ALIBABA CLOUD

阿里云

运维编排
最佳实践

文档版本：20220630

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.模版规范

2.开源集成

2.1. 使用Terraform操作OOS

Terraform是一种开源工具，用于安全高效地预览，配置和管理云基础架构和资源、构建、变更、和安全有效的版本化管理基础设施的工具，可以管理已存在和流行的服务提供商以及定制的内部解决方案。并且支持操作阿里云部分的OpenApi。

运维编排服务是一个以模板的方式来操作阿里云的OpenApi的一种服务，方便您管理已经购买阿里云的资源，或者自动扩充资源等操作。相信读此文章的大家已经大致了解OOS的基本操作。基于以上所述以下是我们为您提供一个简单的教程来展示通过Terraform来操作OOS服务。

安装并配置Terraform

1. 前往Terraform官网，下载适用于您的操作系统的程序包。本文以Linux系统为例。如果您还没有Linux环境，可购买阿里云ECS实例，详情请参见Linux系统实例快速入门。
2. 将程序包解压到/usr/local/bin目录。如果您需要将可执行文件解压到其他目录，请按照以下方法为其定义全局路径：
   - Linux：参见在Linux系统定义全局路径
   - Windows：参见在Windows系统定义全局路径
   - Mac：参见在Mac系统定义全局路径
3. 执行 `terraform` 命令验证路径配置。

```
terraform
Usage: terraform [-version] [-help] <command> [args]
```

4. 安装Golang，在安装以下模块前需要先安装Golang语言。

```
yum install golang
```

5. 安装terraform-provider-alicloud，执行以下命令，或者参考详细操作请参考安装简介文档。

```
yum install git
go get golang.org/x/tools/cmd/goimports
mkdir -p $GOPATH/src/github.com/aliyun
cd $GOPATH/src/github.com/aliyun
git clone https://github.com/aliyun/terraform-provider-alicloud
cd $GOPATH/src/github.com/aliyun/terraform-provider-alicloud
make build
```

> 注意
>
> 在环境中配置以下变量
>
> export ALICLOUD_ACCESS_KEY=xxx
>
> export ALICLOUD_SECRET_KEY=xxx
>
> export ALICLOUD_REGION=xxx

Terraform支持OOS的资源模块

资源类型	模块	使用场景
date	alicloud_oos_templates	查询模版
	alicloud_oos_executions	查询执行
resource	alicloud_oos_template	创建模版、更新模版
	alicloud_oos_execution	创建执行

Terraform操作OOS模板的创建、更新、查询、删除

1、创建模版

1.1 首先创建一个测试testTerraform文件夹，进入此文件夹，初始化Terraform后，在此文件夹内创建一个createTemplate.tf文件，并进入此文件的编辑模式，如下命令。

```
mkdir testTerraform
cd testTerraform
terraform init
vi createTemplate.tf
```

1.2 在编辑模式下，将以下内容复制进createTemplate.tf文件中。

```
resource "alicloud_oos_template" "example" {
        tags={
          "Created" = "TF",
          "For" = "template Test"
        }
        content= <<EOF
        {
        "FormatVersion": "OOS-2019-06-01",
        "Description": "Update Describe instances of given status",
        "Parameters":{
            "Status":{
                "Type": "String",
                "Description": "(Required) The status of the Ecs instance."
            }
        },
        "Tasks": [
            {
                "Properties" :{
                    "Parameters":{
                        "Status": "{{ Status }}"
                    },
                    "API": "DescribeInstances",
                    "Service": "Ecs"
                },
                "Name": "foo",
                "Action": "ACS::ExecuteApi"
            }]
        }
        EOF
        template_name="terraform-test"
}
```

1.3 保存并退出。

1.4 执行以下命令，使用Terraform创建OOS模板。

```
terraform plan
terraform apply
```

1.5 在OOS控制台查看模板是否创建成功，如下所示根据上面提供的例子发现根据自定义的参数，模板创建成功。

2、更新模版

2.1 在执行创建模板操作后，找上面的已经创建完成的createTemplate.tf文件，并编辑此文件。

```
vi createTemplate.tf
```

