Elastic Compute Service

Best Practices

MORE THAN JUST CLOUD | Alibaba Cloud

Best Practices

Use OpenAPI to manage ECS

Use OpenAPI to Create Instance

In addition to the ECS Console or Buy Page, you can also use OpenAPI code to elastically create and manage ECS instances. This article describes how to create an ECS instance using Python.

When creating an ECS instance, pay attention to the following APIs:

- Create an ECS instance
- Query an instance list
- Start an ECS instance
- Allocate a public IP address

Create a Pay-As-You-Go ECS instance

Mandatory attributes:

- SecurityGroupId: Security group ID. A security group is used to implement the configurations of a group of instances based on firewall rules to protect the network access requests of the instances. It is recommended that only necessary access rules, rather than all access rules, be enabled when you configure security group access rules. You can create a security group on the ECS Console.
- InstanceType: Instance type. Refer to the ECS Buy Page. The option “one-core 2GB n1.small” indicates that the input parameter is “ecs.n1.small”.
- ImageId: Image ID. Refer to the image list on the ECS console. You can filter public images or custom images.

For more parameter settings, refer to Create an ECS instance.

Create an ECS instance

The following code shows creating an I/O optimized classic-network ECS instance with SSD as system disk and “cloud_ssd” as disk parameter.

```
# create one after pay ecs instance.
def create_after_pay_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')
response = _send_request(request)
instance_id = response.get('InstanceId')
logging.info("instance %s created task submit successfully.", instance_id)
return instance_id;
```

An instance ID is returned after the ECS instance is created successfully. If creation fails, an error code is returned. Since there are many parameters, you can make adjustments by visiting the ECS Buy Page.

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

ECS lifecycle

For details about the operations in different ECS status, refer to ECS Instance Lifecycle.

Only when an instance is in the Stopped status, can the Start operation be performed, and only when it is in the Running status, can the Stop operation be performed. To query the ECS status, you can filter the instance list by inputting the parameter Instance ID. When you call DescribeInstancesRequest, input a JSON array of strings to query the resource status. When you query the status of a single instance, we suggest using DescribeInstances rather than DescribeInstanceAttribute, because the former API returns more attributes and content than the latter.

The following code is used to check the instance status. The system returns instance details only when the instance status conforms to the input parameters.

```
# 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;
```

Start an ECS instance

After an ECS instance is created successfully, the default instance status is Stopped. To change to the Running status, send the Start command.

```

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

Stop an ECS instance

To stop an ECS instance, just use the input instance ID.

```

def stop_instance(instance_id):
request = StopInstanceRequest()
request.set_InstanceId(instance_id)
_send_request(request)
```

Enable “ECS automatic startup” when creating an ECS instance

The ECS Start and Stop operations are asynchronous. You can perform the operation when the script is creating an ECS instance and detecting if it is in an appropriate status.

After you obtain the ID of a successfully created ECS instance, check whether the instance is in the Stopped status. If it is in the Stopped status, send the Start ECS command and wait until the ECS status changes to 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 < 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 < 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;
```

Allocate a public IP address

If you specify the public network bandwidth when creating an ECS instance, you need to call an API to allocate a public IP address to the instance for public network access. For details, refer to Allocate a public IP address.

Create an ECS instance in the Subscription mode

OpenAPI also supports creating ECS instances in the Subscription mode, in addition to Pay-As-You-Go ECS instances. The process for creating an ECS instance in the Subscription mode is different from that on Alibaba Cloud’ s website. Fees are automatically deducted for an ECS instance created in the Subscription mode. Before you create an ECS instance, ensure that you have sufficient account balance or credit amount, so that the fees can be deducted directly during creation.

When creating an ECS instance in Subscription mode, you only need to specify the payment option and duration. In the following code, the duration is set to one month.

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

The complete code for creating an ECS instance in the Subscription mode is as follows:

```
# 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;
```

Complete code

See the complete code as follows. You can use your resource parameters for configuration.

```
#  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_type=INSTANCE_TYPE,
security_group_id=SECURITY_GROUP_ID)
check_instance_running(instance_id=instance_id)

# create one after pay ecs instance.
def create_after_pay_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')
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 < 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 < 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()
```

In addition to using Alibaba Cloud’ s ECS Console for resource creation and daily management, you can also use OpenAPI to manage and customize resources. OpenAPI allows you to manage and configure ECS instances with greater flexibility.

Alibaba Cloud encapsulates OpenAPI in an SDK to integrate ECS instance management into existing systems. This article describes how to manage ECS instances through OpenAPI based on Python development. You can develop ECS instances easily even if you do not have Python development experience.

