

Object Storage Service

Best Practices

Issue 01
Date 2025-09-26



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 Overview.....	1
2 Using Backup Software to Back Up Local Data to OBS.....	3
2.1 Overview.....	3
2.2 Using Commvault to Back Up Local Data in SAP HANA.....	3
3 Using a User-Defined Domain Name to Host a Static Website.....	6
3.1 Overview.....	6
3.2 Resources and Costs.....	7
3.3 Static Website Hosting Process.....	8
3.4 Procedure.....	9
3.4.1 Making Preparations.....	9
3.4.2 Uploading Static Website Files.....	10
3.4.3 Configuring Static Website Hosting.....	11
3.4.4 Configuring a User-Defined Domain Name.....	12
3.4.5 Creating and Configuring Domain Name Hosting.....	13
3.4.6 Verifying the Hosting.....	14
3.4.7 (Follow-up Operation) Updating a Static Website.....	15
4 Optimizing the Performance.....	18
5 Using the PostObject API to Upload Data from a Web Client to OBS.....	19
6 Uploading Data from Mobile Apps to OBS.....	26
6.1 Overview.....	26
6.2 Using a Temporary Security Credential to Upload Data to OBS.....	26
6.3 Using a Presigned URL to Upload Data to OBS.....	30
7 Uploading Data from Mini Programs to OBS.....	36
8 Accessing OBS Through an NGINX Reverse Proxy.....	43
9 Using OBS to Decouple Storage from Compute in Big Data Scenarios.....	49
9.1 Overview.....	49
9.2 Process.....	53
9.3 Connecting Big Data Platforms to OBS.....	53
9.3.1 Supported Big Data Platforms.....	54
9.3.2 Connecting MRS to OBS.....	54

9.3.3 Connecting Cloudera CDH to OBS.....	55
9.3.4 Connecting Hortonworks HDP to OBS.....	57
9.4 Connecting OBS to Big Data Components.....	59
9.4.1 Supported Big Data Components.....	59
9.4.2 Connecting Hadoop to OBS.....	60
9.4.3 Connecting Hive to OBS.....	66
9.4.4 Connecting Spark to OBS.....	67
9.4.5 Connecting Presto to OBS.....	67
9.4.6 Connecting Flume to OBS.....	71
9.4.7 Connecting DataX to OBS.....	73
9.4.8 Connecting Druid to OBS.....	75
9.4.9 Connecting Flink to OBS.....	77
9.4.10 Connecting Logstash to OBS.....	78
9.5 Migrating HDFS Data to OBS.....	79
A Change History.....	81

1 Overview

This document summarizes practices in common application scenarios of Object Storage Service (OBS). Each practice case is given detailed solution description and operation guidance, helping you easily build your storage services based on OBS.

Table 1-1 OBS best practices

Best Practice	Description
Using Backup Software to Back Up Local Data to OBS	Describes why you should back up local data to OBS and what backup software you can use, and uses Commvault as an example to describe how to back up local data to OBS.
Using a User-Defined Domain Name to Host a Static Website	Describes how to use a user-defined domain name to host static websites on OBS, for you to quickly launch personal and enterprise static websites without setting up website servers.
Optimizing the Performance	Describes how to add random prefixes to object names to implement horizontal expansion for high-speed requests, and thus improve the access rate and shorten the access latency.
Using the PostObject API to Upload Data from a Web Client to OBS	Describes how to use the PostObject API to directly upload files from a web client to OBS, that is, to upload files to OBS through a browser. With this method, you can directly upload data to OBS, without having to upload data to the app server first. This makes the transmission faster and does not impose pressure on the server. Additionally, it is more secure to adopt direct transmission with a signature returned by the server.
Uploading Data from Mobile Apps to OBS	Describes two methods for app clients to access OBS, to better protect data and prevent data leakage and unauthorized access.
Uploading Data from Mini Programs to OBS	Demonstrates how to upload files to OBS using a mini program.

Best Practice	Description
Accessing OBS Through an Nginx Reverse Proxy	Describes how to configure the Nginx reverse proxy on an ECS, so that you can use a fixed IP address to access OBS.

2 Using Backup Software to Back Up Local Data to OBS

2.1 Overview

In traditional backup and restore solutions, data needs to be first written to storage devices (like tapes) and then transported to a data center. In this process, data security and integrity are subject to many factors, such as hardware performance and personnel. In addition, data center deployment and maintenance result in complex management and high costs.

Cloud storage is easy-to-use, secure, efficient, and cost-effective, making it an attractive substitute for traditional storage devices such as tapes. OBS is such a cloud storage service that is scalable and stores a massive amount of data. The service and storage nodes of OBS are deployed in distributed clusters, greatly improving its scalability. Data redundancy and consistency check make data stored on OBS secure and reliable. With OBS, you pay for what you use so that costs are easy to estimate.

Third-party backup software, such as Commvault and AnyBackup Cloud, can be connected to OBS for data backup. With such backup software, you can customize backup policies for secure and efficient backups.

2.2 Using Commvault to Back Up Local Data in SAP HANA

SAP HANA is a high-performance real-time data computing platform based on in-memory computing. It is commonly used in enterprises that have large amounts of real-time business data to process. Commvault is seamlessly integrated with SAP HANA and OBS to support backups for online databases and logs. If your SAP HANA system becomes faulty or data migration is required, Commvault can help you quickly and easily restore data, thereby providing enterprise-level data protection for SAP HANA.

NOTE

Commvault V11 is recommended in this scenario.

Logical Architecture

In the following example, Commvault is used to back up the SAP HANA locally deployed on a single node. **Figure 2-1** shows the logical architecture.