2.2 进入编辑模式，将需要修改的文件内容复制到文件内。如下例子：

```
resource "alicloud_oos_template" "example" {
        tags={
          "Created" = "TF",
          "For" = "template Test"
        }
        content= <<EOF
        {
        "FormatVersion": "OOS-2019-06-01",
        "Description": "Update Describe instances of given status",
        "Parameters":{},
        "Tasks": [
            {
                "Properties" :{
                    "Parameters":{
                        "Status": "Running"
                    },
                    "API": "DescribeInstances",
                    "Service": "Ecs"
                },
                "Name": "foo",
                "Action": "ACS::ExecuteApi"
            }]
        }
        EOF
        template_name="terraform-test"
}
```

2.3 将更改的文件输入完毕，保存并退出。

2.4 执行以下命令进行更新资源。

```
terraform plan
terraform apply
```

3、查看模版

3.1 参考此文档，或ListTemplates来查看搜索操作其它具体的参数。

3.2 创建一个dataTest.tf的文件，进入编辑模式。

```
vi dataTest.tf
```

3.3 将以下查看资源的内容复制进入dataTest.tf文件内，保存并退出。

```
data "alicloud_oos_templates" "example" {
  ids=["terraform-test"]
}

output "first_template" {
  value = "${data.alicloud_oos_templates.example.templates.0}"
}
```

3.4 完成上述操作后，输入以下命令来查看资源的具体信息。

```
terraform plan
terraform apply
```

3.5 如下图所示，将上述命令输入完成后，显示的结果为当前资源的具体信息。

4、删除模版

4.1 使用上面的操作命令创建完成模板后，如果想要删除创建的模板，请参考以下命令来删除模板。

```
terraform show
terraform destroy
```

Terraform操作OOS模板的执行

1、创建执行

1.1 在上面提供的创建模版的例子基础上，首先创建一个测试testExecution.tf的文件，并进入编辑模式

```
vi createTemplate.tf
```

1.2 在编辑模式下，将以下内容复制进testExecution.tf文件中。

```
resource "alicloud_oos_execution" "exampleExecution"{
        template_name = "terraform-test"
        description = "From TF Test"
        parameters = <<EOF
          {"Status":"Running"}
        EOF
}
```

1.3 编辑完成后保存并退出。

1.4 执行以下命令，使用Terraform执行OOS已经创建完成的模版。

```
terraform plan
terraform apply
```

1.5 在OOS控制台查看执行是否启动，如下图所示发现执行创建成功。

2、查看执行

2.1 参考此文档，或ListTemplates来查看搜索操作其它具体的参数。

2.2 创建一个dataExecutionTest.tf的文件，进入编辑模式。

```
vi dataTest.tf
```

2.3 将以下查看资源的内容复制进入dataExecutionTest.tf文件内，保存并退出。

```
data "alicloud_oos_executions" "example" {
  ids = ["execution_id"]
  template_name = "name"
}

output "first_execution_id" {
  value = "${data.alicloud_oos_executions.example.executions.0.id}"
}
```

2.4 完成上述操作后，输入以下命令来查看资源的具体信息。

```
terraform plan
terraform apply
```

2.5 如下图所示，将上述命令输入完成后，在您执行命令的控制台显示的结果为当前资源的具体信息。

3、删除执行

3.1 在使用Terraform创建完执行后，如果想要删除OOS上的执行资源，使用以下命令即可。

```
terraform show
terraform destroy
```

2.2. 使用Ansible创建OOS模版与执行

Ansible是一个开源配置管理工具，可以使用它来自动化执行任务，部署应用来实现IT基础架构。

OOS是一个以模板的方式管理阿里云产品来实现自动化运维的一个服务。相信读此教程的大家已经了解OOS的基本功能与使用方法，本教程将指导您如何使用Ansible创建阿里云的OOS运维模版，以及如何通过Ansible执行创建的模板来管理阿里云产品。

教程概览

本教程将创建和执行模版拆分成了不不同的Ansible playbooks，方便您了解如何通过YAML格式声明配置。您可以参考提供简单的完整示例，运行Playbook来创建并执行一个OOS模版。

Ansible包含OOS的四个模块，你可以使用ali_oos_template、ali_oos_template_info、ali_oos_execution、ali_oos_execution_info四个模块来进行以下操作。

