分钟每分钟触发一次

（4）/：表示起始时间开始触发，然后每隔固定时间触发一次。例如在Minutes域使用5/20,则意味着5分钟触发一次，而25，45等分别触发一次.

（5），：表示列出枚举值。例如：在Minutes域使用5,20，则意味着在5和20分每分钟触发一次。

（6）L：表示最后，只能出现在DayofWeek和DayofMonth域。如果在DayofWeek域使用5L,意味着在最后的一个星期五触发。

（7）W:表示有效工作日(周一到周五),只能出现在DayofMonth域，系统将在离指定日期的最近的有效工作日触发事件。例如：在 DayofMonth使用5W，如果5日是星期六，则将在最近的工作日：星期五，即4日触发。如果5日是星期天，则在6日(周一)触发；如果5日在星期一到星期五中的一天，则就在5日触发。另外一点，W的最近寻找不会跨过月份 。

（8）LW:这两个字符可以连用，表示在某个月最后一个工作日，即最后一个星期五。

（9）#:用于确定每个月第几个星期几，只能出现在DayofMonth域。例如在4#2，表示某月的第二个星期四。

Cron表达式示例

"0 0 12 * * ?" 每天中午12点触发

"0 15 10 ? * *" 每天上午10:15触发

"0 15 10 15 * ?" 每月15日上午10:15触发

"0 0-5 14 * * ?" 在每天下午2点到下午2:05期间的每1分钟触发

"0 15 10 ? * 5L" 每月的最后一个星期五上午10:15触发

"0 15 10 ? * 5#3" 每月的第三个星期五上午10:15触发

相关产品

云服务器BCC

弹性调整EIP实例带宽

概览

运维编排OOS的报警事件运维能力可以解决报警或者事件发生后都需要人工参与故障处理过程运维效率低的问题，本实践通过在云监控BCM设置EIP实例的带宽使用率指标报警策略，并关联运维模板，实现带宽使用率过高时自动化的增加带宽，带宽使用率过低时自动调低带宽，降低EIP实例成本的同时提升管理效率降低运维管理成本。

需求场景描述

很多企业的业务量会随时间变化和难以预料的原因出现强烈的峰谷水位变化，大规模网络流量给弹性公网IP带来巨大负载，一旦没有及时调整EIP带宽产生网络拥挤的情况，会导致业务响应延迟甚至是中断。以前要应对类似情况，需要运维值班人员24小时关注网络流量和EIP使用率的变化情况并手动调整网络带宽，导致运维成本居高不下。而且由于运维动作不及时的问题难以避免，运维质量也难以保障。如果您也遇到类似的问题，可以跟随本文的方法，借助运维编排OOS的报警事件运维能力，通过策略自动地调整EIP带宽，提升运维效率，降低运维人力成本。

前提条件

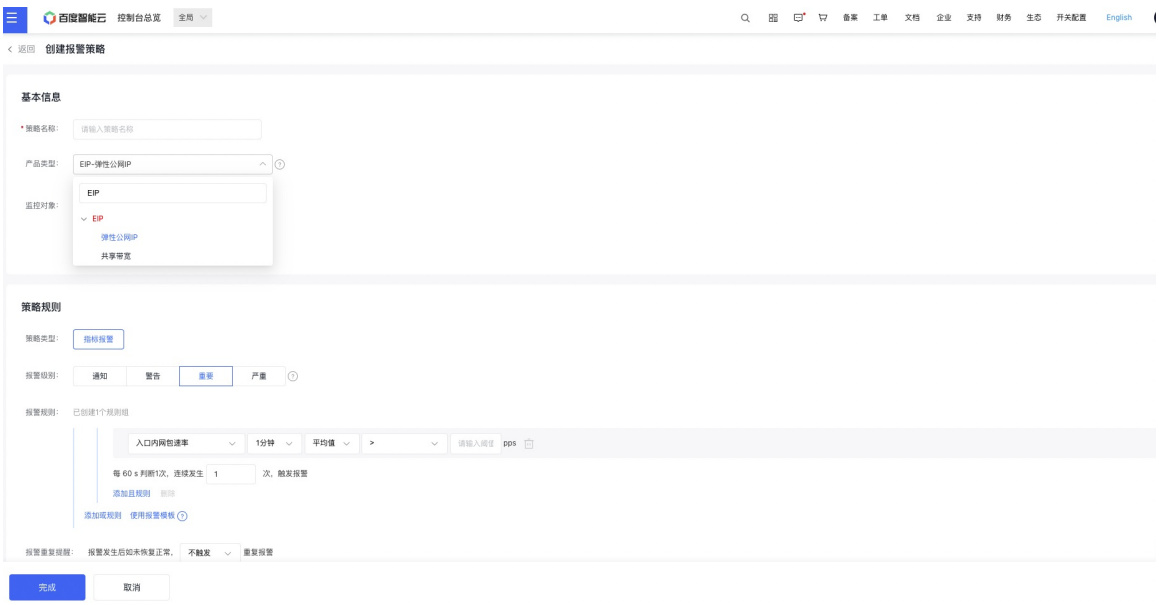
1. 开通了运维编排OOS服务、弹性公网IP EIP服务和云监控BCM服务。
2. 已经创建了EIP实例。

方案概述

百度智能云提供的云监控BCM能力可以实时监控云上资源的实时运行数据，并根据您设置的监控指标及报警策略记录和发送报警信息。运维编排OOS提供了报警事件运维能力，您可以通过把云监控BCM的指标报警和云产品事件信息作为触发运维操作的条件，一旦发生您预设的报警或事件时，运维编排OOS会自动化的执行对应的运维模板，实现自动化运维。

配置步骤

1. 登录百度智能云。
2. 通过产品服务点击“云监控BCM”进入云监控总览页面，在左侧导航栏中点击**报警管理-报警策略**，进入报警策略列表页面，点击**添加策略**，设置监控对象、报警触发策略规则（例如设置出带宽使用率1分钟平均值大于90%时触发报警）、通知方式等信息。如果您已经创建了报警策略，可以跳过这一步。



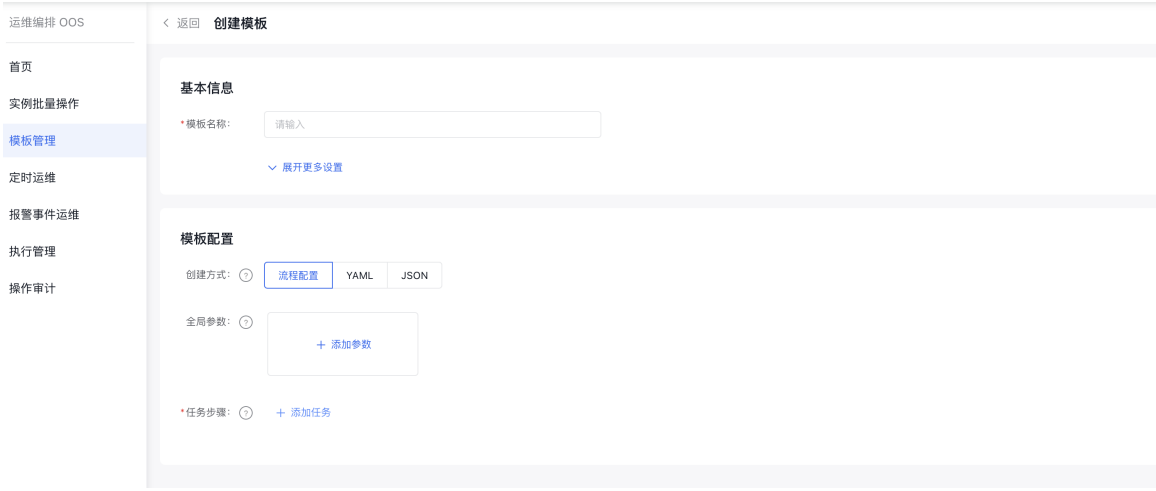
> 说明： 本实践中，产品类型需要选择EIP-弹性公网IP，监控对象选择您需要自动调整带宽的EIP。共享带宽EIP等其他网络资

> 源的配置方式与EIP相似。
>
> 报警规则支持且和或，如果您需要满足所有规则触发时请使用且规则，如果满足任意规则就触发时请选择或规则。
>
> 一个报警策略只能选择一种产品类型，但可以选择同类型的一个或多个实例，请选择您希望弹性调整带宽的EIP实例。 更多配置细节的参见管理报警

3. 通过产品服务点击“运维编排OOS”进入运维编排页面，选择**模板管理** > **我的模板** > **创建模板**。在**基本信息**填入**模板名称**，如本实践中您可以把名称填写为“自动调整EIP带宽”，以便后续定位和使用这一模版，然后按需可选配置标签和描述信息。

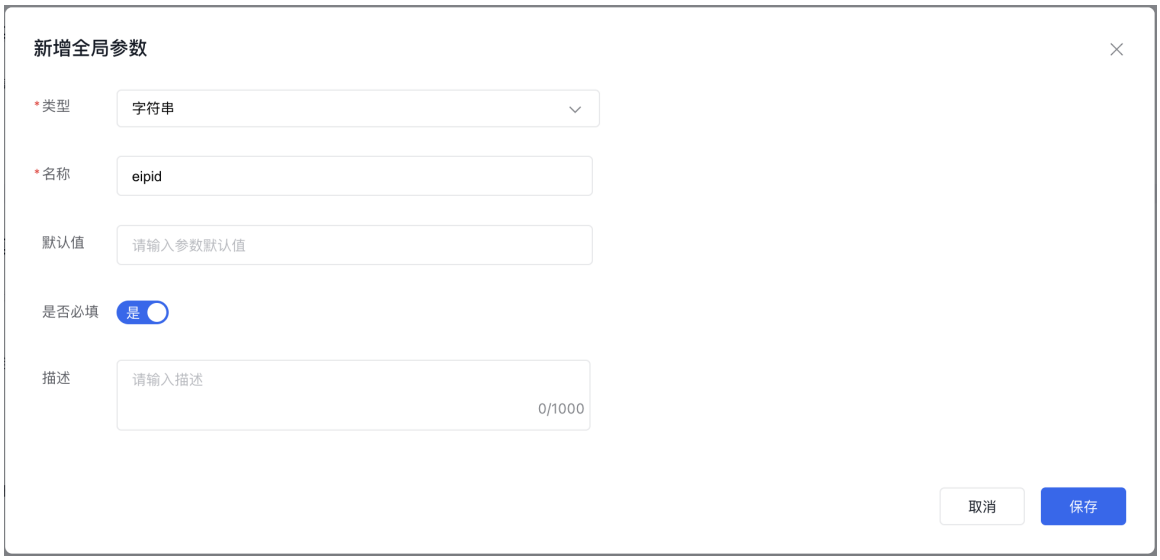
> 说明： 模板名称支持大小写字母、数字、中文以及-_ /.特殊字符，必须以字母或中文开头，长度1-200

