

# Alibaba Cloud Object Storage Service

## Best Practices

Issue: 20190202

## Legal disclaimer

---

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use








or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.



## Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 <b>Danger:</b> Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 <b>Warning:</b> Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 <b>Notice:</b> Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 <b>Note:</b> You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
<b>Bold</b>	It is used for buttons, menus, page names, and other UI elements.	Click <b>OK</b> .
Courier font	It is used for commands.	Run the <code>cd /d C:/windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[ ] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand   slave}</code>



# Contents

---

Legal disclaimer.....	I
Generic conventions.....	I
<b>1 Migrate data to OSS.....</b>	<b>1</b>
1.1 Migrate data from Amazon S3 to Alibaba Cloud OSS.....	1
1.2 Use OssImport to migrate data.....	5
1.3 Back up an HDFS to OSS for disaster tolerance.....	8
1.4 Migrate data from AWS S3 to Alibaba Cloud OSS by using IPSec VPN and Express Connect.....	11
<b>2 Direct upload to OSS from Web.....</b>	<b>28</b>
2.1 Direct transfer after adding a signature on the server.....	28
<b>3 Application server.....</b>	<b>35</b>
3.1 Permission control.....	35
<b>4 Data backup and recovery.....</b>	<b>42</b>
4.1 Back up buckets.....	42
<b>5 Bucket management.....</b>	<b>44</b>
5.1 Storage class conversion.....	44
5.2 Anti-leech.....	47
5.3 Static website hosting.....	56
<b>6 Data security.....</b>	<b>60</b>
6.1 Check data transmission integrity by using 64-bit CRC.....	60
6.2 Protect data through client encryption.....	62
<b>7 Terraform.....</b>	<b>67</b>
7.1 Introduction.....	67
7.2 Use Terraform to manage OSS.....	68



# 1 Migrate data to OSS

---

## 1.1 Migrate data from Amazon S3 to Alibaba Cloud OSS

OSS provides S3 API compatibility that allows seamless migration of data from Amazon S3 to Alibaba Cloud OSS. After data is migrated from Amazon S3 to OSS, you can still use S3 APIs to access OSS. You only need to configure your S3 client application as follows:

1. Acquire the AccessKeyId and AccessKeySecret of your OSS primary account and sub-account, and configure the acquired AccessKeyID and AccessKeySecret in the client and SDK you are using.
2. Configure the endpoint for client connection to OSS endpoint. For more information, see [Regions and endpoints](#).

### Migration procedures

For details about migration procedures, see [Use OssImport to migrate data](#).

### Use S3 APIs to access OSS after migration

Take note of the following when you use S3 APIs to access OSS after the migration from S3 to OSS.

- Path style and virtual hosted style

*Virtual hosted style* supports accessing OSS by placing the bucket into the host header. For security reasons, OSS only supports virtual hosted style access. Therefore, configurations on your client application are required after the migration from S3 to OSS. Some S3 tools use path style access by default, which also requires proper configurations. Otherwise, OSS may report errors and prohibit access.

- Permission definitions in OSS are not quite the same as they are in S3. You may adjust the permissions as necessary after the migration. See the following table for the main differences between the two.



#### Note:

- See [OSS access](#) for more information on the differences.

- OSS supports only three canned ACL modes in S3: private, public-read, and public-read-write.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
Bucket	READ	With the List permission on the bucket	For all objects under the bucket , if no object permission is set for an object, the object is readable.
	WRITE	Objects in the bucket are writable or overwritable.	<ul style="list-style-type: none"> <li>- Writable for objects not existing under the bucket.</li> <li>- If no object permission is set for an object existing in the bucket , the object is overwritable.</li> <li>- Initiate multipart upload is allowed.</li> </ul>
	READ_ACP	Read bucket ACLs.	Read bucket ACLs . Only the bucket owner and the authorized sub-account have the permission of reading bucket ACLs.
	WRITE_ACP	Configure bucket ACLs.	Configure bucket ACLs. Only the bucket owner and the authorized sub-account have the permission of configuring bucket ACLs.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
Object	READ	Objects are readable.	Objects are readable.
	WRITE	N/A	Objects are overwritable.
	READ_ACP	Read object ACLs.	Read object ACLs . Only the bucket owner and the authorized sub-account have the permission of reading object ACLs.
	WRITE_ACP	Configure object ACLs.	Configure object ACLs. Only the bucket owner and the authorized sub-account have the permission of configuring object ACLs.

- Storage classes

OSS supports the Standard, IA, and Archive storage classes, which correspond to STANDARD, STANDARD\_IA, and GLACIER respectively in Amazon S3.

Different from Amazon S3, OSS does not support specifying the storage class directly when uploading an object. The storage class of the object is determined by that of the bucket. OSS supports three bucket storage classes: Standard, IA, and Archive. You can use the lifecycle rules to automatically transition objects between storage classes.

To read an Archive object in OSS, restore it first by initiating a restore request. Different from S3, OSS does not allow setting the lifetime of the restored (active ) copy. Therefore, OSS ignores the lifetime (Days) set in the S3 API. The restored state lasts for one day by default, and can be prolonged to seven days at most. After that, the object enters the frozen state again.

- ETag

- For the object uploaded by using a PUT request, the ETag of an OSS object and that of an Amazon S3 object differ in case sensitivity. The ETag is in upper case for an OSS object and in lower case for an S3 object. If your client has ETag validation, ignore case.
- For the objects uploaded by Multipart Upload, OSS takes the ETag calculation method that is different from S3.

#### Compatible S3 APIs

- **Bucket operations:**
  - Delete Bucket
  - Get Bucket (list objects)
  - Get Bucket ACL
  - Get Bucket lifecycle
  - Get Bucket location
  - Get bucket Logging
  - Head Bucket
  - Put Bucket
  - Put Bucket ACL
  - Put Bucket lifecycle
  - Put Bucket logging
- **Object operations:**
  - Delete Object
  - Delete Objects
  - Get Object
  - Get Object ACL
  - Head Object
  - Post Object
  - Put Object
  - Put Object Copy
  - Put Object ACL
- **Multipart operations:**
  - Abort Multipart Upload
  - Complete Multipart Upload

- **Initiate Multipart Upload**
- **List Parts**
- **Upload Part**
- **Upload Part Copy**

## 1.2 Use OssImport to migrate data

This topic describes how to migrate data from third-party storage products (or from another OSS source) to OSS by using OssImport.

### Environment configuration

OssImport can be deployed in two modes: standalone mode and distributed mode.

- Standalone mode applies to small-scale data migration scenarios where the data size is smaller than 30 TB.
- Distributed mode applies to large-scale data migration scenarios.

For example, you need to migrate 500 TB of data from an AWS S3 bucket in the Tokyo region to an OSS bucket in the China East 1 (Hangzhou) region within a week. Before migrating the data, you must configure environments to deploy OssImport in distributed mode as follows:

- **Activate OSS.**
  1. Use your Alibaba Cloud account to create an OSS bucket in China East 1 (Hangzhou).
  2. Create a RAM user in the RAM console, and then grant OSS access permissions to the RAM user. You also need to securely store the AccessKeyID and AccessKeySecret of the RAM user.
- **Purchase ECS instances.**

Purchase ECS instances with two CPUs and 4 GB of memory in the China East 1 (Hangzhou) region (in which the OSS bucket is also created). If you want to release the ECS instances after data migration, we recommend that you select Pay-As-You-Go as the billing method when purchasing the instances.

The number of required ECS instances can be calculated as follows:  $X/Y/(Z/100)$ . In the formula, X indicates the size of data that needs to be migrated, Y indicates the number of days that the migration requires, and Z indicates the transfer speed of a single ECS instance (MB/s), that is, how much TB of data can be migrated by a

single ECS instance each day (calculated as  $Z/100$ ). Assume that the transfer speed of an ECS instance is 200 MB/s (that is, an ECS instance can migrate 2 TB of data each day). This means you must purchase 36 ECS instances in total (calculated from  $500/7/2$ ).

- **Configure OssImport**

For the large-scale data migration requirement in this example, you must deployed OssImport in ECS instances in distributed mode. For the configuration information about distributed mode, such as `conf/job.cfg`, `conf/sys.properties`, and concurrency control, see [Architecture and configuration](#). For more information about OssImport distributed deployment, such as the downloading method of OssImport and the troubleshooting of OssImport configuration, see [Distributed deployment](#).

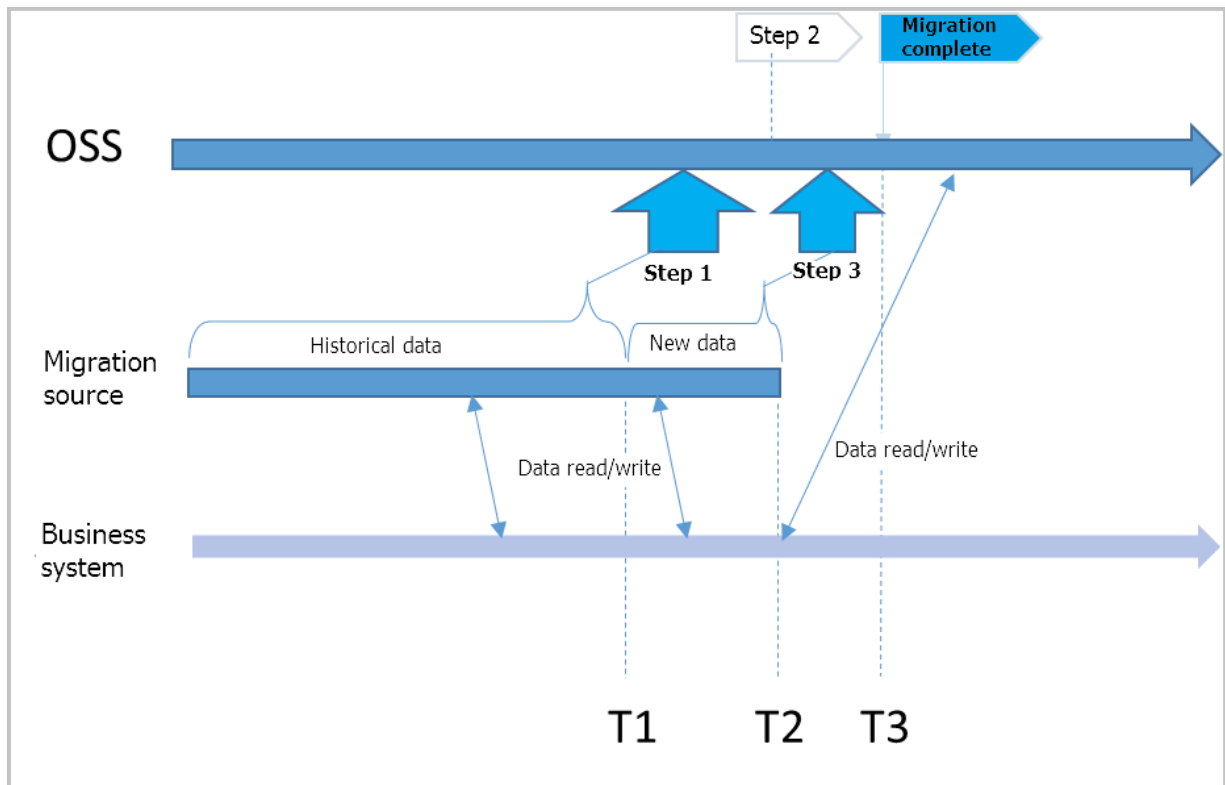
### Procedure

You can use OssImport in distributed mode to migrate data from AWS S3 to OSS as follows:



**Note:**

After deploying OssImport in distributed mode in the ECS instances, use OssImport to download data from the AWS S3 bucket in the Tokyo region to the ECS instances in China East 1 (Hangzhou). We recommend you download the data through Internet. Use OssImport to upload the data from the ECS instances to the OSS bucket in China East 1 (Hangzhou). We recommend you upload the data through the intranet.



1. Fully migrate the historical data in AWS S3 before the time T1. For more information, see the Running section in *Distributed deployment*.

T1 is a UNIX timestamp, namely, the number of seconds elapsed since UTC 00:00, January 1, 1970, and can be obtained by running the `date +%s` command).

2. In the OSS console, enable *Back-to-Origin* for the target bucket and set the access URL of the AWS S3 as the origin URL.
3. Switch all read and write operations on AWS S3 to OSS, and record the time (T2).

In this way, all historical data before T1 is directly read from the OSS bucket, and the data stored between T1 and T2 is read from AWS S3 through the mirroring back-to-origin function of OSS.

After T2, all new data is written to OSS and no new data is written to AWS S3.

4. Modify the item `importSince=T1` in the configuration file `job.cfg`, and then start a migration task again to migrate the data added between T1 and T2.



Note:

- After step 4, all read and write operations are performed on the target OSS bucket. Data stored in AWS S3 is historical data, which can be retained or deleted as needed.

- OssImport only migrates and verifies data but does not delete it.

Various costs are incurred during data migration, including the cost of ECS instances, traffic costs, storage costs, and time-dependent costs. Furthermore, if the size of the data to be migrated is larger than 1 TB, the storage cost increases due to the time required for the migration. However, the storage cost generally remains lower than the costs associated with network traffic and ECS instances. You can reduce the time needed for migration by using more ECS instances.

## References

For more information about OssImport, see the following documentations:

[Distributed deployment](#)

[Architecture and configuration](#)

[Data migration](#)

[FAQ](#)

## 1.3 Back up an HDFS to OSS for disaster tolerance

### Background

Currently, many data centers are constructed using Hadoop, and in turn an increasing number of enterprises want to smoothly migrate their services to the cloud.

Object Storage Service (OSS) is the most widely-used storage service on Alibaba Cloud. The OSS data migration tool, `ossimport2`, allows you to sync files from your local devices or a third-party cloud storage service to OSS. However, `ossimport2` cannot read data from Hadoop file systems. As a result, it becomes impossible to make full use of the distributed structure of Hadoop. In addition, this tool only supports local files. Therefore, you must first download files from your Hadoop file system (HDFS) to your local device and then upload them using the tool. This process consumes a great deal of time and energy.

To solve this problem, Alibaba Cloud's E-MapReduce team developed a Hadoop data migration tool `emr-tools`. This tool allows you to migrate data from Hadoop directly to OSS.

This chapter introduces how to quickly migrate data from HDFS to OSS.



## Prerequisites

Make sure your current machine can access your Hadoop cluster. That is, you must be able to use Hadoop commands to access HDFS.

```
hadoop fs -ls /
```

## Migrate Hadoop data to OSS

1. Download [emr-tools](#).



### Note:

emr-tools is compatible with Hadoop versions 2.4.x, 2.5.x, 2.6.x, and 2.7.x. If you require compatibility with other Hadoop versions, [open a ticket](#).

2. Extract the compressed tool to a local directory.

```
tar jxf emr-tools.tar.bz2
```

3. Copy HDFS data to OSS.

```
cd emr-tools
./hdfs2oss4emr.sh /path/on/hdfs oss://accessKeyId:accessKeySecret@
bucket-name.oss-cn-hangzhou.aliyuncs.com/path/on/oss
```

The relevant parameters are described as follow.

Parameters	Description
accessKeyId	The key used to access OSS APIs.
accessKeySecret	For more information, see <a href="#">How to obtain AccessKeyId and AccessKeySecret</a> .
bucket-name.oss-cn-hangzhou.aliyuncs.com	The OSS access domain name, including the bucket name and endpoint address.

The system enables a Hadoop MapReduce task (DistCp).

4. After the task is completed, local data migration information is displayed. This information is similar to the following sample.

```
17/05/04 22:35:08 INFO mapreduce.Job: Job job_1493800598643_0009
completed successfully
17/05/04 22:35:08 INFO mapreduce.Job: Counters: 38
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=859530
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=263114
    HDFS: Number of bytes written=0
```

```

HDFS: Number of read operations=70
HDFS: Number of large read operations=0
HDFS: Number of write operations=14
OSS: Number of bytes read=0
OSS: Number of bytes written=258660
OSS: Number of read operations=0
OSS: Number of large read operations=0
OSS: Number of write operations=0
Job Counters
  Launched map tasks=7
  Other local map tasks=7
  Total time spent by all maps in occupied slots (ms)=60020
  Total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=30010
  Total vcore-milliseconds taken by all map tasks=30010
  Total megabyte-milliseconds taken by all map tasks=45015000
Map-Reduce Framework
  Map input records=10
  Map output records=0
  Input split bytes=952
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=542
  CPU time spent (ms)=14290
  Physical memory (bytes) snapshot=1562365952
  Virtual memory (bytes) snapshot=17317421056
  Total committed heap usage (bytes)=1167589376
File Input Format Counters
  Bytes Read=3502
File Output Format Counters
  Bytes Written=0
org.apache.hadoop.tools.mapred.CopyMapper$Counter
  BYTESCOPIED=258660
  BYTESEXPECTED=258660
  COPY=10
copy from /path/on/hdfs to oss://accessKeyId:accessKeySecret@bucket-
name.oss-cn-hangzhou.aliyuncs.com/path/on/oss does succeed !!!