Figure 2-1 Logical architecture

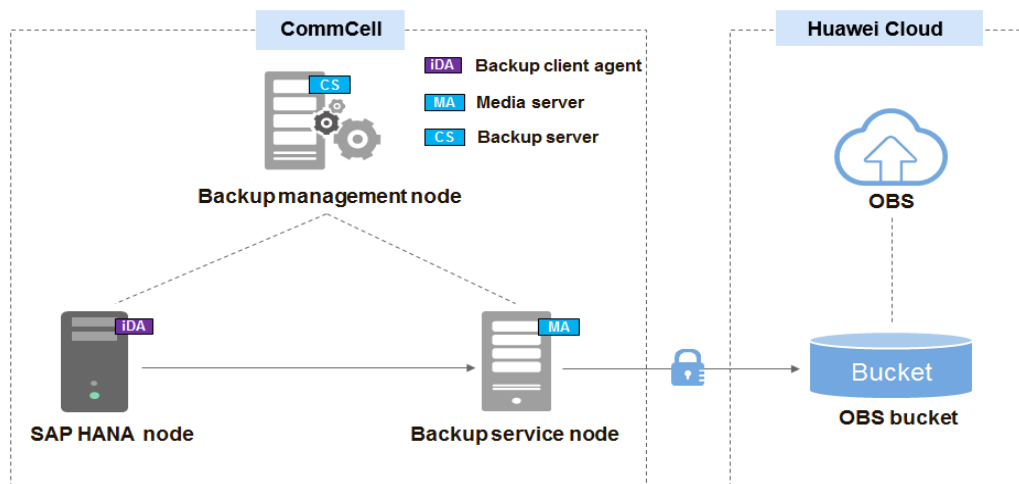


Table 2-1 describes the components in the logical architecture.

Table 2-1 Component description

Component	Description
iDataAgent (iDA)	Backup client agent, which is deployed on the SAP HANA node to obtain data to be backed up from SAP HANA.
CommServe (CS)	Backup server, which is deployed on the backup management node and is responsible for formulating global backup policies and scheduling backup services.
MediaAgent (MA)	Backup media, which is deployed on the backup service node and stores backup data to OBS.
OBS	Stores backup data in OBS buckets.

NOTE

A CommCell is a backup management domain and a logical grouping of all software components that obtain, move, and manage data and information.

Backup Process

1. Install and pre-configure the backup software.
Install and configure the backup server (CommServe), backup media (MediaAgent), and SAP HANA backup client agent (iDataAgent).
2. Create backup storage space (an OBS bucket).
 - a. Log in to OBS Console and create a bucket for storing backup data. For details about how to create a bucket, see [Creating a Bucket](#).
 - b. Create a cloud repository on CommCell Console. Enter the OBS endpoint address, access keys, and the bucket name to associate the MediaAgent of Commvault with OBS.

NOTE

CommCell Console is a graphical user interface for managing CommCell environments, monitoring and controlling activity jobs, and viewing activity-related events.

3. Create a Commvault backup policy.
Create a backup policy on CommCell Console and specify the backup period, time, and encryption method.
4. Check the backup execution status.
During the execution of a backup policy, view the backup execution status on CommCell Console.
5. (Optional) Restore data.
Restore data to the SAP HANA source host.

NOTE

For detailed Commvault operations, see [Commvault Documentation](#).

3 Using a User-Defined Domain Name to Host a Static Website

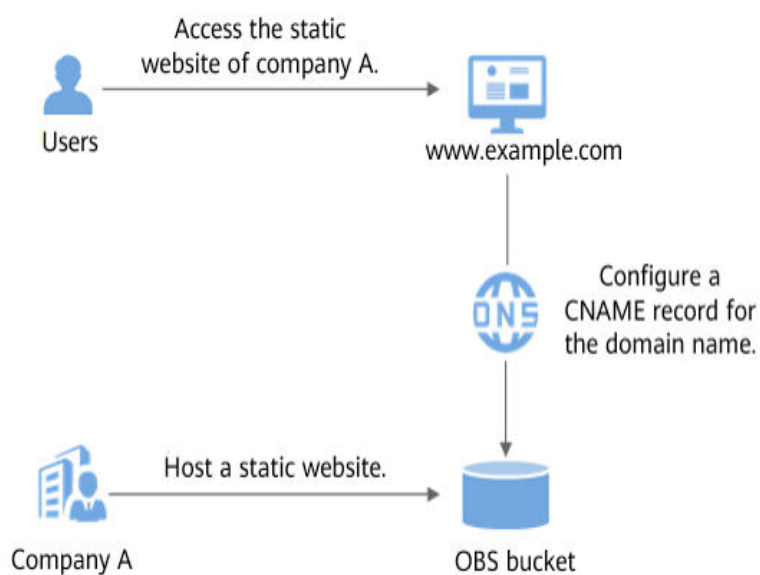
3.1 Overview

Application Scenario

A company has a large number of static websites for users to access, but does not want to set up servers. In this case, this company can host its static websites in an OBS bucket, so that users can access the hosted static websites using the domain name bound to the OBS bucket.

Solution Architecture

Figure 3-1 Using a user-defined domain name to access hosted static resources



1. Company A hosts its static websites in an OBS bucket and binds a user-defined domain name to the bucket.
2. Users access the static websites hosted in the bucket by accessing the bound domain name.

Before starting the configuration, you may need to learn more about [static website hosting](#).

Solution Advantages

- You can quickly build static websites in a simple way, with low operation costs.
- Static websites can be easily rolled out without the need to set up servers.

3.2 Resources and Costs

The table below describes the resources that you need in this practice.