Get the access key for a RAM sub-account

An access key (Access Key ID and Access Key Secret) is required when you want to use OpenAPI to manage ECS instances. To keep your cloud service secure, you have to create a RAM sub-account and generate an access key for it, and authorize the sub-account to manage ECS resources only. Then, you can use the RAM sub-account and its access key to manage ECS resources by using OpenAPI.

Follow the steps to get the access key for a RAM sub-account.

1. Create a RAM sub-account and get the access key.
2. Grant permissions to the RAM sub-account directly. To manage ECS resources, you have to grant **AliyunECSFullAccess** to the sub-account.

Install the ECS Python SDK

Ensure that the Python runtime environment has been installed. This article uses Python 2.7+.

```
pip install aliyun-python-SDK-ecs
```

If you do not have the permission, switch to sudo to continue.

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

The SDK version is **2.1.2**.

Hello Alibaba Cloud

Create the file **hello_ecs_api.py**. To use SDK, you have to use the access key of the RAM sub-account to instantialize an AcsClient object.

> The access key allows the RAM sub-account to access Alibaba Cloud APIs and give you full access to the sub-account. Keep them safe.

```
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 Secret', 'cn-beijing')
```

You can develop your first application after the AcsClient object is instantiated. Query the list of regions that your account supports. For details, refer to Query the list of available regions.

```
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()
```

In the command line, run python hello_ecs_api.py to obtain a list of supported regions. The output is similar to the following.

```
[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']
```

Query the list of ECS instances in the current region

The process for querying the instance list is similar to the region list. You only need to replace the input parameter DescribeRegionsRequest with DescribeInstancesRequest. For a full list of query parameters, refer to Query an instance list.

```
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
```

The output is as follows.

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

For a full list of APIs, refer to ECS API overview. If you want to query a list of disks, replace DescribeInstancesRequest with DescribeDisksRequest.

Complete code

The following is the complete code of the operations described in this document.

```
#  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 Secret', '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()
```

If you want to learn other API operations in ECS, refer to ECS API operation.

One important feature of ECS is on-demand resource creation. You can create custom resources elastically on demand during peak service hours, and then release those resources after service computing is completed. This document describes how to easily release ECS instances and achieve elasticity.

This document covers the following APIs:

- DeleteInstance
- ModifyInstanceAutoReleaseTime
- StopInstance
- Instance list query API

After an ECS instance is released, the physical resources used by the instance are recycled, including disks and snapshots. The data of the instance is completely lost and can never be recovered. If you want to retain the data, we recommend that you create snapshots of disks before releasing the ECS instance. The snapshots can be directly used to create a new ECS instance.

To release an ECS instance, you must stop it first. If any application is affected after the ECS instance is stopped, restart the instance.

Stop an ECS instance

Use the **StopInstance** interface to stop an ECS instance, regardless of the billing method of the instance. The stop command is as follows. When the ForceStop parameter is set to true, the ECS instance is stopped directly but data is not necessarily written to a disk, similar to power failure. Therefore, if you want to release an instance, set ForceStop to 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)
```

Release an ECS instance

Use the **DeleteInstance** interface to release an ECS instance.

When the ECS instance is in the **Stopped** status, you can release it. The API has only two request parameters:

- InstanceId: Instance ID
- Force: If this parameter is set to “true”, the ECS instance is released forcibly even when it is not in the **Stopped** status. Use caution when setting this parameter. Release by mistake may affect your services.

```
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)
```

The following response is returned when an ECS instance is released successfully:

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

If you release an ECS instance when it is not in the **Stopped** status, an error occurs:

```
{"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."}
```

Set the automatic release time for an ECS instance

You can set the automatic release time for an ECS instance to simplify instance management. When the set time is reached, Alibaba Cloud releases your ECS instance automatically. Use the **ModifyInstanceAutoReleaseTime** to set the automatic release time for an ECS instance.

Note:
The automatic release time follows the ISO8601 standard in UTC time. The format is yyyy-MM-ddTHH:mm:ssZ. If the seconds place is not 00, it is automatically set to start from the current minute.
The automatic release time must be at least half an hour later than the current time, and must not be more than 3 years since the current time.

```
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)
```

Run the command set_instance_auto_release_time('i-1111', '2017-01-30T00:00:00Z') to set the time. Then you can use the DescribeInstances to query the automatic release time.

```
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
```

If you want to cancel the automatic release due to service changes, run the set_instance_auto_release_time('i-1111') command to set the automatic release time to null.

Complete example code

Note: Proceed with caution when releasing ECS instances.