模块	使用场景
ali_oos_template	创建模版
	更新模版
	删除模版
ali_oos_template_info	获取模版

ali_oos_execution	开始执行
	取消执行
	删除执行
	审批动作
ali_oos_execution_info	获取执行

前提条件

在执行前需要安装Ansible，操作如下：

1.执行以下命令安装Ansible。

```
sudo yum install ansible
```

更多详细信息，请参见Ansible文档。

2.执行以下命令查看安装的Ansible版本。

```
ansible -version
```

3.执行以下命令安装Ansible阿里云模块。

```
sudo pip install ansible_alicloud
```

4.可选：当Ansible阿里云模块版本过低时，执行以下命令升级阿里云模块的版本。

```
sudo pip install footmark
sudo pip install ansible_alicloud
```

5.执行以下命令配置访问密钥来访问阿里云资源。

```
export ALICLOUD_ACCESS_KEY="your_accesskey"
export ALICLOUD_SECRET_KEY="your_accesskey_secret"
```

关于如何生成访问密钥，请参见创建AccessKey。

OOS的PlayBook

1、创建模版

```
- name: Create oos template
  ali_oos_template:
    alicloud_region: '{{ alicloud_region }}'
    template_name: '{{ template_name }}'
    content: '{{ content }}'
  register: create_template
```

2、获取模版

```
- name: Get oos template
  ali_oos_template_info:
    alicloud_region: '{{ alicloud_region }}'
    name_prefix: '{{ name_prefix }}'
  register: get_template
```

3、删除模版

```
- name: Delete oos template
  ali_oos_template:
    state: absent
    alicloud_region: '{{ alicloud_region }}'
    template_name: '{{ template_name }}'
```

4、更新模版

```
- name: Update oos template
  ali_oos_template:
    content: '{{ content }}'
    alicloud_region: '{{ alicloud_region }}'
    template_name: '{{ template_name }}'
```

5、执行模版

```
- name: Start a execution
  ali_oos_execution:
    alicloud_region: '{{ alicloud_region }}'
    template_name: '{{ template_name }}'
    safety_check: Skip
    parameters:
      Status: '{{ status }}'
  register: start_execution
```

6、取消执行

```
- name: Cancel a execution
  ali_oos_execution:
    state: cancel
    alicloud_region: '{{ alicloud_region }}'
    execution_id: '{{ execution_id }}'
  register: cancel_execution
```

7、获取执行

```
- name: Get executions
  ali_oos_execution_info:
    alicloud_region: '{{ alicloud_region }}'
    name_prefix: '{{ name_prefix }}'
  register: get_executions
```

8、删除执行

```
- name: Delete a execution
  ali_oos_execution:
    state: absent
    alicloud_region: '{{ alicloud_region }}'
    execution_id: '{{ execution_id }}'
  register: delete_execution
```

9、审批动作

```
- name: Notify a execution
  ali_oos_execution:
    state: notify
    notify_type: Approve
    alicloud_region: '{{ alicloud_region }}'
    execution_id: '{{ execution_id }}'
  register: notify_execution
```

> **注意**
>
> 如果有其他的参数需求，参考提供的参数样式，根据OOS提供的简介来添加您的参数。

运行PlayBook操作OOS步骤

以下为提供的一个简单例子。请根据步骤完成以下操作来创建一个模版，以及执行创建的模版。并根据您的实际需求进行参数替换。

1、编写一个alicloud_describe_instances.yml

```
vi alicloud_describe_instances.yml
```

2、在编辑模式下将以下PlayBook复制进alicloud_describe_instances.yml中

```
---
- name: test create template and execute template
  hosts: localhost
  remote_user: root
  tasks:
    - name: Create oos template
      ali_oos_template:
        alicloud_region: 'cn-hangzhou'
        template_name: 'test-ansible-template'
        content: '{"Description": "Example template, describe instances in some status", "FormatVersion": "OOS-2019-06-01", "Parameters": {"Status": {"Description": "(Required) Running or Stopped", "Type": "String"}}, "Tasks": [{"Name": "describeInstances", "Action": "ACS::ExecuteAPI", "Description": {"zh-cn": "desc", "en": "desc-en"}, "Properties": {"Service": "ECS", "API": "DescribeInstances", "Parameters": {"Status": "\{\{ Status \}\}"}}, "Outputs": {"InstanceIds": {"Type": "List", "ValueSelector": "Instances.Instance[].InstanceId"}}}], "Outputs": {"InstanceIds": {"Type": "List", "Value": "\{\{ describeInstances.InstanceIds \}\}"}}}'
      register: create_template
    - name: Describe instances by status
      ali_oos_execution:
        alicloud_region: 'cn-hangzhou'
        template_name: 'test-ansible-template'
        safety_check: Skip
        description: test execution from ansible.
        parameters:
          Status: 'Running'
      register: start_execution
```