Table 3-1 Resource and cost description

Resource	Description	Cost
OBS	An OBS bucket must be created for hosting static website files, and the bucket must have the static website hosting configured and a user-defined domain name bound.	Fees charged for using OBS: • Storage fee generated for storing static website files in OBS • Request fee generated when users access static website files stored in OBS • Traffic fee generated when users use a user-defined domain name to access OBS over the Internet Actual fees vary depending on the size of stored files, the number of requests, and the traffic volume.
Static website files	• Static website homepage: Index page (homepage) that is returned when the hosted static website is accessed. Example: index.html • 404 error page: Page that is returned when the accessed static website path is incorrect. Example: error.html	Free

Resource	Description	Cost
User-defined domain name	The user's domain name that needs to be bound to the OBS bucket. As required by the MIIT, the user must complete the ICP filing, if the bucket to which the user-defined domain name is bound is in any of the following regions: Example: www.example.com	Domain name registration fees charged by the registrar
DNS	A CNAME record must be configured on DNS for the domain name bound to the bucket.	Free

In this example, the static website files are as follows:

- Content in the **index.html** file

```
<html>
<head>
  <title>Hello OBS!</title>
  <meta charset="utf-8">
</head>
<body>
  <p>Welcome to OBS static website hosting.</p>
  <p>This is the homepage.</p>
</body>
</html>
```

- Content in the **error.html** file

```
<html>
<head>
  <title>Hello OBS!</title>
  <meta charset="utf-8">
</head>
<body>
  <p>Welcome to OBS static website hosting.</p>
  <p>This is the 404 error page.</p>
</body>
</html>
```

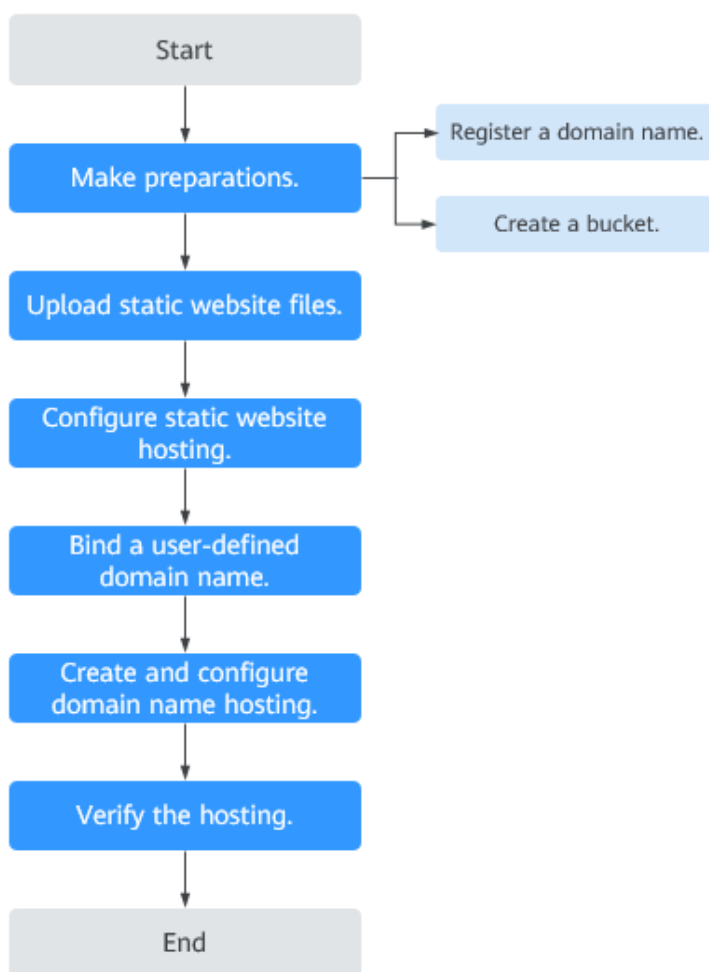
3.3 Static Website Hosting Process

The procedure is as follows:

- Create a bucket on OBS Console for storing static website resources and enable static website hosting for the bucket.
- Bind a user-defined domain name to the bucket using the domain name binding function of OBS.
- Create and configure domain name hosting using Domain Name Service (DNS) so that the user-defined domain name can be used to access the static website hosted on OBS.

Specific operations are as follows:

Figure 3-2 Process of using a user-defined domain name to host a static website



3.4 Procedure

3.4.1 Making Preparations

Registering a Domain Name

If you already have a registered domain name, skip this step.

If you do not have such domain name, register one with a registrar. In this example, the example domain name **www.example.com** is used. In practice, you need to replace the domain name with the one you actually planned.

Creating a Bucket

There are no special requirements on bucket names. Follow the instructions on the console to create a bucket for storing static website files. The following example describes how to create a bucket named **example**:

Step 1 Log in to [OBS Console](#).

Step 2 Click **Create Bucket** in the upper right corner of the page.

Step 3 Configure the following parameters:

- **Region:** Select a region closest to you.
- **Default Storage Class:** Select **Standard** (recommended).

 NOTE

You can also select **Infrequent Access** or **Archive** based on how frequently the website is accessed or how fast the website should respond. For details about storage classes, see [Storage Classes](#).

- **Bucket Name:** Enter **example**.
- **Bucket Policy:** Select **Public Read** to allow any user to access objects in the bucket.
- **Server-Side Encryption:** Select **Disable**.
- **Enterprise Project:** The default project is **default**. You can also [create an enterprise object](#) and then choose it for the bucket you are creating. Only an enterprise account can configure enterprise projects.

Step 4 Click **Create Now**. The bucket is created.

----End

3.4.2 Uploading Static Website Files

Prepare all static website files to be uploaded and repeat the following steps on OBS Console until all files are uploaded to the bucket created in [Making Preparations](#).

Step 1 Click the name of the created bucket to go to the **Objects** page.

Step 2 Click **Upload Object**. A dialog box shown in [Figure 3-3](#) is displayed.

Figure 3-3 Uploading objects

Upload Object [How to Upload a File Larger than 5 GB?](#)

① Upload Object ——— ② (Optional) Configure Advanced Settings

i Upload actions will generate [requests](#) . After the upload, you will be billed for [data storage](#) .

Storage Class

Standard Infrequent Access Archive

Optimized for frequently accessed (multiple times per month) data such as small and essential files that require low latency. If you do not change this setting, your uploaded objects will be stored using the default storage class you selected during bucket creation. [Learn more](#)

Upload Object

i The file or folder you newly upload will overwrite any existing file or folder with the same name. To keep different versions of the same file or folder, enable versioning for the current bucket.