```
#  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.ModifyInstanceAutoReleaseTimeRequest 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) > 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)
```

If you want to learn other API operations in ECS, see ECS API operation.

Lifecycle is important to ECS instances of the Subscription billing method. If you fail to renew your ECS instance on time, the instance may be locked or even released, thus affecting your service continuity. In addition to the ECS console or the ECS purchase page, Alibaba Cloud provides you with APIs to view the resource expiration time and renew your instance.

This document covers the following APIs:

- DescribeInstances
- ModifyInstanceAutoRenewAttribute

Query ECS instances by expiration time

Use the DescribeInstances interface to query the instances that will expire within the specified time range by setting the filter parameters ExpiredStartTime and ExpiredEndTime. The time parameters follow the ISO8601 standard in UTC time, using the format yyyy-MM-ddTHH:mmZ. The system returns a list of instances that will expire within the specified time range.

Note: If you want to filter by security group, add the security group ID.

```
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_group_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_UTC_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)
```

Query and enable automatic ECS instance renewal

You can use the ModifyInstanceAutoRenewAttribute interface to query and set automatic renewal. The API supports only ECS instances of the Subscription billing method. If you use the API on a Pay-As-You-Go instance, an error will be returned.

Query automatic renewal setting

To query the automatic renewal setting, only the instance ID is required. You can query the automatic renewal status of up to 100 ECS instances of the Subscription billing method at a time. Use commas to separate multiple instance IDs.

```
# 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')
```

The following content is returned:

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

If automatic renewal is set, the returned attribute AutoRenewEnabled is true. If automatic renewal is not set, the attribute is false.

Enable automatic renewal for ECS instances

To enable automatic renwal for ECS instances, three input parameters are required:

- InstanceId: You can set automatic renewal for up to 100 ECS instances of the Subscription billing method at a time. Use commas to separate multiple instance IDs.
- Duration: Set to 1, 2, 3, 6, or 12, in unit of Month.
- AutoRenew: Set to true to enable automatic renewal.

  Note: Set to false to disable automatic renewal.

```
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)
```

When the operation is successful, the following response is returned:

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

You can perform a query after successful renewal. The system will return the renewal duration and the status of automatic renewal (true/false).

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

Complete example code

```
#  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.DescribeInstanceAutoRenewAttributeRequest import \
DescribeInstanceAutoRenewAttributeRequest
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import DescribeInstancesRequest
from aliyunsdkecs.request.v20140526.ModifyInstanceAutoRenewAttributeRequest 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_group_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_UTC_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_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') + ','
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

# Renew an instance for a month
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!")
# Query whether there is any instance that needs to be renewed within the specified time range.
describe_need_renew_instance()
# Renew an instance by direct fee deduction
renew_instance('i-1111')
# Query the status of automatic renewal
# describe_instance_auto_renew_setting('i-1111,i-2222')
# Set automatic instance renewal
# setting_instance_auto_renew('i-1111,i-2222')
```

If you want to learn other API operations in ECS, see ECS API operation.

Instance custom data

User-defined scripts are a type of script provided by Alibaba Cloud for users to customize the startup behaviors of ECS instances. For details, refer to User-defined data.

This document takes a Linux instance for example to introduce how to use user-defined scripts to configure your own yum repository, NTP service, and DNS service when creating a Linux instance. User-defined scripts also enables you to configure NTP service and DNS service for a Windows instance.

Scenario

Currently, when a Linux instance is started, Alibaba Cloud automatically configures pre-defined yum repository, NTP service, and DNS service for the instance. However, if you want to have your own yum repository, NTP service, and DNS service, use user-defined scripts to implement this requirement.

Note:

- If you are using a custom yum repository, Alibaba Cloud does not provide support for it.
- If you are using a custom NTP service, Alibaba Cloud does not provide time service.

Customize yum repository, NTP service, and DNS service

Follow the steps below to customize yum repository, NTP service, and DNS service for a Linux instance when creating it.

Log on to the ECS console and create an instance. Configure the instance as follows:

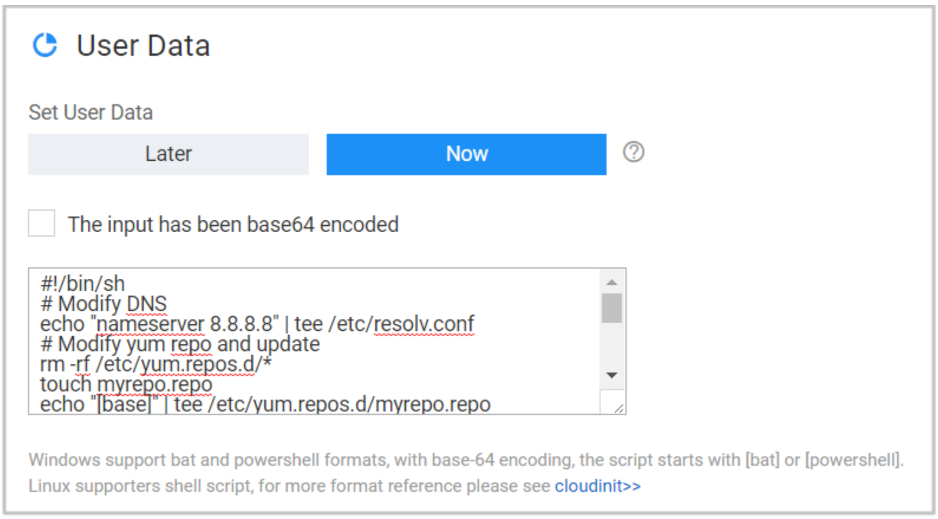
- **Network Type**: Select **VPC**.
- **Instance Type**: Select an I/O-optimized instance.
- **Operating System**: Select CentOS 7.2 in **Public Image** tab.
- **Security Setup**: Select one to meet your requirement.

Enter the following script in the **User Data** box on the instance creation page.