3、保存后退出编辑模式

4、运行Ansible PlayBook创建模版并执行。

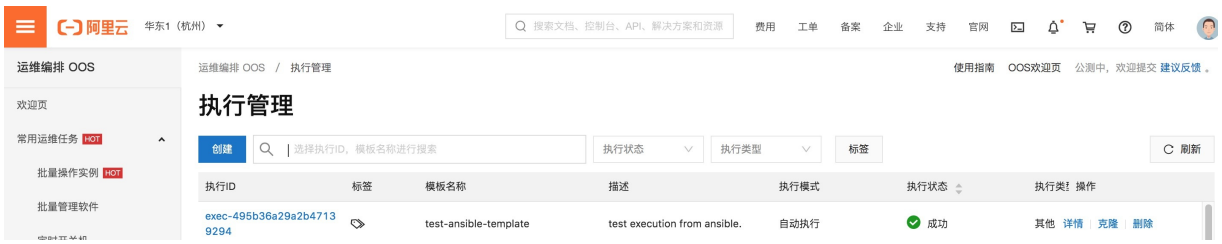
```
ansible-playbook alicloud_describe_instances.yml
```

5. 执行结果

5.1查看Ansible的执行结果，检测Ansible是否执行成功。

```
$ ansible-playbook alicloud_describe_instances.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not
match 'all'

PLAY [test create template and execute template] ***************************************************************

TASK [Gathering Facts] *****************************************************************************************
ok: [localhost]

TASK [Create oos template] *************************************************************************************
changed: [localhost]

TASK [Describe instances by status] ****************************************************************************
changed: [localhost]

PLAY RECAP *****************************************************************************************************
localhost                  : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

5.2 如果命令中的Ansible执行成功， 可以登录OOS控制台，查看使用Ansible执行的task是否实际生效。



2.3. 通过OOS使用Packer更新ECS镜像

环境准备

1. 为目标实例安装并配置AliyunCLI

- 安装CLI

```
wget https://aliyuncli.alicdn.com/aliyun-cli-linux-3.0.60-amd64.tgz

tar -zxvf aliyun-cli-linux-3.0.60-amd64.tgz

mv aliyun /usr/local/bin
```

- 配置CLI，指定profile为：default

```
aliyun configure
```

```
[yincunkundeMacBook-Air:~ yck$ aliyun configure
Configuring profile 'default' in 'AK' authenticate mode...
Access Key Id [*************lU9]:
Access Key Secret [***************************XBD]:
Default Region Id [cn-hangzhou]: cn-shenzhen
Default Output Format [json]: json (Only support json)
Default Language [zh|en] zh:
Saving profile[default] ...Done.