```

### 5. You can use osscmd to view information about OSS data.

```
osscmd ls oss://bucket-name/path/on/oss
```

## Migrate OSS data to Hadoop

If you have already created a Hadoop cluster on Alibaba Cloud, you can use the following command to migrate data from OSS to the new Hadoop cluster.

```
./hdfs2oss4emr.sh oss://accessKeyId:accessKeySecret@bucket-name.oss-cn-
hangzhou.aliyuncs.com/path/on/oss /path/on/new-hdfs
```

## More scenarios

In addition to offline clusters, you can also use emr-tools for Hadoop clusters constructed on ECS. This allows you to quickly migrate a self-built cluster to the [E-MapReduce](#) service.

If your cluster is already on ECS, but in a classic network, it will not provide good interoperability with services in Virtual Private Cloud (VPC). In this case, migrate the cluster to a VPC instance. Follow these steps to migrate the cluster:

1. Use `emr-tools` to migrate data to OSS.
2. Create a new cluster (create it yourself or use E-MapReduce) in the VPC environment.
3. Migrate data from OSS to the new HDFS cluster.

If you use E-MapReduce, on the Hadoop cluster, you can directly access OSS using [Spark](#), [MapReduce](#) and [Hive](#). This not only avoids one data copy operation (from OSS to HDFS), but also greatly reduces storage costs. For more information about cost reduction, see [EMR+OSS: Separated storage and computing](#).

## 1.4 Migrate data from AWS S3 to Alibaba Cloud OSS by using IPsec VPN and Express Connect

This topic describes how to migrate data from a AWS S3 bucket to a Alibaba Cloud OSS bucket by using IPsec VPN tunnel and ExpressConnect.

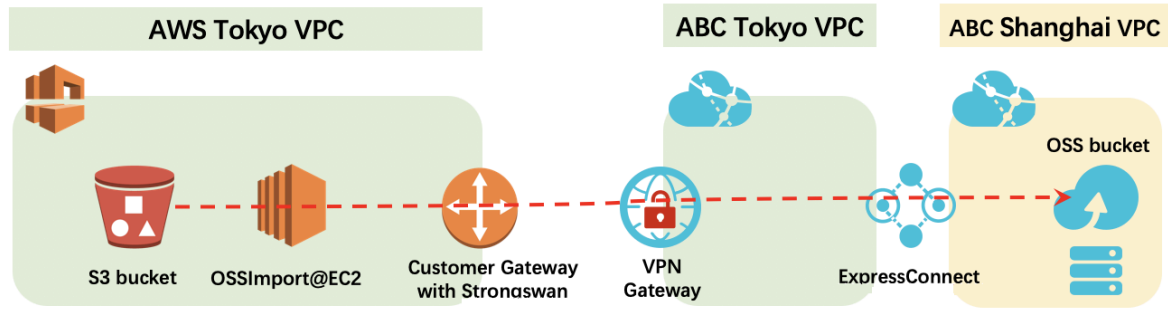
### Background information

When customers migrate data from AWS S3 to Alibaba Cloud OSS, especially cross countries, there are usually two network architectures for them to choose from:

- Transfer data based on the Internet.
- Transfer data based on private network.

This topic focuses on the second network architecture. The IPSEC VPN tunnel and the Alibaba Cloud ExpressConnect product portfolio are combined to meet the customer's needs of transmitting data through internal networks and increasing the security of data transmission.

The following figure shows the overall network architecture.



The advantage of this network architecture is that the data in the S3 bucket is first moved to the Alibaba Cloud Network VPC in the same region as the S3 bucket, and then the data is retransmitted to the OSS bucket of the destination region by means of the Alibaba Cloud ExpressConnect cross-region high-speed network. This network architecture accelerates the transmission speed of cross-country data migration.

### Preparation

The following table describes the information about the network entities involved in the migration process.

Provider	Network entity	Value
AWS	VPC in Tokyo	172.16.0.0/16
	S3 bucket Endpoint	s3.ap-northeast-1.amazonaws.com
	S3 bucket name	eric-s3-tokyo
	OssImport IP address in the EC2 instance	Internal IP address: 172.16.1.183 Public IP address: 3.112.29.59
	Customer Gateway with Strongswan	Public IP address: 3.112.29.59
Alibaba Cloud	VPN Gateway	Public IP address: 47.74.46.62
	VPC in Tokyo Japan	172.24.0.0/16
	Subnet in VPC in Tokyo Japan	172.24.0.0/20
	VPC in Shanghai China	172.19.0.0/16
	Subnet in VPC in Shanghai China	172.19.48.0/20

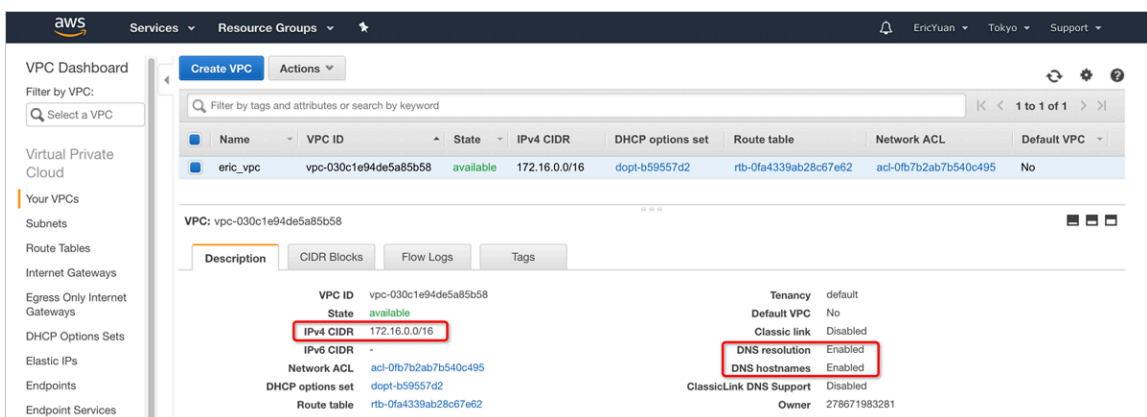
Provider	Network entity	Value
	OSS bucket endpoint	http://oss-cn-shanghai-internal.aliyuncs.com
	OSS bucket name	eric-oss-datastore-shanghai

**Step 1: Install and configure Strongswan.**

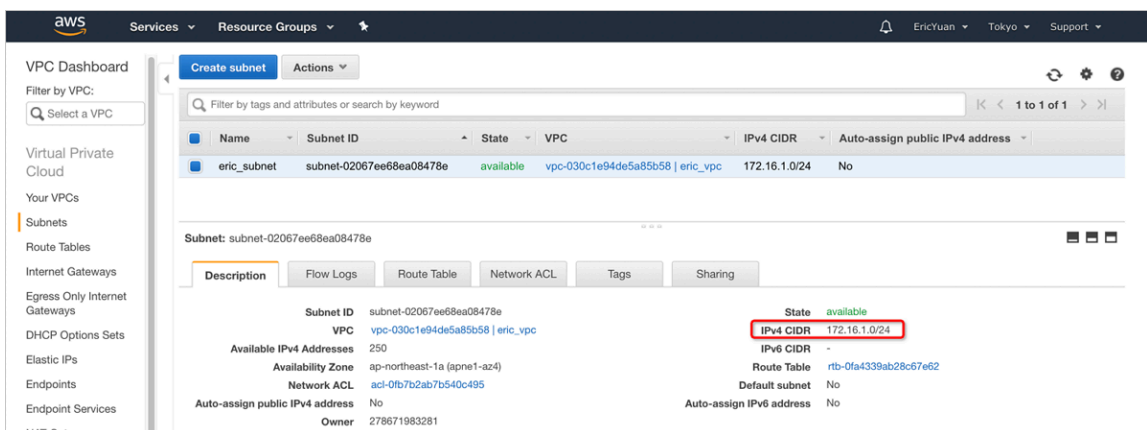
Perform the following operations in AWS to install Strongswan.

**1. Prepare a VPC and related resources.**

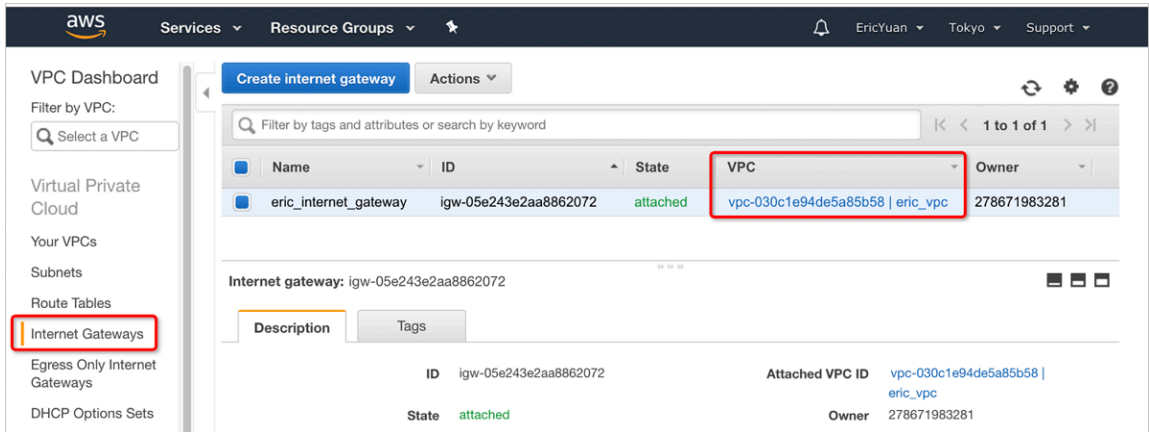
**a. Create a VPC with the following settings.**



**b. Create a subnet with the following settings.**

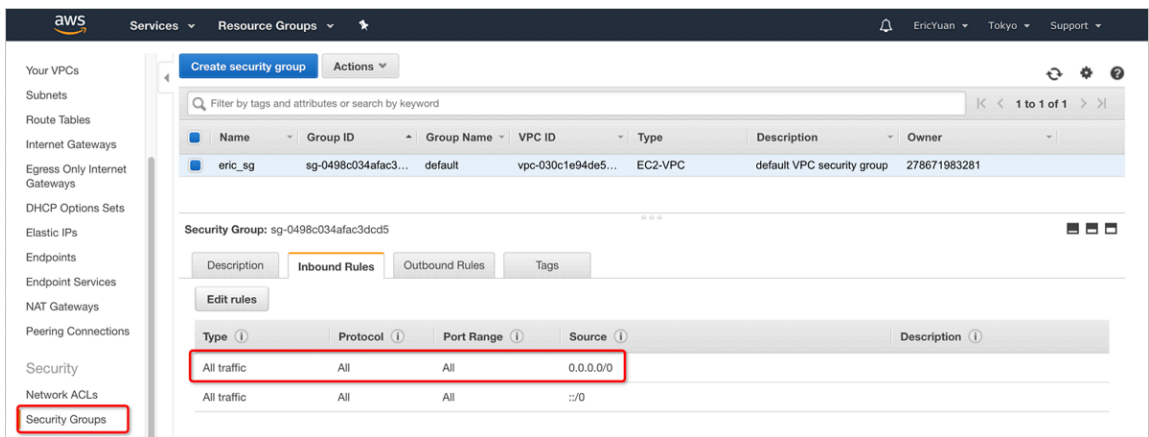


**c. Create an Internet gateway with the following settings.**



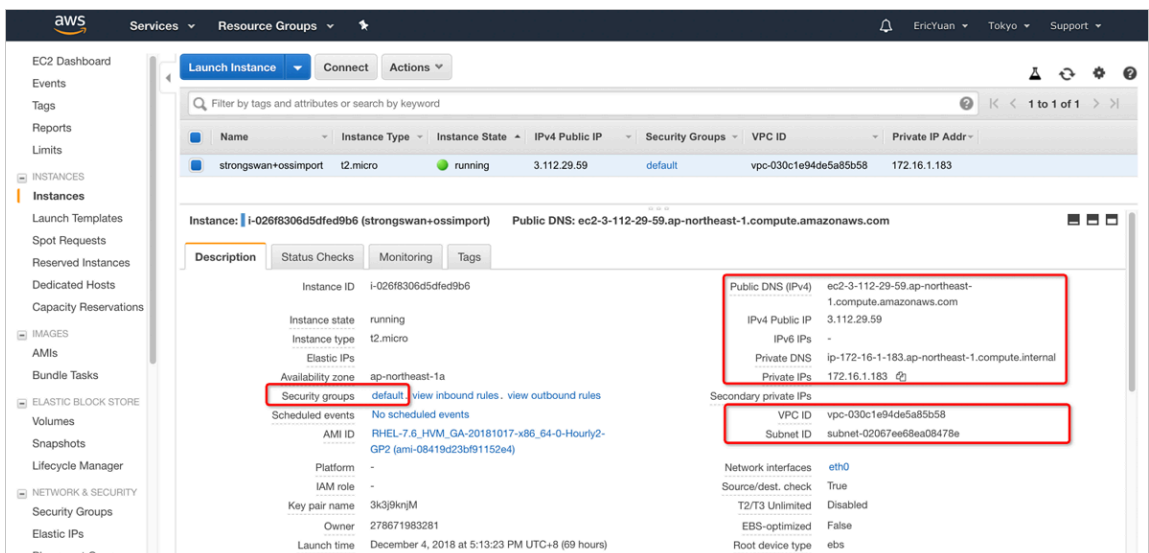
**Note:**  
 If you want to access EC2 via Internet, such as SSH client, then this gateway is necessary, attach this Internet gateway to the VPC that you just created.

d. Create a security group and allow the traffic.



2. Create an EC2 instance for Strongswan and OssImport.

a. Launch an EC2 instance in the VPC, subnet and security group.



**Note:**

If you want to access the EC2 instance with SSH client, you need to save the \*.pem file to local compute devices.

**b. Test to access EC2 with SSH client.**

Run the following command to log on the the EC2 instance via SSH:

```
ssh -i 3k3j***M.pem ec2-user@ec2-3-112-29-59.ap-northeast-1.compute.amazonaws.com
```

```
[root@EC-Demo-Japan ~]#
[root@EC-Demo-Japan ~]# ssh -i 3k3j***M.pem ec2-user@ec2-3-112-29-59.ap-northeast-1.compute.amazonaws.com
Last login: Fri Dec 7 04:25:32 2018 from 42.120.75.133

[ec2-user@ip-172-16-1-183 ~]$
[ec2-user@ip-172-16-1-183 ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 06:50:0a:0c:9a:ac brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.183/24 scope global noprefixroute dynamic eth0
        valid_lft 3144sec preferred_lft 3144sec
    inet6 fe80::450:abff:fe9c:9aac/64 scope link
        valid_lft forever preferred_lft forever
[ec2-user@ip-172-16-1-183 ~]$
```

If you cannot access the EC2 instance via SSH client, it may be because that you need to add a route entry.