4. 在**模板配置**按您的使用习惯选择**流程配置**、**YAML**、**JSON**中的一种方式配置，例如选择**流程配置**



> 说明： 如果您已经通过YAML或JSON方式完成了模版，可以选择对应的方式，直接粘贴模版内容到代码输入框中，经过模版校验即可。

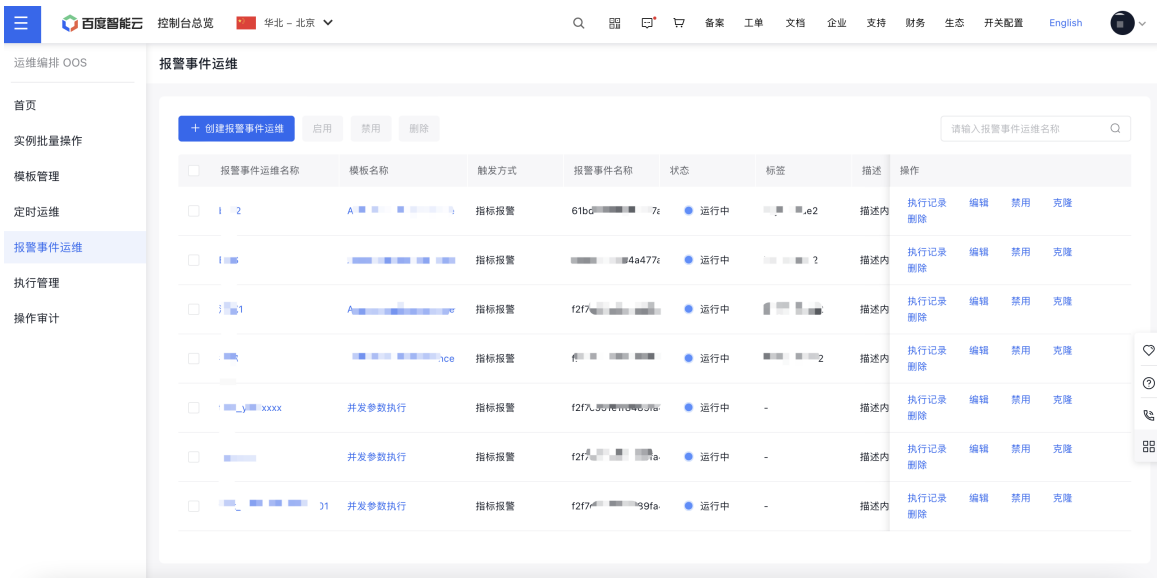
5. 此步骤仅适用于把EIP作为全局参数在多处引用的场景，如果您仅需要监控和操作某一个特定EIP，可以不设置全局参数并跳过此步骤。在全局参数中点击**添加参数**，设置“eipid”为参数名称，这一参数可在新增任务时作为全局参数使用。



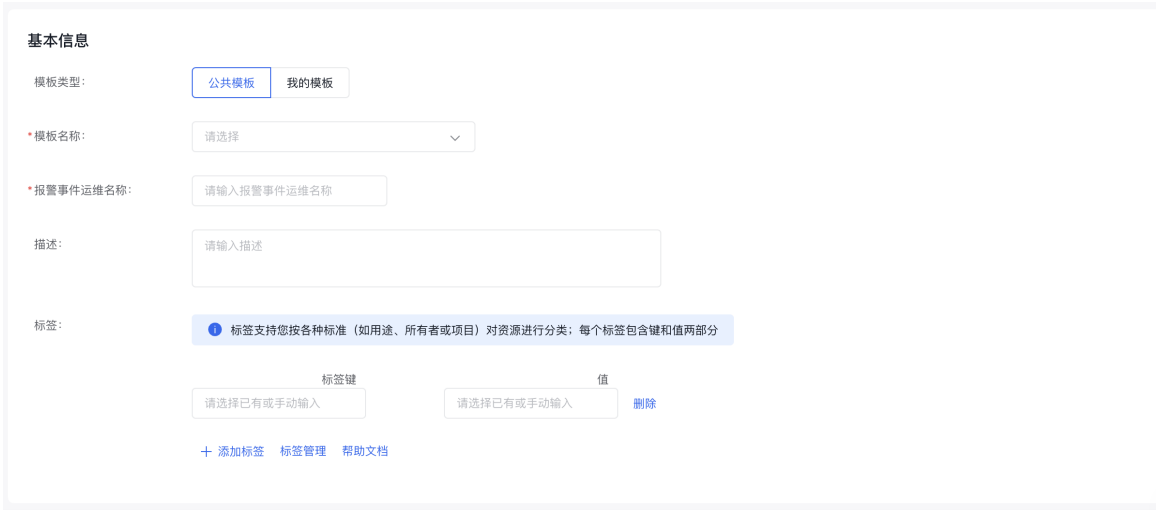
6. 点击**添加任务**,任务类型选择**调整EIP实例带宽**，输入任务名称（如自动调整EIP带宽），按需设置重试次数和超时时间等信息，详细信息可以参考创建模版的说明。

7. 切换到参数,如果您仅需要针对一个特定EIP进行监控报警运维，此处填入对应的EIP公网IP地址即可。如果您针对告警数据的中特定EIP执行调整带宽的动作，需要把实例ID设置为引用函数的方式，填如全局参数，即{"Ref": "eipid"} 。然后调整方式选择为“增加”，并按需填入要增加的公网带宽，如1Mbps。完成模板配置中的参数信息后点击保存。

8. 点击保存模板，即可创建完成模板。
9. 在左侧导航栏中点击**报警事件运维**，进入报警事件运维列表页面，点击**创建报警事件运维**，设置触发方式、模版、参数等信息。



10. 触发方式选择指标报警，云产品类型选择弹性公网EIP。
11. 选择您在云监控BCM创建的指标报警，此时监控对象与报警规则显示了您在云监控BCM控制台创建监控报警策略时定义的数据。
12. 模板类型选择**我的模板**，选择您在模版管理中创建的**自动调整EIP带宽**模板。
13. 填入**报警事件运维名称**和其他参数，其他参数可按需填写。



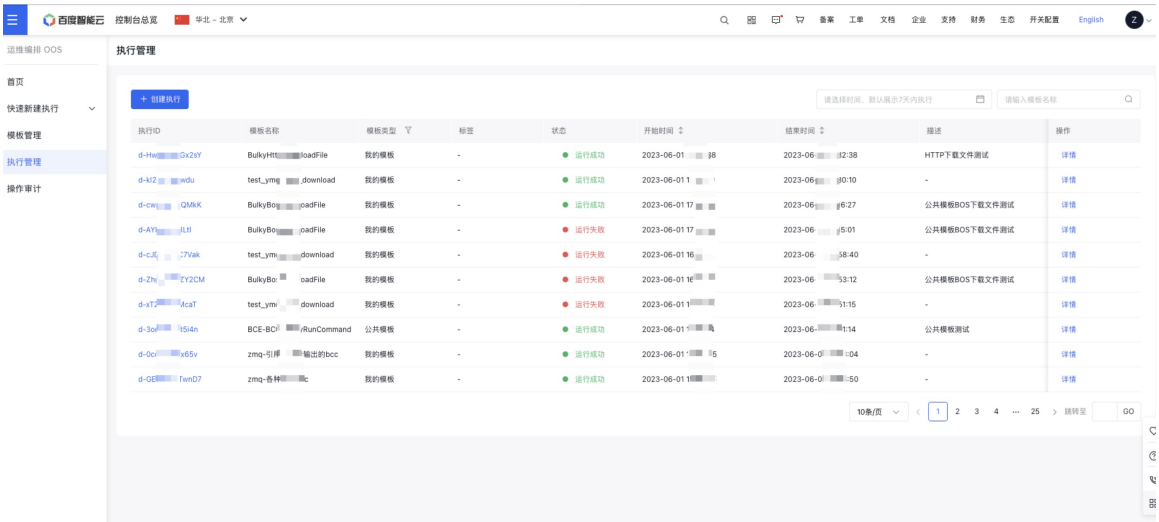
> 说明： 执行名称只能输入200个字符以内的中英文字符、数字和- _，且只能以中文汉字或英文字母开头。

14. 单击下一步，设置要执行操作的弹性公网IP和调整的带宽。公网IP地址可以通过手动指定固定IP的方式，即直接填入要监控的公网IP地址

> 说明： 如果您需要根据报警事件消息中的参数确定要调整带宽的EIP，则可直接在文本输入框内输入{{.消息体名.参数名}}，来引提取消息字段，例如需要从报警详情（metric）中提取BCC实例id，参数值应该使用表达式{{ .metric.resourceId }} 部分常用消息体及其可引用的参数参见模板参数填写说明。
>
> 部分常用消息体及其可引用的参数参见模板参数填写说明，查看更多云产品指标信息，查看更多云产品事件信息。

15. 点击提交，返回到列表即可查看到已经创建的报警事件运维。

16. 当报警事件运维执行后，可以通过点击操作列中的**执行记录**，跳转到**执行管理**页面，并查看相关的全部执行记录。



17. 点击**详情**，可以查看到的执行结果和输入参数等信息。

相关产品

云监控BCM

BSM-agent

云助手客户端概述

云助手客户端（BSM-Agent），是百度智能云为实现云服务器BCC、BBC等实例自动化运维打造的工具，云助手客户端安装到实例后，后续无需输入用户名密码，无需使用跳板机，就可以实现免登录控制实例执行命令和上传文件等，还可以进一步实现更复杂的定时执行自动化运维脚本、配置运行环境、安装更新卸载插件、补丁和软件等操作。

注意：百度智能云官方镜像自动安装了云助手客户端，如您的云服务器BCC实例使用了私有镜像或其他原因导致云助手客户端离线，建议重新安装云助手客户端，管理云助手客户端可参考安装BSM-Agent

安装BSM-Agent

本文记录了在Linux平台和Windows平台安装、卸载云助手客户端的相关命令，您需要登录到云服务器操作系统中执行对应的命令。

Linux平台

第一部分适用于未自动安装BSM-Agent的Linux操作系统的云服务器BCC。

支持范围

目前运维编排OOS服务支持的操作系统包括如下：

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

检查服务

输入`systemctl status bsm-agent-upgrader.service`查看相关服务是否存在，如果不存在，请您手动安装。

手动安装

执行如下命令安装：

```
sh -c  "$(curl -fsSL http://download.bcm.baidubce.com/bsm-agent/bin/install.sh)"
```

通过tail -f -n 100 /opt/bsm-agent/log/baidu-oss-agent.log 查看是否启动成功，有连接，注册成功和开始处理任务结果的日志，说明启动成功

卸载

执行如下命令卸载bsm-agent **ubuntu系统**

```
 #### 停止bsm-agent服务
systemctl stop bsm-agent-upgrader.service
systemctl disable bsm-agent-upgrader.service
#### 删除启动服务脚本
rm /lib/systemd/system/bsm-agent-upgrader.service
rm /usr/bin/bsm-agent-daemon.sh
#### 删除数据
rm -rf /opt/bsm-agent
```

