Elastic Compute Service

Best Practices

MORE THAN JUST CLOUD | Alibaba Cloud

Best Practices

Use API to run 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. We recommend 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 in the ECS console.
- InstanceType: Instance type. See the ECS Buy Page. The option “one-core 2GB n1.small” indicates that the input parameter is “ecs.n1.small”.
- ImageId: Image ID. See the image list in the ECS console. You can filter public images or custom images.

For more parameter settings, see 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 more information about the operations in different ECS status, see 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, 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 more information, see 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, make sure 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 ECS Console for resource creation and daily management, you can also use API to manage and customize resources. API allows you to manage and configure ECS instances with greater flexibility.

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

Get the AccessKey for a RAM user

An AccessKey (AccessKey ID and AccessKey Secret) is required when you want to use API to manage ECS instances. To keep your cloud service secure, you have to create a RAM user and generate an AccessKey for it, and authorize the RAM user to manage ECS resources only. Then, you can use the RAM user and its AccessKey to manage ECS resources by using API.

Follow these steps to get the AccessKey for a RAM user.

1. Create a RAM user and get the AccessKey.
2. Grant permissions to the RAM user directly. To manage ECS resources, you have to grant **AliyunECSFullAccess** to the RAM user.

Install the ECS Python SDK

Make sure 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 AccessKey of the RAM user to instantialize an AcsClient object.

> The AccessKey allows the RAM user to access Alibaba Cloud APIs and give you full access to the user. 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 Secrect', 'cn-beijing')
```

You can develop your first application after the AcsClient object is instantiated. Query the list of regions that your account supports. For more information, see 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, see 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, see 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 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()
```

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

This document describes how to use Alibaba Cloud ECS SDKs to quickly create and manage spot instances.

Preparation

Before you begin, make sure that:

- You know which instance types and regions meet your business requirements.
- You have a basic understanding of Alibaba Cloud ECS SDKs and calling methods. For more information, see SDK documentation.

NOTE:
The ECS SDK version for spot instances is 4.2.0 and later. Here is an example of how to change the pom dependency.

```
<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>
```

Query regions and available instance types

Use the DescribeZones interface to query the regions where you can create spot instances and the available instance types. The sample code is as follows.

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");//You can use DescribeRegionsRequest to get the RegionId of
each region
request.setSpotStrategy("SpotWithPriceLimit");//This field must be entered to query the availability of
instance types
request.setInstanceChargeType("PostPaid");//Post-paid mode, spot instances must be post-paid
DescribeZonesResponse response = caller.doAction(request);
System.out.println(JSON.toJSONString(response));
}
}
```

In the following output result, you can see the instance types, disk types, and network types available in each region.

```
{
"requestId": "388D6321-E587-470C-8CFA-8985E2963DAE",
"zones": [
{
"localName": "China North 3 Zone 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"
]
}
]
}
```

Query spot instance price history

Use the DescribeSpotPriceHistory interface to query the price changes of a spot instance type over the last 30 days, so you can find the most cost efficient regions and instance types. The sample code (DescribeSpotPriceHistorySample.java) is as follows.

```
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 DescribeSpotPriceHistoryRequest();
request.setRegionId("cn-hangzhou");//You can use DescribeRegionsRequest to get the RegionId of each region
where spot instances are available
request.setZoneId("cn-hangzhou-b");//You must enter the zone
request.setInstanceType("ecs.sn2.medium");//See the instance types returned by DescribeZones, this field is
mandatory
request.setNetworkType("vpc");//See the network types returned by DescribeZones, this field is mandatory
// request.setIoOptimized("optimized");//Determines if the instance is I/O optimized, see IoOptimized returned by
DescribeZones, this field is optional
// request.setStartTime("2017-09-20T08:45:08Z");//The price start time, optional, default value: within 3 days
// request.setEndTime("2017-09-28T08:45:08Z");//Price end time, optional
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 {
}
}
}
```

Returned results sample.

```
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
```

Repeat this process to find the price trends and recent prices of the instance type in each zones.

Note:
You can use average price or maximum price to determine if you can afford this spot instance. You can also use more rational data models to analyze historical price data and adjust your instance types and zones at will for maximum cost effectiveness.

Create a spot instance

You must complete the following work before creating a spot instance:

- To use a custom image to create a spot instance, you must have already created the custom image.
- Create a security group in the console or use the CreateSecurityGroup to create a security group. Then, retrieve the security group ID (SecurityGroupId).
- In the console, create a VPC and VSwitch, or use the CreateVpc and CreateVSwitch interfaces to do so. Then, retrieve the VSwitch ID (VSwitchId).

Use the CreateInstance to create a spot instance. The sample code (CreateInstaneSample.java) is as follows.

```
public class CreateInstaneSample {
public static void main(String[] args) {
OpenApiCaller caller = new OpenApiCaller();
CreateInstanceRequest request = new CreateInstanceRequest();
request.setRegionId("cn-hangzhou");//The region ID
request.setZoneId("cn-hangzhou-b"); //The zone ID
request.setSecurityGroupId("sg-bp11nhf94ivkdxwb2gd4");//The ID of the security group
request.setImageId("centos_7_03_64_20G_alibase_20170818.vhd");//We recommend that you select a custom image you have prepared in this region
request.setVSwitchId("vsw-bp164cyonthfudn9kj5br");//For VPC, the VSwitch ID is required
request.setInstanceType("ecs.sn2.medium"); //Enter the instance type you want to purchase
request.setIoOptimized("optimized");//See the parameters returned by DescirbeZones
request.setSystemDiskCategory("cloud_ssd");//See the parameters returned by DescirbeZones, select one: cloud_ssd, cloud_efficiency, or cloud
request.setSystemDiskSize(40);
request.setInstanceChargeType("PostPaid");//Post-paid is required for spot instances
request.setSpotStrategy("SpotWithPriceLimit");//SpotWithPriceLimit: bid mode, SpotAsPriceGo: no bids, the maximum Pay-As-You-Go price
request.setSpotPriceLimit(0.25F);//This applies only to SpotWithPriceLimit. Set the maximum price you are willing to pay, units: USD/hour. An instance is created when this price is higher than the current market price
CreateInstanceResponse response = caller.doAction(request);
System.out.println(response.getInstanceId());
}
}
```

Recover a spot instance

Mandatory recovery can be imposed on spot instances due to changes in price or supply and demand. At such a time, the operation of the spot instance is suspended. Before being released, the spot instance enters the locked status and a prompt notifies you that the instance will be automatically recovered. You can design a withdrawal logic to automatically process instances in the recovery status.

Now, you can use the following methods to determine the suspension and lock statuses of spot instances:

Obtain this information from the instance metadata. Run the following command.

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

If no result is returned, it indicates the instance can continue to be used. If the returned result contains UTC time stamp information in the format of YYYY-MM-DDTHH:mm:ssZ (for example 2015-01-05T18:02:00Z), it indicates the instance will be released at the specified time.

You can use the OperationLocks information returned by the **DescribeInstances** to see if an instance is in the **Awaiting Recovery** status. The sample code (DescribeInstancesSample.java) is as follows.

```
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 DescribeInstancesRequest();
request.setRegionId("cn-hangzhou");
request.setInstanceIds(allInstances.toJSONString());//Specify the instance ID, maximum efficiency
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;
}
}
}
}
```

The output result when recovery is triggered is as follows.

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

Other operations

You can start, stop, and release spot instances. These operations are the same as for Pay-As-You-Go instances. For more information, see the API documentation:

- Start an instance: StartInstance
- Stop an instance: StopInstance
- Release an instance: DeleteInstance

User-defined 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, see User-defined data.

This example uses a Linux instance to demonstrate how to use a user-defined script to configure your own yum repository, NTP service, and DNS service when creating a Linux instance. User-defined scripts also enable you to configure NTP service and DNS service for a Windows instance.

Scenarios

When a Linux instance is started, Alibaba Cloud automatically configures a 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 them.

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.

Procedure

To customize your yum repository, NTP service, and DNS service for a Linux instance when creating it, follow these steps:

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.

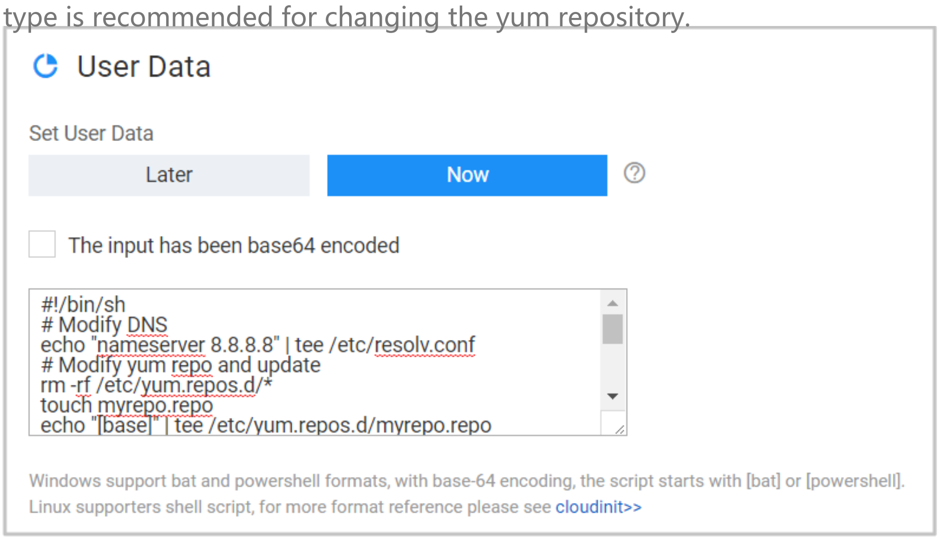
- **Operating System**: Select CentOS 7.2 in **Public Image** tab.
- **Security Setup**: Select one to meet your requirements.

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. 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.



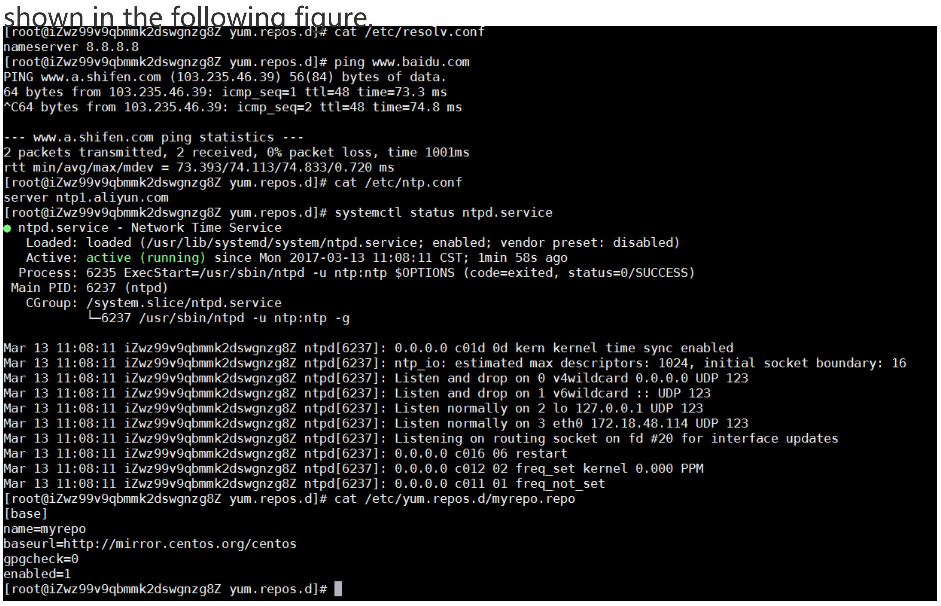
The input has been base64 encoded

```
#!/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 you complete the configuration, click **Buy Now** and activate the instance following the instructions on the page.

