阿里云
云服务器 ECS

SDK 参考

文档版本：20190220

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 SDK

云服务器 ECS 提供了以下编程语言的 SDK。

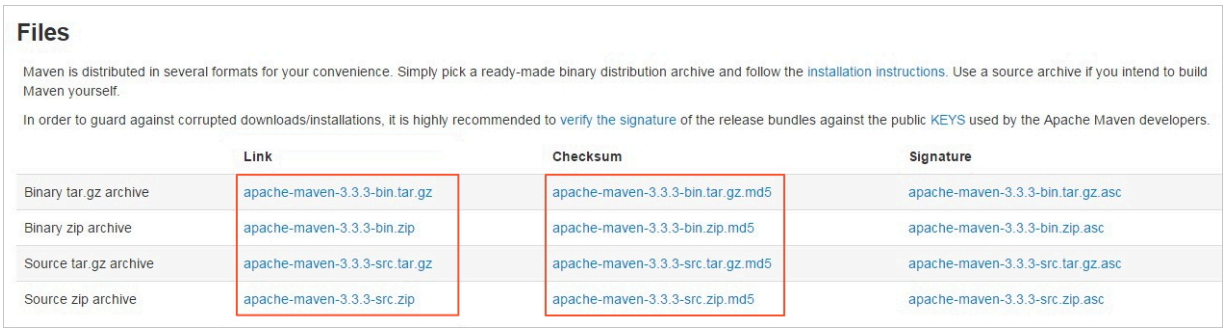
- *Java*
- *Python*
- *PHP*
- *C++*
- *.NET*

2 安装新版 Java SDK

相较于旧版 Java SDK 直接开放下载，阿里云新版 Java SDK 采用了 Maven 的方式分发。为方便统一管理，阿里云所有的 Java SDK 均统一在同一个 Maven 项目中。

下载 Maven 软件

访问 *Maven* 官方下载页面 下载对应操作系统的 Maven 软件。Checksum 文件可供您校验下载文件是否正确无误。以下截图来自于 Windows 7 64 位操作系统，使用环境是 Eclipse Luna。



Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

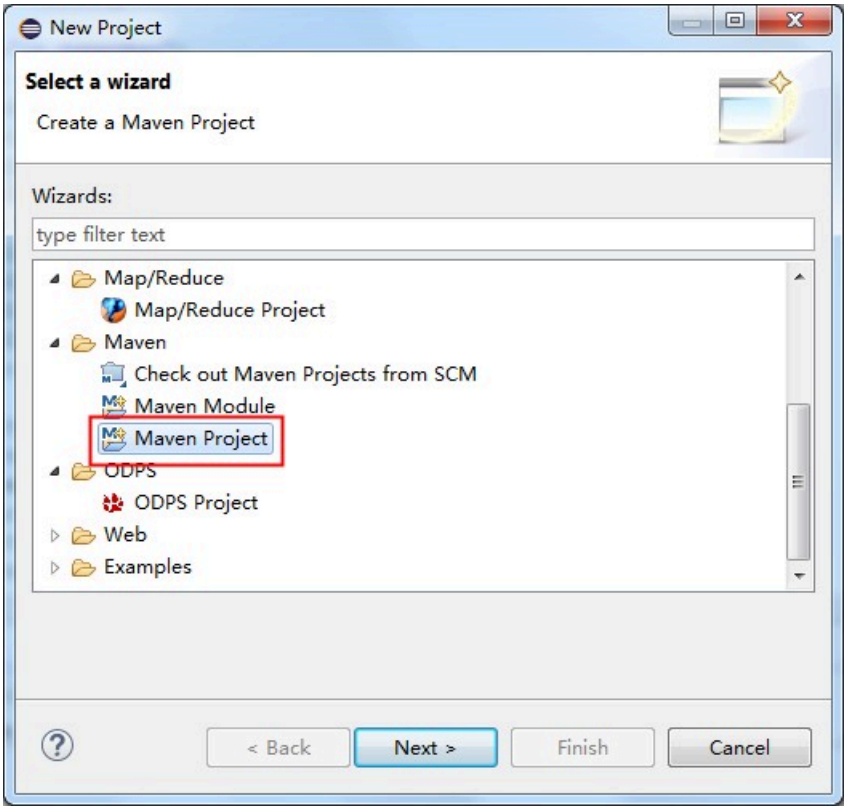
	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.3-bin.tar.gz	apache-maven-3.3.3-bin.tar.gz.md5	apache-maven-3.3.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.3-bin.zip	apache-maven-3.3.3-bin.zip.md5	apache-maven-3.3.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.3-src.tar.gz	apache-maven-3.3.3-src.tar.gz.md5	apache-maven-3.3.3-src.tar.gz.asc
Source zip archive	apache-maven-3.3.3-src.zip	apache-maven-3.3.3-src.zip.md5	apache-maven-3.3.3-src.zip.asc

安装 Java SDK

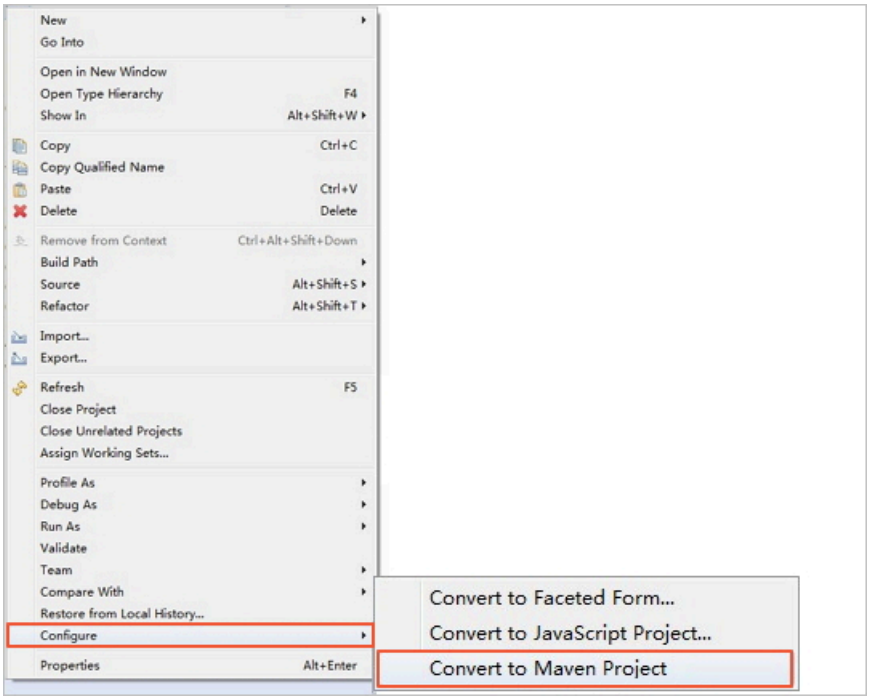
1. 打开 *Java SDK* 下载页面，将阿里云的 SDK 存放的 Maven 库加入到 Maven 软件中。
2. 找到之前下载的 Maven 压缩包并解压，将下列 Maven 库信息添加到 conf 文件夹下的 settings.xml 文件中。

```
<repositories>
 <repository>
  <id>sonatype-nexus-staging</id>
  <name>Sonatype Nexus Staging</name>
  <url>https://oss.sonatype.org/service/local/staging/deploy/maven2/
</url>
  <releases>
   <enabled>true</enabled>
  </releases>
  <snapshots>
   <enabled>true</enabled>
  <snapshots>
 </repository>
</repositories>
```