**3. Install Strongswan.**

Run the following code to install Strongswan and verify the version of Strongswan:

```
$ wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/s/strongswan-5.7.1-1.el7.x86_64.rpm
$ sudo yum install gcc
$ sudo yum install trousers
$ sudo rpm -ivh strongswan-5.7.1-1.el7.x86_64.rpm
$ strongswan version
Linux strongSwan U5.7.1/K3.10.0-957.el7.x86_64
University of Applied Sciences Rapperswil, Switzerland
See 'strongswan --copyright' for copyright information.
```

**4. Configure Strongswan.**

Run the following code to configure Strongswan:

```
$ sudo vi /etc/strongswan/ipsec.conf
Paste following setting into file:
conn %default
    authby=psk
    type=tunnel
    keyexchange=ikev2
    auto=start
    ike=aes-sha1-modp1024
```

```

ikelifetime=86400s
esp=aes-sha1-modp1024
lifetime=86400s
conn abc_shanghai_oss
left=172.16.1.183 //// Local IP address of EC2(Strongswan installed
)
leftsubnet=172.16.1.0/24 //// AWS Tokyo VPC CIDR.
leftid=3.112.29.59 //// Public IP address as ID.
right=47.74.46.62 //// Public IP address of Alibaba Cloud VPN
Gateway
rightid=47.74.46.62 //// Public IP address as ID.
rightsubnet=100.118.102.0/24 //// Note(1)

```



#### Note:

When you ping the intranet endpoint of the OSS bucket in some ECS instances in the Alibaba Cloud Shanghai VPC, you can get the IP address which is belonged to this CIDR block. Therefore this CIDR block is used as the right subnet.

```

[root@EC-Demo-Shanghai ~]# ping -c 3 eric-oss-datastore-shanghai.oss-cn-shanghai-internal.aliyuncs.com
PING oss-cn-shanghai-internal.aliyuncs.com (100.118.102.33) 56(84) bytes of data.
64 bytes from 100.118.102.33 (100.118.102.33): icmp_seq=1 ttl=102 time=1.23 ms
64 bytes from 100.118.102.33 (100.118.102.33): icmp_seq=2 ttl=102 time=1.26 ms
64 bytes from 100.118.102.33 (100.118.102.33): icmp_seq=3 ttl=102 time=1.26 ms

--- oss-cn-shanghai-internal.aliyuncs.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.235/1.252/1.263/0.042 ms
[root@EC-Demo-Shanghai ~]# ping -c 3 eric-oss-datastore-shanghai.oss-cn-shanghai-internal.aliyuncs.com
PING oss-cn-shanghai-internal.aliyuncs.com (100.118.102.35) 56(84) bytes of data.
64 bytes from 100.118.102.35 (100.118.102.35): icmp_seq=1 ttl=102 time=1.08 ms
64 bytes from 100.118.102.35 (100.118.102.35): icmp_seq=2 ttl=102 time=1.12 ms
64 bytes from 100.118.102.35 (100.118.102.35): icmp_seq=3 ttl=102 time=1.12 ms

--- oss-cn-shanghai-internal.aliyuncs.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.089/1.114/1.128/0.042 ms
[root@EC-Demo-Shanghai ~]# █

```

## 5. Start Strongswan.

### a. Run the following commands:

```

$ sudo su - root
# echo 1 > /proc/sys/net/ipv4/ip_forward

```



#### Note:

Then you need to add `net.ipv4.ip_forward=1` into `/etc/sysctl.conf`.

### b. Run the following commands to start Strongswan:

```

# systemctl enable strongswan
# systemctl start strongswan
# systemctl status strongswan

```

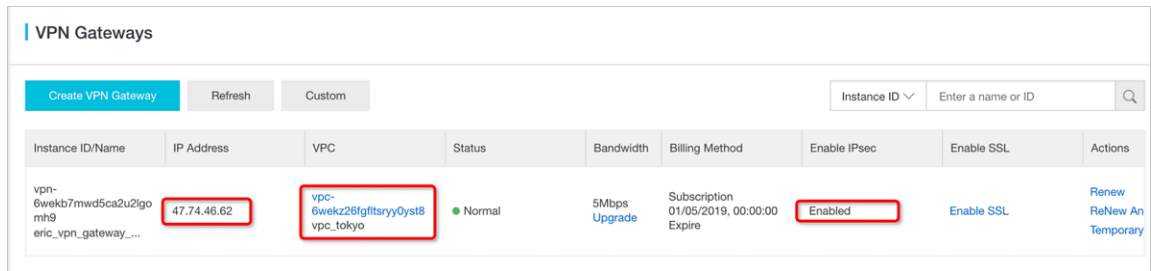
Step 2: Create VPN gateways and IPSec connections.

Follow these steps to create VPN gateways and IPSec connections.



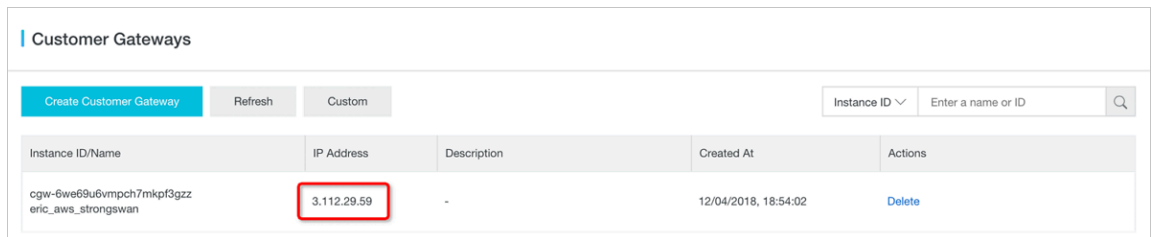
## 1. Create a VPN gateway and a customer gateway.

### a. Create a VPN gateway.



Instance ID/Name	IP Address	VPC	Status	Bandwidth	Billing Method	Enable IPsec	Enable SSL	Actions
vpn-6wekb7mwd5ca2u2lgo mh9 eric_vpn_gateway_...	47.74.46.62	vpc-6wekz26fgfltsryy0yst8 vpc_tokyo	Normal	5Mbps Upgrade	Subscription 01/05/2019, 00:00:00 Expire	Enabled	Enable SSL	Renew ReNew An Temporary

### b. Create a customer gateway.



Instance ID/Name	IP Address	Description	Created At	Actions
cgw-6we69u6vmpch7mkpf3gzz eric_aws_strongswan	3.112.29.59	-	12/04/2018, 18:54:02	Delete



#### Note:

In this practice, Strongswan@AWS EC2 is used as the customer gateway.

## 2. Create an IPsec connection.

### a. Make basic settings.

**Modify IPsec Connections** ⓘ ✕

• **Name** ⓘ

ipsec\_aws\_abc 13/128

• **VPN Gateway**

eric\_vpn\_gateway\_tokyo ▾

• **Customer Gateway**

eric\_aws\_strongswan ▾

• **Local Network** ⓘ

0.0.0.0/0

+ Add Local Network

• **Remote Network** ⓘ

172.16.1.0/24

+ Add Remote Network

**Effective Immediately** ⓘ

Yes  No

b. Make advanced settings.

### Modify IPsec Connections

**Advanced Configuration**

IKE Configurations

**Pre-Shared Key** ?

3k3jM

**Version**

ikev2

**Negotiation Mode** ?

main

**Encryption Algorithm**

aes

**Authentication Algorithm**

sha1

**DH Group**

group2

### Modify IPsec Connections

**SA Life Cycle (seconds)**  
86400

**LocalId** ?  
47.74.46.62

**Remoteld** ?  
3.112.29.59

**IPsec Configurations**

**Encryption Algorithm**  
aes

**Authentication Algorithm**  
sha1

**DH Group**  
group2

**SA Life Cycle (seconds)**  
86400

3. Check the IPsec connection status.

a. Check the IPsec connection status on the Alibaba Cloud console.

Instance ID/Name	VPN Gateway	Customer Gateway	Connection Status	Created At	Actions
vco-6weluco95888x12nnf6l ipsec_aws_abc	vpn-6wekb7mwd5ca2u2lgomh9 eric_vpn_gateway_tokyo	cgw-6we69u6vmpch7mkpf3gzz eric_aws_strongswan	● Phase 2 of IKE Tunnel Negotiation Succeeded	12/04/2018, 18:55:47	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Download Remote Configuration</a> <a href="#">View Logs</a>

b. Check the IPsec connection status on Strongswan by running the following command:

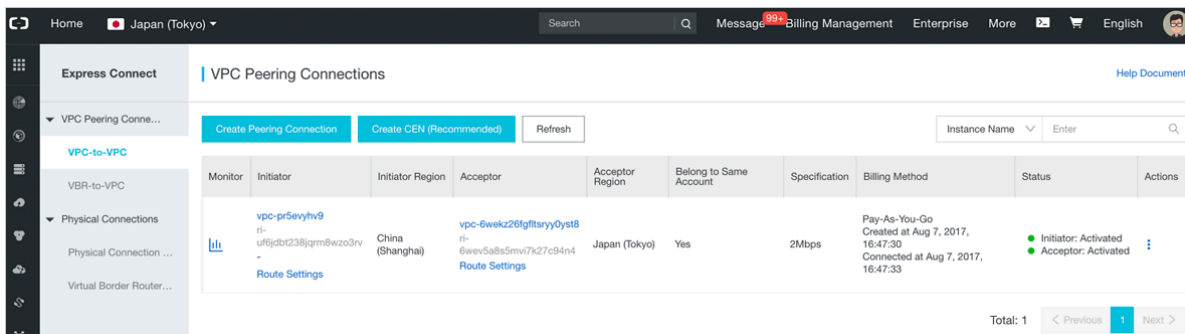
```
# systemctl status strongswan
```

```
[root@ip-172-16-1-183 ossimport]# systemctl status strongswan
● strongswan.service - strongswan IPsec IKEv1/IKEv2 daemon using ipsec.conf
   Loaded: loaded (/usr/lib/systemd/system/strongswan.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2018-12-07 07:55:07 UTC; 9s ago
     Main PID: 22547 (starter)
    CGroup: /system.slice/strongswan.service
            22547 /usr/libexec/strongswan/starter --daemon charon --nofork
            22557 /usr/libexec/strongswan/charon

Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] authentication of '47.74.46.62' with pre-shared key successful
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] IKE_SA abc_shanghai_oss[1] established between 172.16.1.183[3.112.29.59]...47.74.46.62[47.74.46.62]
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] IKE_SA abc_shanghai_oss[1] established between 172.16.1.183[3.112.29.59]...47.74.46.62[47.74.46.62]
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] scheduling reauthentication in 85466s
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] maximum IKE_SA lifetime 86006s
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[CFG] selected proposal: ESP:AES_CBC_128/HMAC_SHA1_96/NO_EXT_SEQ
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] CHILD_SA abc_shanghai_oss[1] established with SPIs cdf3c91e_i c5fcc5f4_o and TS 172.16.1.0/24 === 100.118.102.0/24
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] CHILD_SA abc_shanghai_oss[1] established with SPIs cdf3c91e_i c5fcc5f4_o and TS 172.16.1.0/24 === 100.118.102.0/24
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] received AUTH_LIFETIME of 85603s, scheduling reauthentication in 85603s
Dec 07 07:55:08 ip-172-16-1-183.ap-northeast-1.compute.internal charon[22557]: 10[IKE] peer supports MOBIKE
[root@ip-172-16-1-183 ossimport]#
```

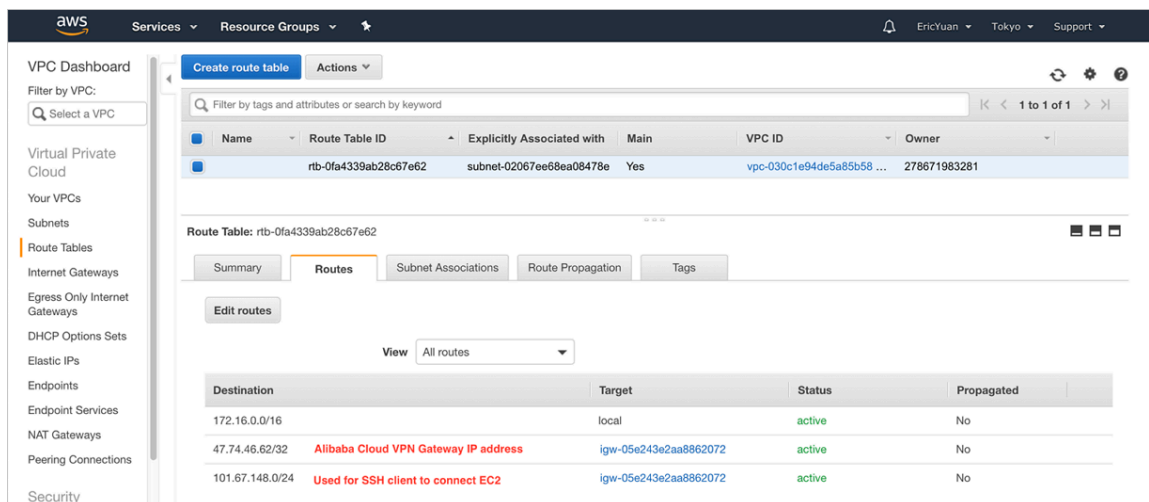
Step 3: Create VPC Peering Connection.

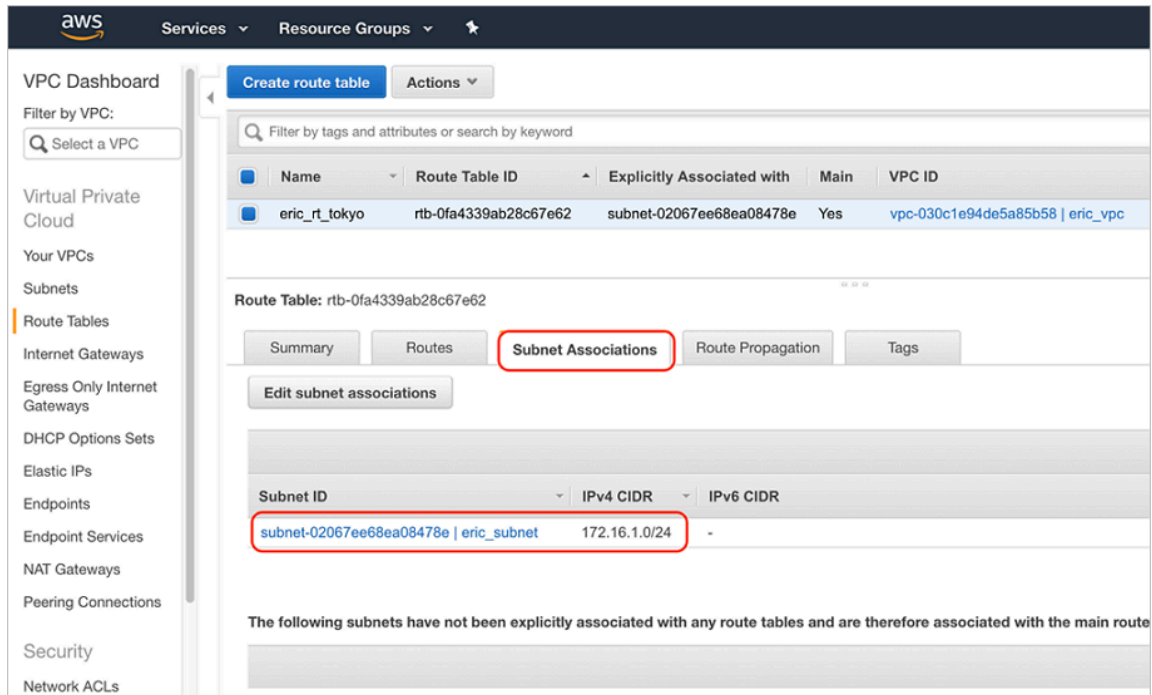
- 1. Follow the steps in *Interconnect two VPCs under the same account* to create a VPC peering connection on the Alibaba Cloud Express Connect console.



- 2. Add the following routing entries in the route settings both on the AWS VPC and Alibaba Cloud VPC.