After the instance is created, you can connect 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 to enable users to customize the startup behavior of ECS instances. For details, see User-defined data.

This example uses a Linux instance to demonstrate how to use a user-defined script to create a new account, with the root user privilege, when creating a Linux instance. User-defined scripts can also be used to create a new account with the administrator privilege for a Windows instance.

Scenarios

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 root user privileges are disabled.
- Create a new account with the root user privilege and customize the account name.
- Use only SSH key pairs, but not user passwords, for remote logon to manage the instance by using the new account with the root user privilege.
- If this new account is required to perform operations that can only be done by a user with root user privilege, the sudo command can be used without a password for privilege escalation.

Procedure

To create a new account with the root user privilege, follow these steps:

Create a Linux instance. Configure the instance as follows:

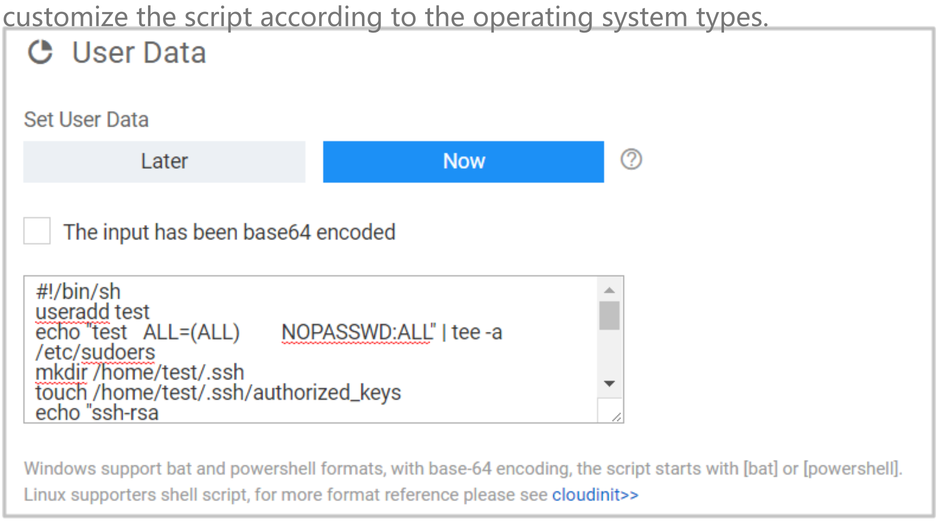
- **Network Type**: Select **VPC**.
- **Instance Type**: Select an I/O-optimized instance.
- **Operating System**: Select CentOS 7.2 in **Public Image** tab.
- **Security Setup**: select **Later**.

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.

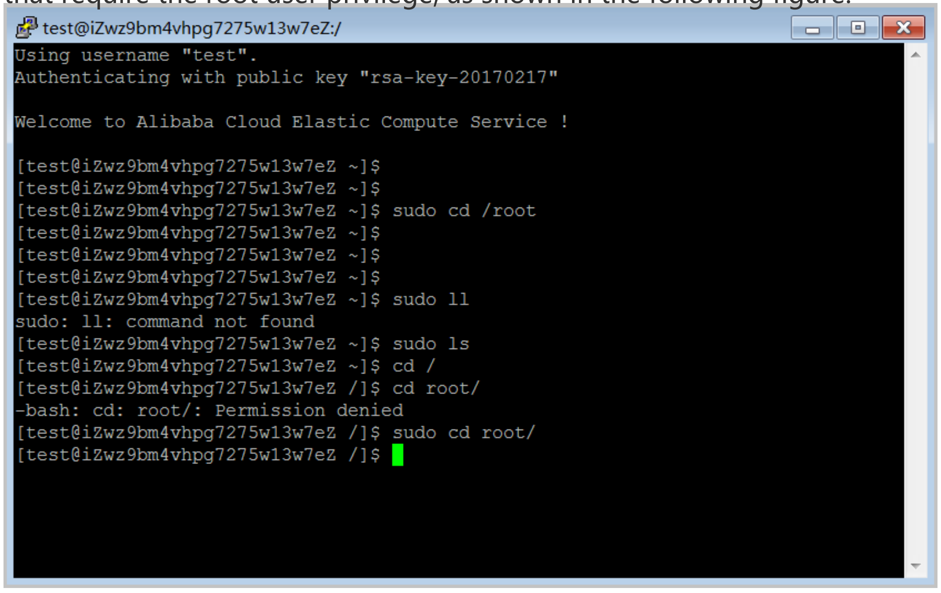


```
#!/bin/sh
useradd test
echo "test   ALL=(ALL)       NOPASSWD:ALL" | tee -a
/etc/sudoers
mkdir /home/test/.ssh
touch /home/test/.ssh/authorized_keys
echo "ssh-rsa
```

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 you finish the configuration, click **Buy Now** and activate the instance by following instructions on the page.

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



```
test@iZwz9bm4vhpg7275w13w7eZ:/
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 AccessKey ID and AccessKey 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. 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 is 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, follow these 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, follow these 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 more information, 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 in 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, follow these 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, follow these 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
```

FaaS instances best practices

This document introduces how to use Open Computing Language (OpenCL) to create an image file, and then download the image to an FPGA chip.

Note:
We strongly recommend that you use an f1 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to perform essential actions only. You must create

a role for the RAM user and grant temporary permissions to the role to access the OSS buckets. If you want to encrypt the IP address, grant the RAM user to use Key Management Service (KMS). If you want the RAM user to check permissions, authorize the RAM user to view the resources of an account.

Prerequisites

Before use OpenCL to create an image file, you must finish the following operations:

- Create an f1 instance.
- If you operate an f1 instance as a RAM user, you must do the following operations:
  - Create a RAM user and grant permissions.
  - Create a RAM role and grant permissions.
  - Create an AccessKey.
- Log on to the ECS console, and in the instance detail page of the f1 instance, obtain the instance ID.

Procedure

To configure the environment of FPGA Server Example, follow these steps.

Step 1. Connect to your f1 instance

Connect to the Linux instance.

Step 2. Install the basic environment

Run the script to install the basic environment.

```
source /opt/dcp1_0/script/f1_env_set.sh
```

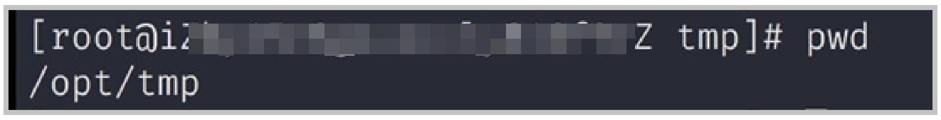
Step 3. Download the OpenCL Example

To download the OpenCL Example, follow these steps:

Create the /opt/tmp directory, and change the current directory to it.

```
mkdir -p /opt/tmp
cd /opt/tmp
```

Now, you are at the /opt/tmp directory.

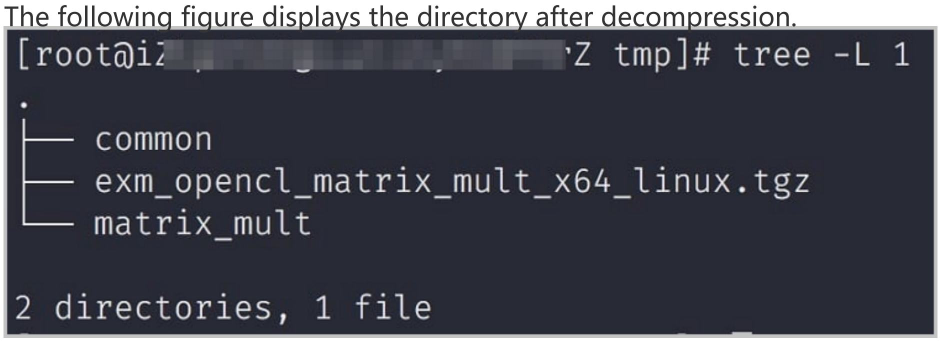


```
[root@i[redacted]Z tmp]# pwd
/opt/tmp
```

Run the commands one by one to download and decompress the OpenCL Example file.

```
wget https://www.altera.com/content/dam/altera-
www/global/en_US/others/support/examples/download/exm_opencl_matrix_mult_x64_linux.tgz
tar -zxvf exm_opencl_matrix_mult_x64_linux.tgz
```

The following figure displays the directory after decompression.



```
[root@i[redacted]Z tmp]# tree -L 1
.
├── common
├── exm_opencl_matrix_mult_x64_linux.tgz
└── matrix_mult

2 directories, 1 file
```

Change the current directory to the matrix_mult directory and run the command for compilation.

```
cd matrix_mult
aoc -v -g --report ./device/matrix_mult.cl
```

The process of compilation takes several hours. You can open a new console, and run the top command to monitor processes and system resource usage on the instance and view the status of the compilation process.

Step 4. Upload the configuration file to the OSS bucket

Run the commands to initialize the faascmd.

```
# If needed, add the environment variable and grant the permission to run the commands
export PATH=$PATH:/opt/dcp1_0/script/
chmod +x /opt/dcp1_0/script/faascmd
# Replace hereIsMySecretId with your AccessKey ID. Replace hereIsMySecretKey with your AccessKey
Secret
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey
# Replace hereIsMyBucket with the bucket name of your OSS in the Region China East 1.
faascmd auth --bucket=hereIsMyBucket
```

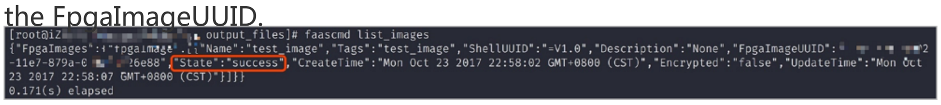
Change the current directory to the matrix_mult/output_files directory, and upload the configuration file.

```
cd matrix_mult/output_files # Now you are accessing /opt/tmp/matrix_mult/matrix_mult/output_files
faascmd upload_object --object=afu_fit.gbs --file=afu_fit.gbs
```

Use gbs to create an FPGA image.

```
# Replace hereIsFPGAImageName with your image name. Replace hereIsFPGAImageTag with the tag of your image.
faascmd create_image --object=afu_fit.gbs --fpgatype=intel --name=hereIsFPGAImageName --tags=hereIsFPGAImageTag --encrypted=false --shell =V1.0
```

Run the faascmd list_images command to check whether the image is created. In the returned result, if "State":"success" is displayed, it means the image is created. Record the FpgaImageUUID.



```
[root@iZ[redacted] output_files]# faascmd list_images
{"FpgaImages":[redacted]"Name":"test_image","Tags":"test_image","ShellUUID":"=V1.0","Description":"None","FpgaImageUUID":[redacted]2
-11e7-879a-0[redacted]26e88","State":"success","CreateTime":"Mon Oct 23 2017 22:58:02 GMT+0800 (CST)","Encrypted":"false","UpdateTime":"Mon Oct
23 2017 22:58:07 GMT+0800 (CST)"}]}}
0.171(s) elapsed
```

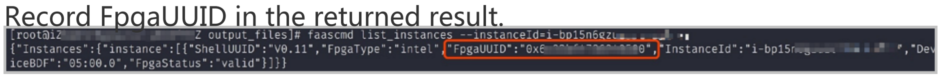
Step 5. Download the image to your f1 instance

To download the image to your f1 instance, follow these steps:

Run the command to obtain FPGA ID.

```
# Replace hereIsYourInstanceId with your f1 instance ID.
faascmd list_instances --instanceId=hereIsYourInstanceId
```

Record FpgaUUID in the returned result.



```
[root@iZ[redacted]Z output_files]# faascmd list_instances --instanceId=i-bp15n6gzu[redacted]
{"Instances":{"instance":[{"ShellUUID":"V0.11","FpgaType":"intel","FpgaUUID":"0x6[redacted]0","InstanceId":"i-bp15n[redacted]","Dev
iceBDF":"05:00.0","FpgaStatus":"valid"}]}}
```

Run the command to download the image to your f1 instance.