其他Linux系统

```
 #### 停止bsm-agent服务
systemctl stop bsm-agent-upgrader.service
systemctl disable bsm-agent-upgrader.service
#### 删除启动服务脚本
rm /usr/lib/systemd/system/bsm-agent-upgrader.service
rm /usr/libexec/bsm-agent-daemon.sh
#### 删除数据
rm -rf /opt/bsm-agent
```

Windows平台

本附录适用于未自动安装BSM-Agent的Windows操作系统的云服务器BCC。

手动安装

用管理员身份打开cmd命令，执行如下命令：

```
#### 下载文件 并 执行文件
powershell (new-object System.Net.WebClient).DownloadFile('http://download.bcm.baidubce.com/bsm-agent/bin/install.bat','install.bat') && install.bat & del /F install.bat
```

手动卸载

用管理员身份打开cmd命令，执行如下命令：

```
cd c:\Program Files\bsm-agent
nssm stop bsm-agent-daemon
nssm remove bsm-agent-daemon
```

如果此时任务管理器还有bsm-agent任务，可以在任务管理器中手动点击结束任务。

API参考

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oos.{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

OOS 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

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，凡是HTTP标准中规定的表示日期和时间字段用GMT，其他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：

- 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。
- 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。
- 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`。

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

服务域名

OOS API的服务域名为：

地域	域名	端口	协议
北京	oos.bj.baidubce.com	80	HTTP
广州	oos.gz.baidubce.com	80	HTTP
苏州	oos.su.baidubce.com	80	HTTP
保定	oos.bd.baidubce.com	80	HTTP
武汉	oos.fwh.baidubce.com	80	HTTP
成都	oos.cd.baidubce.com	80	HTTP

模板任务（Operator）接口

模板任务（Operator）列表

请求结构

- method：POST
- URL：/api/logic/oos/v1/operator/list

请求参数

名称	类型	描述	是否必须	参数位置
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

请求示例

```
 POST /api/logic/oos/v1/operator/list
{
   "pageNo":1,
   "pageSize":100
}
```

响应示例

```
 {
   "success":true,
   "msg":"",
   "result":{
      "operators":[
         {
            "name":"BCE::BCC::StopInstance",
            // 结构体字段参考附录中的Operator
         }
      ],
      "orderBy":"createTime",
      "order":"desc",
      "pageNo":1,
      "pageSize":100,
      "totalCount":26
   }
}
```

运维模板（Template）接口

创建运维模板

接口描述

您可以在模板中指定执行所需参数（参考请求示例1），也可以通过全局参数的形式在创建执行的时候再指定（参考请求实例2）

请求结构

- method：POST
- URL：/api/logic/oos/v2/template

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例1

```
POST /api/logic/oos/v2/template
{
    "description":"这是一段描述",
    "name":"test_oos_template1",
    "operators":[
        {
            "name":"stop_bcc",
            "description":"停止BCC实例",
            "operator":"BCE::BCC::StopInstance",
            "label":"停止BCC实例",
            "retries":0,
            "retryInterval":60000,
            "timeout":3600000,
            // 指定需要开机的BCC实例
            "properties": {
                "instanceId": [
                    {
                        "id":"d8293318-****-****-****-b0e72cb3f3ba",
                        "shortId":"i-pqJV5P**"
                    }
                ]
            }
        },
        {
            "name":"start_bcc",
            "description":"启动BCC实例",
            "operator":"BCE::BCC::StartInstance",
            "label":"启动BCC实例",
            "retries":0,
            "retryInterval":60000,
            "timeout":3600000,
            // 指定需要开机的BCC实例
            "properties": {
                "instanceId": [
                    {
                        "id":"d8293318-****-****-****-b0e72cb3f3ba",
                        "shortId":"i-pqJV5P**"
                    }
                ]
            }
        }
    ],
    "linear":true
}
```

请求示例2

```
POST /api/logic/oos/v2/template
{
    "description":"这是一段描述",
    "name":"test_oos_template2",
    "operators":[
        {
            "name":"stop_bcc",
            "description":"停止BCC实例",
            "operator":"BCE::BCC::StopInstance",
            "label":"停止BCC实例",
            "retryInterval":60000,
            "timeout":3600000,
            // 引用全局参数中的instanceId1
            "properties": {
                "instanceId": {
                    "Ref":"instanceId1"
                }
            }
        },
        {
            "name":"start_bcc",
            "description":"启动BCC实例",
            "operator":"BCE::BCC::StartInstance",
            "label":"启动BCC实例",
            "retryInterval":60000,
            "timeout":3600000,
            // 引用全局参数中的instanceId2
            "properties": {
                "instanceId": {
                    "Ref":"instanceId2"
                }
            }
        }
    ],
    // 指定全局参数
    "properties":[
        {
            "required":true,
            "type":"bccInstance",
            "name":"instanceId1"
        },
        {
            "required":true,
            "type":"bccInstance",
            "name":"instanceId2"
        }
    ],
    "linear":true
}
```

响应示例

```
{
  "success":true,
  "msg":"",
  "code":200,
  "result":{
     // 模板唯一标识，用于更新运维模板
     "id":"tpl-tKXV1J**"
  }
}
```

校验运维模板

请求结构

- method：POST
- URL：/api/logic/oos/v2/template/check

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例

```
POST /api/logic/oos/v2/template/check
// 参考创建运维模板接口
```

响应示例

成功响应示例

```
{
  "success":true,
  "msg":"",
  "code":200
}
```

失败响应示例

```
{
  "success":false,
  "requestId":"62e96041-****-****-****-bb442548e54c",
  "code":"ExecutionException",
  "message":{
     "global":"template is invalid: duplicated op name start_bcc"
  }
}
```

更新运维模板

请求结构

- method：PUT
- URL：/api/logic/oos/v2/template

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例

```
PUT /api/logic/oos/v2/template
{
  "id": "tpl-tKXV1J**",
  // 其余字段参考创建模板请求，注意，目前运维模板不支持部分更新，只能全量更新
}
```

删除运维模板

请求结构

- method：DELETE
- URL：/api/logic/oos/v2/template?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
id	String	运维模板ID	是	Query参数

请求示例

```
DELETE /api/logic/oos/v2/template?id=tpl-tKXV1J**
```

查看运维模板

请求结构

- method：GET
- URL：/api/logic/oos/v2/template?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
name	String	运维模板名称	是	Query参数

请求示例

```
GET /api/logic/oos/v2/template?name=test_oos_template
```

响应示例

```
{
  "success":true,
  "msg":"",
  "code":200,
  "result":{
     "id":"tpl-tKXV1J**",
     "name":"test_oos_template",
     "type":"INDIVIDUAL",
     "description":"",
     // 其余字段参考附录中的Template
  }
}
```

查看运维模板列表

请求结构

- method：POST
- URL：/api/logic/oos/v2/template/list

请求参数

名称	类型	描述	是否必须	参数位置
sort	String	排序字段，默认为创建时间	否	RequestBody参数
ascending	boolean	是否升序，默认false	否	RequestBody参数
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

请求示例

```
POST /api/logic/oos/v2/template/list
{
   "sort":"createTime",
   "ascending":false,
   "pageNo":1,
   "pageSize":10
}
```

响应示例

```
{
  "success":true,
  "msg":"",
  "result":{
    "templates":[
      {
        "id":"tpl-4sOeks**",
        "name":"test_oos_template1",
        "type":"INDIVIDUAL",
        "description":""
        // 其余字段参考附录中的Template
      },
      {
        "id":"tpl-tKXV1J**",
        "name":"test_oos_template2",
        "type":"INDIVIDUAL",
        "description":""
        // 其余字段参考附录中的Template
      }
    ],
    "orderBy":"createTime",
    "order":"desc",
    "pageNo":1,
    "pageSize":10,
    "totalCount":149
  }
}
```


执行（Execution）接口

创建执行

接口描述

可以通过如下两种方式创建执行：

1. 引用一个已存在的模板（参考请求示例1）
2. 在创建执行时动态创建一个模板（参考请求示例2）

请求结构

- method：POST
- URL：/api/logic/oos/v2/execution

请求参数

名称	类型	描述	是否必须	参数位置
execution	Execution	执行描述	是	RequestBody参数

请求示例1

```
POST /api/logic/oos/v2/execution
{
    "template":{
        "ref":"tpl-tKXV1J**",
        "name":"test_oos_template",
        "linear":true
    },
    "properties":{
        "strParam":"testParam"
    },
    "description":"这是一段描述",
    "tags":[
        {
            "tagKey":"key2",
            "tagValue":"value2"
        }
    ]
}
```

请求示例2

```
{
    "template": {
        "name": "执行脚本的模板",
        "linear": true,
        "operators": [
            {
                "name": "run_script",
                "description": "这是一段描述",
                "operator": "BCE::Agent::ExecuteShell",
                "retries": 0,
                "retryInterval": 60000,
                "timeout": 3600000,
                "properties": {
                    "content": "ls /",          // 脚本内容，必填
                    "user": "root",             // 以指定的用户身份执行脚本，必填
                    "workDir": "/tmp",          // 在指定的目录执行脚本，必填
                    "__workerSelectors__": [    // 选择执行的虚机列表，必填
                        {
                            "id": "3f367371-****-****-****-c27f57798e7e", // BCC长实例id
                            "shortId": "i-w1ayJ7**", // BCC短实例id
                        }
                    ]
                }
            }
        ]
    }
}
```

响应示例

```
{
  "success":true,
  "msg":"",
  "code":200,
  "result":{
     // 执行ID
     "id":"d-CRpnWZ******"
  }
}
```

查看执行详情

请求结构

- method：GET
- URL：/api/logic/oos/v2/execution?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
id	String	执行ID	是	Query参数

请求示例

```
GET /api/logic/oos/v2/execution?id=d-lkzdgP******
```

响应示例

```
{
  "success":true,
  "msg":"",
  "code":200,
  "result":{
     "id":"d-lkzdgP******",
     // 结构体字段参考附录中的Execution
  }
}
```

查询执行列表

请求结构

- method：POST
- URL：/api/logic/oos/v2/execution/list

请求参数

名称	类型	描述	是否必须	参数位置
template	Template	模板信息，如设置则只返回指定模板触发的执行列表	否	RequestBody参数
state	String	执行状态，如设置则只返回指定状态的执行列表，可选值：RUNNING（执行中）、SUCCESS（执行成功）、FAILED（执行失败）、CANCELED（已取消）、PENDING（等待中）	否	RequestBody参数
eventExecutionName	String	报警事件运维名称，如设置则只返回指定报警事件运维触发的执行列表	否	RequestBody参数
cronExecutionName	String	定时运维名称，如设置则只返回指定定时运维触发的执行列表	否	RequestBody参数
startTime	long	执行开始时间，单位：毫秒，默认值为7天前毫秒时间戳	否	RequestBody参数
endTime	long	执行结束时间，单位：毫秒，默认值为当前毫秒时间戳	否	RequestBody参数
sort	String	排序字段，默认为执行开始时间，可选值：startTime，endTime	否	RequestBody参数
ascending	boolean	是否升序，默认false	否	RequestBody参数
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100，最小值：1	是	RequestBody参数