Drag files or folders here to upload. Or [add file](#)
(A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.)

Server-Side Encryption

Disable SSE-KMS

Next: (Optional) Configure Advanced Settings

Upload Cancel

Step 3 Add the file to be uploaded.

NOTE

- Static website files cannot be encrypted for upload.
- It is recommended that you select **Standard** for the storage class. If you store the static website files in the **Archive** storage class, you need to restore the files before accessing them. For details, see [Restoring an Archive File](#).
- The website homepage file (**index.html**) and 404 error page (**error.html**) must be stored in the root directory of the bucket.

Step 4 Click **Upload** to complete the upload.

----End

3.4.3 Configuring Static Website Hosting

After the static website files are uploaded, perform the following steps to configure the static website hosting for the bucket.

NOTE

You can also redirect the entire static website to another bucket or domain name. For details, see [Configuring Redirection](#).

Step 1 Click the bucket name to go to the **Objects** page. In the navigation pane, choose **Basic Configurations > Static Website Hosting**.

Step 2 Click **Configure Static Website Hosting**.

Step 3 In the dialog box that is displayed, enable this function and select **Host a static website** for **Hosting Type**. Set **Homepage** to **index.html** and **404 Error Page** to **error.html**, as shown in [Figure 3-4](#).

Figure 3-4 Configuring static website hosting

Configure Static Website Hosting

With static website hosting, your static website contents can be easily accessed through the endpoint provided by OBS.

Status ☒

Hosting Type **Host a static website** Redirect requests [Learn more](#)

This option requires that the bucket policy is Public Read or anonymous users have been granted permissions through an object ACL to read hosted static website files.

Homepage
Only HTML files under the root directory are supported.

404 Error Page
Only HTML, JPG, PNG, BMP, and WEBP files under the root directory are supported.

Redirection Rule

OK Cancel

Step 4 Click **OK**.

----End

3.4.4 Configuring a User-Defined Domain Name

Step 1 Click the bucket name to go to the **Objects** page. In the navigation pane, choose **Domain Name Mgmt.**

Step 2 Click **Configure User Domain Name** in the upper part of the page, or in the lower card area of the page (when no user-defined domain names are available). On the displayed page, enter **www.example.com**, as shown in [Figure 3-5](#).

Figure 3-5 Configuring a user-defined domain name**Configure User Domain Name**

Add User Domain Name

1 Add a domain name so that you can use it to access the files stored in the bucket. ———— 2 Resolve CNAME
Resolve the domain name to finish the binding.

Tip The bound user domain names only support access over HTTP now.

* Origin Server Bucket domain                  

* User Domain Name

[+ Add Domain Name \(1/5\)](#)

Step 3 Click **OK**.

Step 4 Based on the tips, click **Resolve** or manually add a CNAME record set. Then, click **OK**.

NOTE

Clicking **Resolve** will automatically add CNAME record sets for Huawei Cloud domain names. To resolve those domain names not registered with Huawei Cloud, you need to configure resolution rules by yourself.

----End

3.4.5 Creating and Configuring Domain Name Hosting

To facilitate unified management of your user-defined domain names and static websites, you can host your user-defined domain names on Huawei Cloud DNS. After the hosting is configured, you can manage domain names on DNS, including managing record sets and PTR records, as well as creating wildcard DNS records.

NOTE

You can also add a CNAME record to the DNS at the DNS registrar, mapping to the domain name for the website hosted by the bucket. For example, if **www.example.com** is in the **EU-Dublin** region, you need to add a CNAME record whose value is **www.example.com CNAME www.example.com.obs-website.eu-west-101.myhuaweicloud.eu** at your DNS registrar.

To create and configure domain name hosting on DNS, perform the following steps:

Step 1 Add a public zone.

Use the root domain name **example.com** created in [Making Preparations](#) as the name of the public zone to be created. For details, see "Create a Public Zone" in [Routing Internet Traffic to a Website](#).

Step 2 Add a CNAME record.

In DNS, add a record set for the sub-domain name **www.example.com** of the hosted domain name, to map the CNAME of the sub-domain name to the static website domain name hosted by OBS. Configure the parameters as follows:

- **Name:** Enter **www**.
- **Type:** Select **CNAME – Map one domain to another**.
- **Line:** Select **Default**.
- **TTL (s):** Retain the default value.
- **Value:** Domain name to map. If CDN acceleration is disabled when a user-defined domain name is bound, enter the static website hosting domain name of the bucket. If CDN acceleration is enabled, set this parameter to the acceleration domain name (CNAME) provided by CDN.

For details, see [Adding a CNAME Record Set](#).

Step 3 Change the DNS server address at your domain name registrar.

At your domain name registrar, change the DNS server address in the NS record of the root domain name to the cloud DNS server address. The specific address is the NS value of the public zone in DNS.

For details, see section "Change the DNS Servers of the Domain Name" in [Routing Internet Traffic to a Website](#).

NOTE

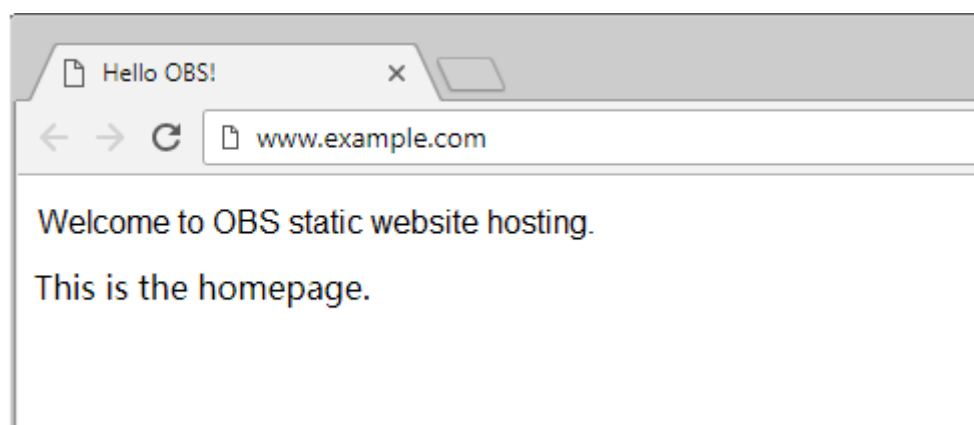
The address change will be effective within 48 hours. The actual time taken varies depending on the domain name registrar.