```
#!/bin/sh
# Modify DNS
echo "nameserver 8.8.8.8" | tee /etc/resolv.conf
# Modify yum repo and update
rm -rf /etc/yum.repos.d/*
touch myrepo.repo
echo "[base]" | tee /etc/yum.repos.d/myrepo.repo
echo "name=myrepo" | tee -a /etc/yum.repos.d/myrepo.repo
echo "baseurl=http://mirror.centos.org/centos" | tee -a /etc/yum.repos.d/myrepo.repo
echo "gpgcheck=0" | tee -a /etc/yum.repos.d/myrepo.repo
echo "enabled=1" | tee -a /etc/yum.repos.d/myrepo.repo
yum update -y
# Modify NTP Server
echo "server ntp1.aliyun.com" | tee /etc/ntp.conf
systemctl restart ntpd.service
```

Note:

- The first line must be #!/bin/sh, with no leading space.
- Do not add unnecessary spaces or carriage return characters in the full text.
- You can customize URLs of your own DNS server, NTP Server, and yum repository based on the instance situations.
- The preceding content applies to CentOS 7.2 images. If you are using other images, modify the scripts as needed.
- You can also define the yum repository in the scripts of the Cloud Config type, but it is not recommended because it is not flexible enough to get adapted to Alibaba Cloud that may pre-configure some yum repository. Scripts of script type is recommended for changing the yum repository.

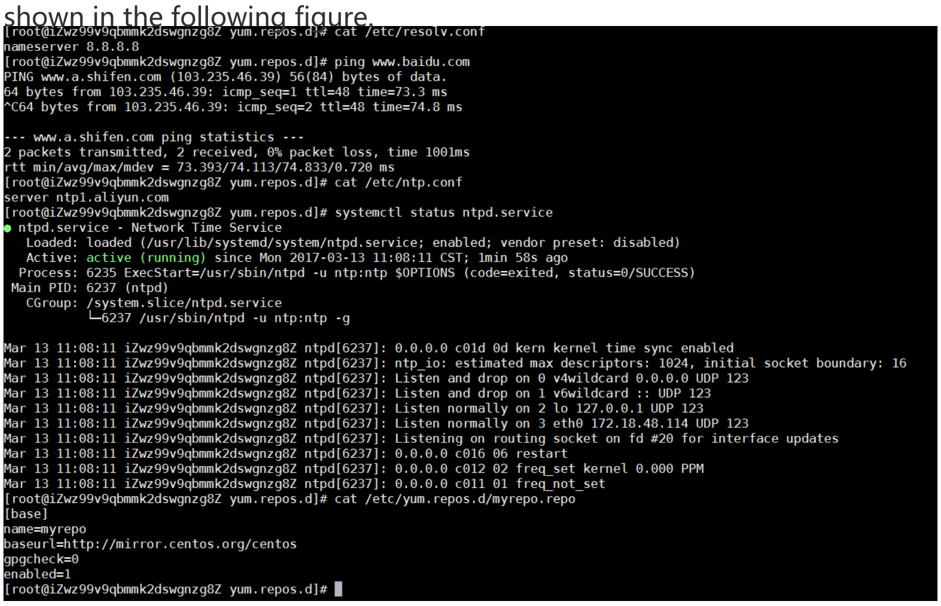


```
#!/bin/sh
# Modify DNS
echo "nameserver 8.8.8.8" | tee /etc/resolv.conf
# Modify yum repo and update
rm -rf /etc/yum.repos.d/*
touch myrepo.repo
echo "[base]" | tee /etc/yum.repos.d/myrepo.repo
```

Windows support bat and powershell formats, with base-64 encoding, the script starts with [bat] or [powershell].
Linux supporters shell script, for more format reference please see cloudinit>>

After the configuration is completed, click **Buy Now** and activate the instance following instructions on the page.

After the instance is created, you can log on to the instance to view the implementation details, as shown in the following figure.