Configure Done!!!
..............88888888888888888888 ........=88888888888888888D=..............
...........888888888888888888888888 ..........D88888888888888888888888I...........
.........,88888888888888ZI: ............................=Z88D88888888888D..........
.........+88888888 .........................................88888888D..........
.........+88888888 .......Welcome to use Alibaba Cloud.......O8888888D..........
.........+88888888 .............. ************* ..............O8888888D..........
.........+88888888 .... Command Line Interface(Reloaded) ....O8888888D..........
.........+88888888.........................................88888888D..........
..........D888888888888DO+. .........................?ND8888888888888D..........
...........O888888888888888888888888.............D888888888888888888888888=...........
.............. .:D888888888888888888.........78888888888888888888O ..............
```

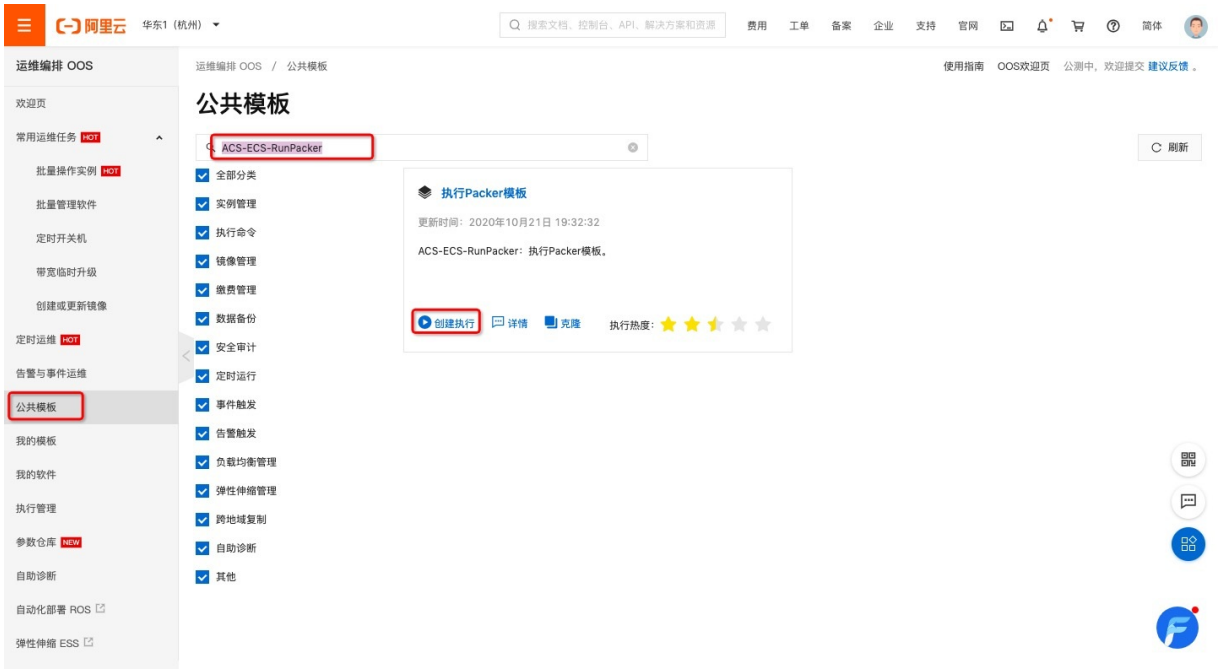
2. 编写packer模板，以更新alicloud镜像为例（配置参数需要根据需求自定义）。