----End

3.4.6 Verifying the Hosting

- Enter **www.example.com** in the address bar of a browser to verify that you can access the default homepage, as shown in [Figure 3-6](#).

Figure 3-6 Default homepage



- In a web browser, enter a static file access address (for example, **www.example.com/imgs**) that does not exist in the bucket to verify that the 404 error page is displayed, as shown in [Figure 3-7](#).

Figure 3-7 404 error page**NOTE**

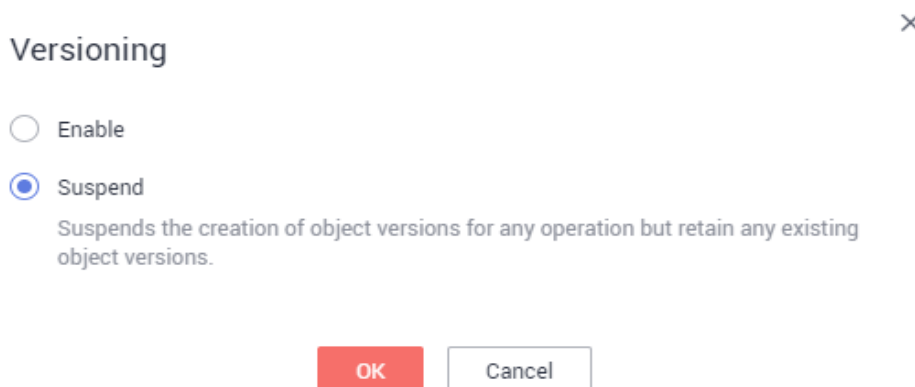
Due to browser caching, you may need to clear the browser cache to view the expected results.

3.4.7 (Follow-up Operation) Updating a Static Website

If you need to update a static file (such as a picture, music file, HTML file, or CSS file) on a static website, you can upload this file again. By default, if you upload a new file using the same name as the file you previously uploaded to the same path of OBS, the new file overwrites the previous one. To avoid this problem, you can enable versioning for OBS. A versioned OBS bucket stores static files of multiple versions, so that you can quickly retrieve and restore different versions or restore data in the event of unintended operations or application failures.

Enabling Versioning

- Step 1** Log in to OBS Console.
- Step 2** In the bucket list, click the bucket you want to go to the **Objects** page. In the navigation pane, click **Overview**.
- Step 3** In the **Basic Information** area, find **Versioning** and click **Edit**. The **Versioning** dialog box is displayed, as shown in [Figure 3-8](#).

Figure 3-8 Configuring versioning

Step 4 Select **Enable** and click **OK** to enable versioning for objects in the bucket.

----End

For more information about versioning, see [Versioning](#).

Updating Static Files

Step 1 Log in to OBS Console.

Step 2 In the bucket list, click the bucket you want to go to the **Objects** page.

Step 3 Click **Upload Object**, or go to the folder where the file you want to update is located and click **Upload Object**. A dialog box is displayed, as shown in [Figure 3-9](#).

Figure 3-9 Uploading objects

Upload Object [How to Upload a File Larger than 5 GB?](#)

① Upload Object ——— ② (Optional) Configure Advanced Settings

i Upload actions will generate requests. After the upload, you will be billed for data storage.

Storage Class

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

Optimized for frequently accessed (multiple times per month) data such as small and essential files that require low latency. If you do not change this setting, your uploaded objects will be stored using the default storage class you selected during bucket creation. [Learn more](#)

Upload Object

i The file or folder you newly upload will overwrite any existing file or folder with the same name. To keep different versions of the same file or folder, enable versioning for the current bucket.



Drag files or folders here to upload. Or [add file](#)
(A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.)

Server-Side Encryption

Disable	SSE-KMS
---------	---------

Next: (Optional) Configure Advanced Settings

Upload Cancel

Step 4 Add the file to be uploaded.

NOTE

- Static website files cannot be encrypted for upload.
- It is recommended that you select **Standard** for the storage class. If you store the static website files in the **Archive** storage class, you need to restore the files before accessing them. For details, see [Restoring an Archive File](#).

Step 5 Click **Upload** to complete the upload.

The most recently uploaded file with the same name as those previously uploaded ones is displayed as the latest version in the object list. Each time, only the latest version is accessed. This way, the static website file can be updated.

----End

4 Optimizing the Performance

OBS manages partitions based on the UTF-8 encoding range of object names and implements horizontal expansion and dynamic load balancing accordingly. If you use sequential prefixes (sorted by timestamp or in alphabetical order) for object naming, object access requests may be concentrated in a specific partition, resulting in access hotspots. This limits the request rate in the hot partition and increases access latency.

Random prefixes are recommended for naming objects, so requests can be evenly distributed across partitions, achieving horizontal expansion.