```
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# cat /etc/resolv.conf
nameserver 8.8.8.8
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# ping www.baidu.com
PING www.a.shifen.com (103.235.46.39) 56(84) bytes of data.
64 bytes from 103.235.46.39: icmp_seq=1 ttl=48 time=73.3 ms
^C64 bytes from 103.235.46.39: icmp_seq=2 ttl=48 time=74.8 ms

--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 73.393/74.113/74.833/0.720 ms
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# cat /etc/ntp.conf
server ntp1.aliyun.com
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# systemctl status ntpd.service
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2017-03-13 11:08:11 CST; 1min 58s ago
  Process: 6235 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 6237 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─6237 /usr/sbin/ntpd -u ntp:ntp -g

Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: 0.0.0.0 c01d 0d kern kernel time sync enabled
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: ntp_io: estimated max descriptors: 1024, initial socket boundary: 16
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: Listen and drop on 1 v6wildcard :: UDP 123
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: Listen normally on 2 lo 127.0.0.1 UDP 123
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: Listen normally on 3 eth0 172.18.48.114 UDP 123
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: Listening on routing socket on fd #20 for interface updates
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: 0.0.0.0 c016 06 restart
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
Mar 13 11:08:11 iZwz99v9qbmmk2dswgnzg8Z ntpd[6237]: 0.0.0.0 c011 01 freq_not_set
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# cat /etc/yum.repos.d/myrepo.repo
[base]
name=myrepo
baseurl=http://mirror.centos.org/centos
gpgcheck=0
enabled=1
[root@iZwz99v9qbmmk2dswgnzg8Z yum.repos.d]# █
```

The preceding figure shows that you have successfully customized the DNS service, the NTP service, and the yum repository.

User-defined scripts are a type of script provided by Alibaba Cloud for users to customize the startup behaviors of ECS instances. For details, refer to Instance custom data.

This document takes a Linux instance as an example to introduce how to use user-defined scripts to create a new account with the root user privilege for a Linux instance when creating the instance. User-defined scripts also enable you to create a new account with the administrator privilege for a Windows instance.

Scenario

Use user-defined scripts of instances if you want to achieve the following results when creating a Linux ECS instance:

- Disable the default **root** account that comes with a Linux ECS instance. You can use the script to customize how to disable the root user and how many privileges of the root user to be disabled.
- Create a new account with the root user privilege and customize the account name.
- Use only SSH key pairs, but not user password, for remote logon to manage the instance using the newly-created account with the root user privilege.
- If this newly-created account requires performing some operations that only can be done by a user with root user privilege, the sudo command can be used without a password for privilege escalation.

Create a new account with the root user privilege

Follow the steps below to create a new account with the root user privilege.

Log on to the ECS console and create a Linux instance. Configure the instance as follows:

- **Network Type**: Select **VPC**.
- **Instance Type**: Select an I/O-optimized instance.
- **Operating System**: Select CentOS 7.2 in **Public Image** tab.
- **Security Setup**: select **Later**.

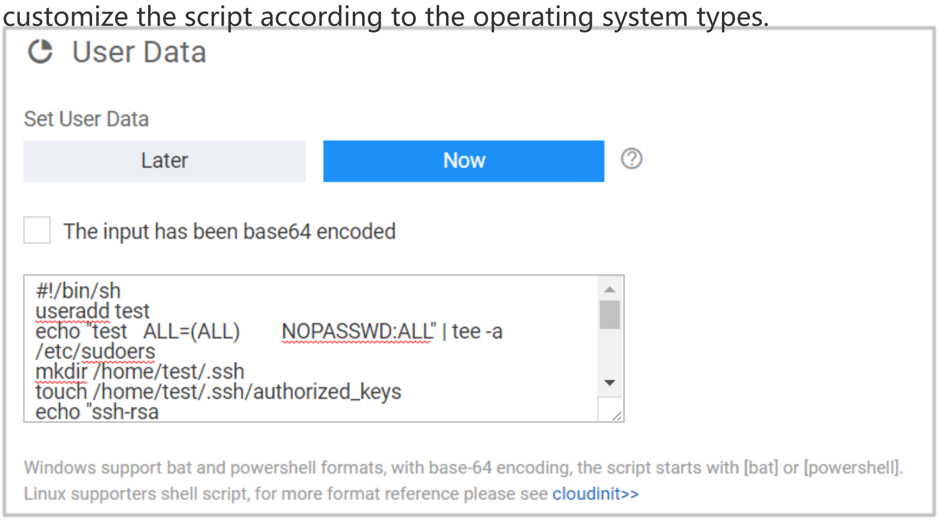
Enter the following script in the **User Data** box on the instance creation page:

```
#!/bin/sh
useradd test
echo "test ALL=(ALL) NOPASSWD:ALL" | tee -a /etc/sudoers
mkdir /home/test/.ssh
touch /home/test/.ssh/authorized_keys
echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAhGqhEh/rGbIMCGItFVtYpsXPQrCaunGJKZVIWtINrGZwusLc290qDZ
93KCeb8o6X1Iby1Wm+psZY8THE+/BsXq0M0HzfkQZD2vXuhRb4xi1z98JHskX+0jnbjqYGY+Brgai9BvKDX
TTSyJtCYUnEKxvcK+d1ZwxbNuk2QZ0ryHESDbSaczlNFgFQEDxhCrvko+zWLjTVnomVUDhdMP2g6fZ0tgF
VwkJFV0bE7oob3NOVcrx2TyhfcAjA4M2/Ry7U2MFADDC+EVkpoVDm0SOT/hYJgaVM1xMDlSeE7kzX7yZ
bJLR1XAWV1xzZkNclY5w1kPnW8qMYuSwhpXzt4gsF0w== rsa-key-20170217" | tee -a
/home/test/.ssh/authorized_keys
```

Note:

- The first line must be #!/bin/sh with no leading space.
- Do not enter unnecessary spaces or carriage return characters in the text.
- The last line is your public key. You can define it.

- You can add other configuration in the script, as you need.
- The example script only applies to CentOS 7.2. If you are using other images, customize the script according to the operating system types.

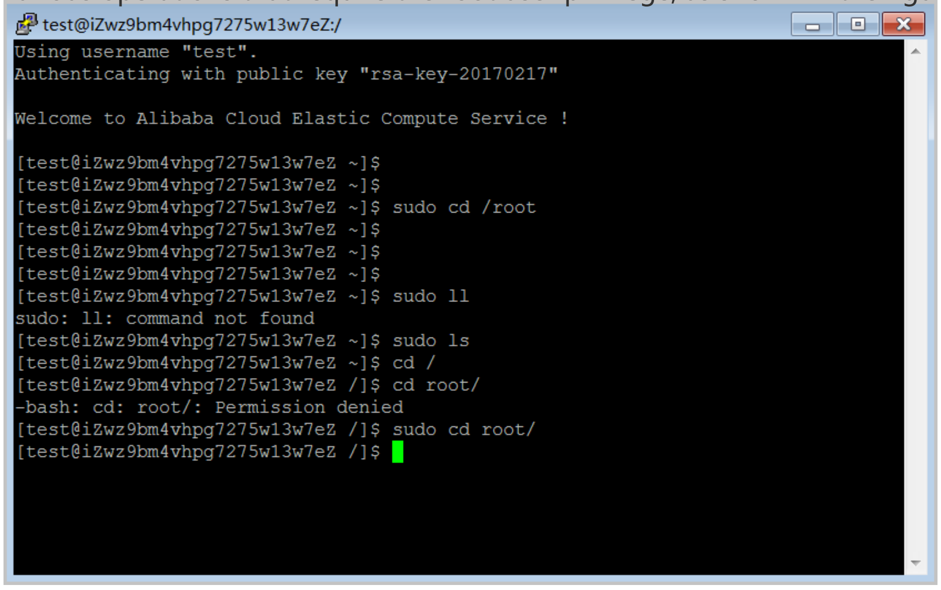


```
#!/bin/sh
useradd test
echo "test   ALL=(ALL)       NOPASSWD:ALL" | tee -a
/etc/sudoers
mkdir /home/test/.ssh
touch /home/test/.ssh/authorized_keys
echo "ssh-rsa
```

Windows support bat and powershell formats, with base-64 encoding, the script starts with [bat] or [powershell].
Linux supporters shell script, for more format reference please see cloudinit>>

After the configuration, click **Buy Now** and activate the instance following instructions on the page.

After the instance is created, you can use the newly-created **test** user to connect to the instance using an SSH private key. You can also escalate the permission using the sudo command and execute various operations that require the root user privilege, as shown in the figure.



```
Using username "test".
Authenticating with public key "rsa-key-20170217"

Welcome to Alibaba Cloud Elastic Compute Service !