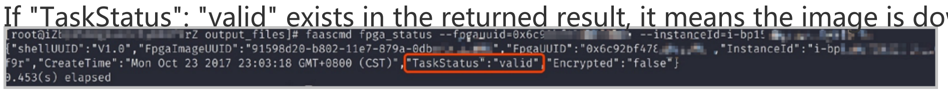
```
# Replace hereIsYourInstanceID with your f1 instance ID. Replace hereIsFpgaUUID with your FpgaUUID. Replace hereIsImageUUID with your FpgaImageUUID.
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --fpgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.0
```

Run the command to check whether the image is downloaded.

```
# Replace hereIsYourInstanceID with your f1 instance ID. Replace hereIsFpgaUUID with your FpgaUUID.
```

```
faascmd fpga_status --fpgauuid=hereIsFpgaUUID --instanceId=hereIsYourInstanceID
```

If "TaskStatus": "valid" exists in the returned result, it means the image is downloaded.



```
[root@iZ[redacted]rZ output_files]# faascmd fpga_status --fpgauuid=0x6c9[redacted] --instanceId=i-bp1[redacted]
{"shellUUID":"V1.0","FpgaImageUUID":"91598d20-b802-11e7-879a-0db[redacted]","FpgaUUID":"0x6c92bf47[redacted] ,"InstanceId":"i-bp[redacted]
f9r","CreateTime":"Mon Oct 23 2017 23:03:18 GMT+0800 (CST)","TaskStatus":"valid","Encrypted":"false"}
0.453(s) elapsed
```

Step 6. Download the FPGA image to an FPGA chip

To download the FPGA image to an FPGA chip, follow these steps:

Open the console in Step 1. If it is closed, repeat Step 1.

Run the following command to configure the runtime environment for OpenCL.

```
sh /opt/dcp1_0/opencl/dcp_opencl_bsp/linux64/libexec/setup_permissions.sh
```

Run the command to go back to the parent directory.

```
cd ../.. # Now, you are at the /opt/tmp/matrix_mult directory
```

Run the command to compile.

```
make
# Output the environment configuration
export CL_CONTEXT_COMPILER_MODE_ALTERA=3
cp matrix_mult.aocx ./bin/matrix_mult.aocx
cd bin
host matrix_mult.aocx
```

If the following result is returned, it means the configuration is successful. Note that the last line must be Verification: PASS.

```
[root@iZbpXXXXXZ bin]# ./host matrix_mult.aocx
Matrix sizes:
A: 2048 x 1024
B: 1024 x 1024
C: 2048 x 1024
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
skx_fpga_dcp_ddr : SKX DCP FPGA OpenCL BSP (acl0)
Using AOCX: matrix_mult.aocx
Generating input matrices
Launching for device 0 (global size: 1024, 2048)
Time: 40.415 ms
```

```
Kernel time (device 0): 40.355 ms
Throughput: 106.27 GFLOPS
Computing reference output
Verifying
Verification: PASS
```

This document describes how to generate and download a custom bitstream file to a specified FPGA chip of an f1 instance.

Note:
We strongly recommend that you use an f1 instance as a RAM user. To avoid unwanted operations, the RAM user must be authorized to perform essential actions only. You must create a role for the RAM user and grant temporary permissions to the role to access the OSS buckets. If you want to encrypt the IP addresses, authorize the RAM user to use Key Management Service (KMS). If you want the RAM user to check permissions, authorize the RAM user to view the resources.

Prerequisites

Before you start, finish the following operations:

- Create an f1 instance.
- Activate OSS to upload your custom bitstream files.
- If you want to encrypt your bitstream, activate Key Management Service (KMS).
- If you operate an f1 instance as a RAM user, you must do the following operations:
    - Create a RAM user and grant permissions.
    - Create a RAM role and grant permissions.
    - Create an AccessKey.

Procedure

To generate and download a bitstream file, follow these steps.

Step 1. Generate a bitstream file

To generate a bistream file, follow these steps:

Upload the project file to the specified OSS bucket. The RAM user must have the permission to access the OSS bucket.

- If you are using an Intel FPGA chip, you must upload the final gbs file to your OSS bucket.

- If you are using a Xilinx FPGA chip, you must upload the DCP file to your OSS bucket.

Call the CreateFpgaImageTask interface in the Python SDK to create a bitstream file. See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKey Secret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import CreateFpgaImageTaskRequest
request = CreateFpgaImageTaskRequest.CreateFpgaImageTaskRequest()
request.set_Bucket(<The OSS bucket for the DCP or bitstream file>)
request.set_Object(<The object name of the DCP or bitstream file>)
request.set_FpgaType(<Fpga type>)
request.set_ShellUUID(<shell type>)
request.set_Name(<Specify a name for the image to ease management>)
request.set_RoleArn(<Create a role for faas-admin>)
request.set_Encrypted(<Encrypted or not, True/False>)
request.set_KeyId(<If encrypted, specify the ID of the key in the KMS>)
result = clt.do_action_with_exception(request)
print result
```

Call the DescribeFpgaImages interface in the Python SDK to check whether the bitstream is generated.

Note:
CreateFpgaImageTask is an asynchronous action. After you send the request, the back-end server performs the security check. If you are using a Xilinx project, the back-end server has to generate a bitstream file from the DCP project, which takes some time.

See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKeySecret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import DescribeFpgaImagesRequest
request = DescribeFpgaImagesRequest.DescribeFpgaImagesRequest()
result = clt.do_action_with_exception(request)
print result
```

Step 2. Download the bitstream file

To download the bitstream file to the specified FPGA chip, follow these steps:

Call the DescribeFpgaInstances interface in the Python SDK to check the FpgaUUID associated with your f1 instance. FpgaUUID is the unique identifier of an FPGA chip. See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKey Secret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import DescribeFpgaInstancesRequest
request = DescribeFpgaInstancesRequest.DescribeFpgaInstancesRequest()
request.set_InstanceId(<Specify the instance ID>)
request.set_RoleArn(<Create a role for faas-admin>)
result = clt.do_action_with_exception(request)
print result
```

Call the DescribeFpgaImages interface to view the information about the bitstream file under your account. The unique identifier of your bitstream file is FpgaImageUUID. See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKey Secret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import DescribeFpgaImagesRequest
request = DescribeFpgaImagesRequest.DescribeFpgaImagesRequest()
result = clt.do_action_with_exception(request)
print result
```

Call the LoadFpgaImageTask interface in the Python SDK to download the specified bitstream file to the specified FPGA chip. See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKey Secret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import LoadFpgaImageTaskRequest
request = LoadFpgaImageTaskRequest.LoadFpgaImageTaskRequest()
request.set_InstanceId(<Specify the instance ID>)
request.set_FpgaUUID(<Specify the FPGA chip>)
request.set_FpgaType(<Fpga type>)
request.set_FpgaImageUUID(<The UUID of the image to be downloaded>)
request.set_FpgaImageType(<Image type>)
request.set_ShellUUID(<Specify shell>)
request.set_RoleArn(<Create a role for faas-admin>)
result = clt.do_action_with_exception(request)
print result
```

Call the DescribeLoadTaskStatus interface in the Python SDK to check whether the bitstream file is downloaded. See the following example code.

```
from aliyunsdkcore import client
clt = client.AcsClient(<Your AccessKey ID>,<Your AccessKey Secret>,'cn-hangzhou')
from aliyunsdkfaas.request.v20170824 import DescribeLoadTaskStatusRequest
request = DescribeLoadTaskStatusRequest.DescribeLoadTaskStatusRequest()
request.set_FpgaUUID(<Specify the FPGA chip>)
request.set_InstanceId(<Specify the instance ID>)
request.set_RoleArn(<Create a role for faas-admin>)
result = clt.do_action_with_exception(request)
```

```
print result
```

This document describes how to use Register Transfer Level (RTL) compiler on an f1 instance.

Note:
We strongly recommend that you use an f1 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to perform essential actions only. You must create a role for the RAM user and grant temporary permissions to the role to access the OSS buckets. If you want to encrypt the IP address, grant the RAM user to use Key Management Service (KMS). If you want the RAM user to check permissions, authorize the RAM user to view the resources of an account.

Prerequisites

Before you start, you must finish the following operations:

- Create an f1 instance.
- If you operate an f1 instance as a RAM user, you must do the following operations:
    - Create a RAM user and grant permissions.
    - Create a RAM role and grant permissions.
    - Create an AccessKey.
- Log on to the ECS console, and in the instance detail page of the f1 instance, obtain the instance ID.

Procedure

To use RTL compiler on an f1 instance, follow these steps.

Step 1. Connect to the f1 instance

Connect to your f1 instance.

Step 2. Configure the basic environment

Run the script to configure the basic environment.

```
source /opt/dcp1_0/script/f1_env_set.sh
```

Step 3. Compile the project

Run the following commands to compile the project.

```
cd /opt/dcp1_0/hw/green_bits/dma_afu/src
run.sh
```

Note:
It takes a long time to compile the project.

Step 4. Create an image

To create an image, follow these steps:

Run the following commands to initialize faascmd.

```
# If needed, add the environment variable and grant permission to run the commands.
export PATH=$PATH:/opt/dcp1_0/script/
chmod +x /opt/dcp1_0/script/faascmd
# Replace hereIsMySecretId with your AccessKey ID. Replace hereIsMySecretKey with your AccessKey
Secret.
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey
# Replace hereIsMyBucket with the OSS bucket name in the China East 1 region.
faascmd auth --bucket=hereIsMyBucket
```

Make sure you are at the /opt/dcp1_0/hw/green_bits/dma_afu/src directory, and run the command to upload the gbs file.

```
faascmd upload_object --object=dma_afu.gbs --file=dma_afu.gbs
```

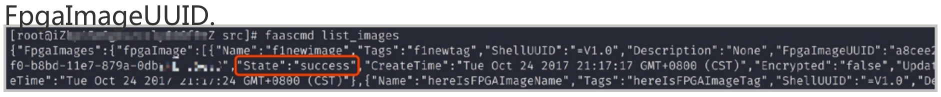
Run the command to create an image.

```
#  Replace hereIsYourImageName with your image name.
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=hereIsYourImageName --
tags=hereIsYourImageTag --encrypted=false --shell =V1.0
```

Step 5. Download the image

To download the image, follow these steps:

Run the faascmd list_images command to check whether the image is created.
If "State":"success" exists in the returned result, it means the image is created. Record the FpgaImageUUID.

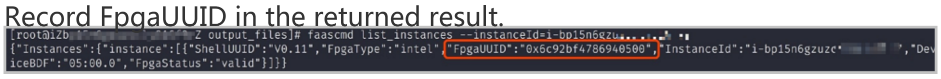


```
[root@iZ[redacted]Z src]# faascmd list_images
{"FpgaImages":{"fpgaImage":[{"Name":"f1newimage","Tags":"f1newtag","ShellUUID":"=V1.0","Description":"None","FpgaImageUUID":"a8cee2
f0-b8bd-11e7-879a-0db[redacted]","State":"success","CreateTime":"Tue Oct 24 2017 21:17:17 GMT+0800 (CST)","Encrypted":"false","Updat
eTime":"Tue Oct 24 2017 21:17:24 GMT+0800 (CST)"},{"Name":"hereIsFPGAImageName","Tags":"hereIsFPGAImageTag","ShellUUID":"=V1.0","De
```

Run the command to obtain FPGA ID.