3. 通过以下任一方式创建 Maven 项目：
   - 方式一：在 Eclipse 添加一个 Maven 项目。

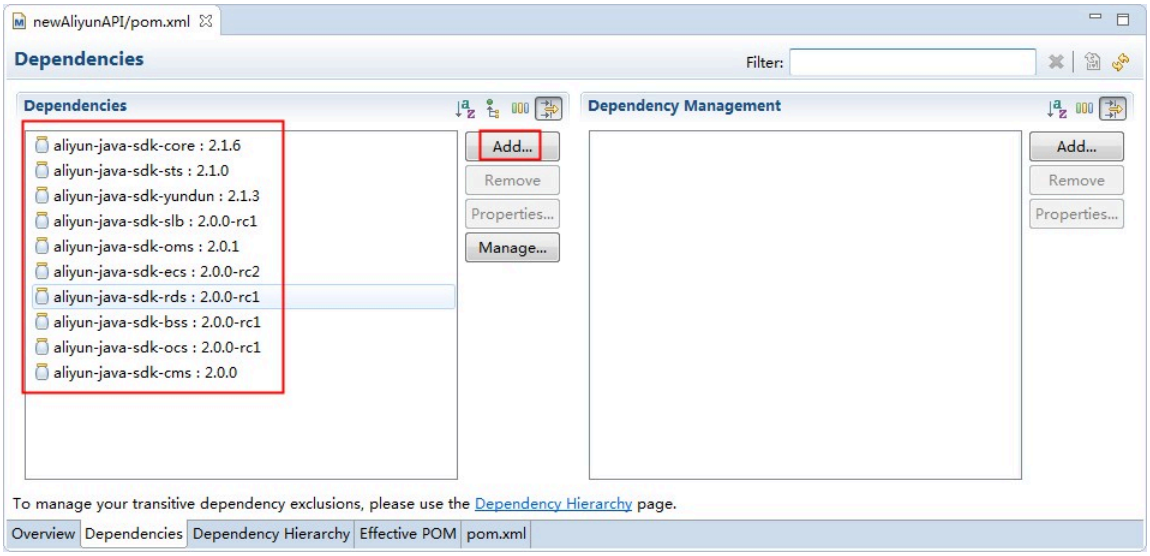


- **方式二：将已有的项目转换为 Maven 项目。**



4. 通过以下任一方式将 dependency 加入 Maven 项目中：

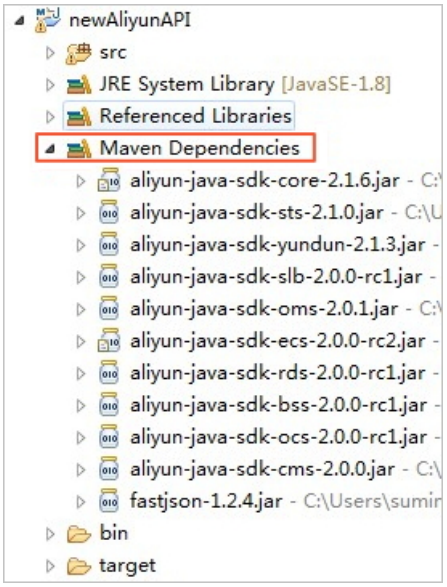
- **方式一：通过图形化界面选择性添加。**



- **方式二：打开 Maven 项目下的 *pom.xml* 文件，添加类似以下内容。您可以根据自身需要添加不同的依赖。**

```
<dependencies>
 <dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>2.1.6</version>
 </dependency>
 <dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-sts</artifactId>
  <version>2.1.0</version>
 </dependency>
 <dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-yundun</artifactId>
  <version>2.1.3</version>
 </dependency>
 <dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-slb</artifactId>
  <version>2.0.0-rc1</version>
 </dependency>
</dependencies>
```

5. **保存更改。Maven Dependencies 中会自动下载并加入 .jar 后缀的相应阿里云 SDK。**



区分旧版 SDK 与新版 SDK

若您使用的是旧版 SDK，建议您切换为新版 SDK，获取最新功能。通过下表中的 SDK 参数，可以识别 SDK 版本。

对比项	旧版 SDK	新版 SDK
如何提交请求	execute()	getAcsResponse()
如何存放 AccessKey 和 AccessKeySecret 的类	AliyunClient	IClientProfile
如何存放凭据对象	new DefaultAliyunClient(APIUrl, Access Key, Access Key Secret)	DefaultProfile.getProfile(RegionId, Access Key, Access Key Secret)
包名前缀	com.aliyun.api	com.aliyuncs

3 SDK使用方法和具体代码编写步骤

新版 SDK 的文件名通常以 aliyun-XXXX-sdk 开头，后面跟上产品名称如 ECS，组成如 aliyun-java-sdk-ecs 的包名。其中有一个核心包 aliyun-java-sdk-core，其中封装了所有产品的 SDK 都会用到的一些类，如 IClientProfile 类、 IAcsClient 类、异常类等。产品相关的类均以产品为单位打包成不同名称的 Jar 包。

您需要准备好您的 AccessKey，用于输出到 创建 *Profile* 中。

Java SDK 使用方法示例

以 ECS Java SDK 查询可用镜像资源的方法 *#unique_7* 为例，介绍 SDK 使用的完整流程，其中 IClientProfile 和 IAcsClient 两个类包含在 aliyun-java-sdk-core 包中，其他的类均包含在 aliyun-java-sdk-ecs 包中。

1. 创建 Profile。生成 IClientProfile 的对象 profile，该对象存放 AccessKeyID 和 AccessKeySecret 和默认的地域信息，如示例中的 `cn-hangzhou`，更多关于地域的信息，参阅 地域和可用区。

   ```
   IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", ak
   ,  aks); #ak 是您的 AccessKey, aks 是您的 AccessKeySecret
   ```

2. 创建 Client。从 IClientProfile 类中再生成 IAcsClient 的对象 client，后续获得 response 都需要从 IClientProfile 中获得。

   ```
   IAcsClient client = new DefaultAcsClient(profile);
   ```

3. 创建 Request。创建一个对应方法的 Request，类的命名规则一般为 API 的方法名加上 `Request`，如获得镜像列表的 API 方法名为 *#unique_7*，那么对应的请求类名就是 DescribeImagesRequest，直接使用构造函数生成一个默认的类 describe。

   ```
   DescribeImagesRequest describe = new DescribeImagesRequest();
   ```

4. 设置 Request 的参数。请求类生成好之后需要通过 Request 类的 `setXxx` 方法设置必要的信息，即 API 参数中必须要提供的信息，*#unique_7* 的 API 方法必须要提供的参数为 `RegionId`，该值可以省略，因为 `IClientProfile` 中已经提供了地域信息，同样的也可以通过 `setXxx`

方法设置其他可选的参数，如这里设置要查询的镜像为自定义镜像，则设置 ImageOwnerAlias 的值为 self，表示查询您的自定义镜像。

```
describe.setImageOwnerAlias("self");
```

5. 参数设置完毕后，通过 IAcsClient 对象获得对应 Request 的响应。

```
DescribeImagesResponse response = client.getAcsResponse(describe);
```

6. 在 Response 中获得返回的参数值。接着可以调用 response 中对应的 getXxx 方法获得返回的参数值了，如获得某个镜像的名字。根据 API 方法的不同，返回的信息中可能会包含多层的信息，如获得镜像列表这个方法，返回的信息中镜像是以一个集合来表示的，集合中存放了每个镜像的信息，对于 Java SDK 而言，那么存放镜像信息的就是一个列表，需要先通过 getImages() 获得 Image 对象的集合，然后再通过遍历等方法取得其中某个镜像的信息，之后调用 getXxx 方法获得具体的信息。

```
for(Image image:response.getImages())
            {
                System.out.println(image.getImageId());
                System.out.println(image.getImageName());
            }
```

至此，一个完整的调用就完成了。

PHP SDK 注意事项

使用 PHP SDK 和 Java SDK 的类似，可以归纳为：

1. 创建 Profile。
2. 创建 Client。
3. 创建 Request。
4. 设置 Request的参数。
5. 使用 Client 对应的方法传入 Request，获得 Response。
6. 在 Response 中获得返回的参数值。

Python SDK 注意事项

使用 Python SDK 省略了创建 Profile 这一步，直接创建 Client，然后执行后面的步骤即可。

参考信息

- 关于 ECS 的所有 API，请参阅 *API* 概览。
- 关于如何创建 AccessKey，请参阅 创建 *AccessKey*。

4 SDK使用案例

4.1 弹性创建ECS实例

除了可以在 ECS 控制台或售卖页创建 ECS 实例外，您还可以使用 API 代码来弹性地创建和管理ECS实例。本页面使用 Python 为例进行说明。

创建 ECS 时需关注以下 API：

- 创建*ECS*实例
- 查询实例列表
- 启动*ECS*实例
- 分配公网*IP*地址

创建按量云服务器

- 创建ECS实例时的必选属性
  - SecurityGroupId：安全组 ID。安全组通过防火墙规则实现对一组实例的配置，保护实例的网络出入请求。在设置安全组出入规则时，建议按需开放而不要默认开放所有的出入规则。您也可以通过 ECS 控制台创建安全组。
  - InstanceType：实例规格。参考 ECS 售卖页的选项，界面上 1 核 2GB n1.small则入参为ecs.n1.small。
  - ImageId：镜像 ID。参考*ECS*控制台的镜像列表，您可以过滤系统公共镜像或者自定义镜像。

  更多参数设置请参考创建*ECS*实例。
- 创建云服务器

  如下面的代码所示，创建一台经典网络的ECS，使用系统盘ssd，盘参数为cloud_ssd，选择io优化实例optimized。

```
# create one after pay ecs instance.
def create_after_pay_instance(image_id, instance_type, security_g
roup_id):
    request = CreateInstanceRequest();
    request.set_ImageId(image_id)
    request.set_SecurityGroupId(security_group_id)
    request.set_InstanceType(instance_type)
    request.set_IoOptimized('optimized')
    request.set_SystemDiskCategory('cloud_ssd')
    response = _send_request(request)
    instance_id = response.get('InstanceId')
```

```
    logging.info("instance %s created task submit successfully.",
instance_id)
    return instance_id;
```

创建成功后将返回相应的实例 ID，失败的话也会有对应的 ErrorCode。由于参数较多，您可以参考 ECS 的售卖页进行调整。

```
{"InstanceId":"i-***","RequestId":"006C1303-BAC5-48E5-BCDF-
7FD5C2E6395D"}
```

- 云服务器生命周期

  对于云服务器的状态操作，请参考云服务器实例生命周期。

  只有Stopped状态的实例可以执行 Start 操作。也只有Running状态的 ECS 可以执行Stop操作。查询云服务器的状态可以通过查询实例列表传入 InstanceId 进行过滤。在DescribeInstancesRequest时可以通过传入一个 JSON 数组格式的 String 就可以查询这个资源的状态。查询单个实例的状态建议使用DescribeInstances而不要使用DescribeInstanceAttribute，因为前者比后者返回更多的属性和内容。