[test@iZwz9bm4vhpg7275w13w7eZ ~]$
[test@iZwz9bm4vhpg7275w13w7eZ ~]$
[test@iZwz9bm4vhpg7275w13w7eZ ~]$ sudo cd /root
[test@iZwz9bm4vhpg7275w13w7eZ ~]$
[test@iZwz9bm4vhpg7275w13w7eZ ~]$
[test@iZwz9bm4vhpg7275w13w7eZ ~]$
[test@iZwz9bm4vhpg7275w13w7eZ ~]$ sudo ll
sudo: ll: command not found
[test@iZwz9bm4vhpg7275w13w7eZ ~]$ sudo ls
[test@iZwz9bm4vhpg7275w13w7eZ ~]$ cd /
[test@iZwz9bm4vhpg7275w13w7eZ /]$ cd root/
-bash: cd: root/: Permission denied
[test@iZwz9bm4vhpg7275w13w7eZ /]$ sudo cd root/
[test@iZwz9bm4vhpg7275w13w7eZ /]$
```

Overview

Previously, applications deployed on an ECS Instance usually needed to use Access Key ID and Access Key Secret (AK) to access APIs of other Alibaba Cloud products. AK is the key to accessing Alibaba Cloud APIs and has all of the permissions of the corresponding accounts. In order to help applications manage the AK, you have to save AK in the configuration files of the application or save it in an ECS instance by using other methods, which makes it more complicated to manage the AK and reduces its confidentiality. What’ s more, if you need concurrent deployment across regions, the AK will be diffused along with the images or instances created by the image, which makes you have to update and re-deploy the instances and images one by one when changing the AK.

Now with the help of the instance RAM role, you can assign a RAM role to an ECS instance. The applications on the instance can then access APIs of other cloud products with the STS credential. The STS credential is automatically generated and updated by the system, and the applications can use the specified meta data URL to obtain the STS credential without special management. Meanwhile, you can modify the RAM role and the authorization policy to grant different or identical access permissions to an instance to different Alibaba Cloud products.

This article introduces how to create an ECS instance that plays a RAM role and how to enable applications on the ECS instance to access other Alibaba Cloud products with the STS credential. In this section, using Python on an ECS instance to access an OSS bucket is used as the example.

Procedure

To enable python on an instance to access an OSS bucket under the same account by using the instance RAM role, perform the following steps:

Step 1. Create a RAM role and attach it to an authorization policy.
Step 2. Create an ECS instance playing the RAM role to create.
Step 3. Within the instance, access the metadata URL to obtain the STS credential.
Step 4. Use Python to access OSS using the STS credential.

Step 1. Create a RAM role and attach it to an authorization policy

Use the CreateRole API to create a RAM role. The required request parameters are:

- **RoleName**: Specify a name for the role. *EcsRamRoleTest* is used in this example.
- **AssumeRolePolicyDocument**: Specify a policy as follows, which indicates that the role to be created is a service role and an Alibaba Cloud product (ECS in this example) is assigned to play this role.

```
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"ecs.aliyuncs.com"
]
}
}
],
"Version": "1"
}
```

Use the CreatePolicy API to create an authorization policy. The required request parameters are:

- **PolicyName**: Specify a name for the authorization policy. *EcsRamRolePolicyTest* is used in this example.
- **PolicyDocument**: Specify a policy as follows, which indicates that the role has OSS read-only permission.