请求示例

```
POST /api/logic/oos/v2/execution/list
{
    "sort": "startTime",
    "state": "SUCCESS",
    "startTime": 1719296223700,
    "endTime": 1719901023700,
    "template": {
        "name": "my_template_123"
    },
    "pageNo": 1,
    "pageSize": 10
}
```

响应示例

```
{
  "success": true,
  "msg": "",
  "result": {
    "executions": [
      {
        "id": "d-kwE00s******",
        "template": {
          "name": "my_template_123",
          "type": "INDIVIDUAL",
          // 结构体字段参考附录中的Template
        },
        "parallelism": 0,
        "createdTimestamp": 1719900946278,
        "updatedTimestamp": 1719901257454,
        "finishedTimestamp": 1719901257454,
        "state": "SUCCESS",
        "reason": ""
      }
    ],
    "orderBy": "startTime",
    "order": "desc",
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 3
  }
}
```


任务（Task）接口

任务详情

请求结构

- method：POST
- URL：/api/logic/oos/v2/task?{Query参数}namespace=${ns}&dagId=${dagId}&taskId=${taskId}&ignoreChildren

请求参数

名称	类型	描述	是否必须	参数位置
dagId	String	执行（Execution）ID	是	Query参数
taskId	String	任务（Task）ID	是	Query参数
namespace	String	名称空间，如果用户未使用自定义名称空间，可以不传此字段，默认为“default”，	否	Query参数
ignoreChildren	boolean	是否忽略子执行，默认为“false”，不忽略子执行，	否	Query参数

请求示例1

```
GET /api/logic/oos/v2/task?namesapce=${ns}&dagId=${dagId}&taskId=${taskId}&ignoreChildren=false
```

响应示例

字段详见Task

```
{
  "success": true,
  "msg": "", // 若失败，返回失败原因
  "result": {
     // 用户ID
     "userId": "******",

     // 当前任务id，全局唯一
     "id": "t-v9ZgQH******",

     // 版本号，当需要更改Task时，需要上传该版本号
     "revision": 10,

     // 当前Task所属执行id
     "execution": {
        "id": "d-nUAtn3******",
     },
     // 当前Task执行对应的Operator
     "operator": {
        "name": "stopFlow", // 节点名称
        "description": "BVS摘流", // 节点描述
     }
     //其余字段参考附录中的Task
  }
}
```

查询任务子执行列表

请求结构

- method：POST
- URL：/api/logic/oos/v2/task/children/list

请求参数

名称	类型	描述	是否必须	参数位置
executionId	String	执行（Execution）ID	是	RequestBody参数
taskId	String	任务（Task）ID	是	RequestBody参数
namespace	String	名称空间，如果用户未使用自定义名称空间，可以不传此字段，默认为“default”，	否	RequestBody参数
states	List<String>	按state进行筛选，选填，若未设置，返回所有状态的子执行	否	RequestBody参数
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100，最小值：1	是	RequestBody参数

请求示例

```
POST /api/logic/oos/v2/task/children/list
{
    "namespace": "default",
    "executionId": "d-98d95c3******",
    "taskId": "t-42a95f55c39******",
    "states": ["RUNNING", "SUCCESS"],
    "pageNo": 1,
    "pageSize": 10
}
```

响应示例

```
{
  "success": true,
  "msg": "", // 若失败，返回失败原因
  "result":{
      // 子执行总数量
     "total": 63
     "pageNo": 1, // 第几页，从1开始计数，必填
     "pageSize": 10, // 每页展示数量，必填，最大值：100
     "orderBy" : "";
     "order": "";
     // 子执行分页列表
     "children": [
        {
          // 当前子执行对应的loops序号
          "loopIndex": 0,
          "createdTimestamp": 1700468603936, // 创建时间戳，Unix时间戳，单位：毫秒
          "updatedTimestamp": 1700468605197, // 最近更新时间戳，Unix时间戳，单位：毫秒
          "finishedTimestamp": 1700468605197, // 执行结束时间戳，Unix时间戳，单位：毫秒，若未结束，该字段填
0

          // 当前子执行的状态，可选值
          // PENDING        - 等待中
          // RUNNING        - 运行中
          // SUCCESS        - 运行成功
          // FAILED         - 运行失败
          // UP_FOR_RETRY   - 等待重试中
          "state": "SUCCESS",

          // 子执行虚机参数
          "properties": {
            "__worker__": {
              "instanceId": "i-1e******",
              "internalIp": "192.168.*.*",
              "name": "****"
            }
          },
           // 当前子执行的执行实例列表。当发生重试时，会产生多个执行实例
          "executions": [
            {
              "id": "d-8763f77******",
              "createdTimestamp": 1659534994121,
              "revision": 10,
            }
          ]
        }
        ...
     ]
  }
}
```

替换参数重新执行Task

请求结构

- method：POST
- URL：/api/logic/oos/v2/task/clear

请求参数

名称	类型	描述	是否必须	参数位置
executionId	String	执行（Execution）ID	是	RequestBody参数
taskId	String	任务（Task）ID	是	RequestBody参数
operator	String	算子名称	是	RequestBody参数
namespace	String	名称空间，如果用户未使用自定义名称空间，可以不传此字段，默认为“default”，	否	RequestBody参数
clearDownstreams	boolean	是否一并重置下游节点	否	RequestBody参数
properties	Map<String, Object>	重新执行task的参数	否	RequestBody参数

请求示例

```
POST /api/logic/oos/v2/task/clear
{
    "namespace": "default",
    "taskId": "t-42a95f55******",
    "executionId": "d-f3b7c******",
    "operator": "BCE::Agent::ExecuteShell",
    "clearDownstreams": true,
    "properties": {
        "content": "echo hello",
        "instances": [
            {
                "instanceId": "i-Ut******",
                "internalIp": "192.168.*.*",
                "name": "***"
            }
        ],
        "user": "root",
        "workDir": "/"
    }
}
```

响应示例

```
{
    "success": true,
    "msg": "", // 若失败，返回失败原因
}
```

附录

附录

Template

名称	类型	描述
description	String	模板描述
name	String	模板名称，不允许重复
operators	List<Operator>	模板任务步骤列表
properties	List<Property>	全局参数列表
linear	boolean	任务是否串行执行，必填
links	List<Link>	描述operator之间的拓扑关系，若linear=false，则该字段必填

Operator

名称	类型	描述
name	String	任务自定义名称，同一个模板下不允许重复
description	String	任务描述，选填
operator	String	任务ID，参考模板任务列表接口返回内容
label	String	任务显示名称
parallelismRatio	double	允许的并行比例，选填，默认值：0，表示串行执行。取值范围[0, 1]，该字段仅当loops字段存在时生效
parallelismCount	int	允许的并行个数，选填，默认值：0，表示串行执行，只允许对parallelismRatio和parallelismCount之一进行设置
allowedFailureRatio	double	允许失败的loops比例，选填，默认值：0，表示不允许失败。取值范围[0, 1]，该字段仅当loops字段存在时生效，表示在循环中允许失败的比例
allowedFailureCount	int	允许失败的loops个数，选填，默认值：0，表示不允许失败
retries	int	重试次数，选填，默认值：0，表示不重试
retryInterval	int	重试间隔，单位：毫秒，选填，默认值：5分钟
timeout	int	超时时长，单位：毫秒，选填，默认值：6小时，如果任务执行时长超过该限制，将会触发重试
scheduleDelayMilli	int	延时启动，单位毫秒，默认值0
manually	boolean	是否需要手动执行，默认值false
pauseOnFailure	boolean	失败后是否暂停，默认值false
category	List	任务所属的类别
properties	Map<String, Object>	任务执行所需参数列表
events	List<Event>	任务关联产生的审计事件列表

Property

名称	类型	描述
type	String	参数类型
name	String	参数名称
required	boolean	是否必填
description	String	参数描述
defaultValue	Object	默认取值
value	Object	实际取值

Execution

名称	类型	描述
id	String	执行ID，全局唯一
template	Template	运维模板内容
createdTimestamp	long	执行开始时间，Unix时间戳，单位：毫秒
updatedTimestamp	long	执行更新时间，Unix时间戳，单位：毫秒
finishedTimestamp	long	执行结束时间，Unix时间戳，单位：毫秒，若未结束，该字段填0
state	String	执行状态，可取值为： PENDING（等待中） RUNNING（运行中） SUCCESS（运行成功） FAILED（运行失败） PAUSED（暂停） CANCELED（已取消） ROLLBACK（回滚中） ROLLBACK_SUCCESS（回滚成功） ROLLBACK_FAILED（回滚失败）
properties	Map<String, Object>	全局参数取值集合
description	String	执行描述
tasks	List<Task>	执行中的任务列表
tags	List<Tag>	执行绑定标签列表

Task

名称	类型	描述
id	String	任务ID，全局唯一
dag	Execution	任务所属的执行
createdTimestamp	long	任务开始时间，Unix时间戳，单位：毫秒
updatedTimestamp	long	任务更新时间，Unix时间戳，单位：毫秒
finishedTimestamp	long	任务结束时间，Unix时间戳，单位：毫秒，若未结束，该字段填0
state	String	任务状态，可取值为： PENDING（等待中） RUNNING（运行中） SUCCESS（运行成功） FAILED（运行失败） UP_FOR_RETRY（等待重试中） UPSTREAM_FAILED（由于上游失败而导致的当前节点失败） SKIPPED（被跳过而未执行） IGNORED（错误被忽略）
operator	Operator	任务详细信息
initContext	Map<String, Object>	任务初始参数
context	Map<String, Object>	任务上下文信息，包含全局参数和输出结果
outputContext	Map<String, Object>	任务输出结果
tries	int	任务尝试次数，从0开始
children	List<Task>	子任务列表

Link

名称	类型	描述
src	String	上游operator的name
dst	String	下游operator的name

Tag

名称	类型	描述
tagKey	String	标签key
tagValue	String	标签value

常见问题

常见问题

BSM-Agent是什么？

云助手客户端（BSM-Agent），是百度智能云为实现BCC实例自动化运维打造的工具，云助手客户端安装到实例后，无需输入用户名密码，无需使用跳板机，就可以实现免登录控制实例执行命令和上传文件等，还可以进一步实现更复杂的定时执行自动化运维脚本、配置运行环境、安装更新卸载插件、补丁和软件等操作。

为什么在我的模版中没有找到之前已经创建的模版呢？

不同区域的运维编排模板不通用，请先切换到其它区域检查下是否存在之前创建的模板。如果依旧没找到对应模板，可以联系技术人员。

运维编排支持针对哪些资源操作呢？

目前运维编排OOS支持以BCC实例（含EBC实例）为主体进行运维编排，暂不支持以CCE实例等其它类型的实例为主体进行运维编排。

实例列表为什么没有我想要的实例？

实例列表中仅展示您当前所在区域的实例，例如您的导航当前在华北-北京区域，在选择实例时，仅能查看到华北-北京区域的实例，无法查看到华北-保定或其它区域的实例。此时如果您需要操作其它区域的实例，可以切换到保定区域再创建执行。