```
# Replace hereIsYourInstanceId with your f1 instance ID.
faascmd list_instances --instanceId=hereIsYourInstanceId
```

Record FpgaUUID in the returned result.



```
[root@iZb[redacted]Z output_files]# faascmd list_instances --instanceId=i-bp15n6gzu[redacted]
{"Instances":{"instance":[{"ShellUUID":"V0.11","FpgaType":"intel","FpgaUUID":"0x6c92bf4786940500","InstanceId":"i-bp15n6gzuzc[redacted]","DeviceBDF":"05:00.0","FpgaStatus":"valid"}]}}
```

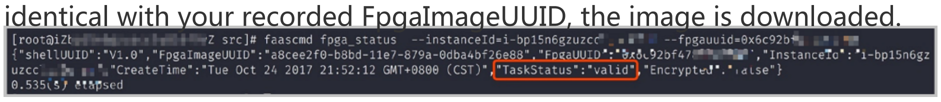
Run the command to download the image to your f1 instance.

```
# Replace hereIsYourInstanceID with your f1 instance ID. Replace hereIsFpgaUUID with your FpgaUUID.
Replace hereIsImageUUID with your FpgaImageUUID.
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --
fpgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.0
```

Run the command to check whether the image is downloaded.

```
# Replace hereIsYourInstanceID with your f1 instance ID. Replace hereIsFpgaUUID with your FpgaUUID.
faascmd fpga_status --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID
```

If "TaskStatus":"valid" exists in the returned result, and the displayed FpgaImageUUID is identical with your recorded FpgaImageUUID, the image is downloaded.



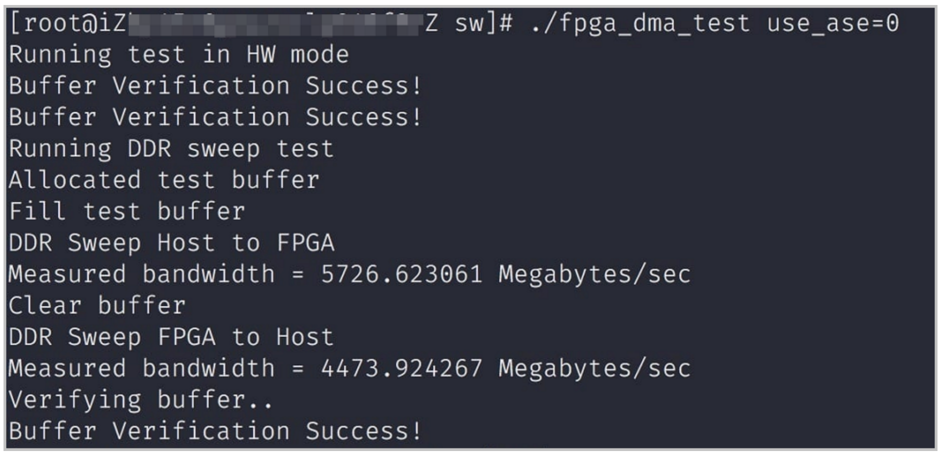
```
[root@iZb[redacted]Z src]# faascmd fpga_status  --instanceId=i-bp15n6gzuzcc[redacted] --fpgauuid=0x6c92b[redacted]
{"shellUUID":"V1.0","FpgaImageUUID":"a8cee2f0-b8bd-11e7-879a-0dba4bf26e88","FpgaUUID":"0x6c92bf47[redacted]","InstanceId":"i-bp15n6gzuzcc[redacted]","CreateTime":"Tue Oct 24 2017 21:52:12 GMT+0800 (CST)","TaskStatus":"valid","Encrypted":"false"}
0.535(s) elapsed
```

Step 6. Test

Run the commands one by one for test.

```
cd /opt/dcp1_0/hw/green_bits/dma_afu/src/sw
make
./fpga_dma_test use_ase=0
```

If the following result is returned, the test is completed.



```
[root@iZ[redacted]Z sw]# ./fpga_dma_test use_ase=0
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 5726.623061 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 4473.924267 Megabytes/sec
Verifying buffer..
Buffer Verification Success!
```

Note:
If the Huge pages feature is not enabled, run the following command to enable it.

```
sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"
```

This document introduces how to use Open Computing Language (OpenCL) to create an image file, and then download image to an FPGA chip.

Note:
We strongly recommend that you use an f2 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to execute essential actions only. You must create a role for the RAM user and grant temporary permissions to the role to access the OSS buckets.

Prerequisites

Before you begin the procedure, you must:

- Create an f2 instance.
- Create a RAM user with granted permissions to play a RAM role with granted permissions, if you use an f2 instance as a RAM user. You must create a pair of AccessKeyID and AccessKeySecret for the RAM user.
- Activate OSS service and create a bucket.
- Log on to the ECS console and obtain the instance ID of your f2 instance.

Procedure

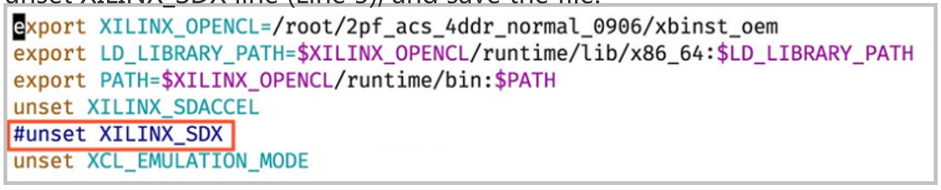
To create an image by using OpenCL on an f2 instance and download it to an FPGA chip, follow these steps.

Step 1. Set up the environment

To set up the environment on an f2 instance, follow these steps.

Connect to an f2 instance.

Run the vim /root/xbinst_oem/setup.sh command to modify the file: comment out the unset XILINX_SDX line (Line 5), and save the file.



```
export XILINX_OPENCL=/root/2pf_acs_4ddr_normal_0906/xbinst_oem
export LD_LIBRARY_PATH=$XILINX_OPENCL/runtime/lib/x86_64:$LD_LIBRARY_PATH
export PATH=$XILINX_OPENCL/runtime/bin:$PATH
unset XILINX_SDACCEL
#unset XILINX_SDX
unset XCL_EMULATION_MODE
```

Run the command to install Screen to keep the terminal session persistent.

```
yum install screen -y
```

Run the command to open a new screen.

```
screen -S f2opencl
```

Run the command to set up a secure environment to download a file.

```
source /root/xbinst_oem/F2_env_setup.sh
```

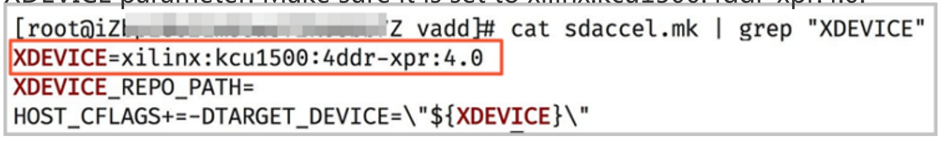
Step 2. Compile a binary file

To compile a binary file, follow these steps.

Run the command to change the directory.

```
cd /opt/Xilinx/SDx/2017.2/examples/vadd
```

Run the cat sdaccel.mk | grep "XDEVICE" command to check the configuration of the XDEVICE parameter. Make sure it is set to xilinx:kcu1500:4ddr-xpr:4.0.



```
[root@iZ[redacted]Z vadd]# cat sdaccel.mk | grep "XDEVICE"
XDEVICE=xilinx:kcu1500:4ddr-xpr:4.0
XDEVICE_REPO_PATH=
HOST_CFLAGS+=-DTARGET_DEVICE=\"${XDEVICE}\"
```

Run the vim ../common/common.mk command to modify the common.mk file: Replace Line 63

```
CLCC_OPT += $(CLCC_OPT_LEVEL) ${DEVICE_REPO_OPT} --platform ${XDEVICE} -o ${XCLBIN}
${KERNEL_DEFS} ${KERNEL_INCS}
```

with

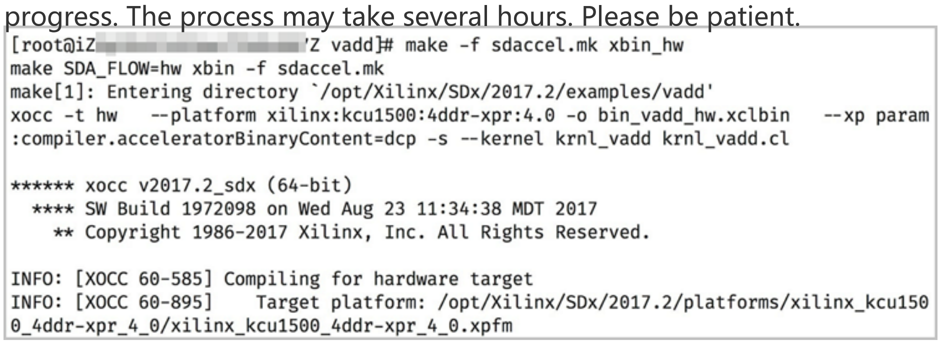
```
CLCC_OPT += $(CLCC_OPT_LEVEL) ${DEVICE_REPO_OPT} --platform ${XDEVICE} -o ${XCLBIN}
${KERNEL_DEFS} ${KERNEL_INCS} --xp param:compiler.acceleratorBinaryContent=dcp
```

> **Note**:
> The parameters may be in lines from Line 60 to Line 62, which is determined by your file.

Run the command to compile the program.

```
export XILINX_SDX=/opt/Xilinx/SDx/2017.2
make -f sdaccel.mk xbin_hw
```

If the following information is displayed, it means the compilation of the binary file is in progress. The process may take several hours. Please be patient.



```
[root@iZ[redacted]Z vadd]# make -f sdaccel.mk xbin_hw
make SDA_FLOW=hw xbin -f sdaccel.mk
make[1]: Entering directory `/opt/Xilinx/SDx/2017.2/examples/vadd'
xocc -t hw   --platform xilinx:kcu1500:4ddr-xpr:4.0 -o bin_vadd_hw.xclbin   --xp param
:compiler.acceleratorBinaryContent=dcp -s --kernel krnl_vadd krnl_vadd.cl

****** xocc v2017.2_sdx (64-bit)
  **** SW Build 1972098 on Wed Aug 23 11:34:38 MDT 2017
    ** Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.

INFO: [XOCC 60-585] Compiling for hardware target
INFO: [XOCC 60-895]   Target platform: /opt/Xilinx/SDx/2017.2/platforms/xilinx_kcu150
0_4ddr-xpr_4_0/xilinx_kcu1500_4ddr-xpr_4_0.xpfm
```

Step 3. Check the packaging script

Run the command to check whether the packaging script exists or not.

```
file /root/xbinst_oem/sdaccel_package.sh
```

> **Note**:
> A returned message containing cannot open (No such file or directory) means no packaging script exist. Then, download the script by running the following command.