  下面的代码会检查实例的状态，只有实例的状态符合入参才会返回实例的详情。

```
# output the instance owned in current region.
def get_instance_detail_by_id(instance_id, status='Stopped'):
    logging.info("Check instance %s status is %s", instance_id,
status)
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps([instance_id]))
    response = _send_request(request)
    instance_detail = None
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        for item in instance_list:
            if item.get('Status') == status:
                instance_detail = item
                break;
        return instance_detail;
```

- 启动云服务器

  创建成功后的 ECS 默认状态是Stopped。如果要启动 ECS 实例为Running状态，只需要发送启动指令即可。

```
def start_instance(instance_id):
    request = StartInstanceRequest()
    request.set_InstanceId(instance_id)
    _send_request(request)
```

- 停止云服务器

  停止云服务器只需传入instanceId即可。

```
def stop_instance(instance_id):
    request = StopInstanceRequest()
```

```
    request.set_InstanceId(instance_id)
    _send_request(request)
```

- 创建时启动“自动启动云服务器”

  服务器的启动和停止都是一个异步操作，您可以在脚本创建并同时检测云服务器符合状态时执行相应操作。

  创建资源后得到实例ID，首先判断实例是否处于Stopped的状态，如果处于Stopped状态，下发Start服务器的指令，然后等待服务器的状态变成Running。

```
def check_instance_running(instance_id):
    detail = get_instance_detail_by_id(instance_id=instance_id,
status=INSTANCE_RUNNING)
    index = 0
    while detail is None and index &lt; 60:
        detail = get_instance_detail_by_id(instance_id=instance_id);
        time.sleep(10)
    if detail and detail.get('Status') == 'Stopped':
        logging.info("instance %s is stopped now.")
        start_instance(instance_id=instance_id)
        logging.info("start instance %s job submit.")
    detail = get_instance_detail_by_id(instance_id=instance_id,
status=INSTANCE_RUNNING)
    while detail is None and index &lt; 60:
        detail = get_instance_detail_by_id(instance_id=instance_id,
status=INSTANCE_RUNNING);
        time.sleep(10)
    logging.info("instance %s is running now.", instance_id)
    return instance_id;
```

- 分配公网IP

  如果在创建云服务器的过程中，指定了公网带宽，若需要公网的访问权限还要调用API来分配公网IP。详情请参考分配公网IP地址。

创建包年包月云服务器

除了创建按量服务的云服务器，您的API还支持创建包年包月的服务器。包年包月的创建和官网的创建流程不同，使用的是自动扣费的模式，也就是说您需要在创建服务器之前确保账号有足够的余额或者信用额度，在创建的时候将直接扣费。

和按量付费的 ECS 相比，只需要指定付费类型和时长即可，下面的时长为1个月。

```
request.set_Period(1)    request.set_InstanceChargeType(‘PrePaid’)
```

创建包年包月实例的整体的代码如下：

```
# create one prepay ecs instance.
def create_prepay_instance(image_id, instance_type, security_group_id
):
    request = CreateInstanceRequest();
    request.set_ImageId(image_id)
    request.set_SecurityGroupId(security_group_id)
    request.set_InstanceType(instance_type)
```

```
    request.set_IoOptimized('optimized')
    request.set_SystemDiskCategory('cloud_ssd')
    request.set_Period(1)
    request.set_InstanceChargeType('PrePaid')
    response = _send_request(request)
    instance_id = response.get('InstanceId')
    logging.info("instance %s created task submit successfully.",
instance_id)
    return instance_id;
```

完整的代码

完整的代码如下，您可以按照自己的资源参数进行设置。

```
#  coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
# make sure the sdk version is 2.1.2, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
import time
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.CreateInstanceRequest import
CreateInstanceRequest
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.StartInstanceRequest import
StartInstanceRequest
# configuration the log output formatter, if you want to save the
output to file,
# append ",filename='ecs_invoke.log'" after datefmt.
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')
clt = client.AcsClient('Your Access Key Id', 'Your Access Key Secrect
', 'cn-beijing')
IMAGE_ID = 'ubuntu1404_64_40G_cloudinit_20160727.raw'
INSTANCE_TYPE = 'ecs.s2.large'  # 2c4g generation 1
SECURITY_GROUP_ID = 'sg-****'
INSTANCE_RUNNING = 'Running'
def create_instance_action():
    instance_id = create_after_pay_instance(image_id=IMAGE_ID,
instance_type=INSTANCE_TYPE,
                                            security_group_id=
SECURITY_GROUP_ID)
    check_instance_running(instance_id=instance_id)
def create_prepay_instance_action():
    instance_id = create_prepay_instance(image_id=IMAGE_ID, instance_t
ype=INSTANCE_TYPE,
                                         security_group_id=SECURITY_G
ROUP_ID)
    check_instance_running(instance_id=instance_id)
# create one after pay ecs instance.
def create_after_pay_instance(image_id, instance_type, security_g
roup_id):
    request = CreateInstanceRequest();
    request.set_ImageId(image_id)
    request.set_SecurityGroupId(security_group_id)
```

```
    request.set_InstanceType(instance_type)
    request.set_IoOptimized('optimized')
    request.set_SystemDiskCategory('cloud_ssd')
    response = _send_request(request)
    instance_id = response.get('InstanceId')
    logging.info("instance %s created task submit successfully.",
instance_id)
    return instance_id;
# create one prepay ecs instance.
def create_prepay_instance(image_id, instance_type, security_group_id
):
    request = CreateInstanceRequest();
    request.set_ImageId(image_id)
    request.set_SecurityGroupId(security_group_id)
    request.set_InstanceType(instance_type)
    request.set_IoOptimized('optimized')
    request.set_SystemDiskCategory('cloud_ssd')
    request.set_Period(1)
    request.set_InstanceChargeType('PrePaid')
    response = _send_request(request)
    instance_id = response.get('InstanceId')
    logging.info("instance %s created task submit successfully.",
instance_id)
    return instance_id;
def check_instance_running(instance_id):
    detail = get_instance_detail_by_id(instance_id=instance_id, status
=INSTANCE_RUNNING)
    index = 0
    while detail is None and index &lt; 60:
        detail = get_instance_detail_by_id(instance_id=instance_id);
        time.sleep(10)
    if detail and detail.get('Status') == 'Stopped':
        logging.info("instance %s is stopped now.")
        start_instance(instance_id=instance_id)
        logging.info("start instance %s job submit.")
    detail = get_instance_detail_by_id(instance_id=instance_id, status
=INSTANCE_RUNNING)
    while detail is None and index &lt; 60:
        detail = get_instance_detail_by_id(instance_id=instance_id,
status=INSTANCE_RUNNING);
        time.sleep(10)
    logging.info("instance %s is running now.", instance_id)
    return instance_id;
def start_instance(instance_id):
    request = StartInstanceRequest()
    request.set_InstanceId(instance_id)
    _send_request(request)
# output the instance owned in current region.
def get_instance_detail_by_id(instance_id, status='Stopped'):
    logging.info("Check instance %s status is %s", instance_id, status
)
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps([instance_id]))
    response = _send_request(request)
    instance_detail = None
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        for item in instance_list:
            if item.get('Status') == status:
                instance_detail = item
                break;
        return instance_detail;
# send open api request
def _send_request(request):
```

```
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
if __name__ == '__main__':
    logging.info("Create ECS by OpenApi!")
    create_instance_action()
    # create_prepay_instance_action()
```

4.2 批量创建ECS实例

RunInstances 批量创建实例接口可以帮助您一次创建多台 ECS 按量付费或预付费实例来完成应用的开发和部署，方便实现弹性的资源创建。

和 *#unique_12* 接口相比，*#unique_18* 接口有下面的优点：

- 单次可以最多创建 100 台实例，避免重复调用。
- 实例创建之后，实例会自动变成 Starting 状态，然后变成 Running 状态，不需要您调用 StartInstance 的操作。
- 创建实例的时候指定了 InternetMaxBandwidthOut，则自动为您分配公网 IP，不需要您再调用分配 IP 的操作。
- 您也可以一次创建 100 台 抢占式实例，充分满足您的弹性需求。
- 创建的参数保持和 CreateInstance 保持兼容，增加了 Amount 参数来设定创建的个数，以及 AutoReleaseTime 参数来设定自动释放时间，不需要您再额外设置自动释放时间。
- 创建返回一个 InstanceIdSets 会记录相关的 InstanceIds，您只需要根据实例 ID 轮询实例状态即可。

前提条件

调用 API 前，您需要 创建 *AccessKey*。

：
禁止使用主账号 AK，因为主账号 AK 泄露会威胁您所有资源的安全。请使用子账号 AK 进行操作，可有效降低 AK 泄露的风险。

安装 ECS Python SDK

首先确保您已经具备 Python 的 Runtime，本文中使用的 Python 版本为 2.7+。

这里以 Python 为示例，其他的版本 *SDK* 大于 4.4.3 即可。

```
pip install aliyun-python-sdk-ecs
```