我需要把北京区域的实例使用保定区域的自定义模板创建定时运维，应该怎么操作？

由于不同区域的自定义模板不互通，您可以先进入华北-保定区域，在模版管理-我的模版中找到您需要执行的模板并进入详情，复制模板内容，随后切换到华北-北京区域，在模版管理-我的模板中通过创建模版粘贴到模版内容输入框中，点击保存模板即可。后续创建定时运维的操作步骤请参见定时运维章节。

我可以导入外部模版吗？

您可以复制外部模板的内容，直接粘贴到模版内容输入框来创建模版。目前暂不支持直接导入和导出模板。

如何查看运维编排在哪一步执行失败了？

运维编排执行失败可能存在多种原因，您可以先进入到执行管理界面查看任务的执行信息，您可以直观地查看到线形流程中哪一步失败了，可以通过点击对应的步骤，查看到具体的输入参数、输出参数和日志信息。目前仅支持线性流程的操作结果可视化展示，暂不支持非线性流程的可视化执行结果展示。

为什么针对某一个BBC实例运维编排操作一直失败？

导致运维编排操作失败原因较多，如果您在执行管理中发现针对某一台实例所有步骤的任务都执行失败，请先检查该实例的BSM-Agent状态是否在线，OOS无法操作您的BSM-Agent状态为离线的实例，您可以通过尝试重装BSM-Agent解决离线的问题，如还不能解决，可以通过工单联系我们。

运维编排OOS收费吗？

运维编排OOS能力本身不收费，但是任务会涉及到创建资源/调整资源配置等影响资源费用的操作，响应费用由对应的云产品收取，具体价格请参见对应产品的规定。

运维编排OOS目前支持哪些任务呢？

运维编排OOS目前支持针对BCC/BBC/EIP/CDS/NAT/ENI等多种云产品进行增删改等近40项操作，您可以在创建模版时按需选用。

运维编排OOS支持什么语法?

支持YAML/JSON，此外还支持可视化流程配置

我不会写运维编排OOS的代码，应该如何使用运维编排呢？

运维编排OOS提供可视化编排的能力，您可以在创建模版时使用流程配置的方式创建模板，此时您可以按照流程步骤依次添加要执行的任务，我们会自动生成对应内容的代码（您无需关注代码内容）完成所有必填参数后，即可创建完成一个模版供后续执行时使用。

报警事件运维有什么作用？

运维编排OOS会实时监测报警和事件信息，当云产品的监控指标达到预先设定的阈值触发报警，或者系统产生了云产品事件时，运维编排OOS会按照预先设置的策略执行对应的运维模板，实现自动化运维。例如您可以使用报警事件运维自动调整后付费模式EIP的带宽或者NAT的规格等。

报警事件运维中没有合适的监控报警应怎么办？

报警事件运维中的指标报警和云产品事件数据均使用了云监控BCM的数据，如果您选择了正确的云产品后，依旧没有发现可用

的数据，请前往云监控BCM控制台自行创建监控策略。

使用云产品事件配置报警事件运维时，需要过滤部分报警事件，应该如何写过滤规则呢？

以云服务器BCC事件为例，如果想配置事件等级为NOTICE类型，并且事件类型是GPU温度过高或BCC状态变化通知的过滤规则，则配置示例如下：

```
 [
{"key":"resource.eventLevel","op":"=","value":"NOTICE"},
{"key":"resource.eventType","op":"in","value":["GpuHighTemperature","InstanceStateChange"]}
]
```

使用指标报警配置报警事件运维设置参数值时，需要引用报警消息中的参数，应该怎么填写模板参数 如果从消息体中提取第一份监控数据中的BCC实例id（更多消息体信息请前往百度智能云文档中心的运维编排OOS查看模板参数填写说明）应该使用表达式

```
 {{ (index .metrics 0).shortInstanceId }}，如果从消息体中提取第二份监控数据中的监控项名称，应该使用表达式 {{ (index .metrics 1).name }}
```

使用运维编排OOS的实例批量操作通过OOS上传文件时，为什么提示先开通对象存储BOS服务？

运维编排OOS的实例批量操作通过OOS上传文件借助的百度智能云的对象存储BOS服务，因此您需要先开通对象存储服务才可以使用，请注意使用对象存储的费用请参见对应云产品的说明。

运维编排OOS支持带分支的流程吗？

运维编排OOS提供条件分支结构，您可以通过写入JSON或YAML格式的文本的方式创建复杂分支的运维流程，目前还不支持通过可视化界面创建这样的负载运维编排模板。

运维编排OOS的执行结果在哪里查看？

在运维编排控制台的执行管理页面，可以查看到一次执行的整体结果。在详情中可以查看到此次执行的详细信息，当一次执行包含多个子执行时，点击展开子执行即可查看更详尽的执行结果。

Python-SDK

概述

概述

本文档主要介绍OOS Python SDK的安装和使用。

在阅读本文档前，您需要先了解OOS的基本知识。若您还不了解OOS，可以参考产品描述和操作指南

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
    │      └──── oos                //OOS目录
    │           ├────oos_client.py      //OOS客户端入口类
   |           ├────oos_model.py       //OOS模型类
    ├──── bce_base_client.py          //BCE客户端入口类的基类
    ├──── bce_client_configuration.py    //BCE客户端的HttpClient配置类
    ├──── bce_response.py             //BCE客户端的请求类
    ├──── exception.py                //BCE客户端的异常类
    ├──── protocol.py                 //网络协议定义
    ├──── region.py                   //区域处理模块
    ├──── retry_policy.py             //BCE服务公共配置类
    └──── utils.py                    //BCE公用工具类
```

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OosClient时将区域对应的Endpoint做为参数填入。

OOS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。

OOSClient

新建OOSClient

OOSClient是OOS服务的客户端，为开发者与运维编排服务进行交互提供了一系列的方法。

通过AK/SK方式访问OOS

1.在新建OosClient之前，需要先创建配置文件对OosClient进行配置，以下将此配置文件命名为`oos_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OosClient的Host，Access Key ID和Secret Access Key
HOST = 'oos.bj.baidubce.com'
AK = 'your-access-key-id'
SK = 'your-secret-access-key'

#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(AK, SK), endpoint=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OOS服务的域名，可以通过赋值给`oos_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OosClient。

```
 #导入OosClient配置文件
import oos_sample_conf

#导入OOS相关模块
from baidubce import exception
from baidubce.services import oos
from baidubce.services.oos.oos_client import OosClient

#新建OosClient
oos_client = OosClient(oos_sample_conf.config)
```

创建模板接口

接口描述

根据需要自定义模板要执行的任务并创建模板。

请求参数

名称	类型	描述	是否必须	参数位置
name	String	模板名称	是	Body体参数
operators	List< OperatorModel >	模板包含的任务列表	是	Body体参数
description	String	模板描述	否	Body体参数

OperatorModel参数

名称	类型	是否必须	描述
name	String	是	任务名称
operator	String	是	任务类型
properties	Map<String, Object>	是	任务执行所需参数
retries	int	否	重试次数，默认不重试
retry_interval	int	否	重试时间间隔，单位：毫秒，默认间隔60秒
timeout	int	否	任务执行超时时间，单位：毫秒，默认超时时间1小时
description	String	否	任务描述

响应参数

名称	类型	描述
id	String	模板ID，全局唯一

请求示例

```
 # params definition
property1 = {
    "instances": [
        {
            "instanceId": "i-khWMkT**"
        }
    ]
}
operator1 = oos_model.OperatorModel("stop_bcc", "BCE::BCC::StopInstance", property1)

# create a oos client
```

```
    oos_client = OosClient(oos_sample_conf.config)

    # create template
    response = oos_client.create_template("test_template_01", [operator1])
    print response
```

创建执行接口

接口描述

可以通过如下两种方式创建任务执行：
* 引用一个已存在的模板（参考请求示例1）
* 在创建执行时动态创建一个模板（参考请求示例2）

请求参数

名称	类型	描述	是否必须	参数位置
template	TemplateModel	要执行的模板	是	Body体参数
properties	Map<String, Object>	模板执行所需参数	否	Body体参数
description	String	执行描述	否	Body体参数
tags	List< TagModel >	执行绑定标签列表	否	Body体参数

TemplateModel参数

名称	类型	是否必须	描述
name	String	是	模板名称
ref	String	否，引用模板创建执行时必填	模板ID
operators	List< OperatorModel >	是，引用模板创建执行时可不填	模板包含任务列表

OperatorModel参数

名称	类型	是否必须	描述
name	String	是	任务名称
operator	String	是	任务类型
properties	Map<String, Object>	是	任务执行所需参数
retries	int	否	重试次数，默认不重试
retry_interval	int	否	重试时间间隔，单位：毫秒，默认间隔60秒
timeout	int	否	任务执行超时时间，单位：毫秒，默认超时时间1小时
description	String	否	任务描述

TagModel参数

名称	类型	是否必须	描述
key	String	是	标签key
value	String	是	标签value

响应参数

名称	类型	描述
id	String	任务执行ID

请求示例1
```python
    # params definition
    template1 = oos_model.TemplateModel("fake_template", ref="tpl-0YIvgR**")

    # create a oos client
    oos_client = OosClient(oos_sample_conf.config)

    # create execution
    response = oos_client.create_execution(template1)
    print response
```

请求示例2

```
     # params definition
    property1 = {
        "content": "ls /",
        "user": "root",
        "workDir": "/",
        "__workerSelectors__": [
            {
                "instanceId": "i-VZIunE**"
            }
        ]
    }
    operator1 = oos_model.OperatorModel("run_script", "BCE::Agent::ExecuteShell", property1)
    template1 = oos_model.TemplateModel("test1", [operator1])

    # create a oos client
    oos_client = OosClient(oos_sample_conf.config)

    # create execution
    response = oos_client.create_execution(template1)
    print response


# Go-SDK

## 概述

##### 概述
本文档主要介绍OOS Go-SDK的安装和使用。

在阅读本文档前，您需要先了解OOS的基本知识。若您还不了解OOS，可以参考产品描述和操作指南


## 安装SDK工具包

 ## 运行环境
GO SDK可以在go1.3及以上环境下运行。
##### 安装SDK
直接从github下载

使用go get工具从github进行下载：
```shell
go get github.com/baidubce/bce-sdk-go
```

SDK目录结构
```

```
 bce-sdk-go
|--auth                 //BCE签名和权限认证
|--bce                  //BCE公用基础组件
|--http                 //BCE的http通信模块
|--services              //BCE相关服务目录
| |--oos                //OOS服务目录
| | |--model             //OOS相关API的数据模型目录
| | |--client.go         //OOS客户端入口
| | |--oos.go            //OOS相关API实现
| |--sts                //STS服务目录
|--util                 //BCE公用的工具实现
```

卸载SDK

预期卸载SDK时，删除下载的源码即可。

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区域进行操作，从而在配置OosClient时将区域对应的Endpoint做为参数填入。

OOS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。

OOSClient

新建OOS Client OOS Client是OOS控制面服务的客户端，为开发者与OOS控制面服务进行交互提供了一系列的方法。

使用AK/SK新建OOS Client 通过AK/SK方式访问OOS，用户可以参考如下代码新建一个OOS Client：