```
wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/sdaccel_package.sh -O /root/xbinst_oem/sdaccel_package.sh
chmod a+x /root/xbinst_oem/sdaccel_package.sh
```

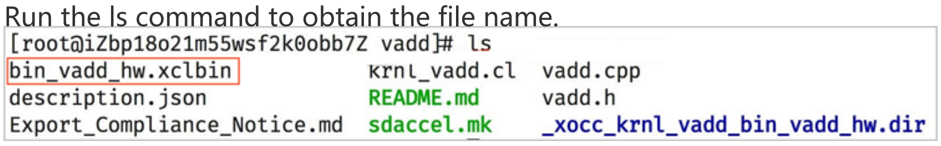
Step 4. Create an image

To create an image, follow these steps.

1. Run the command to set up the OSS environment.

```
# Replace hereIsMySecretId, hereIsMySecretKey, and hereIsMyBucket with your own AccessKeyID,
AccessKeySecret, and OSS Bucket name
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey
faascmd auth --bucket=hereIsMyBucket
```

Run the ls command to obtain the file name.

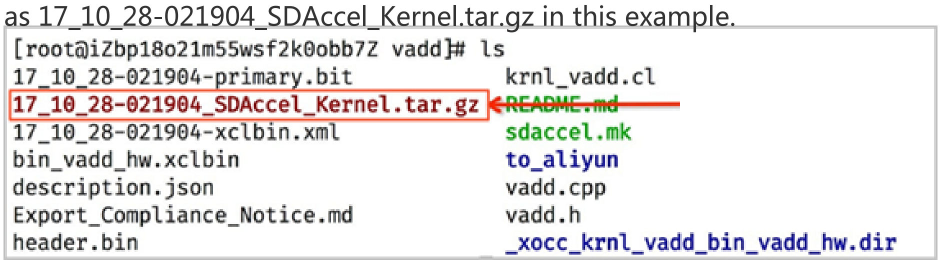


```
[root@iZbp18o21m55wsf2k0obb7Z vadd]# ls
bin_vadd_hw.xclbin           krnl_vadd.cl  vadd.cpp
description.json             README.md     vadd.h
Export_Compliance_Notice.md  sdaccel.mk    _xocc_krnl_vadd_bin_vadd_hw.dir
```

Run the command to package the binary file.

```
/root/xbinst_oem/sdaccel_package.sh -
xclbin=/opt/Xilinx/SDx/2017.2/examples/vadd/bin_vadd_hw.xclbin
```

After the packaging is completed, you will have a packaged file in the same directory, such as 17_10_28-021904_SDAccel_Kernel.tar.gz in this example.



```
[root@iZbp18o21m55wsf2k0obb7Z vadd]# ls
17_10_28-021904-primary.bit              krnl_vadd.cl
17_10_28-021904_SDAccel_Kernel.tar.gz    README.md
17_10_28-021904-xclbin.xml               sdaccel.mk
bin_vadd_hw.xclbin                       to_aliyun
description.json                         vadd.cpp
Export_Compliance_Notice.md              vadd.h
header.bin                               _xocc_krnl_vadd_bin_vadd_hw.dir
```

Run the command to upload the packaged file to your own OSS bucket.

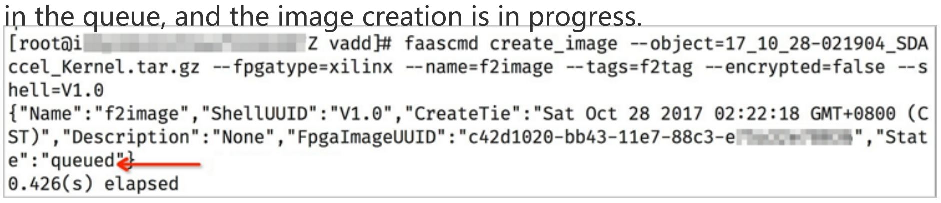
```
# Replace the file name with the name of the packaged file
faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
```

Run the command to create an image.

```
# Replace bit.tar.gz, hereIsFPGAImageName, and hereIsFPGAImageTag with the name of the packaged
file, the image name, and the image tag
faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --name=hereIsFPGAImageName --
```

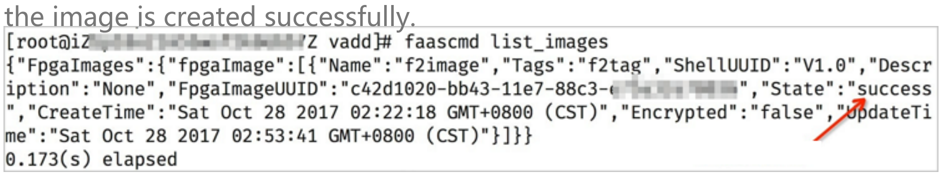
```
tags=hereIsFPGAImageTag --encrypted=false --shell=V1.0
```

The example returned result is displayed as follows. The "State":"queued" means the task is in the queue, and the image creation is in progress.



```
[root@i[redacted]Z vadd]# faascmd create_image --object=17_10_28-021904_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=f2image --tags=f2tag --encrypted=false --shell=V1.0
{"Name":"f2image","ShellUUID":"V1.0","CreateTie":"Sat Oct 28 2017 02:22:18 GMT+0800 (CST)","Description":"None","FpgaImageUUID":"c42d1020-bb43-11e7-88c3-e[redacted]","State":"queued"}
0.426(s) elapsed
```

Note:
Creation of an image consumes time. You can run the faascmd list_images command to check the image status. In the returned result, if "State":"success" displays, it means the image is created successfully.



```
[root@iZ[redacted]Z vadd]# faascmd list_images
{"FpgaImages":{"fpgaImage":[{"Name":"f2image","Tags":"f2tag","ShellUUID":"V1.0","Description":"None","FpgaImageUUID":"c42d1020-bb43-11e7-88c3-[redacted]","State":"success","CreateTime":"Sat Oct 28 2017 02:22:18 GMT+0800 (CST)","Encrypted":"false","UpdateTime":"Sat Oct 28 2017 02:53:41 GMT+0800 (CST)"}]}}
0.173(s) elapsed
```

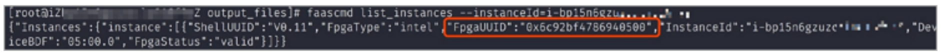
Record the FPGAImageUUID.

Step 5. Download the image

To download the image to an FPGA chip, follow these steps.

Run the command to obtain the FpgaUUID.

```
# Replace hereIsYourInstanceId with your f2 instance ID
faascmd list_instances --instanceId=hereIsYourInstanceId
```



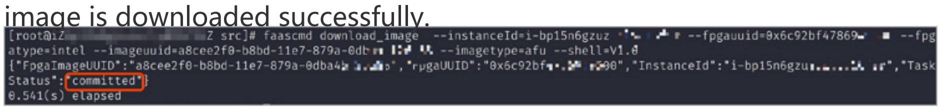
```
[root@iZ[redacted]Z output_files]# faascmd list_instances --instanceId=i-bp15n6gzu[redacted]
{"Instances":{"instance":[{"ShellUUID":"V0.11","FpgaType":"intel","FpgaUUID":"0x6c92bf4786940500","InstanceId":"i-bp15n6gzuzc[redacted]","DeviceBDF":"05:00.0","FpgaStatus":"valid"}]}}
```

Note:
Record the FpgaUUID.

Run the command to download the image.

```
# Replace hereIsYourInstanceID with your f2 instance ID. Replace hereIsFpgaUUID with the recorded FpgaUUID. Replace hereIsImageUUID with the recorded FpgaImageUUID
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --fpgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.0
```

The example returned result is displayed as follows. The "State":"committed" means the image is downloaded successfully.

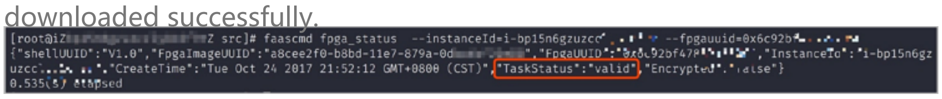


```
[root@iZ[redacted]Z src]# faascmd download_image  --instanceId=i-bp15n6gzuz[redacted] --fpgauuid=0x6c92bf47869[redacted] --fpg
atype=intel --imageuuid=a8cee2f0-b8bd-11e7-879a-0db[redacted] --imagetype=afu --shell=V1.0
{"FpgaImageUUID":"a8cee2f0-b8bd-11e7-879a-0dba4[redacted]","FpgaUUID":"0x6c92bf[redacted]90","InstanceId":"i-bp15n6gzu[redacted]","Task
Status":"committed"}
0.541(s) elapsed
```

Note:
You can run the command to check whether the image is downloaded successfully or not.

```
# Replace hereIsYourInstanceID with your f2 instance ID. Replace hereIsFpgaUUID with your recorded FpgaUUID
faascmd fpga_status --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID
```

The example returned result is displayed as follows. If "TaskStatus":"valid" exists and the displayed FpgaImageUUID is your image FpgaImageUUID, the image is downloaded successfully.



```
[root@iZ[redacted]Z src]# faascmd fpga_status  --instanceId=i-bp15n6gzuzc[redacted] --fpgauuid=0x6c92b[redacted]
{"shellUUID":"V1.0","FpgaImageUUID":"a8cee2f0-b8bd-11e7-879a-0d[redacted]","FpgaUUID":"0x6c92bf47[redacted]","InstanceId":"i-bp15n6gz
uzcc[redacted]","CreateTime":"Tue Oct 24 2017 21:52:12 GMT+0800 (CST)","TaskStatus":"valid","Encrypted":"false"}
0.535(s) elapsed
```

Step 6. Run the Host program

Run the commands to run the Host program.

```
make -f sdaccel.mk host
unset XILINX_SDX
./vadd bin_vadd_hw.xclbin
```

If Test Passed is returned, the test is successful.

Other common commands

In this section, some common commands for an FPGA instance are introduced.

Task	Command
View the help document	make -f ./sdaccel.mk help
Run software simulation	make -f ./sdaccel.mk run_cpu_em
Run hardware simulation	ake -f ./sdaccel.mk run_hw_em
Compile the host code only	make -f ./sdaccel.mk host
Compile and generate files for downloading	make -f sdaccel.mk xbin_hw
Clear a job directory	make -f sdaccel.mk clean
Force clear a job directory	make -f sdaccel.mk cleanall

Note:
During simulation of sdx2017.2, the device must be xilinx_kcu1500_4ddr-xpr_4_0.

This document describes how to use Register Transfer Level (RTL) compiler on an f2 instance.

Note:
We strongly recommend that you use an f2 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to operate essential actions only. You must create a role for the RAM user and grant temporary permissions to the role to access to the OSS buckets.

Prerequisites

Before you begin the procedure, you must:

- Create an f2 instance.
- Create a RAM user with granted permissions and create a RAM role with granted permissions, if you use an f2 instance as a RAM user. You must create a pair of AccessKeyID and AccessKeySecret for the RAM user.
- Activate OSS service and create a bucket.
- Log on to the ECS console and obtain the instance ID of your f2 instance.

Procedure

To use Register Transfer Level (RTL) compiler on an f2 instance, follow these steps.

Connect to an f2 instance.

Note:
It may take 2 or 3 hours to compile a project, thus, we recommend that you use nohup or VNC to connect to your f2 instance to avoid accidental connection interrupt.

Download the reference RTL design from http://faas-ref-design.oss-cn-hangzhou.aliyuncs.com/f2_ddr_prbs_rtl.tar.

Run the scripts to set up the environment on the f2 instance.