如果提示您没有权限，请切换sudo继续执行。

```
sudo pip install aliyun-python-sdk-ecs
```

本文使用的 SDK 版本为 4.4.3， 如果您使用是旧版本的 SDK，需要更新。

批量创建实例

首先创建 RunInstancesRequest 的实例，然后填入相关需要的参数即可。

下面的例子创建了 2 台实例，并且添加了自动每隔 10 秒钟检查一次实例的运行状态。直到实例状态变成Running结束创建流程。

```
# your access key Id
ak_id = "YOU_ACCESS_KEY_ID"
# your access key secret
ak_secret = "YOU_ACCESS_SECRET"
region_id = "cn-beijing"
# your expected instance type
instance_type = "ecs.n4.small"
# 选择的vswitchId
vswitch_id = "vws-xxxxx"
# 使用的镜像信息
image_id = "centos_7_03_64_20G_alibase_20170818.vhd"
# 当前vpc类型的安全组
security_group_id = "sg-xxxxx"
# instance number to launch, support 1-100, default value is 100
amount = 2;
# instance auto delete time 按照 ISO8601 标准表示，并需要使用 UTC 时间。格式为 yyyy-MM-ddTHH:mm:ssZ 。 最短在当前时间之后半小时。最长不能超过当前时间起三年
auto_release_time = "2017-12-05T22:40:00Z"
clt = client.AcsClient(ak_id, ak_secret, 'cn-beijing')
# create instance automatic running
def batch_create_instance():
    request = build_request()
    request.set_Amount(amount)
    _execute_request(request)
def _execute_request(request):
    response = _send_request(request)
    if response.get('Code') is None:
        instance_ids = response.get('InstanceIdSets').get('InstanceId
Set')
        running_amount = 0
        while running_amount < amount:
            time.sleep(10)
            running_amount = check_instance_running(instance_ids)
    print("ecs instance %s is running", instance_ids)
def check_instance_running(instance_ids):
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps(instance_ids))
    response = _send_request(request)
    if response.get('Code') is None:
        instances_list = response.get('Instances').get('Instance')
        running_count = 0
        for instance_detail in instances_list:
```

```
            if instance_detail.get('Status') == "Running":
                running_count += 1
        return running_count
def build_request():
    request = RunInstancesRequest()
    request.set_ImageId(image_id)
    request.set_VSwitchId(vswitch_id)
    request.set_SecurityGroupId(security_group_id)
    request.set_InstanceName("Instance12-04")
    request.set_InstanceType(instance_type)
    return request
# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
```

批量创建实例并自动分配公网 IP

相比批量创建实例的代码，只需要添加一行属性，指定公网的带宽即可。下面的例子中默认给实例都分配了 1 M 的按流量带宽。

```
# create instance with public ip.
def batch_create_instance_with_public_ip():
    request = build_request()
    request.set_Amount(amount)
    request.set_InternetMaxBandwidthOut(1)
    _execute_request(request)
```

批量创建实例并自动设置自动释放时间

相比批量创建实例的代码，只需要添加一行属性，指定实例的自动释放时间即可。自动释放时间按照 *ISO8601* 标准表示，并需要使用 UTC 时间，格式为 `yyyy-MM-ddTHH:mm:ssZ`。最短在当前时间之后半小时，最长不能超过当前时间起三年。

```
# create instance with auto release time.
def batch_create_instance_with_auto_release_time():
    request = build_request()
    request.set_Amount(amount)
    request.set_AutoReleaseTime(auto_release_time)
    _execute_request(request)
```

完整代码示例

以上操作完整的代码示例如下所示。

```
#  coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
```

```
# make sure the sdk version is 4.4.3, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
import time
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.RunInstancesRequest import
RunInstancesRequest
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')
# your access key Id
ak_id = "YOU_ACCESS_KEY_ID"
# your access key secret
ak_secret = "YOU_ACCESS_SECRET"
region_id = "cn-beijing"
# your expected instance type
instance_type = "ecs.n4.small"
# 选择的vswitchId
vswitch_id = "vws-xxxxx"
# 使用的镜像信息
image_id = "centos_7_03_64_20G_alibase_20170818.vhd"
# 当前vpc类型的安全组
security_group_id = "sg-xxxxx"
# instance number to launch, support 1-100, default value is 100
amount = 2;
# instance auto delete time 按照 ISO8601 标准表示，并需要使用 UTC 时间。格式
为 yyyy-MM-ddTHH:mm:ssZ 。 最短在当前时间之后半小时。最长不能超过当前时间起三年
auto_release_time = "2017-12-05T22:40:00Z"
clt = client.AcsClient(ak_id, ak_secret, 'cn-beijing')
# create instance automatic running
def batch_create_instance():
    request = build_request()
    request.set_Amount(amount)
    _execute_request(request)
# create instance with public ip.
def batch_create_instance_with_public_ip():
    request = build_request()
    request.set_Amount(amount)
    request.set_InternetMaxBandwidthOut(1)
    _execute_request(request)
# create instance with auto release time.
def batch_create_instance_with_auto_release_time():
    request = build_request()
    request.set_Amount(amount)
    request.set_AutoReleaseTime(auto_release_time)
    _execute_request(request)
def _execute_request(request):
    response = _send_request(request)
    if response.get('Code') is None:
        instance_ids = response.get('InstanceIdSets').get('InstanceId
Set')
        running_amount = 0
        while running_amount < amount:
            time.sleep(10)
            running_amount = check_instance_running(instance_ids)
    print("ecs instance %s is running", instance_ids)
def check_instance_running(instance_ids):
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps(instance_ids))
    response = _send_request(request)
```

```
    if response.get('Code') is None:
        instances_list = response.get('Instances').get('Instance')
        running_count = 0
        for instance_detail in instances_list:
            if instance_detail.get('Status') == "Running":
                running_count += 1
        return running_count
def build_request():
    request = RunInstancesRequest()
    request.set_ImageId(image_id)
    request.set_VSwitchId(vswitch_id)
    request.set_SecurityGroupId(security_group_id)
    request.set_InstanceName("Instance12-04")
    request.set_InstanceType(instance_type)
    return request
# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
if __name__ == '__main__':
    print "hello ecs batch create instance"
    # batch_create_instance()
    # batch_create_instance_with_public_ip()
    # batch_create_instance_with_auto_release_time()
```

4.3 弹性管理ECS实例

您除了可以通过 ECS管理控制台创建或管理ECS实例外，您也能通过 API 管理或定制开发 ECS 实例。阿里云提供了 SDK 来包装API，将云服务器 ECS 的管理集成到已有系统中。本文基于 Python 的开发来说明如何通过API 管理 ECS 实例。如果您没有 Python 开发经验，也能通过本文完成云服务的开发。

获取 RAM 子账号 AK 密钥

使用API管理ECS实例，您需要能访问ECS资源的API密钥（AccessKey ID 和 AccessKey Secret）。为了保证云服务的安全，您需要创建一个能访问ECS资源的RAM用户，获取该用户的AccessKey密钥，并使用这个RAM用户和API管理ECS实例。

以下是获取RAM用户AccessKey密钥的操作步骤：

1. 创建*RAM*用户并获取*AccessKey*密钥。
2. 为*RAM*用户授权，授予RAM用户管理云服务器服务（ECS）的权限。

安装 ECS Python SDK

首先确保您已经具备Python的Runtime，本文中使用的Python版本为2.7+。

```
pip install aliyun-python-sdk-ecs
```

如果提示您没有权限，请切换sudo继续执行。

```
sudo pip install aliyun-python-sdk-ecs
```

本文使用的SDK版本为2.1.2。

Hello Alibaba Cloud

创建文件 *hello_ecs_api.py*。为了使用SDK，首先实例化AcsClient对象，这里需要RAM用户的AccessKey ID和AccessKey Secret。

说明:

AccessKey ID和AccessKey Secret是RAM用户访问阿里云ECS服务API的密钥，具有该账户完全的权限，请妥善保管。

```
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.DescribeRegionsRequest import
DescribeRegionsRequest
clt = client.AcsClient('Your Access Key Id', 'Your Access Key Secrect
', 'cn-beijing')
```

完成实例化后可以进行第一个应用的开发。查询当前账号支持的地域列表。具体的文档参见 查询可用地域列表。

```
def hello_aliyun_regions():
    request = DescribeRegionsRequest()
    response = _send_request(request)
    region_list = response.get('Regions').get('Region')
    assert response is not None
    assert region_list is not None
    result = map(_print_region_id, region_list)
    logging.info("region list: %s", result)
def _print_region_id(item):
    region_id = item.get("RegionId")
    return region_id
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
```

```
hello_aliyun_regions()
```

在命令行运行 `python hello_ecs_api.py` 会得到当前支持的 Region列表。类似的输出如下：

```
[u'cn-shenzhen', u'ap-southeast-1', u'cn-qingdao', u'cn-beijing', u'cn
-shanghai', u'us-east-1', u'cn-hongkong', u'me-east-1', u'ap-southeast
-2', u'cn-hangzhou', u'eu-central-1', u'ap-northeast-1', u'us-west-1']
```

查询当前的 Region 下的 ECS 实例列表

查询实例列表和查询 Region 列表非常类似，替换入参对象为`DescribeInstancesRequest` 即可，更多的查询参数参考 查询实例列表。

```
def list_instances():
    request = DescribeInstancesRequest()
    response = _send_request(request)
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        result = map(_print_instance_id, instance_list)
        logging.info("current region include instance %s", result)
def _print_instance_id(item):
    instance_id = item.get('InstanceId');
    return instance_id
```