```
import (
    "github.com/baidubce/bce-sdk-go/services/oos"   //导入OOS服务模块
)

func main() {
    // 用户的Access Key ID和Secret Access Key
    ACCESS_KEY_ID, SECRET_ACCESS_KEY := <your-access-key-id>, <your-secret-access-key>

    // 用户指定的Endpoint
    ENDPOINT := <domain-name>

    // 初始化一个OosClient
    oosClient, err := oos.NewClient(AK, SK, ENDPOINT)
}
```

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。

模板任务（Operator）接口

模板任务（Operator）列表

请求参数

名称	类型	描述	是否必须	参数位置
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

请求示例

```
POST /api/logic/oos/v1/operator/list
req := &model.BasePageRequest{
    PageNo:   1,
    PageSize:  100,
}
result, err := oosClient.GetOperatorList(req)
```

响应示例

```
{
    "success": true,
    "msg": "",
    "result": {
        "operators": [
            {
                "name": "BCE::Agent::ExecuteHttp",
                "label": "虚机执行Http请求",
                "description": "虚机执行Http请求",
                "operator": "BCE::Agent::ExecuteHttp",
                "retries": 0,
                "retryInterval": 300000,
                "timeout": 21600000,
                "parallelismRatio": 0.0,
                "parallelismCount": 0,
                "allowedFailureRatio": 0.0,
                "allowedFailureCount": 0,
                "manually": false,
                "scheduleDelayMilli": 0,
```

```
        "pauseOnFailure": false,
        "properties": [
            {
                "name": "method",
                "required": true,
                "type": "string",
                "label": "请求方法类型",
                "multiple": false,
                "description": "",
                "options": [
                    "GET",
                    "POST",
                    "PUT",
                    "DELETE"
                ],
                "defaultValue": "",
                "properties": []
            },
            {
                "name": "url",
                "required": true,
                "type": "string",
                "label": "请求地址",
                "multiple": false,
                "description": "",
                "options": [],
                "defaultValue": "",
                "properties": []
            },
            {
                "name": "headers",
                "required": false,
                "type": "list",
                "label": "请求头",
                "multiple": false,
                "description": "",
                "options": [],
                "defaultValue": "",
                "properties": []
            },
            {
                "name": "body",
                "required": false,
                "type": "list",
                "label": "请求体",
                "multiple": false,
                "description": "",
                "options": [],
                "defaultValue": "",
                "properties": []
            },
            {
                "name": "timeoutMill",
                "required": false,
                "type": "number",
                "label": "请求连接超时时间",
                "multiple": false,
                "description": "",
                "options": [],
                "defaultValue": 10000,
                "unit": "ms",
                "properties": []
```

```
                },
                {
                    "name": "__workerSelectors__",
                    "required": true,
                    "type": "bccInstance",
                    "label": "执行脚本虚机列表",
                    "multiple": true,
                    "description": "",
                    "options": [],
                    "defaultValue": "",
                    "properties": []
                }

            ],
            "initContext": {}
        }
        // 省略剩余
    ],
    "orderBy": "createTime",
    "order": "desc",
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 39
  }
}
```

运维模板（Template）接口

创建运维模板

接口描述

您可以在模板中指定执行所需参数（参考请求示例1），也可以通过全局参数的形式在创建执行的时候再指定（参考请求实例2）

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

响应参数

名称	类型	描述
id	String	模板ID，全局唯一

请求示例 1

```
 req := &model.Template{
Name:        "test_go_sdk",
Description: "des",
Tags:        nil,
Operators: []*model.Operator{
 {
  Name:        "start_bcc",
  Description: "启动BCC实例",
  Operator:    "BCE::BCC::StartInstance",
  Properties: map[string]interface{}{
   "instances": []map[string]string{
    {
     "instanceId": "i-xxxxxx",
     "name":       "instance-xxxxx",
    },
   },
  },
 },
},
Properties: []*model.Property{
 {
  Name:     "test_param",
  Type:     "string",
  Required: false,
 },
},
Linear: true,
}
result, err := oosClient.CreateTemplate(req)
```

响应示例 1

```
 {
   "success": true,
   "msg": "",
   "code": 200,
   "result": {
       "id": "tpl-******DK"
   }
}
```

请求示例 2

```
 req := &model.Template{
Name:        "test_go_sdk_2",
Description: "des",
Tags:        nil,
Operators: []*model.Operator{
 {
  Name:        "start_bcc",
  Description: "启动BCC实例",
  Operator:    "BCE::BCC::StartInstance",
  Properties: map[string]interface{}{
   "instances": []map[string]string{
    {
     "Ref":"test_param",
    },
   },
  },
 },
},
Properties: []*model.Property{
 {
  Name:     "test_param",
  Type:     "bccInstance",
  Required: true,
  DefaultValue: map[string]interface{}{
   "instanceId": "i-xxxxx",
  },
 },
},
Linear: true,
}
result, err := oosClient.CreateTemplate(req)
```

响应示例 2

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
     "id": "tpl-******"
  }
}
```

校验运维模板

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例

```
 req := &model.Template{
Name:        "test_go_sdk",
Description: "des",
Tags:        nil,
Operators: []*model.Operator{
 {
  Name:        "start_bcc",
  Description: "启动BCC实例",
  Operator:    "BCE::BCC::StartInstance",
  Properties: map[string]interface{}{
   "instances": []map[string]string{
    {
     "instanceId": "i-xxxxxx",
     "name":       "instance-xxxxx",
    },
   },
  },
 },
},
Properties: []*model.Property{
 {
  Name:     "test_param",
  Type:     "string",
  Required: false,
 },
},
Linear: true,
}
result, err := oosClient.CheckTemplate(req)
```

响应示例

成功响应示例

无

失败响应示例

```
[Code: InvalidRequest; Message: template is invalid: op name is required; RequestId: eb******]
```

更新运维模板

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例

```
 req := &model.Template{
ID:          "tpl-26ENudO2",
Name:        "test_go_sdk_2",
Description: "des",
Tags:        nil,
Operators: []*model.Operator{
 {
  Name:        "start_bcc",
  Description: "启动BCC实例",
  Operator:    "BCE::BCC::StartInstance",
  Properties: map[string]interface{}{
   "instances": []map[string]string{
    {
     "Ref": "test_param",
    },
    {
     "instanceId": "i-lQwXU07Z",
     "name":       "instance-3nd0wnlr",
    },
   },
  },
 },
},
Properties: []*model.Property{
 {
  Name:     "test_param",
  Type:     "bccInstance",
  Required: true,
  DefaultValue: map[string]interface{}{
   "instanceId": "i-lQwXU07Z",
  },
 },
},
Linear: true,
}
result, err := oosClient.UpdateTemplate(req)
```

删除运维模板

请求参数

名称	类型	描述	是否必须	参数位置
id	String	运维模板ID	是	Query参数

请求示例

```
result, err := oosClient.DeleteTemplate("tpl-6UUp6jpj")
```

查看运维模板

请求参数

名称	类型	描述	是否必须	参数位置
name	String	运维模板名称	是	Query参数
templateType	String	运维模板类型：GLOBAL-全局模板；INDIVIDUAL-个人模板	是	Query参数

请求示例

```
result, err := oosClient.GetTemplateDetail("test_go_sdk", string(model.TemplateTypeIndividual))
```

响应示例

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
     "id": "tpl-******",
     "name": "test_go_sdk",
     "description": "",
     "linear": true,
     "operators": [
        {
           "name": "start_bcc",
           "description": "",
           "operator": "BCE::BCC::StopInstance",
           "retries": 0,
           "retryInterval": 60000,
           "timeout": 3600000,
           "parallelismControl": {
              "ratio": 0.0,
              "count": 0
           },
           "allowedFailureControl": {
              "ratio": 0.0,
              "count": 0
           },
           "manually": false,
           "scheduleDelayMilli": 0,
           "pauseOnFailure": false,
           "properties": {
              "instances": [
                 {
                    "instanceId": "i-******"
                 }
              ]
           },
           "initContext": {}
        }
     ]
  }
}

}
```

查看运维模板列表

请求参数

名称	类型	描述	是否必须	参数位置
sort	String	排序字段，默认为创建时间	否	RequestBody参数
ascending	boolean	是否升序，默认false	否	RequestBody参数
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

请求示例

```
 req := &model.GetTemplateListRequest{
BasePageRequest: model.BasePageRequest{
 Ascending: false,
 PageNo:    1,
 PageSize:  100,
},
}
result, err := oosClient.GetTemplateList(req)
```

响应示例

```
{
  "success": true,
  "msg": "",
  "result": {
    "templates": [
      {
        "id": "tpl-******",
        "name": "test_go_sdk",
        "type": "INDIVIDUAL",
        "description": "",
        "linear": true,
        "operators": [
          {
            "name": "start_bcc",
            "description": "",
            "operator": "BCE::BCC::StopInstance",
            "retries": 0,
            "retryInterval": 60000,
            "timeout": 3600000,
            "parallelismControl": {
              "ratio": 0.0,
              "count": 0
            },
            "allowedFailureControl": {
              "ratio": 0.0,
              "count": 0
            },
            "manually": false,
            "scheduleDelayMilli": 0,
            "pauseOnFailure": false,
            "properties": {
              "instances": [
                {
                  "instanceId": "i-******Pi"
                }
              ]
            },
            "initContext": {}
          }
        ]
      }
      //省略剩余的
    ],
    "orderBy": "createTime",
    "order": "desc",
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 126
  }
}
```


任务执行（Execution）接口

创建任务执行

接口描述

可以通过如下两种方式创建任务执行：

1. 引用一个已存在的模板（参考请求示例1）

2. 在创建执行时动态创建一个模板（参考请求示例2）

请求参数

名称	类型	描述	是否必须	参数位置
execution	Execution	任务执行	是	RequestBody参数

请求示例1

```
 req := &model.Execution{
  Description: "创建执行测试1",
  Template: &model.Template{
        Name:   "test"
   Ref:    "tpl-n******",
   Linear: true,
  },
 }
result, err := oosClient.CreateExecution(req)
```

请求示例2