- AWS VPC in Tokyo Japan





• **Alibaba Cloud VPC in Tokyo Japan**

Basic Information

Route Table ID `vtb-6weu938phhidvxesg4n6m` Router ID `vrt-6wedstuws7iq50c636ycd`  
 Created At Nov 8, 2016, 17:03:12

[Add Route](#) [Refresh](#)

Route Table ID	Destination Subnet	Status	Next Hop Instance	Next Hop Type	Route Type
vtb-6weu938phhidvxesg4n6m	100.118.102.0/24	Available	vpc-pr5evyhv9	VPC	Custom
vtb-6weu938phhidvxesg4n6m	172.16.1.0/24	Available	vpn-6wekb7mwd5ca2u2lgomh9	-	Custom
vtb-6weu938phhidvxesg4n6m	172.19.48.0/20	Available	vpc-pr5evyhv9	VPC	Custom
vtb-6weu938phhidvxesg4n6m	172.24.0.0/20	Available	-	-	System-Defined
vtb-6weu938phhidvxesg4n6m	100.64.0.0/10	Available	-	-	System-Defined

The subnets in the preceding figure are described as follows:

- 100.118.102.0/24: VPC endpoint of OSS bucket in the China Shanghai region
  - 172.16.1.0/24: AWS VPC in Tokyo Japan
  - 172.19.48.0/20: Alibaba Cloud VPC in Shanghai China
- **Alibaba Cloud VPC in Shanghai China**

**Basic Information**

Route Table ID `vtb-ii9hhd1d5` Router ID `vrt-3i9xt5789`  
 Created At `Aug 31, 2016, 16:35:21`

[Add Route](#) [Refresh](#)

Route Table ID	Destination Subnet	Status	Next Hop Instance	Next Hop Type	Route Type
vtb-ii9hhd1d5	172.16.1.0/24	Available	vpc-6wekz26fgfltsryy0yst8	VPC	Custom
vtb-ii9hhd1d5	192.168.1.0/24	Available	vpc-251e3e6m4	VPC	Custom
vtb-ii9hhd1d5	172.24.0.0/20	Available	vpc-6wekz26fgfltsryy0yst8	VPC	Custom
vtb-ii9hhd1d5	172.19.18.0/24	Available	-	-	System-Defined
vtb-ii9hhd1d5	172.19.0.0/20	Available	-	-	System-Defined
vtb-ii9hhd1d5	172.19.224.0/20	Available	-	-	System-Defined
vtb-ii9hhd1d5	172.19.48.0/20	Available	-	-	System-Defined
vtb-ii9hhd1d5	100.64.0.0/10	Available	-	-	System-Defined

The subnets in the preceding figure are described as follows:

- 172.16.1.0/24: AWS VPC in Tokyo Japan
- 172.24.0.0/20: Alibaba Cloud VPC in Tokyo Japan

### 3. Test the connection between AWS VPC in Tokyo Japan and the Alibaba Cloud OSS bucket in Shanghai.

```
[ec2-user@ip-172-16-1-183 ~]$
[ec2-user@ip-172-16-1-183 ~]$ ping -c 5 eric-oss-datastore-shanghai.oss-cn-shanghai-internal.aliyuncs.com
PING oss-cn-shanghai-internal.aliyuncs.com (100.118.102.36) 56(84) bytes of data:
64 bytes from 100.118.102.36 (100.118.102.36): icmp_seq=1 ttl=101 time=34.2 ms
64 bytes from 100.118.102.36 (100.118.102.36): icmp_seq=2 ttl=101 time=34.2 ms
64 bytes from 100.118.102.36 (100.118.102.36): icmp_seq=3 ttl=101 time=34.1 ms
64 bytes from 100.118.102.36 (100.118.102.36): icmp_seq=4 ttl=101 time=34.9 ms
64 bytes from 100.118.102.36 (100.118.102.36): icmp_seq=5 ttl=101 time=34.2 ms

--- oss-cn-shanghai-internal.aliyuncs.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 34.155/34.369/34.902/0.357 ms
[ec2-user@ip-172-16-1-183 ~]$
```

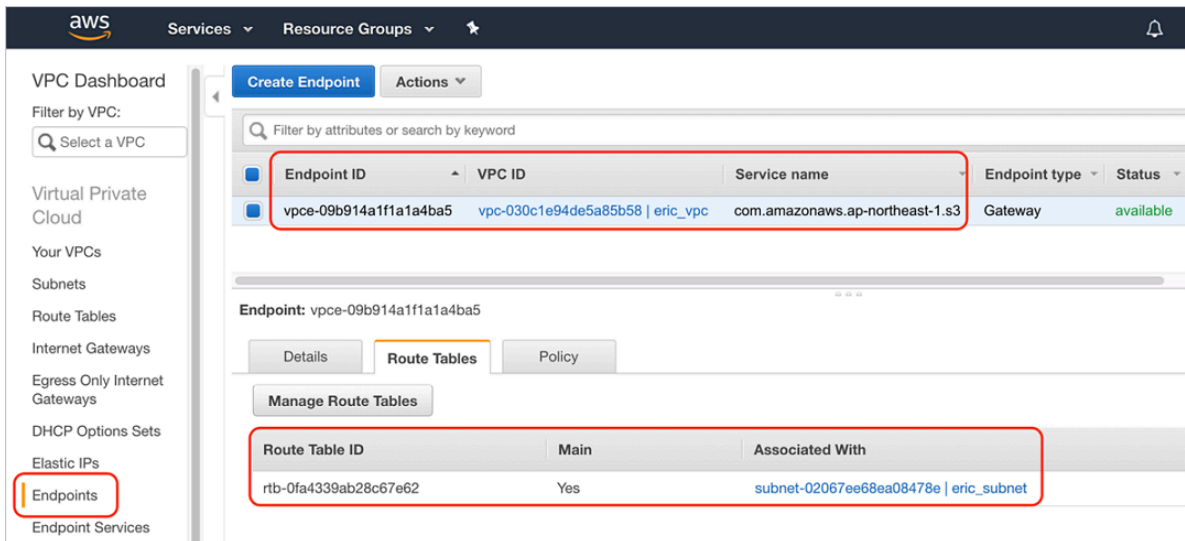
The test results in the preceding figure show that the connection between the AWS EC2 instance and the Alibaba Cloud OSS bucket is connected. You can deploy OssImport and migrate objects from the AWS S3 bucket to the Alibaba Cloud OSS bucket.

#### Step 4: Create a VPC endpoint for AWS S3 bucket.

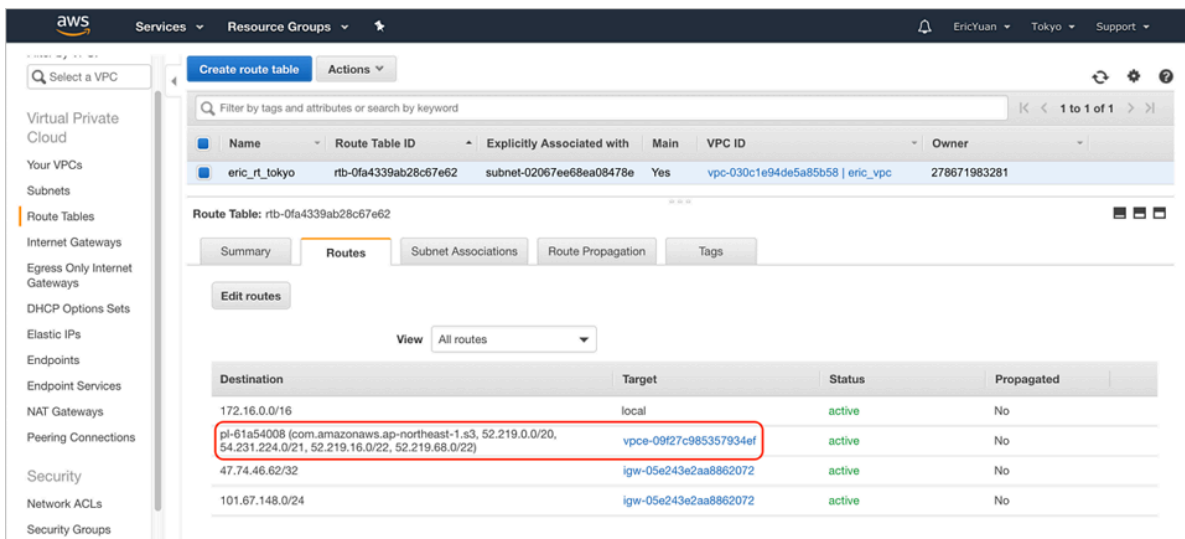
In this practice, OssImport deployed on the EC2 instance in VPC is used to move data from the AWS S3 bucket to the OSS bucket. We recommend you create a VPC endpoint

for the AWS S3 bucket and associate it with the route table so that OssImport can visit the AWS S3 bucket within the VPC instead of visiting the Internet IP address of the AWS S3 bucket.

1. Create a VPC endpoint for the AWS S3 bucket.



2. Check the VPC route table.



3. Verify the connection between the AWS S3 VPC endpoint and the EC2 instance.

```
[root@ip-172-16-1-183 ossimport]#
[root@ip-172-16-1-183 ossimport]# ping -c 3 eric-s3-tokyo.s3.ap-northeast-1.amazonaws.com
PING s3.ap-northeast-1.amazonaws.com (52.219.68.24) 56(84) bytes of data:
64 bytes from s3-ap-northeast-1.amazonaws.com (52.219.68.24): icmp_seq=1 ttl=60 time=0.185 ms
64 bytes from s3-ap-northeast-1.amazonaws.com (52.219.68.24): icmp_seq=2 ttl=60 time=0.216 ms
64 bytes from s3-ap-northeast-1.amazonaws.com (52.219.68.24): icmp_seq=3 ttl=60 time=0.253 ms

--- s3.ap-northeast-1.amazonaws.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.185/0.218/0.253/0.027 ms
[root@ip-172-16-1-183 ossimport]# a
```



Step 5: Migrate Data from the AWS S3 bucket to the OSS bucket by using OssImport.

Follow the steps in [Use OssImport to migrate data](#) to deploy and configure OssImport.

1. Configure the `local_job.cfg` file as follows:

```
srcType=s3
srcAccessKey=AK*****A
srcSecretKey=+RW*****iM3
srcDomain=s3.ap-northeast-1.amazonaws.com
srcBucket=eric-s3-tokyo
srcPrefix=source_folder/
destAccessKey=L*****ic
destSecretKey=nEtM*****iDx
destDomain=http://oss-cn-shanghai-internal.aliyuncs.com
destBucket=eric-oss-datastore-shanghai
destPrefix=destination_folder/
```

2. Execute the `import.sh` script and check the migration status as follows:

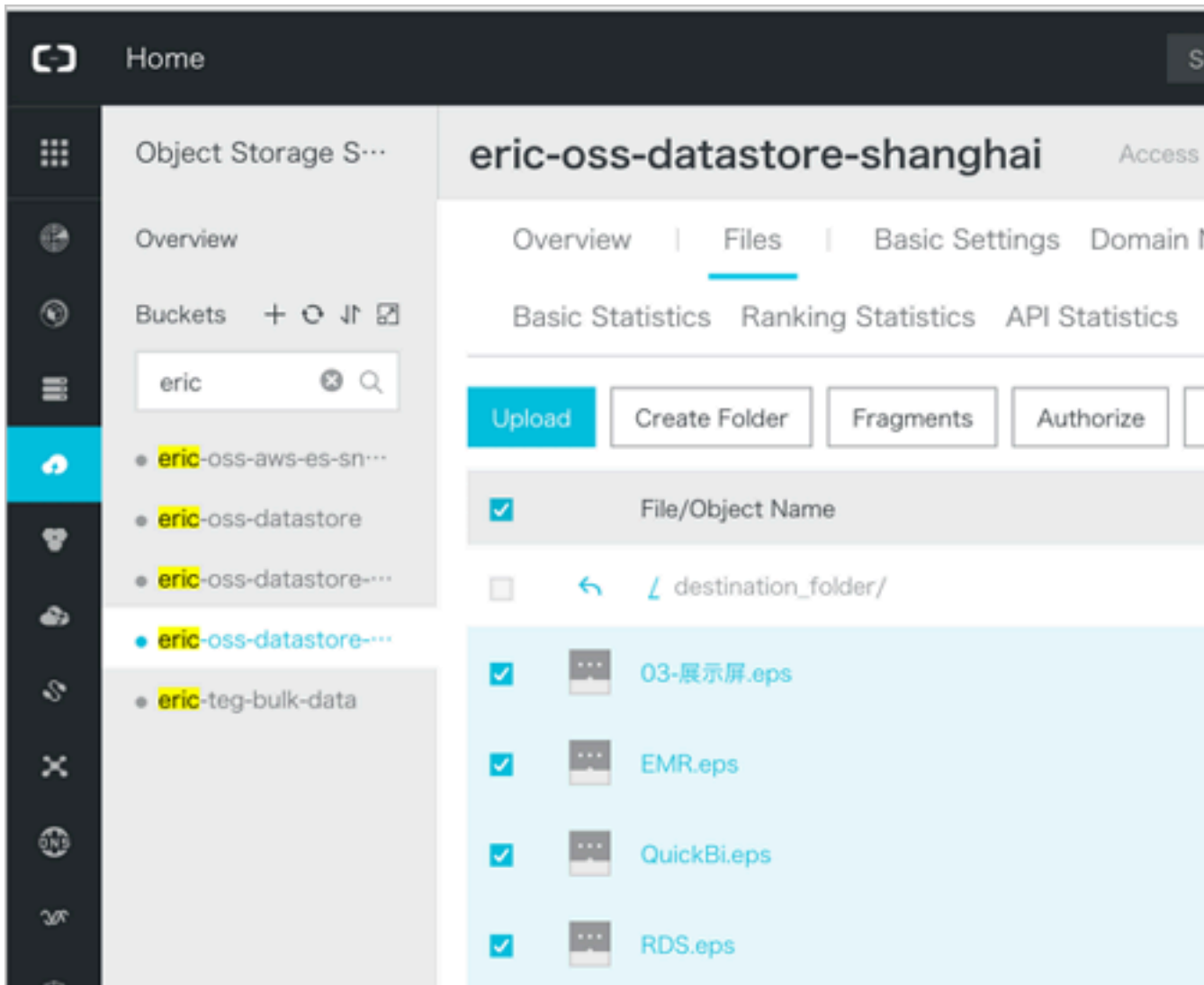
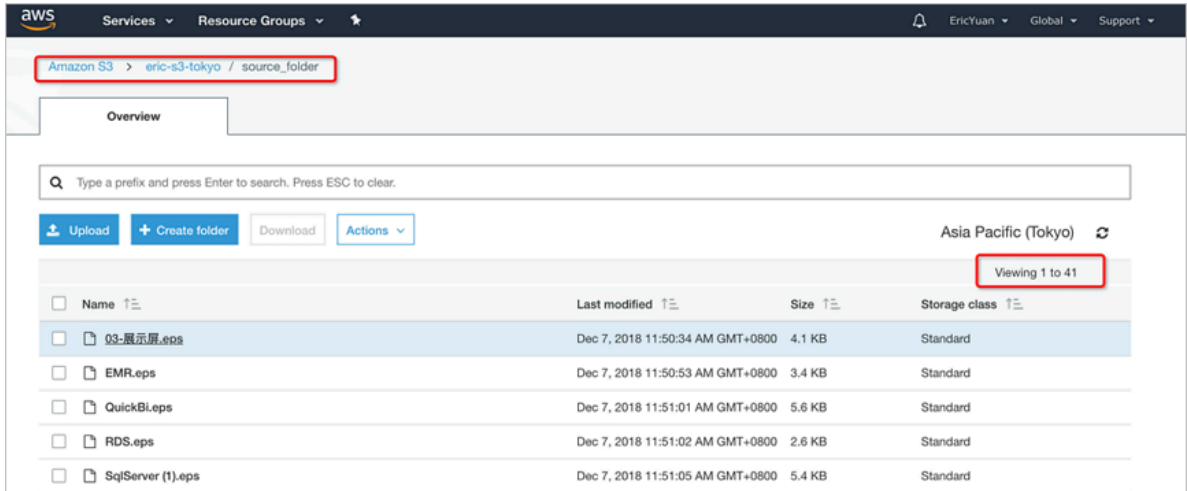
```
[root@ip-172-16-1-183 ~]# cd /home/ec2-user/ossimport
[root@ip-172-16-1-183 ossimport]# ./import.sh
Clean the previous job, Yes or No: yes
Stop import service completed.
delete jobs dir:./master/jobs/local_test/
Clean job:local_test completed.
submit job:/home/ec2-user/ossimport/conf/local_job.cfg
submit job:local_test success!
Start import service completed.
.....
----- job stats -----
----- job stat -----
JobName:local_test
JobState:Running
PendingTasks:0
DispatchedTasks:1
RunningTasks:1
SucceedTasks:0
FailedTasks:0
ScanFinished:true
RunningTasks Progress:
8528637A126676A4FD0D2F981ED5E0EF_1544176618182:0/191048 1/42
-----
----- job stats -----
----- job stat -----
JobName:local_test
JobState:Running
PendingTasks:0
DispatchedTasks:1
RunningTasks:1
SucceedTasks:0
FailedTasks:0
ScanFinished:true
RunningTasks Progress:
8528637A126676A4FD0D2F981ED5E0EF_1544176618182:191048/191048 42/42
[root@ip-172-16-1-183 ossimport]#
```



Note:

You can find detail logs in the `ossimport/logs` folder.

### 3. Compare the files in the AWS S3 bucket and those in the OSS bucket.



As shown in the preceding figure, if the files in the AWS S3 bucket and those in the OSS bucket are the same, the migration is successful.

If the migration fails at any preceding steps, please contact technical support by [opening a ticket](#).

## References

- [Data migration](#)
- [Migration Technical Guide](#)

## 2 Direct upload to OSS from Web

---

### 2.1 Direct transfer after adding a signature on the server

#### Background

Direct signature by JS clients has a serious hidden security hazard in that the OSS AccessId/AccessKey are exposed on the frontend which may be accessible to others. This document explains how to get a signature from and upload a policy to the backend PHP code.

The logic for uploading a signature to the backend is as follows:

1. Before uploading an image, the client obtains the uploaded policy and signature from the application server.
2. The client directly uploads the obtained signature to the OSS.

#### Signature sample uploaded to the backend

- Download sample:
  - Click [here](#) to download a test sample on a PC browser.
  - You can test whether the upload was effective on a mobile phone. You can use a mobile phone app (such as WeChat, QQ, and mobile browsers) to scan the QR code.