Example:

In a typical scenario of log archiving, the names of objects to be uploaded are as follows:

```
yourbucket/obslog/20190610-01.log.tar.gz
yourbucket/obslog/20190610-02.log.tar.gz
yourbucket/obslog/20190610-03.log.tar.gz
yourbucket/obslog/20190610-04.log.tar.gz
...
yourbucket/obslog/20190611-01.log.tar.gz
yourbucket/obslog/20190611-02.log.tar.gz
yourbucket/obslog/20190611-03.log.tar.gz
yourbucket/obslog/20190611-04.log.tar.gz
```

You are advised to add a hexadecimal hash prefix with three or more digits to each object name.

```
yourbucket/6ac-obslog/20140610-01.log.tar.gz
yourbucket/b42-obslog/20140610-02.log.tar.gz
yourbucket/17f-obslog/20140610-03.log.tar.gz
yourbucket/ac9-obslog/20140610-04.log.tar.gz
...
yourbucket/95d-obslog/20140611-01.log.tar.gz
yourbucket/4a5-obslog/20140611-02.log.tar.gz
yourbucket/ea2-obslog/20140611-03.log.tar.gz
yourbucket/ba3-obslog/20140611-04.log.tar.gz
```

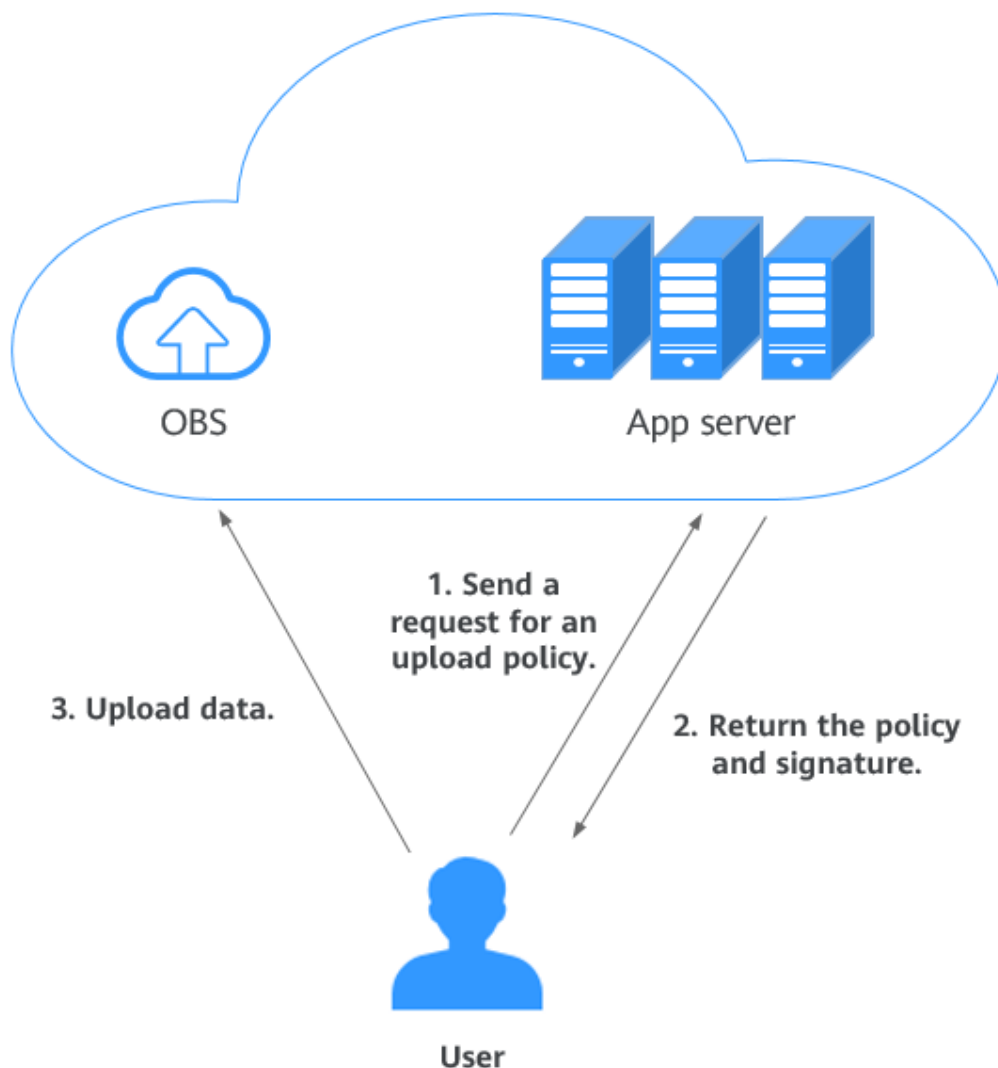
5 Using the PostObject API to Upload Data from a Web Client to OBS

Context

Files are usually uploaded to the app server through a browser and then to OBS. Data transfer on the app server results in a low efficiency. In addition, the app server will be heavily loaded if multiple tasks are concurrently uploaded.

This section describes how to use the PostObject API to directly upload files from a web client to OBS, that is, to upload files to OBS through a browser. As shown in [Figure 5-1](#), you can directly upload data to OBS, without having to upload data to the app server first. This makes the transmission faster and does not impose pressure on the server. Additionally, direct transmission with a signature returned by the server is more secure.

Figure 5-1 Direct transfer using the PostObject API



Prerequisites

An OBS bucket is available. For details about how to create a bucket, see [Creating a Bucket](#).

Procedure

The configuration consists of the following two steps:

Step 1: Configure CORS.

In web page requests, website scripts and contents in one origin cannot interact with those in another origin because of Same Origin Policies (SOPs).