```
 req := &model.Execution{
   Description: "创建执行测试2",
   Template: &model.Template{
      Name: "test_template_02",
      Operators: []*model.Operator{
         {
            Name:          "stop_bcc",
            Description:   "停止BCC实例",
            Operator:      "BCE::BCC::StopInstance",
            Retries:       3,
            RetryInterval: 60000,
            Timeout:       3600000,
            ParallelismControl: &model.RateControl{
               Ratio: 0,
               Count: 1,
            },
            Properties: map[string]interface{}{
               "instances": []map[string]string{
                  {
                     "instanceId": "i-jOS24U0I",
                     "name":       "instance-3nd0wnlr",
                  },
               },
            },
         },
      },
      Linear: true,
   },
}
result, err := oosClient.CreateExecution(req)
```

响应示例

```
{
  "success":true,
  "msg":"",
  "code":200,
  "result":{
     // 任务执行ID
     "id":"d-C******"
  }
}
```

查看任务执行

请求参数

名称	类型	描述	是否必须	参数位置
id	String	任务执行ID	否	Query参数

请求示例

```
result, err := oosClient.GetExecutionDetail("d-*********")
```

响应示例

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
     "id": "d-*********",
     "template": {
        "name": "test_oos_template",
        "type": "INDIVIDUAL",
        "tags": [],
        "linear": true,
        "links": [],
        "operators": [
           {
              "name": "stop_bcc",
              "operator": "BCE::BCC::StopInstance",
              "retries": 0,
              "retryInterval": 60000,
              "timeout": 3600000,
              "manually": false,
              "scheduleDelayMilli": 0,
              "pauseOnFailure": false,
              "properties": {
                 "instanceId": {
                    "Ref": "instanceId"
                 },
                 "instances": [
                    {
                       "instanceId": "i-******kK"
                    }
                 ]
              },
              "initContext": {}
           }
        ],
        "properties": []
```

```
        },
        "createdTimestamp": 1713941313268,
        "updatedTimestamp": 1713941326534,
        "finishedTimestamp": 1713941326534,
        "state": "SUCCESS",
        "properties": {},
        "tasks": [
          {
            "id": "t-*****************",
            "revision": 79917077,
            "createdTimestamp": 1713941313269,
            "updatedTimestamp": 1713941326532,
            "finishedTimestamp": 1713941326532,
            "state": "SUCCESS",
            "operator": {
              "name": "stop_bcc",
              "operator": "BCE::BCC::StopInstance",
              "retries": 0,
              "retryInterval": 60000,
              "timeout": 3600000,
              "manually": false,
              "scheduleDelayMilli": 0,
              "pauseOnFailure": false,
              "properties": {},
              "initContext": {
                "instanceId": {
                  "Ref": "instanceId"
                },
                "instances": [
                  {
                    "instanceId": "i-******kK"
                  }
                ]
              }
            },
            "initContext": {
              "instanceId": {
                "Ref": "instanceId"
              },
              "instances": [
                {
                  "instanceId": "i-******kK"
                }
              ]
            },
            "context": {},
            "outputContext": {
              "instances": [
                {
                  "instanceId": "i-******kK"
                }
              ]
            },
            "tries": 0,
            "children": []
          }
        ],
        "tags": [],
        "trigger": "MANUAL"
    }
}
```

Java-SDK

概述

概述

本文档主要介绍OOS Java-SDK的安装和使用。

在阅读本文档前，您需要先了解OOS的基本知识。若您还不了解OOS，可以参考产品描述和操作指南

安装SDK工具包

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
│   └── oos                             //OOS服务相关类
│       ├── model                       //OOS内部model，如Request或Response
│       ├── OosClient.class             //OOS客户端入口类
│       └── OosClientConfiguration.class    //针对OOS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
```

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OosClient时将区域对应的Endpoint做为参数填入。

OOS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。

OosClient

OOS Client是OOS服务的客户端，为开发者与OOS服务进行交互提供了一系列的方法。

新建OosClient

使用AK/SK新建OosClient

通过AK/SK方式访问OOS，用户可以参考如下代码新建一个OosClient：

```
 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
    String ENDPOINT = <domain-name>;                            // 用户自己指定的域名

    // 初始化一个OosClient
    OosClientConfiguration config = new OosClientConfiguration();
    config.setCredentials(new DefaultBceCredentials(ACCESS_KEY_ID, SECRET_ACCESS_KEY));
    config.setEndpoint(ENDPOINT);
    OosClient client = new OosClient(config);
}
}
```

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《操作指南 管理ACCESSKEY》。

配置OosClient

如果用户需要配置 OosClient 的一些细节的参数，可以在构造 OosClient 的时候传入 OosClientConfiguration 对象。OosClientConfiguration 是OOS服务的配置类，可以为客户端配置代理，最大连接数等参数。

设置网络参数

用户可以用 OosClientConfiguration 对基本网络参数进行设置：

```
 OosClientConfiguration config = new OosClientConfiguration();

// 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.setConnectionTimeoutInMillis(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 OosClientConfiguration 能指定的所有参数如下表所示：

参数	说明
CnameEnabled	使用cname访问OOS资源
ConnectionTimeoutInMillis	建立连接的超时时间（单位：毫秒）
Credentials	客户端用于签署HTTP请求的BCE凭据
EnableHttpAsyncPut	异步put
Endpoint	访问域名
LocalAddress	本地地址
MaxConnections	允许打开的最大HTTP连接数
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
ProxyPassword	代理服务器验证的密码
ProxyPort	代理服务器端口
ProxyPreemptiveAuthenticationEnabled	是否设置用户代理认证
ProxyUsername	代理服务器验证的用户名
ProxyWorkstation	NTLM代理服务器的Windows工作站名称
Region	地域
RetryPolicy	连接重试策略
SocketBufferSizeInBytes	Socket缓冲区大小
SocketTimeoutInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）
StreamBufferSize	流文件缓冲区大小
UserAgent	用户代理，指HTTP的User-Agent头
RedirectsEnabled	是否开启HTTP重定向。默认开启

模板任务（Operator）接口

请求结构

- method：POST
- URL：/api/logic/oos/v1/operator/list

请求参数

名称	类型	描述	是否必须	参数位置
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

响应示例

请求示例

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
OperatorListRequest request = new OperatorListRequest();
request.setPageNo(1);
request.setPageSize(10);
// call
OperatorListResponse operatorList = oosClient.getOperatorList(request);
```

响应示例


```
{
  "metadata": {
    "bceRequestId": "64da1e89-4b7c-4d7e-b276-6fa18c5b02dd",
    "contentLength": 5340,
    "contentType": "application/json;charset=UTF-8",
    "date": 1713876907000,
    "server": "BWS"
  },
  "success": true,
  "msg": "",
  "result": {
    "operators": [
      {
        "name": "BCE::Agent::ExecuteHttp",
        "label": "虚机执行Http请求",
        "description": "虚机执行Http请求",
        "operator": "BCE::Agent::ExecuteHttp",
        "retries": 0,
        "retryInterval": 300000,
        "timeout": 21600000,
        "parallelismRatio": 0.0,
        "parallelismCount": 0,
        "allowedFailureRatio": 0.0,
        "allowedFailureCount": 0,
        "manually": false,
        "scheduleDelayMilli": 0,
        "pauseOnFailure": false,
        "properties": [
          {
            "name": "method",
            "required": true,
            "type": "string",
            "label": "请求方法类型",
            "multiple": false,
            "description": "",
            "options": [
              "GET",
              "POST",
              "PUT",
              "DELETE"
            ],
            "defaultValue": "",
```

```
            "properties": []
        },
        {
            "name": "url",
            "required": true,
            "type": "string",
            "label": "请求地址",
            "multiple": false,
            "description": "",
            "options": [],
            "defaultValue": "",
            "properties": []
        },
        {
            "name": "headers",
            "required": false,
            "type": "list",
            "label": "请求头",
            "multiple": false,
            "description": "",
            "options": [],
            "defaultValue": "",
            "properties": []
        },
        {
            "name": "body",
            "required": false,
            "type": "list",
            "label": "请求体",
            "multiple": false,
            "description": "",
            "options": [],
            "defaultValue": "",
            "properties": []
        },
        {
            "name": "timeoutMill",
            "required": false,
            "type": "number",
            "label": "请求连接超时时间",
            "multiple": false,
            "description": "",
            "options": [],
            "defaultValue": 10000,
            "unit": "ms",
            "properties": []
        },
        {
            "name": "__workerSelectors__",
            "required": true,
            "type": "bccInstance",
            "label": "执行脚本虚机列表",
            "multiple": true,
            "description": "",
            "options": [],
            "defaultValue": "",
            "properties": []
        }

    ],
    "initContext": {}
}
```

```
            // 省略剩余
        ],
        "orderBy": "createTime",
        "order": "desc",
        "pageNo": 1,
        "pageSize": 10,
        "totalCount": 39
    }
}
```

运维模板（Template）接口

创建运维模板

接口描述

您可以在模板中指定执行所需参数（参考请求示例1），也可以通过全局参数的形式在创建执行的时候再指定（参考请求实例2）

请求结构

- method：POST
- URL：/api/logic/oos/v2/template

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例 1

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseTemplateRequest request = new BaseTemplateRequest();
request.setName("test_oos_template4");
request.setLinear(true);
Operator stopBcc = new Operator();
stopBcc.setName("stop_bcc");
stopBcc.setOperator("BCE::BCC::StopInstance");
stopBcc.setRetries(0);
stopBcc.setRetryInterval(60000);
stopBcc.setTimeout(3600000);

Map<String, Object> properties = new HashMap();
properties.put("instances", Collections.singletonList(
        Collections.singletonMap("instanceId", "i-******Pi") // 待关机的实例 Id
));
stopBcc.setProperties(properties);
List<Operator> operators = new ArrayList();
operators.add(stopBcc);
request.setOperators(operators);
// call
BaseTemplateResponse template = oosClient.createTemplate(request);
```

响应示例 1

```
 {
    "success": true,
    "msg": "",
    "code": 200,
    "result": {
        "id": "tpl-******DK"
    }
}
```

请求示例 2

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseTemplateRequest request = new BaseTemplateRequest();
request.setName("test_oos_template_param");
request.setLinear(true);
Operator stopBcc = new Operator();
stopBcc.setName("stop_bcc");
stopBcc.setOperator("BCE::BCC::StopInstance");
stopBcc.setRetries(0);
stopBcc.setRetryInterval(60000);
stopBcc.setTimeout(3600000);

Map<String, Object> properties = new HashMap();
// 引用全局参数
properties.put("instances", Collections.singletonList(
        Collections.singletonMap("Ref", "instanceId")
));
stopBcc.setProperties(properties);
List<Operator> operators = new ArrayList();
operators.add(stopBcc);
request.setOperators(operators);
// 设置全局参数
Property instanceId = new Property();
instanceId.setName("instanceId");
instanceId.setRequired(true);
instanceId.setType("bccInstance");
request.setProperties(Collections.singletonList(instanceId));
// call
BaseTemplateResponse template = oosClient.createTemplate(request);
```

响应示例 2