输出结果为如下：

```
current region include instance [u'i-****', u'i-****'']
```

更多的API参考 *ECS API* 概览，您可以尝试作一个 查询磁盘列表，将实例的参数替换为 `DescribeDisksRequest`。

完整代码示例

以上操作完整的代码示例如下所示。

```
#  coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
# make sure the sdk version is 2.1.2, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.DescribeRegionsRequest import
DescribeRegionsRequest
# configuration the log output formatter, if you want to save the
output to file,
# append ",filename='ecs_invoke.log'" after datefmt.
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')
```

```
clt = client.AcsClient('Your Access Key Id', 'Your Access Key Secrect
', 'cn-beijing')
# sample api to list aliyun open api.
def hello_aliyun_regions():
    request = DescribeRegionsRequest()
    response = _send_request(request)
    if response is not None:
        region_list = response.get('Regions').get('Region')
        assert response is not None
        assert region_list is not None
        result = map(_print_region_id, region_list)
        logging.info("region list: %s", result)
# output the instance owned in current region.
def list_instances():
    request = DescribeInstancesRequest()
    response = _send_request(request)
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        result = map(_print_instance_id, instance_list)
        logging.info("current region include instance %s", result)
def _print_instance_id(item):
    instance_id = item.get('InstanceId');
    return instance_id
def _print_region_id(item):
    region_id = item.get("RegionId")
    return region_id
# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
if __name__ == '__main__':
    logging.info("Hello Aliyun OpenApi!")
    hello_aliyun_regions()
    list_instances()
```

如您想了解 ECS 中 API 的其它操作，请参考 *ECS*中的*API*操作。

4.4 弹性释放ECS实例

云服务器 ECS 的一个重要特性就是按需创建资源。您可以在业务高峰期按需弹性地进行自定义资源创建，完成业务计算时释放资源。本篇将提供若干 Tips 帮助您更加便捷地完成云服务器的释放以及弹性设置。

本文将涉及到几个重要功能和相关API：

- **释放按量付费的云服务器**
- **设置按量付费实例的自动释放时间**
- **停止服务器**
- **查询实例列表**

释放后，实例所使用的物理资源将被回收，包括磁盘及快照，相关数据将全部丢失且永久不可恢复。如果您还想继续使用相关的数据，建议您释放云服务器之前一定要对磁盘数据做快照，下次创建 ECS 时可以直接通过快照创建资源。

释放云服务器

释放服务器，首先要求您的服务器处于停止状态。当服务器停止后，若影响到应用，您可以将服务器重新启动。

停止云服务器

停止服务器的指令非常简单，且对于按量付费和包年包月都是一样的。停止云服务器的一个参数是ForceStop，若属性设置为 true，它将类似于断电，直接停止服务器，但不承诺数据能写到磁盘中。如果仅仅为了释放服务器，这个可以设置为 true。

```
def stop_instance(instance_id, force_stop=False):
    '''
    stop one ecs instance.
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param force_stop: if force stop is true, it will force stop the
server and not ensure the data
    write to disk correctly.
    :return:
    '''
    request = StopInstanceRequest()
    request.set_InstanceId(instance_id)
    request.set_ForceStop(force_stop)
    logging.info("Stop %s command submit successfully.", instance_id)
    _send_request(request)
```

释放云服务器

如果您没有停止服务器直接执行释放，可能会有如下报错：

```
{"RequestId":"3C6DEAB4-7207-411F-9A31-6ADE54C268BE","HostId":"ecs-cn-
hangzhou.aliyuncs.com","Code":"IncorrectInstanceStatus","Message":"The
 current status of the resource does not support this operation."}
```

当服务器处于Stopped状态时，您可以执行释放服务器。释放服务器的方法比较简单，参数如下：

- InstanceId: 实例的 ID。
- force: 如果将这个参数设置为 true，将会执行强制释放。即使云服务器不是Stopped状态也可以释放。执行的时候请务必小心，以防错误释放影响您的业务。

释放云服务器的Request如下：

```
def release_instance(instance_id, force=False):
    '''
    delete instance according instance id, only support after pay
instance.
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param force:
```

```
    if force is false, you need to make the ecs instance stopped, you
can
    execute the delete action.
    If force is true, you can delete the instance even the instance is
 running.
    :return:
    '''
    request = DeleteInstanceRequest();
    request.set_InstanceId(instance_id)
    request.set_Force(force)
    _send_request(request)
```

释放云服务器成功的 Response 如下：

```
{"RequestId":"689E5813-D150-4664-AF6F-2A27BB4986A3"}
```

设置云服务器的自动释放时间

为了更加简化对云服务器的管理，您可以自定义云服务器的释放时间。当定时时间到后，阿里云将自动为您完成服务器的释放，无需手动执行释放。

说明：

自动释放时间按照 ISO8601 标准表示，并需要使用 UTC 时间。 格式为：yyyy-MM-ddTHH:mm:ssZ。 如果秒不是 00，则自动取为当前分钟开始时。自动释放的时间范围：当前时间后 30 分钟 ~ 当前时间起 3 年。

```
def set_instance_auto_release_time(instance_id, time_to_release = None
):
    '''
    setting instance auto delete time
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param time_to_release: if the property is setting, such as '2017-
01-30T00:00:00Z'
    it means setting the instance to be release at that time.
    if the property is None, it means cancel the auto delete time.
    :return:
    '''
    request = ModifyInstanceAutoReleaseTimeRequest()
    request.set_InstanceId(instance_id)
    if time_to_release is not None:
        request.set_AutoReleaseTime(time_to_release)
    _send_request(request)
```

执行 `set_instance_auto_release_time('i-1111', '2017-01-30T00:00:00Z')` **后完成设置。**

执行设置成功后，您可以通过`DescribeInstances`**来查询自动释放的时间设置。**

```
def describe_instance_detail(instance_id):
    '''
    describe instance detail
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :return:
    '''
```

```
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps([instance_id]))
    response = _send_request(request)
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        if len(instance_list) > 0:
            return instance_list[0]
def check_auto_release_time_ready(instance_id):
    detail = describe_instance_detail(instance_id=instance_id)
    if detail is not None:
        release_time = detail.get('AutoReleaseTime')
        return release_time
```

取消自动释放设置

如果您的业务有变化，需要取消自动释放设置。只需执行命令将自动释放时间设置为空即可。

```
set_instance_auto_release_time('i-1111')
```

完整代码

说明:

释放云服务器需谨慎。

```
#  coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
# make sure the sdk version is 2.1.2, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.DeleteInstanceRequest import
DeleteInstanceRequest
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.ModifyInstanceAutoReleaseTimeR
equest import \
    ModifyInstanceAutoReleaseTimeRequest
from aliyunsdkecs.request.v20140526.StopInstanceRequest import
StopInstanceRequest
# configuration the log output formatter, if you want to save the
output to file,
# append ",filename='ecs_invoke.log'" after datefmt.
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')
clt = client.AcsClient('Your Access Key Id', 'Your Access Key Secrect
', 'cn-beijing')
def stop_instance(instance_id, force_stop=False):
    '''
    stop one ecs instance.
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param force_stop: if force stop is true, it will force stop the
server and not ensure the data
    write to disk correctly.
```

```
    :return:
    '''
    request = StopInstanceRequest()
    request.set_InstanceId(instance_id)
    request.set_ForceStop(force_stop)
    logging.info("Stop %s command submit successfully.", instance_id)
    _send_request(request)
def describe_instance_detail(instance_id):
    '''
    describe instance detail
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :return:
    '''
    request = DescribeInstancesRequest()
    request.set_InstanceIds(json.dumps([instance_id]))
    response = _send_request(request)
    if response is not None:
        instance_list = response.get('Instances').get('Instance')
        if len(instance_list) &gt; 0:
            return instance_list[0]
def check_auto_release_time_ready(instance_id):
    detail = describe_instance_detail(instance_id=instance_id)
    if detail is not None:
        release_time = detail.get('AutoReleaseTime')
        return release_time
def release_instance(instance_id, force=False):
    '''
    delete instance according instance id, only support after pay
instance.
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param force:
    if force is false, you need to make the ecs instance stopped, you
can
    execute the delete action.
    If force is true, you can delete the instance even the instance is
 running.
    :return:
    '''
    request = DeleteInstanceRequest();
    request.set_InstanceId(instance_id)
    request.set_Force(force)
    _send_request(request)
def set_instance_auto_release_time(instance_id, time_to_release = None
):
    '''
    setting instance auto delete time
    :param instance_id: instance id of the ecs instance, like 'i-***'.
    :param time_to_release: if the property is setting, such as '2017-
01-30T00:00:00Z'
    it means setting the instance to be release at that time.
    if the property is None, it means cancel the auto delete time.
    :return:
    '''
    request = ModifyInstanceAutoReleaseTimeRequest()
    request.set_InstanceId(instance_id)
    if time_to_release is not None:
        request.set_AutoReleaseTime(time_to_release)
    _send_request(request)
    release_time = check_auto_release_time_ready(instance_id)
    logging.info("Check instance %s auto release time setting is %s.
 ", instance_id, release_time)
def _send_request(request):
    '''
    send open api request
```

```
    :param request:
    :return:
    '''
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
if __name__ == '__main__':
    logging.info("Release ecs instance by Aliyun OpenApi!")
    set_instance_auto_release_time('i-1111', '2017-01-28T06:00:00Z')
    # set_instance_auto_release_time('i-1111')
    # stop_instance('i-1111')
    # release_instance('i-1111')
    # release_instance('i-1111', True)
```