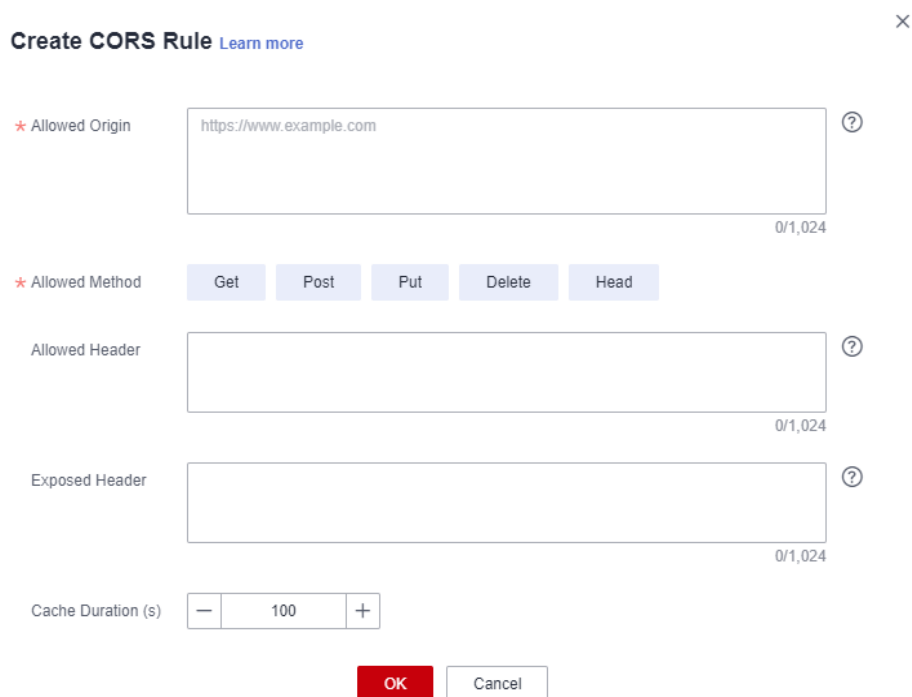
Cross Origin Resource Sharing (CORS) is a browser-standard mechanism. It defines how a web client in one origin interacts with resources in another one. OBS supports CORS rules and allows resources in OBS to be accessed across origins.

- Step 1** In the navigation pane of OBS Console, choose **Object Storage**.
- Step 2** In the bucket list, click the bucket you want to go to the **Objects** page.
- Step 3** In the navigation pane, choose **Permissions** > **CORS Rules**.
- Step 4** Click **Create**. The **Create CORS Rule** dialog box is displayed. See [Figure 5-2](#) for details.

 NOTE

A bucket can have a maximum of 100 CORS rules configured.

Figure 5-2 Creating a CORS rule



The image shows the 'Create CORS Rule' dialog box in the OBS Console. The dialog has a title bar with 'Create CORS Rule' and a 'Learn more' link. It contains several input fields and a 'Cache Duration' section. The 'Allowed Origin' field is a text input with the value 'https://www.example.com' and a character count of '0/1,024'. The 'Allowed Method' section has five buttons: 'Get', 'Post', 'Put', 'Delete', and 'Head'. The 'Allowed Header' and 'Exposed Header' fields are text inputs, both with a character count of '0/1,024'. The 'Cache Duration (s)' section has a numeric input with the value '100' and minus/plus buttons. At the bottom are 'OK' and 'Cancel' buttons.

Create CORS Rule [Learn more](#)

★ Allowed Origin 0/1,024

★ Allowed Method

Allowed Header 0/1,024

Exposed Header 0/1,024

Cache Duration (s)

- Step 5** In the **Create CORS Rule** dialog box, configure **Allowed Origin**, **Allowed Method**, **Allowed Header**, **Exposed Header**, and **Cache Duration (s)**.

 NOTE

Table 5-1 CORS rule parameters

Parameter	Description	Recommended Configurations
Allowed Origin	Mandatory. It specifies the origin from which the requests can access the bucket. You can enter multiple origins, with one separated from another using a line break. Each origin can contain one wildcard character (*) at most. An example is as follows: http://rds.example.com https://*.vbs.example.com	*
Allowed Method	Mandatory. It specifies the allowed cross-origin request methods, same as the operation types of buckets and objects. Request methods include Get, Post, Put, Delete, and Head.	Select all of them.
Allowed Header	Optional. It specifies the allowed headers for cross-origin requests. Only CORS requests matching the allowed headers are valid. You can enter multiple allowed headers, with one separated from another using a line break. Each header can contain one wildcard character (*) at most. Spaces, ampersands (&), colons (:), and less-than signs (<) are not allowed.	*

Parameter	Description	Recommended Configurations
Exposed Header	Optional. It specifies the supplemented header in CORS responses, providing additional information for clients. You can enter multiple exposed headers, with one separated from another using a line break. Spaces, wildcard characters (*), ampersands (&), colons (:), and less-than signs (<) are not allowed.	• ETag • x-obs-request-id • x-obs-api • Content-Type • Content-Length • Cache-Control • Content-Disposition • Content-Encoding • Content-Language • Expires • x-obs-id-2 • x-reserved-indicator • x-obs-version-id • x-obs-copy-source-version-id • x-obs-storage-class • x-obs-delete-marker • x-obs-expiration • x-obs-website-redirect-location • x-obs-restore • x-obs-version • x-obs-object-type • x-obs-next-append-position
Cache Duration (s)	Mandatory. It specifies the duration (in seconds) that your browser can cache CORS responses. The default value is 100 .	Configure it based on your service needs.

Step 6 Click **OK**.

Message "The CORS rule created successfully." is displayed. The CORS configuration will take effect within two minutes.

Then, only the addresses specified in **Allowed Origin** can access the OBS bucket over the methods specified in **Allowed Method**. Suppose you need to configure a CORS rule for bucket **testbucket** and you set **Allowed Origin** to **https://www.example.com**, **Allowed Method** to **GET**, **Allowed Header** to *****, **Exposed Header** to **ETag**, and **Cache Duration (s)** to **100**. Then, only GET requests from **https://www.example.com** are allowed to access bucket **testbucket**. In addition, there are no limits put on headers in the requests, the ETag value can be returned

in the response, and the client where the requests are from can cache the CORS response for 100 seconds.

----End

Step 2: Upload data through a browser.

The following describes how to use the BrowserJS SDK to calculate a signature.

A browser-based upload is to upload objects in HTML form to a bucket. The uploaded object cannot exceed 5 GB in size.

You can call **ObsClient.createPostSignatureSync** to generate request parameters for a browser-based upload. Click [post-object-sample](#) to download the sample code for using BrowserJS to perform a browser-based upload. Alternatively, you can perform the following operations to implement a browser-based upload:

Step 1 Call **ObsClient.createPostSignatureSync** to generate request parameters for authentication.