```
{
  "builders": [
    {
      "type": "alicloud-ecs",
      "region": "cn-hangzhou",
      "profile": "default",
      "image_name": "OOS_packer_update_image",
      "source_image": "<source_image_id>",
      "ssh_username": "root",
      "instance_type": "ecs.g6.xlarge",
      "io_optimized": "true",
      "system_disk_mapping": {
          "disk_category": "cloud_essd",
          "disk_size": 40
      },
      "internet_charge_type": "PayByTraffic",
      "image_force_delete": "true"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "inline": ["sleep 30", "yum install redis.x86_64 -y"]
    }
  ]
}
```

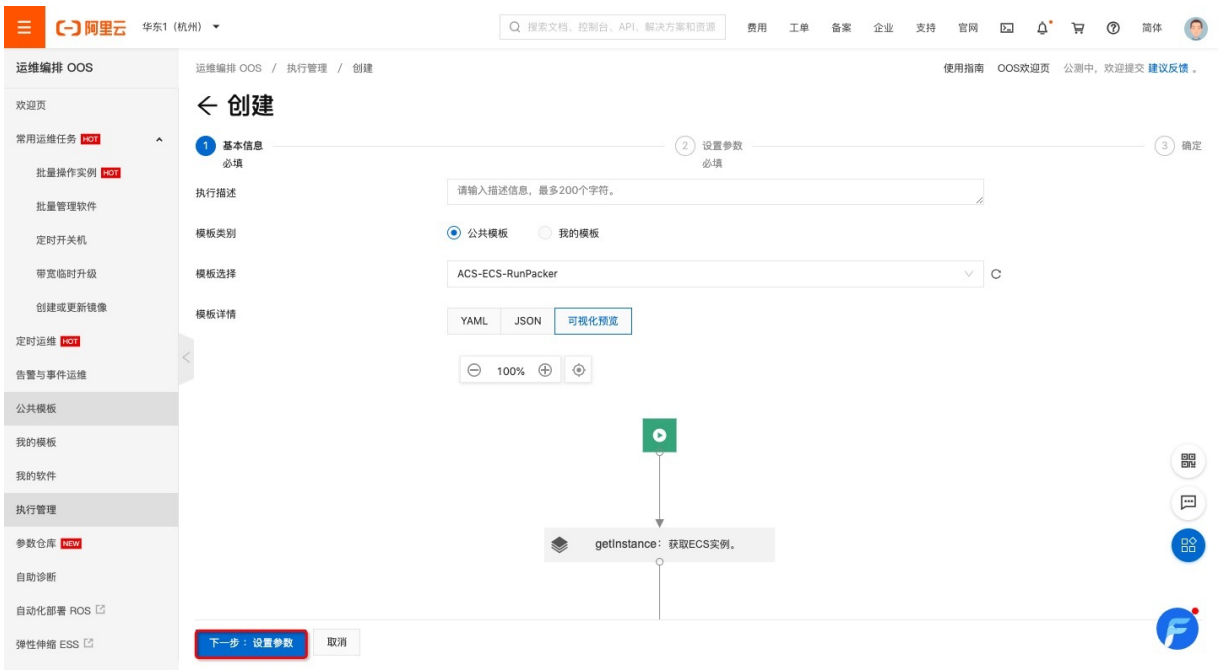
2. 登录OSS（对象存储）控制台，上传packer模板到Bucket。

执行模板

1.登录OOS控制台，点击**公共模板**，搜索**ACS-ECS-RunPacker**，点击**创建执行**。



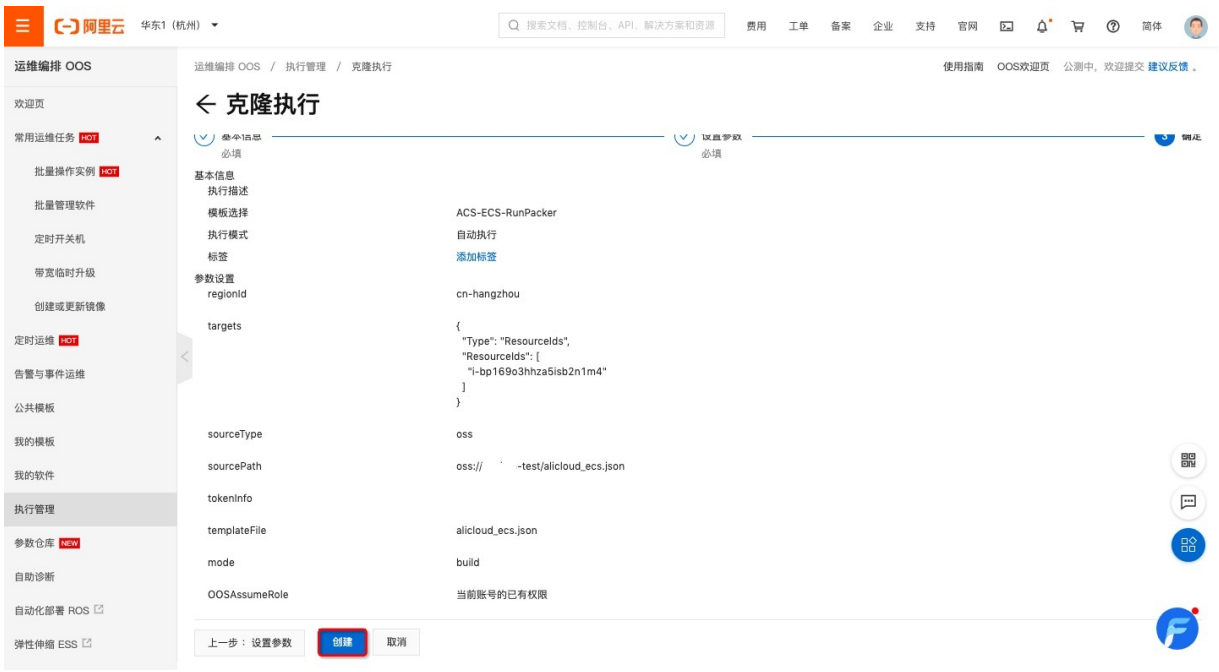
2.点击**下一步：设置参数**。



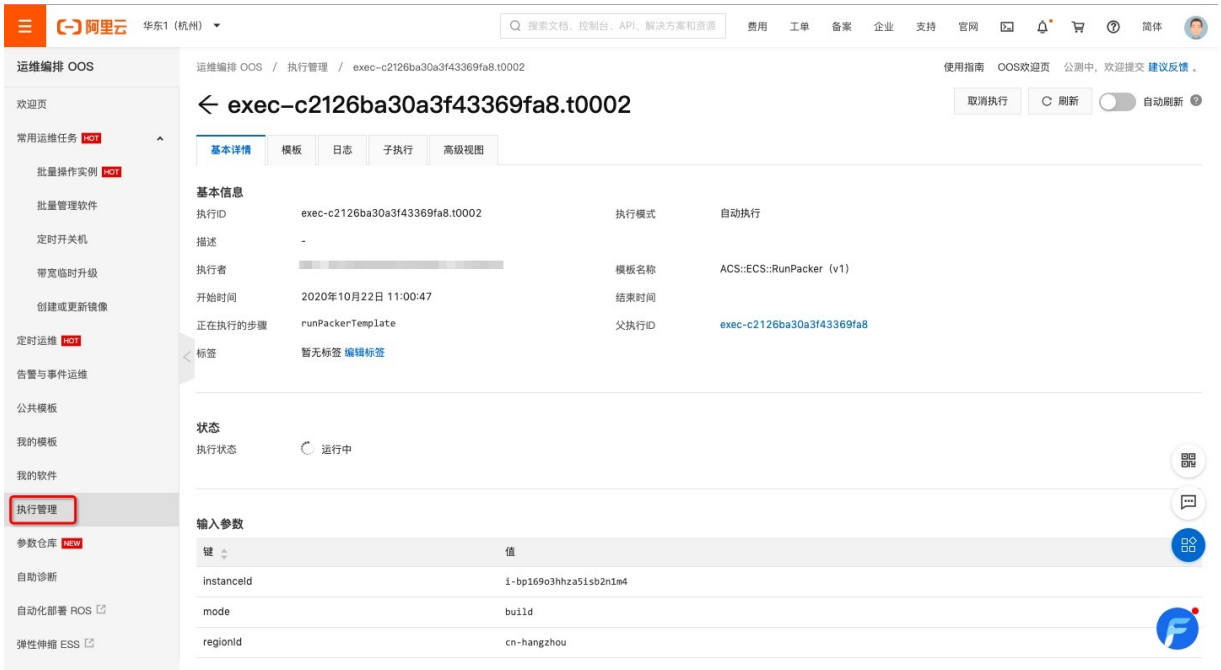
3.选择上文中已配置AliyunCLI的实例（同时所选实例必须配置实例角色，并且实例角色中包含OSS读权限，详情参考批量下载文件到实例中角色配置和授予部分）,点击**下一步：确定**。

- sourceType选择oss
- sourcePath输入上文中模板文件存储的目录
- templateFile输入packer模板的名称
- mode选择build

4.点击**创建**。



5.在**执行详情**中查看模板执行的情况，packer更新镜像过程较耗时，需要耐心等待一段时间。



流程说明

Packer更新AliCloud镜像流程