This is not an advertisement, but a QR code for the preceding URL. This operation allows you to see whether the service works as intended on mobile phones.

- Download code:

Click [here](#) to download the code.

This example adopts the backend signature, and uses PHP language.

- Click [here](#) for the example of a backend signature using Java language.
- Click [here](#) for the example of a backend signature using Python language.
- Click [here](#) for the example of a backend signature using Go language.

Usage of other languages:

1. Download the corresponding language example.

2. Modify the example code. For instance, set the listening port, and then start running.
3. At `upload.js` in `oss-h5-upload-js-php.zip`, change the variable `serverUrl` to the address configured at step 2. For example, `serverUrl=http://1.2.3.4:8080` or `serverUrl=http://abc.com/post/`.

### Principle of constructing a Post signature on the server end

The OSS PostObject method is used for uploads. You can construct a PostObject request in the browser using Plupload and send the request to the OSS. Signatures are implemented on the server in PHP. In the same principle, the server can be compiled in Java, .NET, Ruby, Go, or Python language. The core logic is to construct a Post signature. The Java and PHP examples are provided here. The following steps are required:

1. The webpage requests the signature through JavaScript from the server end.
2. After JavaScript gets the signature, it uploads the signature to the OSS through Plupload.

#### · Implementation

1. Populate the fields with your ID, key, and bucket.

Modify `php/get.php`:

- Set the variable `$id` to `AccessKeyId`.
- Set `$key` to `AccessKeySecret`.
- Set `$host` to `bucket+endpoint`.



#### Note:

For information on the endpoint, see [Basic OSS concepts](#).

```
$id= 'xxxxxx';  
$key= 'xxxxx';  
$host = 'http://post-test.oss-cn-hangzhou.aliyuncs.com'
```

2. You must set CORS for the bucket to guarantee browser safety.



#### Note:

Make sure that the CORS settings of the bucket attribute support the POST method. This is because, HTML directly uploads data to OSS and produces

a cross-origin request in the process. Hence, you must allow cross-original requests in the bucket attributes.

For procedure, see [Set CORS](#). The settings are as follows:



Note:

In earlier-version IE browsers, Plupload is executed in flash. You must set `crossdomain.xml`.

#### Details of core logic

- Set random object names

You often need to name uploaded objects randomly, if they have the same suffix as the objects on the client. In this example, two radios are used to differentiate. If you want to fix the settings to apply random names to the uploaded objects, you can change the function to the following:

```
function check_object_radio() {
    g_object_name_type = 'random_name';
}
```

If you want to set uploads to the user's objects, you can change the function to the following:

```
function check_object_radio() {
    g_object_name_type = 'local_name';
}
```

- Set the upload directory

The upload directory is specified by the server end (in PHP), which enhances security. Each client is only allowed to upload objects to a specific directory. This guarantees security by isolation. The following code changes the upload directory address to `abc/` (the address must end with `/`).

```
$dir = 'abc/';
```

- Set the filtering conditions for uploaded objects

We often need to set the filtering conditions for uploads. For example, only allowing image uploads, setting the size of uploaded objects, and disallowing repeated uploads. You can use the `filters` parameter for this.

```
var uploader = new plupload.Uploader({
    .....
    filters: {
```

```

        mime_types : [ //Only images and zip objects are allowed to
be uploaded
        { title : "Image files", extensions : "jpg,gif,png,bmp" },
        ],
        max_file_size : '400kb', //Only objects with a maximum size
of 400 KB are allowed to be uploaded
        prevent_duplicates : true //Repeated objects are not allowed
to be selected
    },

```

Use the Plupload attribute filters to set filtering conditions.

Explanations of the preceding setting values:

- **mime\_types**: Restrict extensions of the uploaded objects.
- **max\_file\_size**: Restrict the size of the uploaded objects.
- **prevent\_duplicates**: Restrict repeated uploads.



**Note:**

The filter conditions are not required. You can comment out the filtering condition, if you do not need it.

- Get uploaded object names

If you want to know the name of the uploaded object, you can use Plupload to call the FileUploaded event, as follows:

```

FileUploaded: function(up, file, info) {
    if (info.status == 200)
    {
        document.getElementById(file.id).getElement
sByTagName('b')[0].innerHTML = 'upload to oss success, object name:'
+ get_uploaded_object_name(file.name);
    }
    else
    {
        document.getElementById(file.id).getElement
sByTagName('b')[0].innerHTML = info.response;
    }
}

```

You can use the following functions to get the names of the objects uploaded to OSS . The file.name property records the names of the uploaded local objects.

```
get_uploaded_object_name(file.name)
```

- Upload signatures

JavaScript can get the policyBase64, accessid, and signature variables from the backend. The following is the core code for getting the three variables:

```
phpUrl = './php/get.php'
```

```
xmlhttp.open( "GET", phpUrl, false );
xmlhttp.send( null );
var obj = eval ("(" + xmlhttp.responseText+ ")");
host = obj['host']
policyBase64 = obj['policy']
accessid = obj['accessid']
signature = obj['signature']
expire = parseInt(obj['expire'])
key = obj['dir']
```

Parse `xmlhttp.responseText` (the following only serves as an example. The actual format may vary, but the values of `signature`, `accessid`, and `policy` must exist).

```
{"accessid":"6MK0qxGiGU4AUk44",
"host":"http://post-test.oss-cn-hangzhou.aliyuncs.com",
"policy":"eyJleHBpcmF0aW9uIjoimjAxNS0xMS0wNVQyMDoyMzoyMloiLC
Jjxb25kaXRpb25zIjpbWyJjcb250ZW50LWxlbmd0aC1yYW5nZSIsMCwxMDQ4
NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwjJGtleSIsInVzZXItZGlyXC8iXV19",
"signature":"I2u57FWjTKqX/AE6doIdyff151E=",
"expire":1446726203,"dir":"user-dir/"}
```

- **accessid:** It is the Accessid of the user request. However, disclosing Accessid does not impact data security.
- **host:** The domain name to which the user wants to send an upload request.
- **policy:** A policy for uploading user forms. It is a Base64-encoded string.
- **signature:** A signature string for the policy variable.
- **expire:** It is the expiration time of the current upload policy. This variable is not sent to OSS, because it is already indicated in the policy.

Parse `policy`. The decoded content of the policy is as follows:

```
{"expiration":"2015-11-05T20:23:23Z",
"conditions":["content-length-range",0,1048576000],
["starts-with","$key","user-dir/"]}
```

For more information about Policy, see [Policy basic elements](#).

The key content of the `PolicyText` specifies the final expiration time of this policy. Before its expiry, this policy may be used to upload objects. Therefore, it is not necessary to obtain a signature from the backend for each upload.

Here, we use the following designs:

- For initial uploads, a signature is obtained for each object upload.
- For subsequent uploads, the current time is compared with the signature time to see whether the signature has expired.

■ If the signature expires, a new signature is obtained.



- If the signature has not expired, the same signature is used. The expired variable is used here.

The core code is as follows:

```
now = timestamp = Date.parse(new Date()) / 1000;
[color=#000000]//This determines whether the time specified by the
  expire variable is earlier than the current time. If so, a new
  signature is obtained. 3s is the buffer duration.[/color]
  if (expire < now + 3)
  {
    .....
    phpUrl = './php/get.php'
    xmlhttp.open( "GET", phpUrl, false );
    xmlhttp.send( null );
    .....
  }
return .
```

We see that starts-with has been added to the policy content. This indicates the name of the object to be uploaded must start with the user-dir (this string can be customized).

This setting is added because, in many scenarios, one bucket is used for one app and contains the data of different users. To prevent the data from being overwritten, a specific prefix is added to the objects uploaded by a specific user to OSS.

However, an issue occurs. Once the users obtains this policy, they can modify the upload prefix before the expiration time to upload objects to another user's directory. To resolve this issue, you can set the application server to specify the prefix of the uploaded objects by a specific user at the time of upload. In this case, no one can upload objects with another user's prefix even after obtaining the policy. This guarantees data security.

## Summary

In the sample mentioned in this document, the webpage end requests the signature from the server end during uploads from the webpage end, and then objects are uploaded directly, with no pressure on the server end. This approach is safe and reliable.

However in this sample, the backend program is not immediately aware of the number or identity of objects uploaded. You can use upload callback to see which objects were uploaded. This sample cannot implement multipart and breakpoint.

## Related Documents

- [\*Basic concepts\*](#)
- [\*Set Cross-Origin Resource Sharing \(CORS\)\*](#)
- [\*Overview of direct transfer on Web client\*](#)
- [\*Directly add a signature on the server, transfer the file, and set upload callback\*](#)
- [\*Set up direct data transfer for mobile apps\*](#)

## 3 Application server

---

### 3.1 Permission control

This document elaborates how to configure different policies to implement different permission controls based on the app server mentioned in [Set up direct data transfer for mobile apps](#) by taking the app-base-oss bucket in the Shanghai region as an example.



#### Note:

- The following illustration assumes you have already activated STS and have thoroughly read the [Set up direct data transfer for mobile apps](#) document.
- The policies mentioned in the following content are covered in the specified policy file in the config.json file mentioned in the previous section.
- The operations on OSS upon retrieving the STS token indicate the process of specifying the policy for the app server, the app server retrieving a temporary credential from the STS and the app using the temporary credential to access OSS.

#### Common policies

- Full authorization policy

For the ease of demonstration, the default policy is shown as follows. This policy indicates that the app is allowed to perform any operation on OSS.



#### Note:

This policy is neither secured nor recommended to use for mobile apps.

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

```

}
    
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Successful
Upload the object without a prefix, test.txt.	Successful
Download the object without a prefix, test.txt.	Successful
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Read-only policies with or without any prefixes

This policy indicates the app can list and download all objects in the bucket app-base-oss.

```

{
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:ListObjects"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app-base-oss/*", "acs:oss:*:*:app-base-oss"]
    }
  ],
  "Version": "1"
}
    
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Successful

Operations on OSS upon retrieving STS token	Result
Upload the object with a prefix, user1/test.txt.	Failed
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Read-only policies with a specified prefix

This policy indicates the app can list and download all objects with the prefix of **\*\*user1/\*\*** in the bucket **\*\*app-base-oss\*\***. However, the policy does not specify to download any objects with another prefix. By this way, different apps corresponding to different prefixes are spatially isolated in the bucket.

```
{
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:ListObjects"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app-base-oss/user1/*", "acs:oss:*:*:app-base-oss"]
    }
  ],
  "Version": "1"
}
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Failed
Download the object with a prefix, user1/test.txt.	Successful

Operations on OSS upon retrieving STS token	Result
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Write-only policies with no specified prefixes

This policy indicates that the app can upload all objects in the bucket app-base-oss.

```
{
  "Statement": [
    {
      "Action": [
        "oss:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app-base-oss/*", "acs:oss:*:*:app-base-oss"]
    }
  ],
  "Version": "1"
}
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Successful
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Write-only policies with a specified prefix

This policy indicates the app can upload all objects with the user1/ prefix in the bucket app-base-oss. The app cannot upload any object with another prefix. In this

way, different apps corresponding to different prefixes are spatially isolated in the bucket.

```
{
  "Statement": [
    {
      "Action": [
        "oss:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app-base-oss/user1/*", "acs:oss:*:*:app-base-oss"]
    }
  ],
  "Version": "1"
}
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Failed
List the object without a prefix, test.txt.	Failed
List the object with a prefix, user1/test.txt.	Failed

- Read/write policies with or without any prefixes

This policy indicates that the app can list, download, upload, and delete all objects in the bucket `app-base-oss`.

```
{
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:ListParts",
        "oss:AbortMultipartUpload",
        "oss:ListObjects"
      ],
      "Effect": "Allow",
    }
  ]
}
```

```

    "Resource": ["acs:oss:*:*:app-base-oss/*", "acs:oss:*:*:app-
base-oss"]
  }
],
  "Version": "1"
}

```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Successful
Download the object without a prefix, test.txt.	Successful
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Read/write policies with a specified prefix

This policy indicates the app can list, download, upload, and delete all objects with a prefix of `user1/` in the bucket `app-base-oss`. The policy does not specify to read or write any objects with another prefix. In this way, different apps corresponding to different prefixes are spatially isolated in the bucket.

```

{
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:ListParts",
        "oss:AbortMultipartUpload",
        "oss:ListObjects"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app-base-oss/user1/*", "acs:oss
:*:*:app-base-oss"]
    }
  ],
  "Version": "1"
}

```