```
{
"Statement": [
{
"Action": [
"oss:Get*",
"oss:List*"
],
"Effect": "Allow",
"Resource": "*"
}
],
"Version": "1"
}
```

Use the AttachPolicyToRole API to attach the authorization policy to the role. The required request parameters are:

- **PolicyType**: Set it to *Custom*.
- **PolicyName**: Use the policy name specified in step 2. Use *EcsRamRolePolicyTest* in this example.
- **RoleName**: Use the role name specified in step 1. Use *EcsRamRoleTest* in this example.

Step 2. Create an ECS instance playing the RAM role

You can use either method to create an ECS instance playing the RAM role:

- Attach a RAM role to an existing VPC instance
- Create a VPC instance with the RAM role

Attach a RAM role to an existing VPC instance

Use the AttachInstanceRamRole API to attach a RAM role to an existing VPC instance. The parameters are as follows:

- **RegionId**: The ID of the region where the instance is located.
- **RamRoleName**: The name of a RAM role. In this example, *EcsRamRoleTest* is used.
- **InstanceIds**: The IDs of VPC instances that you want to attach the RAM role to, in the format of [“i-bXXXXXXXX”] for one instance, or [“i-bXXXXX” , “i-cXXXXX” , “i-dXXXXX” …] for multiple instances.

Create a VPC ECS instance with the RAM role

You must have a VPC network before creating an ECS instance with the RAM role.

To create a VPC instance with the RAM role, perform the following steps:

Use the CreateInstance API to create an ECS instance. The required request parameters are:

- **RegionId**: The region of the instance. In this example, *cn-hangzhou* is used.
- **ImageId**: The image of the instance. In this example, *centos_7_03_64_40G_alibase_20170503.vhd* is used.
- **InstanceType**: The type of the instance. In this example, *ecs.xn4.small* is used.
- **VSwitchId**: The virtual switch of the VPC network where the instance is located.

  Because the instance RAM role only supports VPC network, VSwitchId is required.

- **RamRoleName**: The name of RAM Role. In this example, *EcsRamRoleTest* is used.

  If you want to authorize a sub account to create an ECS instance playing the specified RAM role, besides the permission to create an ECS instance, the sub account must have the PassRole permission. Therefore, you must customize an authorization policy as follows and attach it to the sub account. If the action is creating an ECS instance only, set *[ECS RAM Action]* to ecs:CreateInstance. You can grant more permissions to meet your needs. For details, see Actions in RAM that can be authorized to an ECS instance. If you want to grant all ECS action permissions to the sub account, set *[ECS RAM Action]* to ecs:*.

```
{
"Statement": [
{
"Action": "[ECS RAM Action]",
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:PassRole",
"Resource": "*",
"Effect": "Allow"
],
"Version": "1"
}
```

Set the password and start the instance.

Set the ECS instance to access the Internet by using API or on the ECS console.

To set an ECS instance in a VPC network to access the Internet on a console, see Bind an Elastic IP address (EIP) in the *Quick Start* of Virtual Private Cloud.

Step 3. Access the metadata URL within the instance to obtain the STS credential

To obtain the STS credential of the instance, perform the following steps:

Connect to the instance.

Access the following URL to obtain the STS credential.
http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest
The last part of the URL is the RAM role name, which must be replaced with the one you create.

In this example, we run the curl command to access the URL. If you are using a Windows ECS instance, see Use metadata of an instance in ECS the *User Guide* to obtain the STS credential.

The return parameters are as follows.

```
[root@local ~]# curl http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest
{
"AccessKeyId" : "STS.J8XXXXXXXXXX4",
"AccessKeySecret" : "9PjfXXXXXXXXXBf2XAW",
"Expiration" : "2017-06-09T09:17:19Z",
"SecurityToken" : "CAIXXXXXXXXXXXwmBkleCTkyI+",
"LastUpdated" : "2017-06-09T03:17:18Z",
"Code" : "Success"
}
```

Step 4. Use Python SDK to access OSS with the STS credential

In this example, with the STS credential, we use Python to list 10 files in an OSS bucket that is in the same region with the instance.

Prerequisites

- You have remotely connected to the ECS instance.
- Python has been installed on the ECS instance. If you are using a Linux ECS instance, pip must be installed.
- A bucket has been created in the region of the instance, and the bucket name and the Endpoint have been acquired. In this example, the bucket name is *ramroletest*, and the endpoint is *oss-cn-hangzhou.aliyuncs.com*.

Procedure

To use Python to access the OSS bucket, perform the following steps:

Run the command pip install oss2 to install OSS Python SDK.

If you are using a Windows ECS instance, see Installation in the *Python-SDK* Reference of Object Storage Service.

Run the following commands to test, of which:

- The three parameters in oss2.StsAuth must be set to the values of the return parameters: AccessKeyId, AccessKeySecret, and SecurityToken.
- The last two parameters in oss2.Bucket are the bucket name and the endpoint.

```
import oss2
from itertools import islice
auth = oss2.StsAuth(<AccessKeyId>, <AccessKeySecret>, <SecurityToken>)
bucket = oss2.Bucket(auth, <your Endpoint>, <your Bucket name>)
for b in islice(oss2.ObjectIterator(bucket), 10):
print(b.key)
```

The output result is displayed as follows.

```
[root@local ~]# python
Python 2.7.5 (default, Nov 6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import oss2
>>> from itertools import islice
>>> auth = oss2.StsAuth("STS.J8XXXXXXXXXX4", "9PjfXXXXXXXXXBf2XAW",
"CAIXXXXXXXXXXXwmBkleCTkyI+")
>>> bucket = oss2.Bucket(auth, "oss-cn-hangzhou.aliyuncs.com", "ramroletest")
>>> for b in islice(oss2.ObjectIterator(bucket), 10):
... print(b.key)
...
ramroletest.txt
test.sh
```