```
==> alicloud-ecs: Prevalidating source region and copied regions...
==> alicloud-ecs: Force delete flag found, skipping prevalidating image name.
    alicloud-ecs: Found image ID: m-bp12i2mrva8etqnugcxn
==> alicloud-ecs: Creating temporary keypair: packer_5f90fbd1-27f5-3f5f-a7c3-9470b51183e2
==> alicloud-ecs: Creating vpc...
    alicloud-ecs: Created vpc: vpc-bp1m0tkybn06u04v32ryn
==> alicloud-ecs: Creating vswitch...
    alicloud-ecs: Created vswitch: vsw-bp1f67vlm6h72osjr2le5
==> alicloud-ecs: Creating security group...
```

```
==> alicloud-ecs: Creating security group...
    alicloud-ecs: Created security group: sg-bp11tp7gcrgwtqlq9nhf
==> alicloud-ecs: Creating instance...
    alicloud-ecs: Created instance: i-bp16p08isghg3hbcwijx
==> alicloud-ecs: Allocating eip...
    alicloud-ecs: Allocated eip: 47.97.112.43
    alicloud-ecs: Attach keypair packer_5f90fbd1-27f5-3f5f-a7c3-9470b51183e2 to instance: i
-bp16p08isghg3hbcwijx
==> alicloud-ecs: Starting instance: i-bp16p08isghg3hbcwijx
==> alicloud-ecs: Using ssh communicator to connect: 47.97.112.43
==> alicloud-ecs: Waiting for SSH to become available...
==> alicloud-ecs: Connected to SSH!
==> alicloud-ecs: Provisioning with shell script: /tmp/packer-shell177009608
    alicloud-ecs: CentOS-8 - AppStream                              517 kB/s | 4.3 kB     00:
00
    alicloud-ecs: CentOS-8 - AppStream                               38 MB/s | 5.8 MB     00:
00
    alicloud-ecs: CentOS-8 - Base                                   498 kB/s | 3.9 kB     00:
00
    alicloud-ecs: CentOS-8 - Base                                    28 MB/s | 2.2 MB     00:
00
    alicloud-ecs: CentOS-8 - Extras                                 196 kB/s | 1.5 kB     00:
00
    alicloud-ecs: CentOS-8 - Extras                                 247 kB/s | 8.1 kB     00:
00
    alicloud-ecs: Extra Packages for Enterprise Linux 8 - x86_64  614 kB/s | 4.7 kB     00:
00
    alicloud-ecs: Extra Packages for Enterprise Linux 8 - x86_64   80 MB/s | 8.1 MB     00:
00
    alicloud-ecs: Dependencies resolved.
    alicloud-ecs: ================================================================================
=======
    alicloud-ecs:  Package  Arch      Version                                  Repository
Size
    alicloud-ecs: ================================================================================
=======
    alicloud-ecs: Installing:
    alicloud-ecs:  redis    x86_64    5.0.3-2.module_el8.2.0+318+3d7e67ea      AppStream
925 k
    alicloud-ecs: Enabling module streams:
    alicloud-ecs:  redis              5
    alicloud-ecs:
    alicloud-ecs: Transaction Summary
    alicloud-ecs: ================================================================================
=======
    alicloud-ecs: Install  1 Package
    alicloud-ecs:
    alicloud-ecs: Total download size: 925 k
    alicloud-ecs: Installed size: 3.2 M
    alicloud-ecs: Downloading Packages:
    alicloud-ecs: redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_6  11 MB/s | 925 kB     00:
00
    alicloud-ecs: --------------------------------------------------------------------------------
-------
    alicloud-ecs: Total                                             11 MB/s | 925 kB     00:
```

```
00
    alicloud-ecs: Running transaction check
    alicloud-ecs: Transaction check succeeded.
    alicloud-ecs: Running transaction test
    alicloud-ecs: Transaction test succeeded.
    alicloud-ecs: Running transaction
    alicloud-ecs:   Preparing        :
1/1
    alicloud-ecs:   Running scriptlet: redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_64
1/1
    alicloud-ecs:   Installing       : redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_64
1/1
    alicloud-ecs:   Running scriptlet: redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_64
1/1
    alicloud-ecs:   Verifying        : redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_64
1/1
    alicloud-ecs:
    alicloud-ecs: Installed:
    alicloud-ecs:   redis-5.0.3-2.module_el8.2.0+318+3d7e67ea.x86_64
    alicloud-ecs:
    alicloud-ecs: Complete!
==> alicloud-ecs: Stopping instance: i-bp16p08isghg3hbcwijx
==> alicloud-ecs: Waiting instance stopped: i-bp16p08isghg3hbcwijx
==> alicloud-ecs: Deleting duplicated image and snapshot in cn-hangzhou: OOS_packer_test2
==> alicloud-ecs: Creating image: OOS_packer_test
alicloud-ecs: Detach keypair packer_5f90fbd1-27f5-3f5f-a7c3-9470b51183e2 from instance: i-b
p16p08isghg3hbcwijx
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished after 6 minutes 28 seconds.

==> Wait completed after 6 minutes 28 seconds

==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:

cn-hangzhou: m-bp1ah9sq4uzj3yfi62il
```