如您想了解 ECS 中 API 的其它操作，请参考 *ECS中的API操作*。

4.5 ECS实例续费

除了通过 ECS控制台或售卖页进行云服务器续费外，阿里云还支持直接通过 API 进行续费查询和续费管理

本文主要涉及如下关键功能：

- 按照过期时间查询云服务器
- 续费实例
- 查询云服务器自动续费时间
- 设置云服务器自动续费时间

对于包年包月的云服务器，生命周期非常重要。如果云服务器资源不能按时续费，将可能导致服务器被锁定甚至被释放，从而影响业务持续性。API 帮助您及时了解和检查资源的到期时间，并完成续费充值功能。

本篇需关注如下 API：

- 查询实例列表
- 续费实例

查询指定范围内到期的云服务器

查询实例列表的 API，通过过滤参数，您可以查询一定时间范围内到期的实例信息。通过设置过滤参数 `ExpiredStartTime` 和 `ExpiredEndTime`（时间参数 按照 ISO8601 标准表示，并需要使

用 UTC 时间。 格式为：yyyy-MM-ddTHH:mmZ），可以方便地查询该时间范围内到期的实例列表。如果需要通过安全组进行过滤，只需加上安全组 ID 即可。

```
INSTANCE_EXPIRED_START_TIME_IN_UTC_STRING = '2017-01-22T00:00Z'
INSTANCE_EXPIRE_END_TIME_IN_UTC_STRING = '2017-01-28T00:00Z'
def describe_need_renew_instance(page_size=100, page_number=1,
instance_id=None,
                                 check_need_renew=True, security_g
roup_id=None):
    request = DescribeInstancesRequest()
    if check_need_renew is True:
        request.set_Filter3Key("ExpiredStartTime")
        request.set_Filter3Value(INSTANCE_EXPIRED_START_TIME_IN
_UTC_STRING)
        request.set_Filter4Key("ExpiredEndTime")
        request.set_Filter4Value(INSTANCE_EXPIRE_END_TIME_IN_UT
C_STRING)
    if instance_id is not None:
        request.set_InstanceIds(json.dumps([instance_id]))
    if security_group_id:
        request.set_SecurityGroupId(security_group_id)
    request.set_PageNumber(page_number)
    request.set_PageSize(page_size)
    return _send_request(request)
```

续费云服务器

续费实例只支持包年包月的服务器类型，不支持按量付费的服务器，同时要求用户必须支持账号的余额支付或信用支付。执行 API 的时候将执行同步的扣费和订单生成。因此，执行 API 的时候必须保证您的账号有足够的资金支持自动扣费。

```
def _renew_instance_action(instance_id, period='1'):
    request = RenewInstanceRequest()
    request.set_Period(period)
    request.set_InstanceId(instance_id)
    response = _send_request(request)
    logging.info('renew %s ready, output is %s ', instance_id,
response)
```

续费实例将会自动完成扣费。在完成续费后，您可以根据InstanceId查询实例的资源到期时间。由于 API 为异步任务，查询资源到期时间可能需要延迟 10 秒才会变化。

开启云服务器自动续费

为了减少您的资源到期维护成本，针对包年包月的 ECS 实例，阿里云还推出了 自动续费功能。 自动续费扣款日为服务器到期前第 9 天的 08:00:00。如果前一日执行自动扣费失败，将会继续下一日定时执行，直到完成扣费或者 9 天后到期资源锁定。您只需要保证自己的账号余额或者信用额度充足即可。

- 查询自动续费设置

您可以通过 OpenAPI 来查询和设置自动续费。该 API 仅支持包年包月的实例，按量付费的实例执行将会报错。查询实例的自动续费状态支持一次最多查询 100 个包年包月的实例，多个实例 ID 以逗号连接。

DescribeInstanceAutoRenewAttribute 的入参为实例 ID。

InstanceId：支持最多查询 100 个包年包月的实例，多个实例 ID 以逗号连接。

```
# check the instances is renew or not
def describe_auto_renew(instance_ids, expected_auto_renew=True):
    describe_request = DescribeInstanceAutoRenewAttributeRequest()
    describe_request.set_InstanceId(instance_ids)
    response_detail = _send_request(request=describe_request)
    failed_instance_ids = ''
    if response_detail is not None:
        attributes = response_detail.get('InstanceRenewAttributes').
get('InstanceRenewAttribute')
        if attributes:
            for item in attributes:
                auto_renew_status = item.get('AutoRenewEnabled')
                if auto_renew_status != expected_auto_renew:
                    failed_instance_ids += item.get('InstanceId') +
 ','

describe_auto_renew('i-1111,i-2222')
```

返回内容如下：

```
{"InstanceRenewAttributes":{"InstanceRenewAttribute":[{"Duration
":0,"InstanceId":"i-1111","AutoRenewEnabled":false},{"Duration":
0,"InstanceId":"i-2222","AutoRenewEnabled":false}]},"RequestId":"
71FBB7A5-C793-4A0D-B17E-D6B426EA746A"}
```

如果设置自动续费，则返回的属性AutoRenewEnabled为 true，否则返回 false。

- 设置和取消云服务器的自动续费

设置自动续费有三个入参：

- InstanceId： 支持最多查询100个包年包月的实例，多个实例 ID 以逗号连接。
- Duration：支持 1、2、3、6、12，单位为月。
- AutoRenew：true/false， true为开启自动续费，false为取消自动续费。

```
def setting_instance_auto_renew(instance_ids, auto_renew = True):
    logging.info('execute enable auto renew ' + instance_ids)
    request = ModifyInstanceAutoRenewAttributeRequest();
    request.set_Duration(1);
    request.set_AutoRenew(auto_renew);
    request.set_InstanceId(instance_ids)
```

```
    _send_request(request)
```

执行成功返回 Response 如下：

```
{"RequestId":"7DAC9984-AAB4-43EF-8FC7-7D74C57BE46D"}
```

续费成功后，您可以再执行一次查询。如果续费成功将返回续费时长以及是否开启自动续费。

```
{"InstanceRenewAttributes":{"InstanceRenewAttribute":[{"Duration
":1,"InstanceId":"i-1111","AutoRenewEnabled":true},{"Duration":
1,"InstanceId":"i-2222","AutoRenewEnabled":true}]},"RequestId":"
7F4D14B0-D0D2-48C7-B310-B1DF713D4331"}
```