```

}

```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

## Summary

With the help of preceding examples, we can understand that:

- You can create different policies for various app scenarios and then achieve differentiated permission control for different apps through slight modifications on the app server.
- You can also optimize apps to save the process of making another request to the app server before the STS token expires.
- Tokens are actually issued by the STS. An app server customizes a policy, requests for a token from the STS, and then delivers this token to the app. Here, token is only a shorthand expression. However, a "token" actually contains an "AccessKeyId", an "AccessKeySecret", an "Expiration" value, and a "SecurityToken". These are used in the SDK provided by OSS to the app. For more information, see the implementation of the respective SDK.

More references:

- [How to use RAM and STS in OSS](#)
- [RAM documentation](#) and [STS documentation](#)

## 4 Data backup and recovery

---

### 4.1 Back up buckets

Alibaba Cloud offers multiple backup methods for data on OSS to suit different scenarios.

The following methods can be used to back up OSS data on the cloud:

- Cross-region replication (set on the console or using APIs or SDK code)
- OssImport tool

Back up data using cross-region replication

- Applicable scenarios

See [Cross-Region replication development guide](#).

- Operation on the console

See [Cross-Region replication operation guide](#).

- FAQ

See [How to synchronize data to OSS](#).



Note:

- The source bucket and target bucket belong to the same user but different regions.
- The source bucket and target bucket do not use archive storage.
- Data synchronization between buckets in the same region can be implemented using OSS SDK/API code.

Back up data using the OssImport tool

The OssImport tool can migrate data stored on local hosts or other cloud storage systems to OSS. It has the following features:

- Supports a vast variety of data sources, including local drives, Qiniu Cloud, Baidu BOS, AWS S3, Azure Blob, Blob, but also cloud, Tencent cloud cos, Golden Mountain ks3, HTTP, OSS, and so on, and can be expanded as needed.
- Supports resumable upload.
- Supports throttling.

- Supports migration of objects generated after a specified time or with a specified prefix.
- Supports parallel data upload and download.
- Supports the standalone and distributed modes. The standalone mode is easy to deploy and use, while the distributed mode is suitable for large-scale data migration.

#### Applicable scenarios

See [Data migration](#).

#### Installation and deployment

See [Architecture and configuration](#), [Standalone deployment](#) , and [Distributed deployment](#).

#### FAQ

See [FAQ](#).

#### NOTE

- If data needs to be migrated between buckets of different user accounts and the data volume exceeds 10 TB, the distributed version is recommended.
- When using the incremental mode to synchronize object changes between OSS buckets, note that OssImport can synchronize only modification operations (put /append/multipart) and cannot synchronize read or delete operations. No SLA guarantee is provided for timely data synchronization in this mode. Therefore, use the incremental mode with caution.
- Cross-region replication is recommended for data synchronization between different regions, if cross-region replication is enabled in these regions.

# 5 Bucket management

---

## 5.1 Storage class conversion

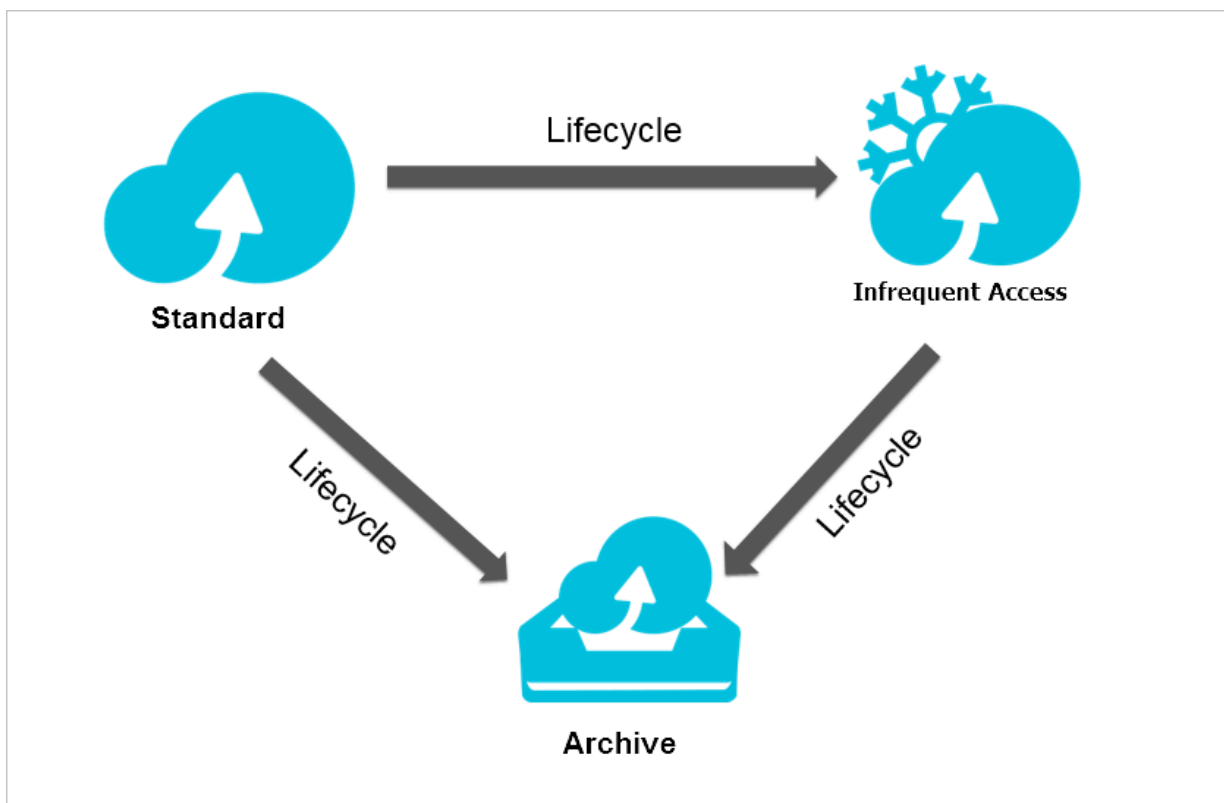
This topic describes how to convert the storage class of an object between Standard, IA, and Archive.

### Lifecycle Object Transition

OSS supports three *storage classes*: Standard, Infrequent Access, and Archive.

The Object Transition mechanism is now available in OSS Lifecycle Management function in all regions across China. The following storage classes are supported for automatic conversion:

- Standard -> Infrequent Access
- Standard -> Archive
- Infrequent Access -> Archive

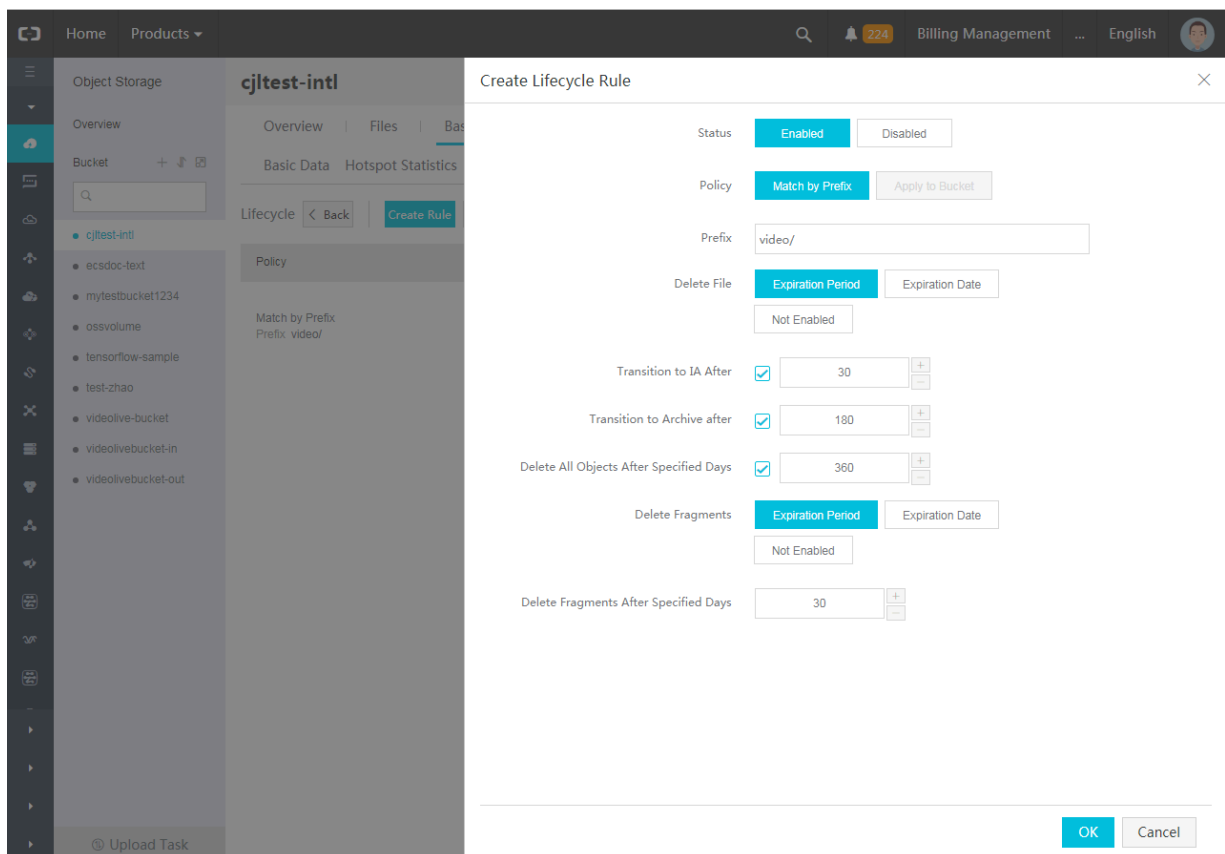


### Examples

You can configure lifecycle policies for objects with a given prefix in one bucket as follows:

- They are converted to Infrequent Access class after being stored for 30 days.
- They are converted to Archive class after being stored for 180 days.
- They are deleted automatically after being stored for 360 days.

You can complete the configuration of the preceding lifecycle policies in the console. For more information, see [Set lifecycle](#).



### Note:

If the following three parameters are configured:

Transition to IA After, Transition to Archive After, and Delete All Objects After Specified Days, then the number of days set for each parameter must meet the following criteria:

Days for converting to Infrequent Access < Days for converting to Archive < Specified days for deleting

### Notes

After the Object type conversion, the storage cost is calculated based on the unit price of converted storage class.

Notes for Infrequent Access and Archive storage types:

- **Minimum billable size:**

Objects smaller than 128 KB are charged as 128 KB.

- **Minimum storage period:**

The stored data is required to be saved for at least 30 days. Charges will be incurred if you delete files that are stored for less than 30 days.

- **Restore time of Archive type:**

It takes one minute for Archive type Object to restore the data to a readable state

. If real-time read is required in the business scenario, we recommend that you convert the file to the Infrequent Access storage class instead of Archive class.

Otherwise, after converting the file to the Archive class, the data cannot be read in real time.

- **Data access charges:**

Both Infrequent Access and Archive classes are required to pay data access charges as a separate charge item to outbound traffic. If the average access frequency per Object is higher than once per month, you are not advised to convert the data to Infrequent Access or Archive class.

Storage classes conversion in other ways

For conversions from Archive type to Standard class or Infrequent Access class, or from Infrequent Access class to Standard class, you can read the Object and rewrite it to the Bucket of corresponding storage class. The default storage class of Object is determined by the Bucket.

For example, for the conversion of Infrequent Access Object in the Bucket of Standard type to Standard Object, you can read and rewrite the Object. Based on the type of the Bucket, the newly-written Object is of Standard storage class.

For the Object that has been converted to Archive class, you can only read it after performing Restore operation and restore it to a readable state.

For more information, see [Create and use the Archive bucket](#).

## 5.2 Anti-leech

### Background

For example, A is the webmaster of a website. Webpages on the website contain links to images and audio/video files. These static resources are stored on *Alibaba Cloud OSS*. For example, A may save an image file on OSS with the URL `http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png`.

For OSS external resource url, see *OSS address* such a URL (without signing) requires the user's bucket permission to read publicly.

B is the webmaster of another website, B use the image resources of the website without permission, use this method to steal space and traffic by placing it in a web page on your website. In this case, the third-party web site user sees the B web site, but it's not clear the source of the pictures on the website. Since OSS charges by usage, so that user A does not get any benefit, instead, the cost of resource use is borne.

This article applies to users who use OSS resources as outer chains in a Web page, it also introduces a-like users who have stored their resources on OSS, how to avoid the use of unnecessary resources by setting up anti-theft chains.

### Implementation method

At present, the methods of anti-theft chain provided by OSS mainly include the following two types:

- Set Referer : The operation is available through the console and the SDK, and the user can choose according to their needs.
- Use signature URL: This is suitable for users who are used to developing.

The following two examples are provided in this article:

- Set the Referer anti-theft chain through the console
- Dynamic generation of signed URL anti-theft chains based on PHP SDK

### Set Referer

This section focuses on what Referer is and how OSS uses Referer for anti-theft chains

- 
- What is Referer?

Referer is HTTP Part of the header that usually comes with a referer when the browser sends a request to the web server, tell the server the source of the link for this request. In the example above, if the web site for user B is userdomain-steal, want to steal a picture link `http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png`. A's website domain name is userdomain.

Suppose the web page of the chain website user domain-steal is as follows:

```
<html>
  <p>This is a test</p>
  
</html>
```

Assume the web page with the source station user domain is as follows:

```
<html>
  <p>This is my test link from OSS URL</p>
  
</html>
```

- When an Internet user uses a browser to access the Web page of B's website `http://userdomain-steal/index.html`, the link in the web page is a picture of the site A. Because a request from one domain name (user domain-steal) jumped to another domain name (maid), the browser takes the Referer with it in the header of the HTTP request, as shown:

You can see that the browser Referer in the HTTP request is `http://userdomain-steal/index.html`. This article mainly uses Chrome's developer mode to view web page requests, as follows:

- The same browser visits `http://userdomain/error.html`, and you can also see that the browser's Referer is `http://userdomain/error.html`.
- If the browser enters the address directly, you can see that Referer is empty in the request.

If a does not have any Referer-related settings on the OSS, all three cases have access to the picture link for user.

- The principle of OSS through Referer anti-theft chain

Thus, when the browser requests the OSS resource, if a page Jump occurs, the browser takes the Referer in the request, and the Referer's value is the URL on the previous page, sometimes Referer is empty.



For both cases, the OSS Referer feature offers two options:

- Sets whether empty Referer access is allowed. It cannot be set separately and needs to be used in conjunction with the Referer whitelist.
- Sets the Referer whitelist.

The details are analyzed as follows:

- Anti-theft chain authentication is performed only if the user is accessing the object through a signed URL or an anonymous access. If the requested header has an "Authorization" field, it does not do anti-theft chain validation.
- A bucket can support multiple Referer parameters.
- The Referer parameter supports wildcard characters '\*' and '?'.
- Users can set up to allow request access for empty referer.
- When the whitelist is empty, the Referer field is not checked for empty ( otherwise all requests will be rejected, because empty Referer will be rejected, for non-empty Referer OSS is also not found on the Referer whitelist ).
- The whitelist is not empty, and a rule is set that does not allow Referer fields to be empty. Only Referer's whitelist of requests is allowed, other requests, including those whose Referer is empty, are rejected.
- The whitelist is not empty, but the rule "allow Referer field to be empty" is set. An empty request with Referer and a whitelist-compliant request are permitted, other requests are rejected.
- Three permissions of bucket (private, public-read, public-read-write) the Referer field is checked.

Wildcard character explanation:

- Asterisks '\*': You can use an asterisks instead of 0 or more characters. If you are looking for a file name that starts with "AEW" , you can enter AEW to search for all types of files with the names starting with "AEW" , for example, AEWT.txt, AEWU.EXE, and AEWI.dll. If you want to narrow down the search scope, you can enter AEW.txt to search for all .txt files with names starting with AEW, such as AEWIP.txt and AEWDF.txt.
- Question mark (?): represents one character. If you enter love?, all types of files with names starting with "love" and ending with a character are displayed, such as lovey and lovei. If you want to narrow the search scope, you can enter

love?.doc to search for all .doc files with names starting with “love” and ending with a character, such as lovey.doc and lovei.doc.

- Anti-leech effects of different Referer settings

The following describes the effects of Referer settings:

- Disable Allow Empty Referer, as shown in the following figure:

**Direct access:** The resources are accessible even when anti-leech protection takes effect. The reason is, if the whitelist is blank, the system does not check whether the Referer field is blank. The Referer setting does not take effect when the whitelist is blank. Therefore, the Referer whitelist must be configured.

- Disable Allow Empty Referer and configure a Referer whitelist.

As shown in the preceding example, the Referer in the browser request is the URL of the current webpage. Therefore, it is necessary to know from which URL the request jumps and then specify the URL.

Referer whitelist setting rules:

- In the example, the Referer is `http://userdomain/error.html`. Therefore, the Referer whitelist can be set to `http://userdomain/error.html`. As the Referer check performed by OSS is based on prefix matching, access to other webpages such as `http://userdomain/index.html` fails. To avoid this problem, you can set the Referer whitelist set to `http://userdomain/`.
- To allow access to other domain names such as `http://img.userdomain/index.html`, add `http://*.userdomain/` to the Referer whitelist.

Both entries are configured as shown in the following figure:

**Anti-leech**

Set HTTP Referer whitelist to prevent leeching. [Learn more](#)

**Referer**


http://www.aliyun.com  
 http://www.\*.com  
 http://www.aliyun?.com

**Allow Empty Referer**

Save
Cancel

After testing, the following results are obtained:

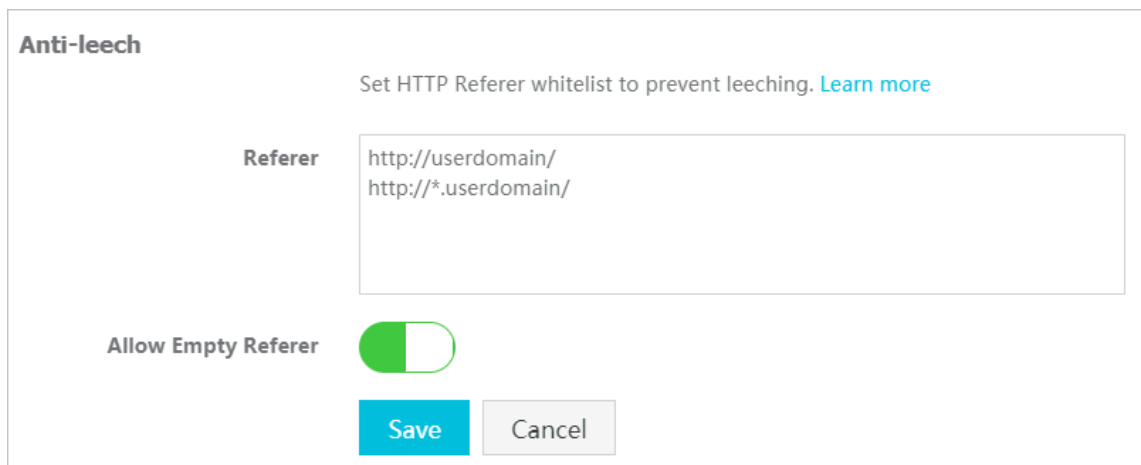
Browser input	Expectation	Result
<a href="http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png">http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png</a>	Expectation for direct access with a blank Referer: Blank Referers are not allowed and OSS returns 403.	As expected
<a href="http://userdomain/error.html">http://userdomain/error.html</a>	Expectation for a request from the origin site: successful access.	As expected
<a href="http://userdomain-steal/index.html">http://userdomain-steal/index.html</a>	Expectation for a request from a leeching site: OSS returns 403. Anti-leech protection is successful.	As expected
<a href="http://img.userdomain/error.html">http://img.userdomain/error.html</a>	Expectation for a request from a third-level domain of the origin site : successful access.	As expected

 **Note:**

- In this test, the domain names only serve as examples, and are not the same as the actual domain names you use. Be sure to differentiate them.
- If the Referer whitelist only contains `http://userdomain/`, and the browser attempts to access the resources through the simulated third-level domain name `http://img.userdomain/error.html`, the third-level domain name fails to match any of the entries in the Referer whitelist, and OSS returns 403.

- Enable Allow Empty Referer and configure a Referer whitelist.

The Referer whitelist contains `http://*.userdomain/` and `http://userdomain`, as shown in the following figure:



After testing, the following results are obtained:

Browser input	Expectation	Result
<a href="http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png">http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png</a>	Expectation for direct access with a blank Referer: successful access	As expected
<a href="http://userdomain/error.html">http://userdomain/error.html</a>	Expectation for a request from the origin site: successful access	As expected
<a href="http://userdomain-steal/index.html">http://userdomain-steal/index.html</a>	Expectation for a request from a leeching site: OSS returns 403. Anti-leech protection is successful.	As expected
<a href="http://img.userdomain/error.html">http://img.userdomain/error.html</a>	Expectation for a request from a third-level domain of the origin site : successful access	As expected

- How to configure Referer on OSS

Functional use reference:

- API: [Put Bucket Referer](#)
- Console: [Anti-leech settings](#)

- Pros and cons of Referer anti-leech protection

Referer anti-leech protection can be easily configured on the console. The main drawback of the Referer anti-leech protection is that it cannot prevent access attempts by the malicious spoofing Referers. If a leecher uses an application to simulate HTTP requests with a spoofing Referer, the Referer can bypass anti-leech protection settings. If you have higher anti-leech protection requirements, consider using signed URL anti-leech protection.

### Signed URLs

For the principles and implementation methods for signed URLs, see [Authorizing third-Party download](#). A signed URL is implemented as follows:

1. Set the bucket permission to private-read.
2. Generate a signature based on the expected expiration time (the time when the signed URL expires).

### Specific implementation

1. Install the latest PHP code by referring to the [PHP SDK documentation](#).
2. Generate a signed URL and add it to the webpage as an external link, for example:

```
<? php
require 'vendor/autoload.php';
#Indicates the automatic loading function provided by the latest
PHP.
use OSS\OssClient;
#Indicates the namespace used.
$accessKeyId="a5etodit71tlznjt3pdx7lch";
#Indicates the AccessKeyId, which must be replaced by the one you
use.
$accessKeySecret="secret_key";
#Indicates the AccessKeySecret, which must be replaced by the one
you use.
$endpoint="oss-cn-hangzhou.aliyuncs.com";
#Indicates the Endpoint, selected based on the region created by
the bucket. In the example, the endpoint is Hangzhou.
$bucket = 'referer-test';
#Indicates the bucket, which must be replaced by the one you use.
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);
$object = "aliyun-logo.png";
#Indicates the object to be signed.
$timeout = 300;
#Indicates the expected link expiration time. The value indicates
that the link is valid for 300 seconds from when this line of code
starts running.
$signedUrl = $ossClient->signUrl($bucket, $object, $timeout); #
Indicates the function used to implement the signed URL.
$img= $signedUrl;
#Indicates dynamically placing the signed URL in image resources
and printing it out.
```

```
$my_html = "<html>";
$my_html .= "<img src=\"\".$img. \"\" />";
$my_html .= "<p>\".$img.\"</p>";
$my_html .= "</html>";
echo $my_html;
? >
```

3. If the browser requests the resource multiple times, different signed URLs may be displayed. This is a normal phenomenon because the signed URL changes once it expires. After expiration time the link is no longer valid. It is displayed in Unix time format, for example, Expires=1448991693. The time can be converted to the local time. In Linux, the command for converting the time is `date -d@1448991693`. You can also find a conversion tool on the Internet.

### Special instructions

Signed URLs can be used with the Referer whitelist function.

If the expiration time of signed URLs is limited to minutes, even when a leecher spoofs a Referer, the leecher needs to obtain the signed URL and complete leeching before the signed URL expires. Compared with the Referer method, this makes leeching more difficult. Using signed URLs with the Referer whitelist function provides enhanced anti-leech protection results.

### Conclusion

Best practices of OSS-based anti-leech protection:

- Use third-level domain name URLs, such as `referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png`, as they are more secure than bound second-level domain names. The third-level domain name access method provides bucket-level cleaning and isolation, enabling you to respond to a burst in leeching traffic while preventing different buckets from affecting each other, thereby increasing service availability.
- If you use custom domain names as links, bind the CNAME to a third-level domain name, with the rule bucket + endpoint. For example, your bucket is named “test” and the third-level domain name is `test.oss-cn-hangzhou.aliyuncs.com`.
- Set the strictest possible permission for the bucket. For example, set a bucket that provides Internet services to public-read or private. Do not set it to public-read-write. For bucket permission information, see [Access control](#).
- Verify access sources and set a Referer whitelist based on your requirement.
- If you need a more rigorous anti-leeching solution, consider using signed URLs.

- Record access logs of the bucket, so that you can promptly discover leeching and verify the effectiveness of your anti-leeching solution. For access log information, see [Access logging configuration](#).

## FAQ

- I have configured anti-leech protection on the OSS Console, but the configuration does not take effect. Access to webpages is blocked, whereas access to players is not. Why? How can this problem be fixed?

Currently, anti-leech protection fails to take effect for audio and video files. When a media player, such as Windows Media Player or Flash Player, is used to request OSS resources, a blank Referer request is sent. This causes anti-leech protection ineffective. To resolve this issue, you can see the preceding signed URL anti-leech protection method.

- What is a Referer? How is it sent? How to deal with HTTPS websites? Does anything else need to be added, like commas?

A Referer is a request header in the HTTP protocol. It is attached to a request that involves a page jump. You must check whether the Referer in the request sent by your browser is `http://` or `https://`. In normal cases, the Referer is `http://`.

- How are signed URLs generated? Is storing the AccessKeySecret on the client secure?

See the individual SDK documentation for the method of signing the URL. It is not recommended that the AccessKeySecret be directly stored on the client. RAM provides the [STS service](#) to solve this problem. Also, see [RAM and STS Guide](#).

- How do I use wildcard characters (\*, ?) to write `a.baidu.com` and `b.baidu.com`?

You can use `http://*.baidu.com`. If the wildcard character represents a single character only, you can also use `http://?.baidu.com`.

- `*.domain.com` can match a second-level domain name, but does not match `domain.com`. Only adding a second entry of `domain.com` does not work either. What settings must be configured?

Note that a Referer generally includes a parameter such as `http`. You can view the request Referer in Chrome's developer mode and then specify the corresponding Referer. As in this case, you may have forgotten to include `http://`, which is required to be `http://domain.com`.

- What must I do if anti-leech protection does not take effect?

We recommend that you use Chrome to solve the problem. Open developer mode and click on the Web page to view the `Referer` specific values in the HTTP request. Check whether the `Referer` value matches the Referer value configured on OSS. If they do not match, set the Referer value configured on OSS to the Referer value in the HTTP request. If the problem persists, open a ticket.

## 5.3 Static website hosting

This document describes the process and procedure about how to build a simple static website based on OSS right from the beginning and also includes FAQs as well. The following are the key steps:

1. Apply for a domain name.
2. Activate OSS and create a bucket.
3. Activate Static Website Hosting on OSS.
4. Access OSS with custom domain names.

### Static website hosting overview

You can build a simple static website page based on OSS. Once you activate this function, OSS provides a default homepage and a default 404 page. For more information, see [Static Website Hosting](#) in the developer guide.

### Procedure

1. Apply for a domain name
2. Activate OSS and create a bucket
  - a. Log on to the OSS console and create a bucket named “imgleo23” in Shanghai with the endpoint `oss-cn-shanghai.aliyuncs.com`. For detailed operation, see [Create a bucket](#).
  - b. Set the bucket permission to public-read. For detailed operation, see [Set bucket ACL](#).
  - c. Upload the content of `index.htm` and `error.htm`. For detailed operation, see [Upload objects](#).
    - Body of `index.html`:

```
<html>
<head>
```



```

<title>Hello OSS! </title>
<meta charset="utf-8">
</head>
<body>
  <p>Welcome to OSS Static Website Hosting.</p>
  <p>This is the homepage.</p>
</body>
</html>

```

- **Body of error.html:**

```

<html>
  <head>
    <title>Hello OSS! </title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>This is an error homepage for OSS Static Website
    Hosting.</p>
  </body>
</html>

```

- `aliyun-logo.png` is a picture.

### 3. Activate static website hosting on OSS

As shown in the following figure, once you log on to the OSS console, set Default Homepage to `index.html` and Default 404 Page to `error.html`. For more information, see [Set static website hosting](#).

**Static Page**

Set your bucket to static website hosting mode. [Learn more](#)

Default Homepage

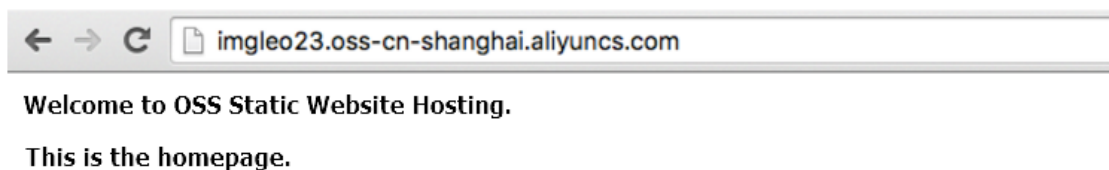
Enter the file name of the default webpage. Only the .html format object under the root directory is supported. If you do not enter a file name, the default homepage will be disabled.

Default 404 Page

Enter the file name of the 404 error default webpage. Only the .html, .jpg, .png, .bmp, and .webp formats are supported. If you do not enter a file name, the 404 error default webpage will be disabled.

To test the Static Website Hosting function, enter the URL as shown in the following figure:

- Display the default homepage:



When a similar URL is entered, the body of `index.html` specified upon activating the function is displayed.

- Display normal files



When a matched file for the entered URL is found, data is read successfully.

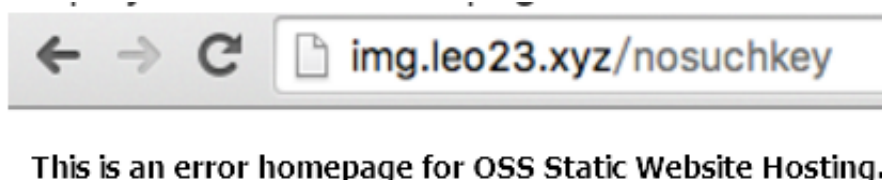
#### 4. Access OSS with custom domain names

For more information about how to access OSS with custom domain names, see [Access OSS with custom domain names](#).

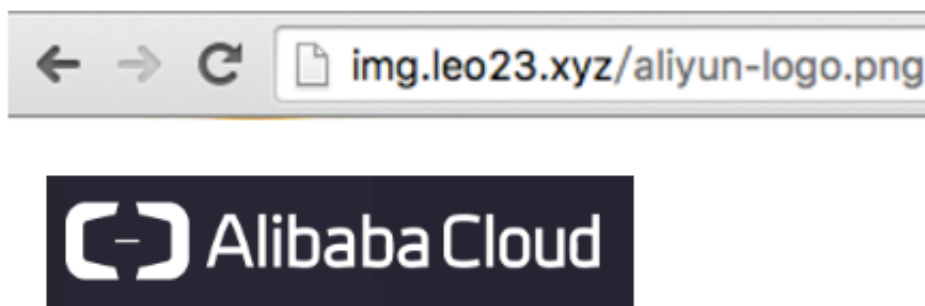
- Display the default homepage



- Display the default 404 page



- Display normal files



 Note:

When you use an OSS endpoint in Mainland China regions or the Hongkong region to access a web file through the Internet, the Content-Disposition: 'attachment=filename;' is automatically added to the Response Header, and the web file is downloaded as an attachment. If you access OSS with a user domain, the Content-Disposition: 'attachment=filename;' will not be added to the Response Header. For more information about using the user domain to access OSS, see [Bind a custom domain name](#).

## FAQ

- What are the benefits of OSS Static Website Hosting?

An ECS instance is saved in case any user needs a relatively small amount of traffic. In the case of larger traffic volumes, CDN can be used.

- How is OSS priced? How does OSS work with CDN?

For pricing, see the OSS and CDN prices on Alibaba Cloud website. For cases on combination of OSS and CDN, see [CDN-based OSS acceleration practices](#).

- Do the default homepage and default 404 page both need to be set?

The default homepage needs to be set, whereas the default 404 page does not need to be set.

- Why does the browser return a 403 error after a URL is entered?

The reason may be that the bucket permission is not public-read, or your Static Website Hosting function is suspended due to overdue payment.

## 6 Data security

---

### 6.1 Check data transmission integrity by using 64-bit CRC

#### Background

An error may occur when data is transmitted between the client and the server. Currently, OSS can return the 64-bit CRC value for an object uploaded in any mode. To check the data integrity, the client can compare the 64-bit CRC value with the locally calculated value.

- OSS calculates 64-bit CRC value for newly uploaded object, stores the result as metadata of the object, and then adds the `x-oss-hash-crc64ecma` header to the returned response header, indicating its 64-bit CRC value. This 64-bit CRC is calculated according to ECMA-182 Standard.
- For the object that already exists on OSS before the 64-bit CRC goes live, OSS does not calculate its 64-bit CRC value. Therefore, its 64-bit CRC value is not returned when such object is obtained.

#### Operation instructions

- Put Object / Append Object / Post Object / Multipart upload part returns the corresponding 64-bit CRC value. The client can get the 64-bit CRC value returned by the server after the upload is completed and can check it against the locally calculated value.
- In the case of Multipart Complete, if all the parts have their respective 64-bit CRC values, then the 64-bit CRC value of the entire object is returned. Otherwise, the 64-bit CRC value is not returned (for example, if a part has been uploaded before the 64-bit CRC goes live).
- Get Object / Head Object / Get ObjectMeta returns the corresponding 64-bit CRC value (if any). After Get Object is completed, the client can get the 64-bit CRC value returned by the server and check it against the locally calculated value.



Note:

The 64-bit CRC value of the entire object is returned for the range get object.

- For copy related operations, for example, Copy Object/Upload Part Copy, the newly generated object/Part may not necessarily have the 64-bit CRC value.

## Python example

An example of complete Python code is as follows. It shows how to check data transmission integrity based on the 64-bit CRC value.

### 1. Calculate the 64-bit CRC value.

```
import oss2
from oss2.models import PartInfo
import os
import crcmod
import random
import string
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)
def check_crc64(local_crc64, oss_crc64, msg="check crc64"):
if local_crc64 != oss_crc64:
print "{0} check crc64 failed. local:{1}, oss:{2}.".format(msg,
local_crc64, oss_crc64)
return False
else:
print "{0} check crc64 ok.".format(msg)
return True
def random_string(length):
return ''.join(random.choice(string.lowercase) for i in range(length
))
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
```

### 2. Verify Put Object.

```
content = random_string(1024)
key = 'normal-key'
result = bucket.put_object(key, content)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(content))
check_crc64(local_crc64, oss_crc64, "put object")
```

### 3. Verify Get Object.

```
result = bucket.get_object(key)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(result.resp.read()))
check_crc64(local_crc64, oss_crc64, "get object")
```

### 4. Verify Upload Part and Complete.