```
source /root/xbinst_oem/F2_env_setup.sh
```

Run the vim /source/misc/xclbin.xml command to modify the xclbin.xml file. Replace 300MHz with 200MHz.

Run the command to compile a project.

```
sh compiling.sh
```

Note:
It may take 2 or 3 hours to compile a project.

Run the command to upload the package to your own OSS bucket.

```
# Replace hereIsMySecretId with your own AccessKeyId. Replace hereIsMySecretKey with your own
AccessKeySecret. Replace hereIsMyBucket with your own OSS bucket name.
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey
faascmd auth --bucket=hereIsMyBucket
faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
```

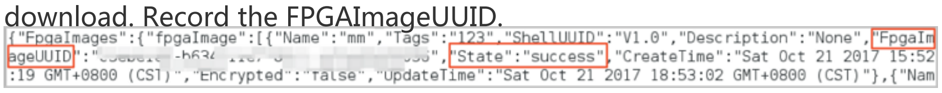
Run the command to upload the package to the OSS of the FaaS service.

```
faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --name=hereIsFPGAImageName --
tags=hereIsFPGAImageTag --encrypted=false --shell=V1.0
```

Run the command to check whether the FPGA image is available for download.

```
faascmd list_images
```

"State":"success" in the returned result means that the FPGA image is available for download. Record the FPGAImageUUID.



```
{"FpgaImages":{"fpgaImage":[{"Name":"mm","Tags":"123","ShellUUID":"V1.0","Description":"None","FpgaIm
ageUUID":"[redacted]","State":"success","CreateTime":"Sat Oct 21 2017 15:52
:19 GMT+0800 (CST)","Encrypted":"false","UpdateTime":"Sat Oct 21 2017 18:53:02 GMT+0800 (CST)"},{"Nam
```

Run the command to obtain FpgaUUID.

```
# Replace hereIsYourInstanceId with your f2 instance ID
faascmd list_instances --instanceId=hereIsYourInstanceId
```

Record FpgaUUID in the returned result.

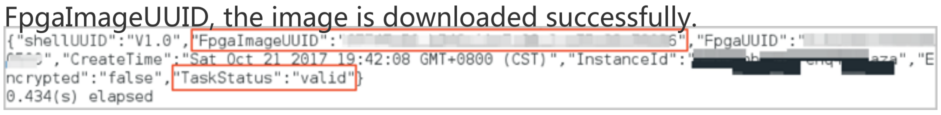
Run the command to download the FPGA image.

```
# Replace hereIsYourInstanceId with your f2 instance ID. Replace hereIsFpgaUUID with the recorded FpgaUUID. Replace hereIsImageUUID with the recorded FPGAImageUUID.
faascmd download_image --instanceId=hereIsYourInstanceId --fpgauuid=hereIsFpgaUUID --fpgatype=xilinx --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.0
```

Run the command to check the download status of the image.

```
# Replace hereIsFpgaUUID with the recorded FpgaUUID. Replace hereIsYourInstanceId with your f2 instance ID.
faascmd fpga_status --fpgauuid=hereIsFpgaUUID --instanceId=hereIsYourInstanceId
```

If ‘TaskStatus’:’valid’ is returned and the returned FpgaImageUUID is your image FpgaImageUUID, the image is downloaded successfully.

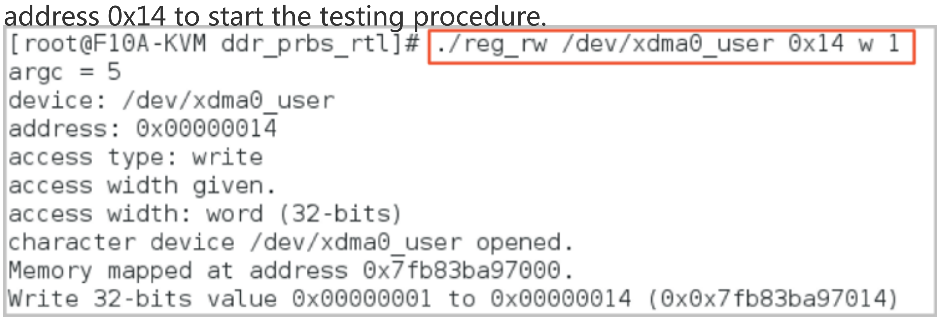


```
{"shellUUID":"V1.0","FpgaImageUUID":"[redacted]6","FpgaUUID":"[redacted]","CreateTime":"Sat Oct 21 2017 19:42:08 GMT+0800 (CST)","InstanceId":"[redacted]aza","Encrypted":"false","TaskStatus":"valid"}
0.434(s) elapsed
```

Run the Host program to run the tests:

Smoke testing on DDR:

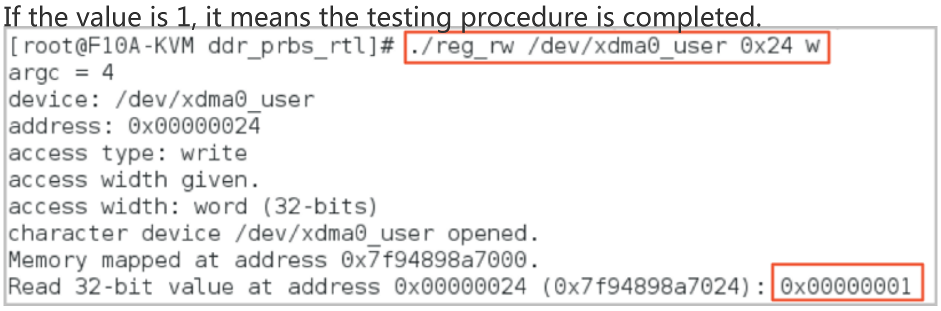
Run the ./reg_rw /dev/xdma0_user 0x14 w 1 command to write 1 to address 0x14 to start the testing procedure.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x14 w 1
argc = 5
device: /dev/xdma0_user
address: 0x00000014
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7fb83ba97000.
Write 32-bits value 0x00000001 to 0x00000014 (0x0x7fb83ba97014)
```

Run the ./reg_rw /dev/xdma0_user 0x24 w command to read from address 0x24.

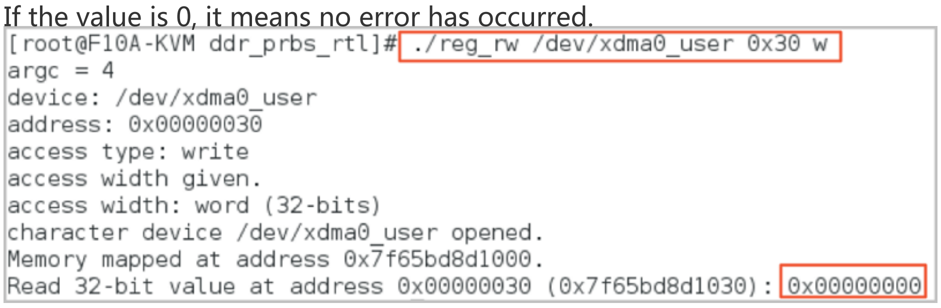
If the value is 1, it means the testing procedure is completed.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x24 w
argc = 4
device: /dev/xdma0_user
address: 0x00000024
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7f94898a7000.
Read 32-bit value at address 0x00000024 (0x7f94898a7024): 0x00000001
```

Run the ./reg_rw /dev/xdma0_user 0x30 w command to read the value of address 0x30.

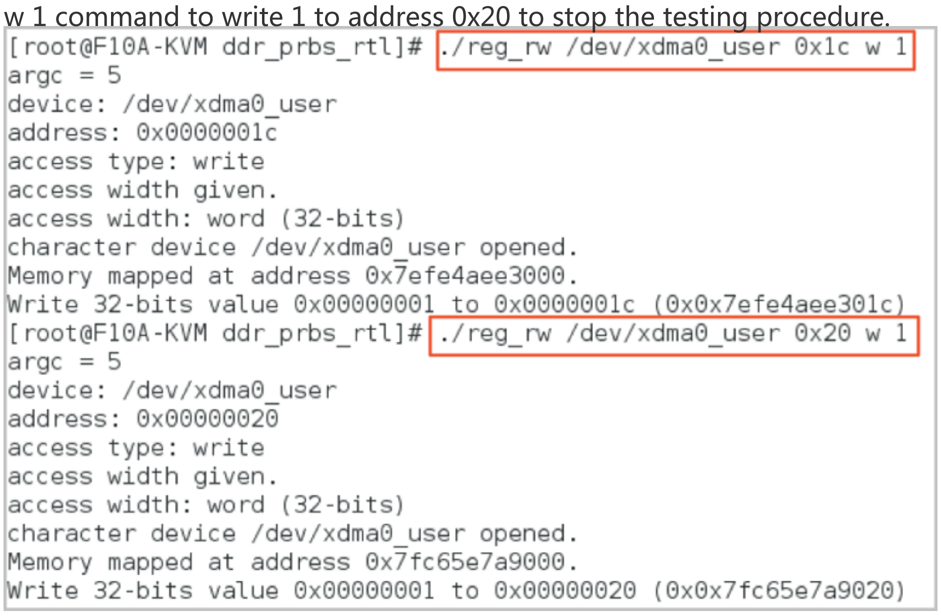
If the value is 0, it means no error has occurred.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x30 w
argc = 4
device: /dev/xdma0_user
address: 0x00000030
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7f65bd8d1000.
Read 32-bit value at address 0x00000030 (0x7f65bd8d1030): 0x00000000
```

PRBS testing on DDR:

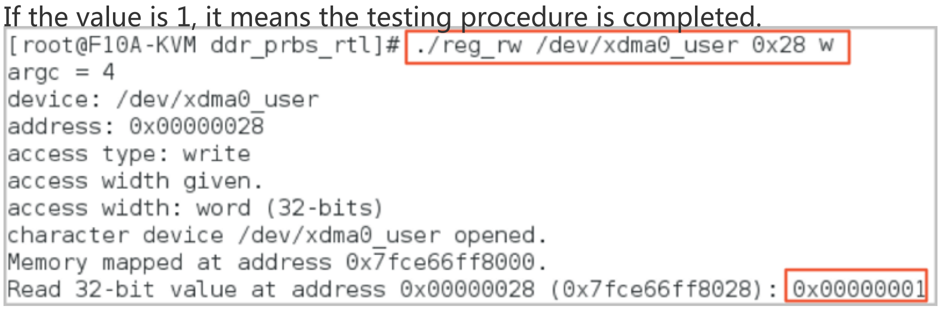
Run the ./reg_rw /dev/xdma0_user 0x1c w 1 command to write 1 to address 0x1c to start the testing procedure. After a period of time, which is determined by yourself, run the ./reg_rw /dev/xdma0_user 0x20 w 1 command to write 1 to address 0x20 to stop the testing procedure.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x1c w 1
argc = 5
device: /dev/xdma0_user
address: 0x0000001c
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7efe4aee3000.
Write 32-bits value 0x00000001 to 0x0000001c (0x0x7efe4aee301c)
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x20 w 1
argc = 5
device: /dev/xdma0_user
address: 0x00000020
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7fc65e7a9000.
Write 32-bits value 0x00000001 to 0x00000020 (0x0x7fc65e7a9020)
```

Run the ./reg_rw /dev/xdma0_user 0x28 w command to read the value of address 0x28.

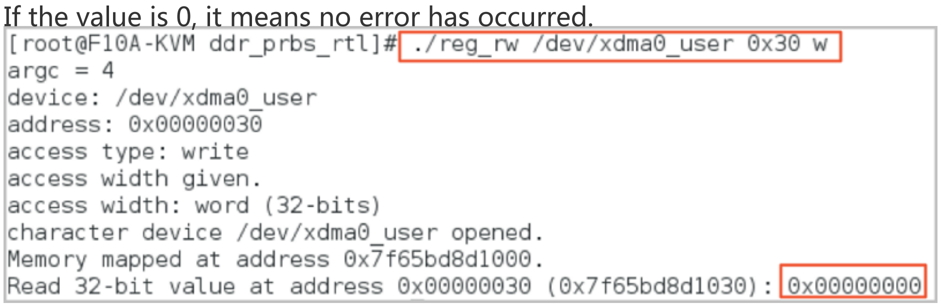
If the value is 1, it means the testing procedure is completed.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x28 w
argc = 4
device: /dev/xdma0_user
address: 0x00000028
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7fce66ff8000.
Read 32-bit value at address 0x00000028 (0x7fce66ff8028): 0x00000001
```

Run the ./reg_rw /dev/xdma0_user 0x30 w command to read the value of address 0x30.