完整代码

```
#  coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
# make sure the sdk version is 2.1.2, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
from aliyunsdkcore import client
from aliyunsdkecs.request.v20140526.DescribeInstanceAutoRenewAttri
buteRequest import \
    DescribeInstanceAutoRenewAttributeRequest
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import
DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.ModifyInstanceAutoRenewAttribu
teRequest import \
    ModifyInstanceAutoRenewAttributeRequest
from aliyunsdkecs.request.v20140526.RenewInstanceRequest import
RenewInstanceRequest
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')
clt = client.AcsClient('Your Access Key Id', 'Your Access Key Secrect
', 'cn-beijing')
# data format in UTC, only support passed the value for minute,
seconds is not support.
INSTANCE_EXPIRED_START_TIME_IN_UTC_STRING = '2017-01-22T00:00Z'
INSTANCE_EXPIRE_END_TIME_IN_UTC_STRING = '2017-01-28T00:00Z'
def renew_job(page_size=100, page_number=1, check_need_renew=True,
security_group_id=None):
    response = describe_need_renew_instance(page_size=page_size,
page_number=page_number,
                                            check_need_renew=
check_need_renew,
                                            security_group_id=
security_group_id)
    response_list = response.get('Instances').get('Instance')
    logging.info("%s instances need to renew", str(response.get('
TotalCount')))
    if response_list > 0:
        instance_ids = ''
        for item in response_list:
            instance_id = item.get('InstanceId')
            instance_ids += instance_id + ','
```

```
            renew_instance(instance_id=instance_id)
        logging.info("%s execute renew action ready", instance_ids)
def describe_need_renew_instance(page_size=100, page_number=1,
instance_id=None,
                                 check_need_renew=True, security_g
roup_id=None):
    request = DescribeInstancesRequest()
    if check_need_renew is True:
        request.set_Filter3Key("ExpiredStartTime")
        request.set_Filter3Value(INSTANCE_EXPIRED_START_TIME_IN
_UTC_STRING)
        request.set_Filter4Key("ExpiredEndTime")
        request.set_Filter4Value(INSTANCE_EXPIRE_END_TIME_IN_UT
C_STRING)
    if instance_id is not None:
        request.set_InstanceIds(json.dumps([instance_id]))
    if security_group_id:
        request.set_SecurityGroupId(security_group_id)
    request.set_PageNumber(page_number)
    request.set_PageSize(page_size)
    return _send_request(request)
# check the instances is renew or not
def describe_instance_auto_renew_setting(instance_ids, expected_a
uto_renew=True):
    describe_request = DescribeInstanceAutoRenewAttributeRequest()
    describe_request.set_InstanceId(instance_ids)
    response_detail = _send_request(request=describe_request)
    failed_instance_ids = ''
    if response_detail is not None:
        attributes = response_detail.get('InstanceRenewAttributes').
get('InstanceRenewAttribute')
        if attributes:
            for item in attributes:
                auto_renew_status = item.get('AutoRenewEnabled')
                if auto_renew_status != expected_auto_renew:
                    failed_instance_ids += item.get('InstanceId') +
 ','
    if len(failed_instance_ids) > 0:
        logging.error("instance %s auto renew not match expect %s.",
failed_instance_ids,
                      expected_auto_renew)
def setting_instance_auto_renew(instance_ids, auto_renew=True):
    logging.info('execute enable auto renew ' + instance_ids)
    request = ModifyInstanceAutoRenewAttributeRequest();
    request.set_Duration(1);
    request.set_AutoRenew(auto_renew);
    request.set_InstanceId(instance_ids)
    _send_request(request)
    describe_instance_auto_renew_setting(instance_ids, auto_renew)
# if using the instance id can be found means the instance is not
renew successfully.
def check_instance_need_renew(instance_id):
    response = describe_need_renew_instance(instance_id=instance_id)
    if response is not None:
        return response.get('TotalCount') == 1
    return False
# 续费一个实例一个月
def renew_instance(instance_id, period='1'):
    need_renew = check_instance_need_renew(instance_id)
    if need_renew:
        _renew_instance_action(instance_id=instance_id, period=period)
        # describe_need_renew_instance(instance_id=instance_id,
check_need_renew=False)
def _renew_instance_action(instance_id, period='1'):
```

```
    request = RenewInstanceRequest()
    request.set_Period(period)
    request.set_InstanceId(instance_id)
    response = _send_request(request)
    logging.info('renew %s ready, output is %s ', instance_id,
response)
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
if __name__ == '__main__':
    logging.info("Renew ECS Instance by OpenApi!")
    # 查询在指定的时间范围内是否有需要续费的实例。
    describe_need_renew_instance()
    # 续费一个实例，直接执行扣费
    renew_instance('i-1111')
    # 查询实例自动续费的状态
    # describe_instance_auto_renew_setting('i-1111,i-2222')
    # 设置实例自动续费
    # setting_instance_auto_renew('i-1111,i-2222')
```

如您想了解 ECS 中 API 的其它操作，请参考 *ECS中的API操作*。

4.6 管理抢占式实例

本文介绍了如何使用阿里云 ECS SDK 合理快速地创建并管理抢占式实例。

准备工作

在执行操作之前，您需要：

- 了解能满足您业务要求的实例规格和地域。
- 熟悉了解阿里云 ECS SDK 的基础知识和调用方法。详细信息，请参考 *SDK 使用说明*。
- 抢占式实例代码需要依赖的 ECS SDK 版本 4.2.0 以上。以 Java POM 依赖为例，修改引入 pom 依赖：

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>3.2.8</version>
</dependency>
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-ecs</artifactId>
  <version>4.2.0</version>
</dependency>
```

查询地域及可用的实例规格

使用 *#unique_39* 查询可以创建抢占式实例的地域以及可用的实例规格。示例代码如下所示。

- OpenApiCaller.java

```
public class OpenApiCaller {
  IClientProfile profile;
  IAcsClient client;
  public OpenApiCaller() {
      profile = DefaultProfile.getProfile("cn-hangzhou", AKSUtil.
accessKeyId, AKSUtil.accessKeySecret);
      client = new DefaultAcsClient(profile);
  }
  public  <T extends AcsResponse> T doAction(AcsRequest<T> var1) {
      try {
          return  client.getAcsResponse(var1);
      } catch (Throwable e) {
          e.printStackTrace();
          return null;
      }
  }
}
```

- DescribeZonesSample.java

```
public class DescribeZonesSample {
  public static void main(String[] args) {
      OpenApiCaller caller = new OpenApiCaller();
      DescribeZonesRequest request = new DescribeZonesRequest();
      request.setRegionId("cn-zhangjiakou");//可以通过 DescribeRe
gionsRequest 获取每个地域的 RegionId
      request.setSpotStrategy("SpotWithPriceLimit");//对于查询是否可购
买抢占式实例此项必填
      request.setInstanceChargeType("PostPaid");//后付费模式，抢占式实例
必须是后付费模式
      DescribeZonesResponse response = caller.doAction(request);
      System.out.println(JSON.toJSONString(response));
  }
}
```

以下为输出结果，可以查看每个地域各个地域可供选择的实例规格、磁盘类型、网络类型等信息。

```
{
  "requestId": "388D6321-E587-470C-8CFA-8985E2963DAE",
  "zones": [
      {
          "localName": "华北 3 可用区 A",
          "zoneId": "cn-zhangjiakou-a",
          "availableDiskCategories": [
              "cloud_ssd",
              "cloud_efficiency"
          ],
          "availableInstanceTypes": [
              "ecs.e4.large",
              "ecs.n4.4xlarge",
              "ecs.sn2.medium",
              "ecs.i1.2xlarge",
              "ecs.se1.2xlarge",
              "ecs.n4.xlarge",
              "ecs.se1ne.2xlarge",
              "ecs.se1.large",
              "ecs.sn2.xlarge",
              "ecs.se1ne.xlarge",
              "ecs.xn4.small",
```

```
                "ecs.sn2ne.4xlarge",
                "ecs.se1ne.4xlarge",
                "ecs.sn1.medium",
                "ecs.n4.8xlarge",
                "ecs.mn4.large",
                "ecs.e4.2xlarge",
                "ecs.mn4.2xlarge",
                "ecs.mn4.8xlarge",
                "ecs.n4.2xlarge",
                "ecs.e4.xlarge",
                "ecs.sn2ne.large",
                "ecs.sn2ne.xlarge",
                "ecs.sn1ne.large",
                "ecs.n4.large",
                "ecs.sn1.3xlarge",
                "ecs.e4.4xlarge",
                "ecs.sn1ne.2xlarge",
                "ecs.e4.small",
                "ecs.i1.4xlarge",
                "ecs.se1.4xlarge",
                "ecs.sn2ne.2xlarge",
                "ecs.sn2.3xlarge",
                "ecs.i1.xlarge",
                "ecs.n4.small",
                "ecs.sn1ne.4xlarge",
                "ecs.mn4.4xlarge",
                "ecs.sn1ne.xlarge",
                "ecs.se1ne.large",
                "ecs.sn2.large",
                "ecs.i1-c5d1.4xlarge",
                "ecs.sn1.xlarge",
                "ecs.sn1.large",
                "ecs.mn4.small",
                "ecs.mn4.xlarge",
                "ecs.se1.xlarge"
            ],
            "availableResourceCreation": [
                "VSwitch",
                "IoOptimized",
                "Instance",
                "Disk"
            ],
            "availableResources": [
                {
                    "dataDiskCategories": [
                        "cloud_ssd",
                        "cloud_efficiency"
                    ],
                    "instanceGenerations": [
                        "ecs-3",
                        "ecs-2"
                    ],
                    "instanceTypeFamilies": [
                        "ecs.mn4",
                        "ecs.sn1",
                        "ecs.sn2",
                        "ecs.sn1ne",
                        "ecs.xn4",
                        "ecs.i1",
                        "ecs.se1",
                        "ecs.e4",
                        "ecs.n4",
                        "ecs.se1ne",
                        "ecs.sn2ne"
```

```
                    ],
                    "instanceTypes": [
                        "ecs.n4.4xlarge",
                        "ecs.sn2.medium",
                        "ecs.i1.2xlarge",
                        "ecs.se1.2xlarge",
                        "ecs.n4.xlarge",
                        "ecs.se1ne.2xlarge",
                        "ecs.se1.large",
                        "ecs.sn2.xlarge",
                        "ecs.se1ne.xlarge",
                        "ecs.xn4.small",
                        "ecs.sn2ne.4xlarge",
                        "ecs.se1ne.4xlarge",
                        "ecs.sn1.medium",
                        "ecs.n4.8xlarge",
                        "ecs.mn4.large",
                        "ecs.mn4.2xlarge",
                        "ecs.mn4.8xlarge",
                        "ecs.n4.2xlarge",
                        "ecs.sn2ne.large",
                        "ecs.sn2ne.xlarge",
                        "ecs.sn1ne.large",
                        "ecs.n4.large",
                        "ecs.sn1.3xlarge",
                        "ecs.sn1ne.2xlarge",
                        "ecs.e4.small",
                        "ecs.i1.4xlarge",
                        "ecs.se1.4xlarge",
                        "ecs.sn2ne.2xlarge",
                        "ecs.sn2.3xlarge",
                        "ecs.i1.xlarge",
                        "ecs.n4.small",
                        "ecs.sn1ne.4xlarge",
                        "ecs.mn4.4xlarge",
                        "ecs.sn1ne.xlarge",
                        "ecs.se1ne.large",
                        "ecs.sn2.large",
                        "ecs.i1-c5d1.4xlarge",
                        "ecs.sn1.xlarge",
                        "ecs.sn1.large",
                        "ecs.mn4.small",
                        "ecs.mn4.xlarge",
                        "ecs.se1.xlarge"
                    ],
                    "ioOptimized": true,
                    "networkTypes": [
                        "vpc"
                    ],
                    "systemDiskCategories": [
                        "cloud_ssd",
                        "cloud_efficiency"
                    ]
                }
            ],
            "availableVolumeCategories": [
                "san_ssd",
                "san_efficiency"
            ]
        }
    ]
```