```
part_info_list = []
key = "multipart-key"
result = bucket.init_multipart_upload(key)
upload_id = result.upload_id
part_1 = random_string(1024 * 1024)
result = bucket.upload_part(key, upload_id, 1, part_1)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_1))
#Check whether the uploaded part 1 data is complete
check_crc64(local_crc64, oss_crc64, "upload_part object 1")
part_info_list.append(PartInfo(1, result.etag, len(part_1)))
part_2 = random_string(1024 * 1024)
result = bucket.upload_part(key, upload_id, 2, part_2)
```

```

oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_2))
#Check whether the uploaded part 2 data is complete
check_crc64(local_crc64, oss_crc64, "upload_part object 2")
part_info_list.append(PartInfo(2, result.etag, len(part_2)))
result = bucket.complete_multipart_upload(key, upload_id,
part_info_list)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_2, do_crc64(part_1)))
#Check whether the final object on the OSS is consistent with the
local file
check_crc64(local_crc64, oss_crc64, "complete object")

```

## OSS SDK support

Part of the OSS SDK already supports the data validation using crc64 for the upload and download, as shown in the following table:

SDK	Support for CRC?	Example
Java SDK	Yes	<a href="#">CRCSample.java</a>
Python SDK	Yes	<a href="#">object_check.py</a>
PHP SDK	No	N/A
C# SDK	No	None
C SDK	Yes	<a href="#">oss_crc_sample.c</a>
JavaScript SDK	No	None
Go SDK	Yes	<a href="#">crc_test.go</a>
Ruby SDK	No	None
iOS SDK	Yes	<a href="#">OSSCrc64Tests.m</a>
Android SDK	Yes	<a href="#">OSSCrc64Tests.m</a>

## 6.2 Protect data through client encryption

Client encryption means that the encryption is completed before the user data is sent to the remote server, whereas the plaintext of the key used for encryption is kept in the local computer only. Therefore, the security of user data can be ensured because others cannot decrypt the data to obtain the original data even if the data leaks.

This document describes how to protect data through client encryption based on the current Python SDK version of OSS.

## Principles

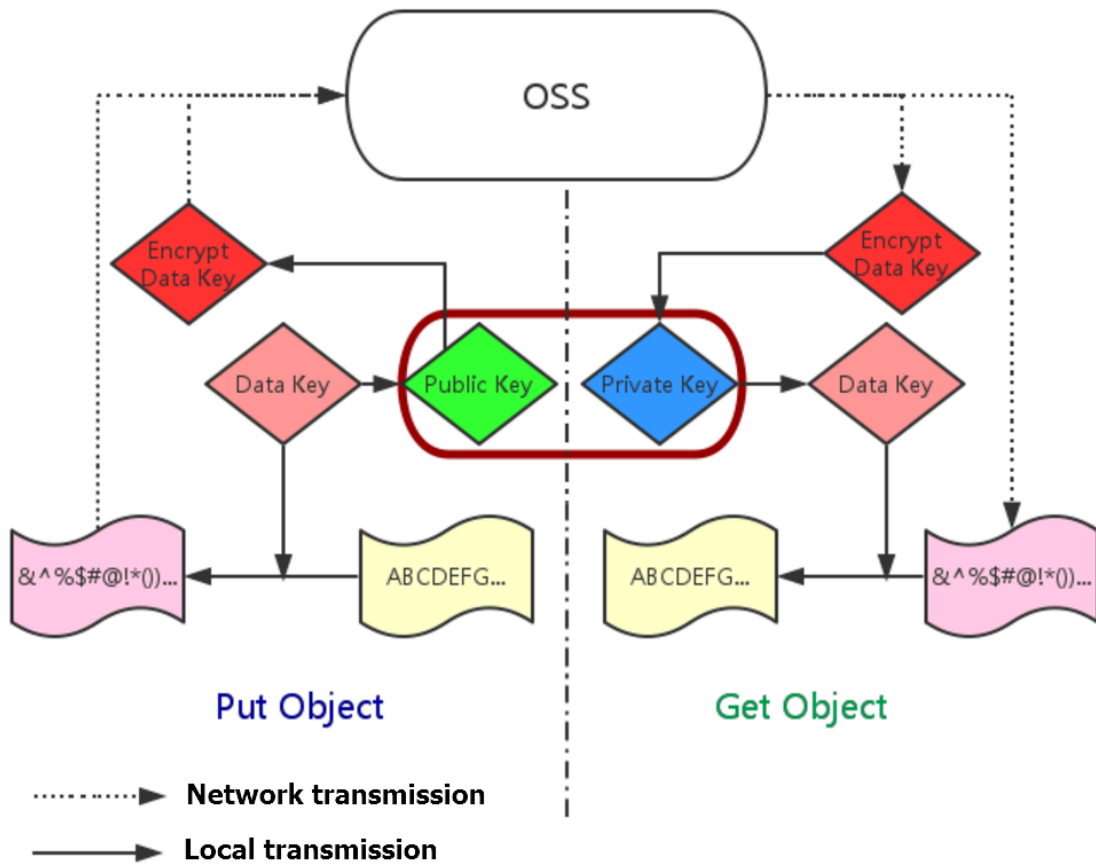
1. The user maintains a pair of RSA keys (`rsa_private_key` and `rsa_public_key`) in the local computer.
2. Each time when any object is uploaded, a symmetric key `data_key` of AES256 type is generated randomly, and then `data_key` is used to encrypt the original content to obtain `encrypt_content`.
3. Use `rsa_public_key` to encrypt `data_key` to obtain `encrypt_data_key`, place it in the request header as the custom meta of the user, and send it together with `encrypt_content` to the OSS.
4. When Get Object is performed, `encrypt_content` and `encrypt_data_key` in the custom meta of the user are obtained first.
5. The user uses `rsa_private_key` to decrypt `encrypt_data_key` to obtain `data_key`, and then uses `data_key` to decrypt `encrypt_content` to obtain the original content.



### Note:

The user's key in this document is an asymmetric RSA key, and the AES256-CTR algorithm is used when object content is encrypted. For more information, see [PyCrypto Document](#). This document describes how to implement client encryption through the custom meta of an object. The user can select the encryption key type and encryption algorithm as required.

Structural diagram



Preparation

1. For installation and usage of the Python SDK, see [Quick Installation of Python SDK](#).
2. Install the PyCrypto library.

```
pip install pycrypto
```

Example of complete Python code

```
# -*- coding: utf-8 -*-
import os
import shutil
import base64
import random
import oss2
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto import Random
from Crypto.Util import Counter
# aes 256, key always is 32 bytes
_AES_256_KEY_SIZE = 32
_AES_CTR_COUNTER_BITS_LEN = 8 * 16
class AESCipher:
    def __init__(self, key=None, start=None):
        self.key = key
```



```

        self.start = start
        if not self.key:
            self.key = Random.new().read(_AES_256_KEY_SIZE)
        if not self.start:
            self.start = random.randint(1, 10)
        ctr = Counter.new(_AES_CTR_COUNTER_BITS_LEN, initial_value=
self.start)
        self.cipher = AES.new(self.key, AES.MODE_CTR, counter=ctr)
        def encrypt(self, raw):
            return self.cipher.encrypt(raw)
        def decrypt(self, enc):
            return self.cipher.decrypt(enc)
# First, initialize the information such as AccessKeyId, AccessKeyS
ecret, and Endpoint.
# Obtain the information through environment variables or replace the
information such as "<Your AccessKeyId>" with the real AccessKeyId,
and so on.

# Use Hangzhou region as an example. Endpoint can be:
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com
# Access using the HTTP and HTTPS protocols respectively.
access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<your AccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<Your
AccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<Your Bucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<Your Access Domain Name>')
# Make sure that all the preceding parameters have been filled in
correctly.
for param in (access_key_id, access_key_secret, bucket_name, endpoint
):
    assert '<' not in param, 'Please set the parameter:' + param
##### 0 prepare #####
# 0.1 Generate the RSA key file and save it to the disk
rsa_private_key_obj = RSA.generate(2048)
rsa_public_key_obj = rsa_private_key_obj.publickey()
encrypt_obj = PKCS1_OAEP.new(rsa_public_key_obj)
decrypt_obj = PKCS1_OAEP.new(rsa_private_key_obj)
# save to local disk
file_out = open("private_key.pem", "w")
file_out.write(rsa_private_key_obj.exportKey())
file_out.close()
file_out = open("public_key.pem", "w")
file_out.write(rsa_public_key_obj.exportKey())
file_out.close()
# 0.2 Create the Bucket object. All the object-related interfaces can
be implemented by using the Bucket object
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
obj_name = 'test-sig-1'
content = "test content"
##### 1 Put Object #####
# 1.1 Generate the one-time symmetric key encrypt_cipher used to
encrypt this object, where key and start are values generated at
random
encrypt_cipher = AESCipher()
# 1.2 Use the public key to encrypt the information for assisting
encryption, and save it in the custom meta of the object. When Get
Object is performed later, we can use the private key to perform
decryption and obtain the original content according to the custom
meta
headers = {}

```

```
headers['x-oss-meta-x-oss-key'] = base64.b64encode(encrypt_obj.encrypt(
    encrypt_cipher.key))
headers['x-oss-meta-x-oss-start'] = base64.b64encode(encrypt_obj.
    encrypt(str(encrypt_cipher.start)))
# 1.3. Use encrypt_cipher to encrypt the original content to obtain
encrypt_content
encrypt_content = encrypt_cipher.encrypt(content)
# 1.4 Upload the object
result = bucket.put_object(obj_name, encrypt_content, headers)
if result.status / 100 != 2:
    exit(1)
#### 2 Get Object ####
# 2.1 Download the encrypted object
result = bucket.get_object(obj_name)
if result.status / 100 != 2:
    exit(1)
resp = result.resp
download_encrypt_content = resp.read()
# 2.2 Resolve from the custom meta the key and start that are
previously used to encrypt this object
download_encrypt_key = base64.b64decode(resp.headers.get('x-oss-meta-x-
    -oss-key', ''))
key = decrypt_obj.decrypt(download_encrypt_key)
download_encrypt_start = base64.b64decode(resp.headers.get('x-oss-meta-
    -x-oss-start', ''))
start = int(decrypt_obj.decrypt(download_encrypt_start))
# 2.3 Generate the cipher used for decryption, and decrypt it to
obtain the original content
decrypt_cipher = AESCipher(key, start)
download_content = decrypt_cipher.decrypt(download_encrypt_content)
if download_content != content:
    print "Error!"
else:
    print "Decrypt ok. Content is: %s" % download_content
```

# 7 Terraform

---

## 7.1 Introduction

Terraform is an open-source automatic resource orchestration tool that supports multiple cloud service providers. Alibaba Cloud (referenced as *terraform-alicloud-provider* in Terraform) allows developers to easily build, update, and version their infrastructure in the Alibaba Cloud Terraform ecosystem by supporting over 90 resources and data sources across more than 20 products and services.

*HashiCorp Terraform* is an automatic IT infrastructure orchestration tool that can manage and maintain IT resources by using code. The easy to use Command Line Interface (CLI) of Terraform allows you to deploy configuration files on Alibaba Cloud or any other supported cloud, and control the versions of the configuration files. The CLI provides code for the infrastructures (such as VMs, storage accounts, and network interfaces) defined in the configuration files that describe the cloud resource topology. The Command Line Interface (CLI) of Terraform provides a simple mechanism, which is used to deploy configuration files on Alibaba Cloud or any other supported cloud and control the versions of the configuration files. Terraform is a highly scalable tool that supports new infrastructures through providers. You can use Terraform to create, modify, or delete multiple resources, such as ECS, VPC, RDS, and SLB.

### Functions of OSS Terraform module

You can use the OSS Terraform module to manage buckets and objects. For example:

- Bucket management functions:
  - Creates a bucket.
  - Configures an ACL for a bucket.
  - Configures Cross-Origin Resource Sharing (CORS) for a bucket.
  - Sets logging for a bucket.
  - Configures static website hosting for a bucket.
  - Configures referers for a bucket.
  - Configures the lifecycle rules of a bucket.
- Object management functions:

- Uploads an object.
- Configures server-end encryption for an object.
- Sets an ACL for an object.
- Sets Object Meta.

## References

- For the installation and usage of Terraform, see [Use Terraform to manage OSS](#).
- To download the OSS Terraform module, see [terraform-alicloud-modules](#).
- For more information about the OSS Terraform module, see [alicloud\\_oss\\_bucket](#).

## 7.2 Use Terraform to manage OSS

This topic describes how to install and configure Terraform and how to use Terraform to manage OSS.

### Install and configure Terraform

Before using Terraform, follow these steps to install and configure Terraform:

1. Download the installation package applicable to your operating system from [Terraform official website](#). In this topic, Terraform is installed and configured in Linux as an example.
2. Extract the installation package to the following path: `/usr/local/bin`. If you extract the executable file to another path, you must add the path to global variables.
3. Run the path verification command. If a list of available Terraform options is displayed, Terraform is installed successfully.

```
[root@test bin]#terraform
Usage: terraform [-version] [-help] <command> [args]
```

4. Create and authorize a RAM user.
  - a. Log on to the [RAM console](#).
  - b. Create a RAM user named `Terraform` and create an AccessKey for the user. For more information, see [Create a RAM user](#).
  - c. Authorize the RAM user. You can add relevant permissions to the `Terraform` RAM user as needed. For detailed steps, see [Authorize RAM users](#).

**Notice:**

To maintain data security, do not use the AccessKey of your Alibaba Cloud account to configure Terraform.

5. You must create a separate directory for each Terraform project. Therefore, create a test directory first: `terraform-test`.

```
[root@test bin]#mkdir terraform-test
```

6. Enter the `terraform-test` directory.

```
[root@test bin]#cd terraform-test
[root@test terraform-test]#
```

7. Create a configuration file. Terraform reads all `*.tf` and `*.tfvars` files in the directory when running. Therefore, you can write configuration information to different files as needed. Some common configuration files are described as follows:

```
provider.tf: Used to configure providers.
terraform.tfvars: Used to configure the variables required to
configure providers.
variable.tf: Used to configure universal variables.
resource.tf: Used to define resources.
data.tf: Used to define package files.
output.tf: Used to configure the output.
```

For example, when you create the `provider.tf` file, you can configure your authentication information as follows:

```
[root@test terraform-test]# vim provider.tf
provider "alicloud" {
  region      = "cn-beijing"
  access_key  = "LTA*****NO2"
  secret_key  = "M0k8x0*****wwff"
```

For more information about configurations, see [alicloud\\_oss\\_bucket](#).

8. Initialize a working directory.

```
[root@test terraform-test]#terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.
hashicorp.com...
- Downloading plugin for provider "alicloud" (1.25.0)...
```

The following providers do not have any version constraints in configuration, so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking changes, it is recommended to add `version = "..."` constraints to the corresponding provider blocks in configuration, with the constraint strings suggested below.

```
* provider.alicloud: version = "~> 1.25"
```

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running `"terraform plan"` to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.



**Notice:**

**After creating a directory and configuration files for a Terraform project, you must initialize the directory.**

You can use Terraform after completing the preceding steps.

## Use Terraform to manage OSS

After Terraform is installed, you can run commands in Terraform to manage OSS.

Some common commands are described as follows:

- `terraform plan`: You can run this command to view the operations to be executed by a configuration file.

For example, you add a configuration file named `test.tf` that is used to create a bucket as follows:

```
[root@test terraform-test]#vim test.tf
resource "alicloud_oss_bucket" "bucket-acl"{
  bucket = "figo-chen-2020"
  acl = "private"
```

```
}
```

In this case, you can run the `terraform plan` command to view the operations to be executed by `test.tf`.

```
[root@test terraform-test]# terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will
not be
persisted to local or remote state storage.
```

---

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
+ create

Terraform will perform the following actions:

```
+ alicloud_oss_bucket.bucket-acl
  id:                <computed>
  acl:                "private"
  bucket:             "figo-chen-2020"
  creation_date:     <computed>
  extranet_endpoint: <computed>
  intranet_endpoint: <computed>
  location:           <computed>
  logging_isenable:  "true"
  owner:              <computed>
  referer_config.#: <computed>
  storage_class:     <computed>
```

Plan: 1 to add, 0 to change, 0 to destroy.

---

Note: You didn't specify an "-out" parameter to save this plan, so Terraform can't guarantee that exactly these actions will be performed if "terraform apply" is subsequently run.

- `terraform apply`: You can run this command to execute a configuration file in the working directory.

For example, if you want to create a bucket named `figo-chen-2020`, you must add a configuration file named `test.tf` that is used to create a bucket as follows:

```
[root@test terraform-test]#vim test.tf
resource "alicloud_oss_bucket" "bucket-acl"{
  bucket = "figo-chen-2020"
  acl = "private"
}
```

Run the `terraform apply` command to execute the configuration file as follows:

```
[root@test terraform-test]#terraform apply
```

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
+ create

Terraform will perform the following actions:

```
+ alicloud_oss_bucket.bucket-acl
  id:                <computed>
  acl:                "private"
  bucket:            "figo-chen-2020"
  creation_date:     <computed>
  extranet_endpoint: <computed>
  intranet_endpoint: <computed>
  location:          <computed>
  logging_isenable:  "true"
  owner:             <computed>
  referer_config.#: <computed>
  storage_class:     <computed>
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

```
alicloud_oss_bucket.bucket-acl: Creating...
acl:                "" => "private"
bucket:             "" => "figo-chen-2020"
creation_date:     "" => "<computed>"
extranet_endpoint: "" => "<computed>"
intranet_endpoint: "" => "<computed>"
location:          "" => "<computed>"
logging_isenable:  "" => "true"
owner:             "" => "<computed>"
referer_config.#:  "" => "<computed>"
storage_class:     "" => "<computed>"
alicloud_oss_bucket.bucket-acl: Creation complete after 1s (ID: figo-chen-2020)
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.



#### Note:

After you have executed the configuration file, a new bucket is created if the *figo-chen-2020* bucket does not exist. If the *figo-chen-2020* bucket already exists and is an empty bucket created by Terraform, the bucket is deleted and a new bucket with the same name is created.

- **terraform destroy:** You can run this command to delete an empty bucket created by Terraform.
- **terraform import:** You can run this command to import a bucket that is not created by Terraform.



First, create a configuration file named *main.tf* and add configurations to the file as follows:

```
[root@test terraform-test]#vim main.tf
resource "alicloud_oss_bucket" "bucket" {
  # (resource arguments)
}
```

Then, run the following command to import an existing bucket:

```
terraform import alicloud_oss_bucket.bucket bucketname
```

## References

- For more bucket configuration examples, see [alicloud\\_oss\\_bucket](#).
- For more object configuration examples, see [alicloud\\_oss\\_bucket\\_object](#).