```
 {
    "success": true,
    "msg": "",
    "code": 200,
    "result": {
        "id": "tpl-******DK"
    }
}
```

校验运维模板

接口描述

用于校验模板是否符合规则

请求结构

- method：POST

- URL：/api/logic/oos/v2/template/check

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数

请求示例

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseTemplateRequest request = new BaseTemplateRequest();
request.setName("test_oos_template_param");
request.setLinear(true);
Operator stopBcc = new Operator();
stopBcc.setName("stop_bcc");
stopBcc.setOperator("BCE::BCC::StopInstance");
stopBcc.setRetries(0);
stopBcc.setRetryInterval(60000);
stopBcc.setTimeout(3600000);

Map<String, Object> properties = new HashMap();
properties.put("instances", Collections.singletonList(
        Collections.singletonMap("Ref", "instanceId")
));
stopBcc.setProperties(properties);
List<Operator> operators = new ArrayList();
operators.add(stopBcc);
request.setOperators(operators);
Property instanceId = new Property();
instanceId.setName("instanceId");
instanceId.setRequired(true);
instanceId.setType("bccInstance");
request.setProperties(Collections.singletonList(instanceId));

// call
CheckTemplateResponse checkTemplateResponse = oosClient.checkTemplate(request);
```

响应示例

```
 {
   "success": true,
   "msg": "",
   "code": 200
}
```

更新运维模板

接口描述

用于更新运维模板，注意改接口只能够全量更新

请求结构

- method：PUT
- URL：/api/logic/oos/v2/template

请求参数

名称	类型	描述	是否必须	参数位置
Template	Template	运维模板	是	RequestBody参数
id	String	运维模板ID	是	RequestBody参数

请求示例

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseTemplateRequest request = new BaseTemplateRequest();
request.setName("test_oos_template_param");
request.setLinear(true);
Operator stopBcc = new Operator();
stopBcc.setName("stop_bcc_update");
stopBcc.setOperator("BCE::BCC::StopInstance");
stopBcc.setRetries(0);
stopBcc.setRetryInterval(60000);
stopBcc.setTimeout(3600000);

Map<String, Object> properties = new HashMap();
properties.put("instances", Collections.singletonList(
        Collections.singletonMap("Ref", "instanceId")
));
stopBcc.setProperties(properties);
List<Operator> operators = new ArrayList();
operators.add(stopBcc);
request.setOperators(operators);
Property instanceId = new Property();
instanceId.setName("instanceId");
instanceId.setRequired(true);
instanceId.setType("bccInstance");
request.setProperties(Collections.singletonList(instanceId));

request.setId("tpl-******wE");
// call
BaseTemplateResponse baseTemplateResponse = oosClient.updateTemplate(request);
```

响应示例

```
{
  "success": true,
  "msg": "",
  "code": 200
}
```

删除运维模板

接口描述

用于删除运维模板

请求结构

- method：DELETE
- URL：/api/logic/oos/v2/template?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
id	String	运维模板ID	是	Query参数

请求示例

```
// config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********ce4";
String ak = "ALTA**********CYG";
String sk = "b2c5**********3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
String templateId = "tpl-******wE";
// call
BaseTemplateResponse baseTemplateResponse = oosClient.deleteTemplate(templateId);
```

响应示例

```
{
  "success": true,
  "msg": "",
  "code": 200
}
```

查看运维模板

接口描述

用于查看运维模板详情

请求结构

- method：GET

- URL：/api/logic/oos/v2/template?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
name	String	运维模板名称	是	Query参数
type	String	运维模板类型，包括GLOBAL（系统）、INDIVIDUAL（个人）	是	Query参数

请求示例

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
String templateName = "test_oos_template4";
TemplateType templateType = TemplateType.INDIVIDUAL;
// call
BaseTemplateResponse baseTemplateResponse = oosClient.getTemplateDetail(templateName, templateType);
```

响应示例

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
    "id": "tpl-******nP",
    "name": "test_oos_template4",
    "description": "",
    "linear": true,
    "operators": [
      {
        "name": "stop_bcc",
        "description": "",
        "operator": "BCE::BCC::StopInstance",
        "retries": 0,
        "retryInterval": 60000,
        "timeout": 3600000,
        "parallelismControl": {
          "ratio": 0.0,
          "count": 0
        },
        "allowedFailureControl": {
          "ratio": 0.0,
          "count": 0
        },
        "manually": false,
        "scheduleDelayMilli": 0,
        "pauseOnFailure": false,
        "properties": {
          "instances": [
            {
              "instanceId": "i-******Pi"
            }
          ]
        },
        "initContext": {}
      }
    ]
  }
}
```


查看运维模板列表

接口描述

用于查看运维模板列表

请求结构

- method：POST
- URL：/api/logic/oos/v2/template/list

请求参数

名称	类型	描述	是否必须	参数位置
sort	String	排序字段，默认为创建时间	否	RequestBody参数
ascending	boolean	是否升序，默认false	否	RequestBody参数
pageNo	int	页数，从1开始计数	是	RequestBody参数
pageSize	int	每页展示数量，最大值：100	是	RequestBody参数

请求示例

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********ce4";
String ak = "ALTA**********CYG";
String sk = "b2c5**********3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
TemplateListRequest request = new TemplateListRequest();
request.setSort("createTime");
request.setAscending(false);
request.setPageNo(1);
request.setPageSize(10);
// call
TemplateListResponse templateList = oosClient.getTemplateList(request);
```

响应示例

```
{
  "success": true,
  "msg": "",
  "result": {
    "templates": [
      {
        "id": "tpl-******nP",
        "name": "test_oos_template4",
        "type": "INDIVIDUAL",
        "description": "",
        "linear": true,
        "operators": [
          {
            "name": "stop_bcc",
            "description": "",
            "operator": "BCE::BCC::StopInstance",
            "retries": 0,
            "retryInterval": 60000,
            "timeout": 3600000,
            "parallelismControl": {
              "ratio": 0.0,
              "count": 0
            },
            "allowedFailureControl": {
              "ratio": 0.0,
              "count": 0
            },
            "manually": false,
            "scheduleDelayMilli": 0,
            "pauseOnFailure": false,
            "properties": {
              "instances": [
                {
                  "instanceId": "i-******Pi"
                }
              ]
            },
            "initContext": {}
          }
        ]
      }
      //省略剩余的
    ],
    "orderBy": "createTime",
    "order": "desc",
    "pageNo": 1,
    "pageSize": 10,
    "totalCount": 126
  }
}
```


任务执行（Execution）接口

创建任务执行

接口描述

可以通过如下两种方式创建任务执行：

- 引用一个已存在的模板（参考请求示例1）

- 在创建执行时动态创建一个模板（参考请求示例2）

请求结构

- method：POST
- URL：/api/logic/oos/v2/execution

请求参数

名称	类型	描述	是否必须	参数位置
execution	Execution	任务执行	是	RequestBody参数

请求示例 1

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseExecutionRequest request = new BaseExecutionRequest();
Template template = new Template();
template.setName("test_oos_template4");
template.setRef("tpl-******nP");
template.setLinear(true);
request.setTemplate(template);

// call
BaseExecutionResponse execution = oosClient.createExecution(request);
```

响应示例 1

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
    "id": "d-******UMgb1M"
  }
}
```

请求示例 2

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
BaseExecutionRequest request = new BaseExecutionRequest();
Template template = new Template();
template.setName("test_oos_template");
template.setLinear(true);
Operator stopBcc = new Operator();
stopBcc.setName("stop_bcc");
stopBcc.setOperator("BCE::BCC::StopInstance");
stopBcc.setRetries(0);
stopBcc.setRetryInterval(60000);
stopBcc.setTimeout(3600000);

Map<String, Object> properties = new HashMap<>();
properties.put("instances", Collections.singletonList(
        Collections.singletonMap("instanceId", "i-******kK") // 待关机的实例 Id
));
stopBcc.setProperties(properties);
template.setOperators(Collections.singletonList(stopBcc));
request.setTemplate(template);
// call
BaseExecutionResponse execution = oosClient.createExecution(request);
```

响应示例 2

```
 {
   "success": true,
   "msg": "",
   "code": 200,
   "result": {
      "id": "d-******UMgb1M"
   }
}
```

查看任务执行

接口描述

查看任务执行的详情

请求结构

- method：GET
- URL：/api/logic/oos/v2/execution?{Query参数}

请求参数

名称	类型	描述	是否必须	参数位置
id	String	任务执行ID	否	Query参数

请求示例 1

```
 // config of client
String endpoint = "http://oos.bj.baidubce.com";
String userId = "a0d04**********************ce4";
String ak = "ALTA******************CYG";
String sk = "b2c5**********************3ac1";
// create oos client
OosClientConfiguration config = new OosClientConfiguration();
config.setCredentials(new DefaultBceCredentials(ak, sk));
config.setEndpoint(endpoint);
OosClient oosClient = new OosClient(config);
// request params definition
String executionId = "d-*********34r";
 // call
BaseExecutionResponse execution = oosClient.getExecutionDetail(executionId);
```

响应示例 1

```
{
  "success": true,
  "msg": "",
  "code": 200,
  "result": {
    "id": "d-*********34r",
    "template": {
      "name": "test_oos_template",
      "type": "INDIVIDUAL",
      "tags": [],
      "linear": true,
      "links": [],
      "operators": [
        {
          "name": "stop_bcc",
          "operator": "BCE::BCC::StopInstance",
          "retries": 0,
          "retryInterval": 60000,
          "timeout": 3600000,
          "manually": false,
          "scheduleDelayMilli": 0,
          "pauseOnFailure": false,
          "properties": {
            "instanceId": {
              "Ref": "instanceId"
            },
            "instances": [
              {
                "instanceId": "i-******kK"
              }
            ]
          },
          "initContext": {}
        }
      ],
      "properties": []
    },
    "createdTimestamp": 1713941313268,
```

```
        "updatedTimestamp": 1713941326534,
        "finishedTimestamp": 1713941326534,
        "state": "SUCCESS",
        "properties": {},
        "tasks": [
          {
            "id": "t-*****************dMGNa",
            "revision": 79917077,
            "createdTimestamp": 1713941313269,
            "updatedTimestamp": 1713941326532,
            "finishedTimestamp": 1713941326532,
            "state": "SUCCESS",
            "operator": {
              "name": "stop_bcc",
              "operator": "BCE::BCC::StopInstance",
              "retries": 0,
              "retryInterval": 60000,
              "timeout": 3600000,
              "manually": false,
              "scheduleDelayMilli": 0,
              "pauseOnFailure": false,
              "properties": {},
              "initContext": {
                "instanceId": {
                  "Ref": "instanceId"
                },
                "instances": [
                  {
                    "instanceId": "i-******kK"
                  }
                ]
              }
            },
            "initContext": {
              "instanceId": {
                "Ref": "instanceId"
              },
              "instances": [
                {
                  "instanceId": "i-******kK"
                }
              ]
            },
            "context": {},
            "outputContext": {
              "instances": [
                {
                  "instanceId": "i-******kK"
                }
              ]
            },
            "tries": 0,
            "children": []
          }
        ],
        "tags": [],
        "trigger": "MANUAL"
    }
}
```