If the value is 0, it means no error has occurred.



```
[root@F10A-KVM ddr_prbs_rtl]# ./reg_rw /dev/xdma0_user 0x30 w
argc = 4
device: /dev/xdma0_user
address: 0x00000030
access type: write
access width given.
access width: word (32-bits)
character device /dev/xdma0_user opened.
Memory mapped at address 0x7f65bd8d1000.
Read 32-bit value at address 0x00000030 (0x7f65bd8d1030): 0x00000000
```

P2V Migration Practice

Alibaba Cloud Migration Tool, often abbreviated as **Cloud Migration Tool**, is a proprietary resource migration tool of Alibaba Cloud. It supports resource migration, such as the operating system, applications, and application data in a computer disk, of physical machines, VMs (virtual machines), or cloud hosts to the image list in the ECS console.

Being lightweight and agile, Alibaba Cloud Migration Tool helps you balance the workload between your local and cloud hosts, or cloud hosts from different cloud platforms.

Scenarios

Alibaba Cloud Migration Tool is applicable in the following scenarios:

Migrate physical servers to Alibaba Cloud ECS console.

Migrate virtual machines to Alibaba Cloud ECS console.

Migrate the cloud hosts from other cloud platforms, such as Amazon Web Services (AWS), Microsoft Azure, Tencent Cloud, or Huawei Cloud, to Alibaba Cloud ECS console.

Applicable operating systems

Alibaba Cloud Migration Tool supports the following 64-bit Windows and Linux operating systems:

- Windows server
  - Windows Server 2003
  - Windows Server 2008

  - Windows Server 2012
  - Windows Server 2016

- Linux server
  - CentOS 5/6/7
  - Ubuntu 12/14/16
  - Debian 7/8/9
  - Red Hat 6/7
  - SUSE 11.4
  - OpenSUSE 13.1
  - Gentoo 13.0

To migrate an operating system that is not listed previously, you must perform with caution and see Migrate to Alibaba Cloud by using Cloud Migration Tool thoroughly.

Billing details

Alibaba Cloud Migration Tool is free of charge.

However, during the migration, an ECS instance is created by default under your Alibaba Cloud account to act as an intermediate station. Billing method of the intermediate ECS instance is Pay-As-You-Go, you must make sure that you have enough credit payment of your linked credit card.

Limits

Before migration, you must make sure that:

The on-premises server can access the Internet to uninterruptedly transfer data to the Alibaba Cloud ECS console.

The on-premises server is synchronized with the actual time. Otherwise, an error indicating abnormal TimeStamp is recorded in the migration log file.

For more information, see Migrate to Alibaba Cloud by using Cloud Migration Tool.

To use Alibaba Cloud Migration Tool

For instructions on how to use Alibaba Cloud Migration Tool, see Migrate to Alibaba Cloud by using Cloud Migration Tool.

References

Currently, the following methods are available for migrating servers to Alibaba Cloud:

- Using the Alibaba Cloud Migration Tool
- Importing images in the ECS console

Attention

To use Alibaba Cloud Migration Tool, consider the following:

Alibaba Cloud Migration Tool does not support migration of incremental data yet. You may select an off-peak period to suspend your on-premises server in which services that require data integrity are running.

During migration, an ECS instance named INSTANCE_FOR_GOTOALIYUN is created by default under your Alibaba Cloud account. It acts as an intermediate station.

Note:

- To avoid migration failure, do not stop, restart, or release the intermediate ECS instance. Moreover, the intermediate ECS instance is automatically released once the migration completes.
- If the migration fails, the intermediate ECS instance is retained in the ECS console for the next migration attempt. You can log on to the ECS console and manually release the instance to avoid unnecessary charges.

Billing method of the intermediate ECS instance is Pay-As-You-Go, you must make sure that no credit limit is set to your credit card and it allows the payment to go through.

After each successful migration, information about the intermediate ECS instance in the ECS console is automatically recorded in the client_data configuration file. For the next migration, you must use the raw client configuration file that you initially downloaded.

Note:
To avoid migration failure, in most cases, no manual modification for file client_data is required.

Migrating servers by using Cloud Migration Tool requires your AccessKeyID and AccessKeySecret, which are important credentials, and you must keep them confidential and secured.

Note:
If the AccessKey that you create belongs to a RAM user, you must make sure that the

specified RAM user is authorized to operate the ECS resources. For more information, see *RAM* document Authorization policies.

Prerequisites

Before you use the Alibaba Cloud Migration Tool, consider the following:

The on-premises server can access the Internet for uninterruptedly transferring data to Alibaba Cloud ECS console.

The system time of the on-premises server is synchronized with the actual time. Otherwise, an error indicating abnormal TimeStamp is recorded in the migration log file.

For on-premises servers running Windows OS

The go2aliyun_client.exe and Rsync\bin\rsync.exe programs are not restricted by firewall on the server.

The system start loader is normal.

Run Alibaba Cloud Migration Tool as an administrator.

For on-premises servers running Linux OS

The go2aliyun_client program is not restricted by firewall on the server.

The Rsync library has been installed.

- CentOS: Run yum install rsync –y.
- Ubuntu: Run apt-get install rsync –y.
- Debian: Run apt-get install rsync –y.
- Other distributions: See the installation documents of the distributions on their official website.

The Xen or Kernel-based Virtual Machine (KVM) driver is installed. For more information about how to install a KVM driver, see Install virtio driver.

SELinux must has been deactivated. You can deactivate SELinux by setting SELINUX=disabled in the /etc/selinux/config file.

Run Alibaba Cloud Migration Tool as a root user.

If the kernel of your on-premises Linux servers is too old and the version of GRUB (GRand Unified Bootloader) is earlier than 1.9. You may update the boot loader GRUB to a version later than 1.9.

Step 1. Download Alibaba Cloud Migration Tool

Log on to the ECS console to apply for migration.

Note:
After you apply for a migration, you must receive an email response. If you do not receive a reply for a longer time, you must check your inbox and spam mails.

After the approval, download and decompress the Alibaba Cloud Migration Tool package according to the Email. The list of files is as follows:

Windows server

File or file folder	Description
Rsync file folder	This folder contains all the applications required for a migration. Do not modify the files manually except for the filter file Rsync\etc\rsync_excludes_win.txt.
client_data	Records of the transmission data during a migration.
user_config.json	Configuration file of the on-premises server
go2aliyun_client.exe	Main program of Cloud Migration Tool

Linux server

File or file folder	Description
client_check	Auxiliary program
client_data	Records of the transmission data during a migration.
user_config.json	The configuration file of the on-premises server

rsync_excludes_linux.txt	This file filters out the directories from the migration.
go2aliyun_client	Main program of Cloud Migration Tool

Step 2. Use Alibaba Cloud Migration Tool

Log on to the server, virtual machine, or cloud host to be migrated.

Decompress the Cloud Migration Tool package to a specified directory.

In the console, create an AccessKey, which is used in file user_config.json.

Configure file user_config.json as needed.

Filter out directories from migration as needed.

Run Cloud Migration Tool:

- **Windows** server: Right-click go2aliyun_client.exe and select **Run as administrator**.
- **Linux** server:
  a. Run chmod +x go2aliyun_client to make go2aliyun_client executable.
  b. Run ./ go2aliyun_client to migrate.

Wait for the results.

- If Goto Aliyun Finished! is displayed, go to the image details page of the ECS console to check the results.
- If Goto Aliyun Not Finished! is displayed, check the log files in the logs folder for troubleshooting. After the problem is rectified, run go2aliyun_client again, and it continues to proceed from where it was suspended during the preceding execution.

Customize user_config.json

The user_config.json configuration file is edited in JSON. The file contains necessary configuration information when you migrate the on-premises server to be migrated, including your AccessKey and target custom image. You must manually configure a few parameters. Make sure that the configuration complies with the JSON syntax. For more information about JSON syntax, see RFC 7159
.

Template of user_config.json

The template for user_config.json is as follows:

```
{
"access_id": "",
"secret_key": "",
"region_id": "",
"image_name": "",
"system_disk_size": 40,
"platform": "",
"architecture": "",
"data_disks": [],
"bandwidth_limit":0
}
```

Parameters in template

Table 1. Parameters for server configuration

Parameter name	Type	Required	Description
access_id	String	Yes	Your AccessKeyID for accessing Alibaba Cloud API. For more information, see Create AccessKey.
secret_key	String	Yes	Your AccessKeySecret for accessing Alibaba Cloud API. For more information, see Create AccessKey.
region_id	String	Yes	The ID of an Alibaba Cloud region to which your server is migrated, for example, cn-hangzhou (China East 1). For more information, see Regions and zones.
image_name	String	Yes	Set a name for your server image, which must be different from the name of existing images in the same region of Alibaba Cloud. - The image name is a string of [2,

			128] English letters. - It must start with an uppercase or lowercase English letter. It can contain numbers, dots (.), underscores (_), and hyphens (-). - The image name is displayed in the ECS console. - It cannot start with http:// or https://.
system_disk_size	int	Yes	Specify the system disk size in the unit of GB. Value range: - [40, 500] - The value must be greater than the actually used space of the system disk on the on-premises server. For example, if the original system disk size is 500 GB and the used space is 100 GB, set this value greater

			than 100 GB..
platform	String	No	Operating system of the on-premises server. Optional values: - CentOS - Ubuntu - SUSE - OpenSUSE - Debian - RedHat - Others Linux - Windows Server 2003 - Windows Server 2008 - Windows Server 2012 - Windows Server 2016 The platform parameter is case sensitive.
architecture	String	No	Computer architecture. Optional values: - i386: 32-bit system architecture - x86_64: 64-bit system architecture
bandwidth_limit	int	No	The maximum bandwidth of data transmission, in the units of measurment KB/s. The default value is 0, and 0 indicates no limits for the bandwidth.
data_disks	Array	No	List of data disks in your on-premises server. A maximum of

			16 data disks are supported. For more information about specific parameters, see Parameters for data disk configuration.

Table 2. Parameters for data disk configuration

Parameter name	Type	Manually set	Description
data_disk_index	int	Yes	Data disk number. Value range: [1, 16] Default value: 1
data_disk_size	int	Yes	Data disk size. Unit: GB. Value range: - [5, 2000] - The value must be greater than the actually used space of the data disk on the on-premises server. For example, if the original data disk size is 500 GB and the used space is 100 GB, set this value to be greater than 100 GB.
src_path	String	Yes	The directory of a data disk. Optional values: - In Windows, specify a drive letter, for example, D:, E:, or F:. - In Linux,

			specify a path, for example, /mnt/disk1, mnt/disk2, or /mnt/disk3. After you migrate an on-premises Linux server, the data disks are not mounted by default. You can run the command ls /dev/vd* to view the data disk devices. You may mount the data disks manually as needed, and edit configuration file /etc/fstab to configure the mounting file systems. For more information, see Linux_Format and mount a data disk.