```
}
```

查询抢占式实例的历史价格

使用 *#unique_40* 查询抢占式实例最近 30 天的价格变化数据，获得最佳性价比的地域和规格信息，示例代码（DescribeSpotPriceHistorySample.java）如下。

```
public class DescribeSpotPriceHistorySample {
    public static void main(String[] args) {
        OpenApiCaller caller = new OpenApiCaller();
        List<DescribeSpotPriceHistoryResponse.SpotPriceType> result =
new ArrayList<DescribeSpotPriceHistoryResponse.SpotPriceType>();
        int offset = 0;
        while (true) {
            DescribeSpotPriceHistoryRequest request = new DescribeSp
otPriceHistoryRequest();
            request.setRegionId("cn-hangzhou");//可以通过 DescribeRe
gionsRequest 获取可购买的每个地域的 RegionId
            request.setZoneId("cn-hangzhou-b");//可用区必填
            request.setInstanceType("ecs.sn2.medium");//参考 DescribeZo
nes 返回的实例类型，必填
            request.setNetworkType("vpc");//参考 DescribeZones 返回的网
络类型，必填
//            request.setIoOptimized("optimized");//是否 I/O 优化类型，
DescribeZones 返回的 IoOptimized，选填
//            request.setStartTime("2017-09-20T08:45:08Z");//价格开始时
间，选填，默认 3 天内数据
//            request.setEndTime("2017-09-28T08:45:08Z");//价格结束时
间，选填
            request.setOffset(offset);
            DescribeSpotPriceHistoryResponse response = caller.
doAction(request);
            if (response != null && response.getSpotPrices() != null)
 {
                result.addAll(response.getSpotPrices());
            }
            if (response.getNextOffset() == null || response.
getNextOffset() == 0) {
                break;
            } else {
                offset = response.getNextOffset();
            }
        }
        if (!result.isEmpty()) {
            for (DescribeSpotPriceHistoryResponse.SpotPriceType
spotPriceType : result) {
                System.out.println(spotPriceType.getTimestamp() + "---
>spotPrice:" + spotPriceType.getSpotPrice() + "---->originPrice:" +
spotPriceType.getOriginPrice());
            }
            System.out.println(result.size());
        } else {
        }
    }
}
```

以下为返回结果示例。

```
2017-09-26T06:28:55Z--->spotPrice:0.24---->originPrice:1.2
2017-09-26T14:00:00Z--->spotPrice:0.36---->originPrice:1.2
2017-09-26T15:00:00Z--->spotPrice:0.24---->originPrice:1.2
```

```
2017-09-27T14:00:00Z--->spotPrice:0.36---->originPrice:1.2
2017-09-27T15:00:00Z--->spotPrice:0.24---->originPrice:1.2
2017-09-28T14:00:00Z--->spotPrice:0.36---->originPrice:1.2
2017-09-28T15:00:00Z--->spotPrice:0.24---->originPrice:1.2
2017-09-29T06:28:55Z--->spotPrice:0.24---->originPrice:1.2
```

重复以上步骤，您可以判断出该规格资源在可用区的价格变化趋势和最近价格。

说明:

您可以通过平均价格和最高价格来决定是否可以接受购买该抢占式实例，也可以通过更加合理的数据模型来分析历史价格数据，随时调整创建资源的规格和可用区，到达最佳性价比。

创建抢占式实例

在创建抢占式实例之前，您需要完成以下工作：

- 如果您使用自定义镜像创建抢占式实例，必须已经 *#unique_41*。
- 在控制台 **创建安全组**，或者使用 *CreateSecurityGroup* 创建安全组，并获取安全组 ID（SecurityGroupId）。
- 在控制台创建 *VPC*和交换机，或者使用 *#unique_46* 和 *#unique_47* 创建，并获取交换机 ID（VSwitchId）。

使用 *#unique_12* 创建抢占式实例。示例代码（CreateInstaneSample.java）如下。

```
public class CreateInstaneSample {
    public static void main(String[] args) {
        OpenApiCaller caller = new OpenApiCaller();
        CreateInstanceRequest request = new CreateInstanceRequest();
        request.setRegionId("cn-hangzhou");//地域 ID
        request.setZoneId("cn-hangzhou-b"); //可用区ID
        request.setSecurityGroupId("sg-bp11nhf94ivkdxwb2gd4");//提前创
建的安全组 ID
        request.setImageId("centos_7_03_64_20G_alibase_20170818.vhd
");//建议选择您自己在该地域准备的自定义镜像
        request.setVSwitchId("vsw-bp164cyonthfudn9kj5br");//VPC 类型需
要交换机 ID
        request.setInstanceType("ecs.sn2.medium"); //填入您询价后需要购买
的规格
        request.setIoOptimized("optimized");//参考 DescirbeZones 返回参
数
        request.setSystemDiskCategory("cloud_ssd");//参考 DescirbeZones
 返回参数，多选一 cloud_ssd，cloud_efficiency，cloud
        request.setSystemDiskSize(40);
        request.setInstanceChargeType("PostPaid");//抢占式实例必须后付费
        request.setSpotStrategy("SpotWithPriceLimit");//SpotWithPr
iceLimit 出价模式，SpotAsPriceGo 不用出价，最高按量付费价格
        request.setSpotPriceLimit(0.25F);//SpotWithPriceLimit 出价模式生
效，您能接受的最高价格，单位为元每小时，必须高于当前的市场成交价才能成功
        CreateInstanceResponse response = caller.doAction(request);
        System.out.println(response.getInstanceId());
    }
```

```
}
```

回收抢占式实例

当抢占式实例可能会因为价格因素或者市场供需变化而被强制回收。此时会触发抢占式实例的中断。释放前，抢占式实例会进入锁定状态，提示实例将会被自动回收。您可以针对实例回收状态自动化处理实例的退出逻辑。

目前，您可以通过以下任一种方式来获取抢占式实例的中断锁定状态：

- 通过 实例元数据 获取。运行以下命令：

  ```
  curl 'http://100.100.100.200/latest/meta-data/instance/spot/
  termination-time'
  ```

  如果返回为空，说明实例可持续使用。如果返回类似 `2015-01-05T18:02:00Z` 格式的信息（UTC 时间），说明实例将于这个时间释放。

- 使用 *#unique_13*，根据返回的 `OperationLocks` 判断实例是否进入 待回收 状态。代码示例（DescribeInstancesSample.java）如下。

  ```
  public class DescribeInstancesSample {
    public static void main(String[] args) throws InterruptedException
   {
        OpenApiCaller caller = new OpenApiCaller();
        JSONArray allInstances = new JSONArray();
        allInstances.addAll(Arrays.asList("i-bp18hgfai8ekoqwo0y2n", "i
  -bp1ecbyds24ij63w146c"));
        while (!allInstances.isEmpty()) {
            DescribeInstancesRequest request = new DescribeIn
  stancesRequest();
            request.setRegionId("cn-hangzhou");
            request.setInstanceIds(allInstances.toJSONString());//指定
  实例 ID，效率最高
            DescribeInstancesResponse response = caller.doAction(
  request);
            List<DescribeInstancesResponse.Instance> instanceList =
  response.getInstances();
            if (instanceList != null && !instanceList.isEmpty()) {
                for (DescribeInstancesResponse.Instance instance :
  instanceList) {
                    System.out.println("result:instance:" + instance.
  getInstanceId() + ",az:" + instance.getZoneId());
                    if (instance.getOperationLocks() != null) {
                        for (DescribeInstancesResponse.Instance.
  LockReason lockReason : instance.getOperationLocks()) {
                            System.out.println("instance:" + instance.
  getInstanceId() + "-->lockReason:" + lockReason.getLockReason() + ",
  vmStatus:" + instance.getStatus());
                            if ("Recycling".equals(lockReason.
  getLockReason())) {
                                //do your action
                                System.out.println("spot instance will
   be recycled immediately, instance id:" + instance.getInstanceId());
                                allInstances.remove(instance.
  getInstanceId());
                            }
  ```

```
                        }
                    }
                }
                System.out.print("try describeInstances again later
 ...");
                Thread.sleep(2 * 60 * 1000);
            } else {
                break;
            }
        }
    }
}
```

触发回收时输出结果如下：

```
instance:i-bp1ecbyds24ij63w146c-->lockReason:Recycling,vmStatus:
Stopped
spot instance will be recycled immediately, instance id:i-bp1ecbyds2
4ij63w146c
```

其他操作

您还可以启动、停止、释放抢占式实例。具体的操作与一般按量付费实例没有区别。可以参考 OpenAPI 文档：

- **启动实例：** *#unique_14*
- **停止实例：** *#unique_33*
- **释放实例：** *#unique_31*