Examples of custom configuration

The following describes how to customize user_config.json based on the configuration file templates in four scenarios:

Scenario 1. Migrate a Windows server without data disk

- Assume that the configuration of your server is as follows:
    - Operating system: Windows Server 2008
    - Used space of system disk: 30 GB
    - Computer architecture: 64-bit
- Migration destination:
    - Target migration region: Alibaba Cloud China East 1 region (cn-hangzhou)
    - Image name: CLIENT_IMAGE_WIN08_01
    - Size of system disk: 50 GB

You can configure the file user_config.json based on the following information:

```
{
"access_id": "YourAccessKeyID",
"secret_key": "YourAccessKeySecret",
"region_id": "cn-hangzhou",
"image_name": "CLIENT_IMAGE_WIN08_01",
"system_disk_size": 50,
"platform": "Windows Server 2008",
"architecture": "x86_64",
"data_disks": [],
"bandwidth_limit":0
}
```

Scenario 2. Migrate a Windows server with data disks

Assume that the three data disks are attached to the Windows server in Scenario 1. The drive letter and sizes of the data disks are as follows:

- D: 100 GB
- E: 150 GB
- F: 200 GB

You can configure the user_config.json file based on the following information:

```
{
"access_id": "YourAccessKeyID",
"secret_key": "YourAccessKeySecret",
"region_id": "cn-hangzhou",
"image_name": "CLIENT_IMAGE_WIN08_01",
"system_disk_size": 50,
"platform": "Windows Server 2008",
"architecture": "x86_64",
"data_disks": [ {
"data_disk_index": 1,
"data_disk_size": 100,
"src_path": "D:"
}, {
"data_disk_index": 2,
"data_disk_size": 150,
"src_path": "E:"
}, {
"data_disk_index": 3,
"data_disk_size": 200,
"src_path": "F:"
}
],
"bandwidth_limit":0
}
```

Scenario 3. Migrate a Linux server without data disk

- Assume that the configuration of your server is as follows:
    - Version: CentOS 7.2
    - Used space of system disk: 30 GB
    - Computer architecture: 64-bit
- Migration destination:

- Target migration region: Alibaba Cloud China East 1 region (cn-hangzhou)
- Image name: CLIENT_IMAGE_CENTOS72_01
- Size of system disk: 50 GB

You can configure the user_config.json file based on the following information:

```
{
"access_id": "YourAccessKeyID",
"secret_key": "YourAccessKeySecret",
"region_id": "cn-hangzhou",
"image_name": "CLIENT_IMAGE_CENTOS72_01",
"system_disk_size": 50,
"platform": "CentOS",
"architecture": "x86_64",
"data_disks": [],
"bandwidth_limit":0
}
```

Scenario 4. Migrate a Linux server with data disks

Assume that the three data disks are attached to the Linux server in Scenario 3. The drive letter and sizes of the data disks are as follows:

- /mnt/disk1: 100 GB
- /mnt/disk2: 150 GB
- /mnt/disk3: 300 GB

You can configure the user_config.json file based on the following information:

```
{
"access_id": "YourAccessKeyID",
"secret_key": "YourAccessKeySecret",
"region_id": "cn-hangzhou",
"image_name": "CLIENT_IMAGE_CENTOS72_01",
"system_disk_size": 50,
"platform": "CentOS",
"architecture": "x86_64",
"data_disks": [ {
"data_disk_index": 1,
"data_disk_size": 100,
"src_path": "/mnt/disk1"
}, {
"data_disk_index": 2,
"data_disk_size": 150,
"src_path": "/mnt/disk2"
}, {
"data_disk_index": 3,
"data_disk_size": 200,
"src_path": "/mnt/disk3"
}
],
"bandwidth_limit":0
}
```

Filter out directories from migration

Alibaba Cloud Migration Tool can be used to filter out files or directories that are not migrated to Alibaba Cloud. Default files to be filtered out are as follows:

- Windows servers:
  - pagefile.sys
  - $RECYCLE.BIN
  - System Volume Information
- Linux servers:
  - /dev/*
  - /sys/*
  - /proc/*
  - /media/*
  - /lost+found/*
  - /mnt/*
  - /var/lib/lxcfs/*

  ...

Note:

- Directory /var/lib/lxcfs/* is only applicable to some versions. For example, when you do not have the access permission on the cache directory for Linux containers of Ubuntu, /var/lib/lxcfs/* must be filtered out before migration.
- Cloud migration is a time-consuming task, and we recommend that you filter out the unnecessary directories on the data disks to be migrated, at the same time, the used storage space of cloud disks after the migration will be also reduced.

You can configure rsync to filter out files and directories from migrating to Alibaba Cloud. On a Linux server, the filtering function is implemented by configuring the **rsync_excludes_linux.txt** text file. On a Windows server, the filtering function is implemented by configuring the **Rsync/etc/rsync_excludes_win.txt** text file. For how to add files and directories to be filtered out in the .txt file, see Documents Related to rsync.

Troubleshooting

Log files

Log records of Alibaba Cloud Migration Tool are stored in the Logs directory under the main program directory. Abnormal interruptions during migration are recorded in log files. When Goto Aliyun Not Finished is displayed, you can check the log for troubleshooting.

FAQ

1. Keyword “TimeStamp” appears in the migration logs.

Check whether the system time is correct or not.

2. Keyword “OperationDenied” appears in the migration logs.

Check whether your Alibaba Cloud account, to which the access_id parameter in the user_config.json configuration file belongs, has applied for the Cloud Migration Tool whitelist.

3. Keyword “check rsync failed” appears in the migration logs.

Check whether the rsync component is installed or not.

4. Keyword “check virtio failed” appears in the migration logs.

Check whether the virtio driver is installed or not.

5. Keyword “check selinux failed” appears in the migration logs.

Check whether SElinux is deactivated or not.

You can set SELINUX=disabled in the /etc/selinux/config file to deactivate SELinux.

6. Keyword “Do Grub Failed” appears in the migration logs of a Linux server.

Check whether the on-premises server has correctly installed the GRUB (GRand Unified Bootloader) or not. You can install a GRUB with the version newer than 1.9 and try again.

7. Keyword “Unknown Error” appears in the migration logs.

Check whether the value of the platform parameter is correct in file user_config.json or not.

8. Keyword “Permission denied” appears in the migration logs.

If rsync: send_files failed to open "...": Permission denied (13) is displayed in the log, Alibaba Cloud Migration Tool has no access permission on the directory or folder, which leads to rsync failure. You can configure rsync_excludes_linux.txt or Rsync/etc/rsync_excludes_win.txt to filter out the specified directory or folder and try again.

9. Keyword “NotEnoughBalance” appears in the migration logs.

The default billing method of the intermediate instance is Pay-As-You-Go, you must make sure that no credit limit is set to your credit card and it allows the payment to go through.

10. Keyword “Forbidden.RAM” appears in the migration logs.

If the AccessKey that you create belongs to a RAM user, you must make sure that the specified RAM user is authorized to operate the ECS resources. For more information, see *RAM* document Authorization policies.

11. Keyword “InvalidImageName.Duplicated” appears in the migration logs.

The specified parameter image_name cannot be the same as an existing image name.

12. Why no data is found in the original data disk directory in the started Linux ECS instances?

After you migrate an on-premises Linux server, the data disks are not mounted by default. You can run the command ls /dev/vd* to view the data disk devices. You may mount the data disks manually as needed, and edit configuration file /etc/fstab to configure the mounting file systems. For more information, see Linux _ Format and mount a data disk.

13. What should I do if the drive letters of data disks are missing or wrong after the Windows server migration?

If the drive letters are missing, you can add the drive letters in the **Disk Management**. For more information, see Windows_Format a data disk.

If the drive letters are wrong, you can replace the drive letters in the **Disk Management**. For more information, see Windows_Format a data disk.

14. What should I do if network service is abnormal when I start the migrated Others Linux instances?

When an image of Others Linux type is imported, Alibaba Cloud performs no configuration, including network configuration and SSH configuration, on ECS instances created by custom images. You can manually modify the configurations according to configure the Customized Linux image.

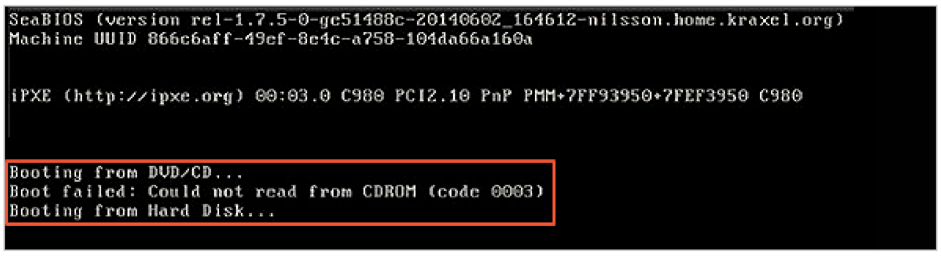
If network configuration fails, you can open a ticket to contact us.

15. Why cannot I start the created ECS instances after Linux server migration?

Check the driver. Before creating the I/O optimized instances, make sure that the virtio driver is installed on the on-premises server.

Check whether the boot configurations of the on-premises server are normal.

Connect to the ECS instance by using the Management Terminal in the ECS console, if the following output appears:



```
SeaBIOS (version rel-1.7.5-0-ge51488c-20140602_164612-nilsson.home.kraxel.org)
Machine UUID 866c6aff-49cf-8c4c-a758-104da66a160a


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+7FF93950+7FEF3950 C980


Booting from DVD/CD...
Boot failed: Could not read from CDROM (code 0003)
Booting from Hard Disk...
```

Perhaps the kernel of your on-premises Linux servers is too old, and the version of GRUB (GRand Unified Bootloader) is earlier than 1.9. You may update the boot loader GRUB to a version later than 1.9.

If the problem persists, you can join the dedicated DingTalk Migration Tool group chat to contact Alibaba Cloud.

Migration Tool
Technical Support

Image test

After a migration, the resource of your on-premises server, such as the operating system, applications, and application data, is displayed as a custom image in the image list of the ECS console. You can create a Pay-As-You-Go instance by using the custom image to test whether the custom image works or not.

Further operations

- You can create an instance by using the custom image.
- You can change the system disk by using the custom image.

Change log

The following table has the updated information about Alibaba Cloud Migration Tool:

Time	Release	Description
January 18, 2018	1.2.0	- Extend the range of data resource to be migrated, and more types of resource can be migrated. - Enhance the efficiency and stability of image creation.
January 01, 2018	1.1.8	- Supports SUSE 12 SP2 server. - Boosts the connection speed. - Optimizes the layout of the migration logs. - Fixes the possible network issue of the NetworkManager.
December 21, 2017	1.1.7	- Supports SUSE 12 SP1 server. - Is able to specify the maximum bandwidth of data transmission.
December 14, 2017	1.1.6	- Scans the latest release of Alibaba Cloud Migration Tool to update. - Fixes the 6144 error of data transmission. - Proofreads the request parameter specified in the user_config.json configuration file.

December 08, 2017	1.1.5	- Fixes the issue of Linux data disk directory. - Highlights the error message in the migration logs.
December 01, 2017	1.1.3	Supports Debian server.