Two request parameters generated:

- **Policy**: corresponding to the **policy** field in the form
- **Signature**: corresponding to the **signature** field in the form

Sample code:

```
// Create an ObsClient instance.
var obsClient = new ObsClient({
  // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and store
  // them in the configuration file or environment variables. In this example, the AK and SK are stored in
  // environment variables for identity authentication. Before running the code in this example, configure
  // environment variables AccessKeyID and SecretAccessKey.
  // Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
  access_key_id: process.env.AccessKeyID,
  secret_access_key: process.env.SecretAccessKey,
  server : 'https://your-endpoint',
  signature : 'obs'
});

// Configure form parameters.
var formParams = {
  // Set the object ACL to public-read.
  'x-obs-acl': obsClient.enums.AclPublicRead,
  // Configure the object's MIME type.
  'content-type': 'text/plain'
};

// Configure the validity period (in seconds) for a browser-based upload request.
var expires = 3600;

var res = obsClient.createPostSignatureSync({Expires:expires, FormParams: formParams});

// Obtain the request parameters.
console.log('\t' + res.Policy);
console.log('\t' + res.Signature);
```

Step 2 Prepare an HTML page.

Sample code for the HTML form is as follows:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
```

```
<form action="http://bucketname.your-endpoint/" method="post" enctype="multipart/form-data">
Object key
<!-- Object name -->
<input type="text" name="key" value="objectname" />
<p>
ACL
<!-- Object ACL -->
<input type="text" name="x-obs-acl" value="public-read" />
<p>
Content-Type
<!-- Object MIME type -->
<input type="text" name="content-type" value="text/plain" />
<p>
<!-- Base64-coded policy -->
<input type="hidden" name="policy" value="*** Provide your policy ***" />
<!-- AK -->
<input type="hidden" name="AccessKeyId" value="*** Provide your access key ***"/>
<!-- Signature string -->
<input type="hidden" name="signature" value="*** Provide your signature ***"/>

<input name="file" type="file" />
<input name="submit" value="Upload" type="submit" />
</form>
</body>
</html>
```

NOTE

- Values of **policy** and **signature** in the HTML form are obtained from the result returned by **ObsClient.createPostSignatureSync**.
- Click [PostDemo](#) to download an HTML form example.

Step 3 Enter the request parameters on the HTML page.

Step 4 Select a file from your local PC and upload it using the form.

----End

6 Uploading Data from Mobile Apps to OBS

6.1 Overview

OBS is widely used as the storage for mobile Android and iOS apps. When accessing OBS from Android or iOS apps, do not directly save [access keys \(AK/SK\)](#). If the access keys are saved, they may be cracked by hackers, and as a result, data stored in the cloud storage may be stolen or even tampered with.

To better protect data and prevent data leakage and unauthorized access after attacks, you can use the following two methods:

- [Using a Temporary Security Credential to Upload Data to OBS](#)
- [Using a Presigned URL to Upload Data to OBS](#)

In method 1, a temporary AK/SK pair is used to avoid leakage. You are advised to use a temporary security credential to directly upload data to OBS.

6.2 Using a Temporary Security Credential to Upload Data to OBS

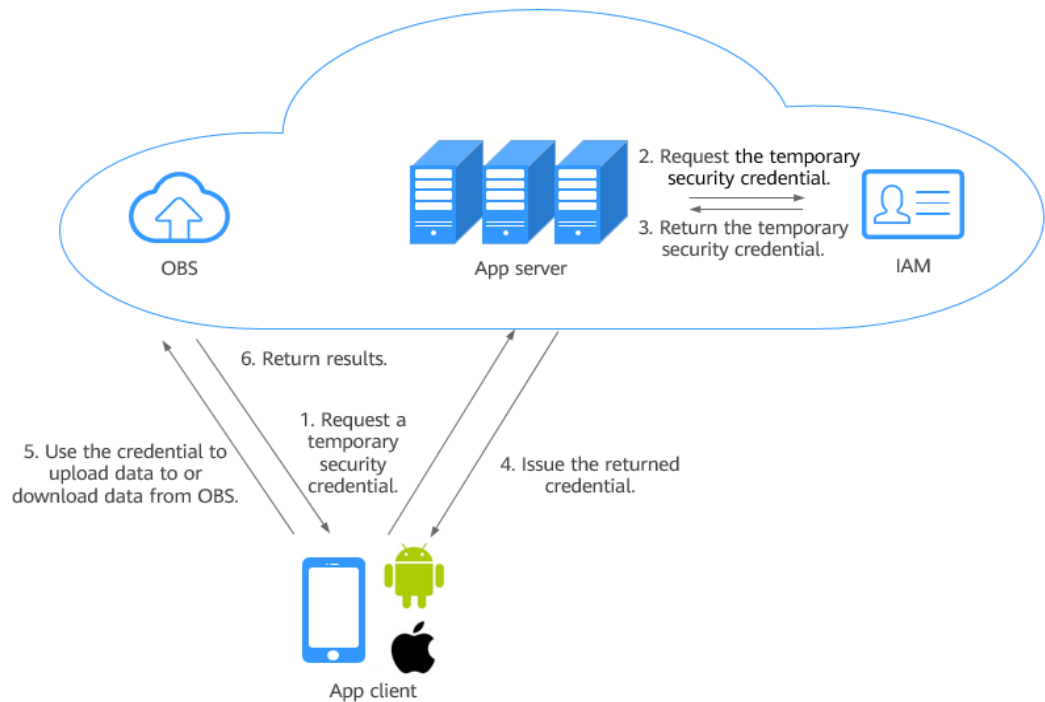
Solution Architecture

Upload data from your apps to OBS or download your data from OBS. [Figure 6-1](#) describes the process.

OBS allows you to use a temporary security credential (temporary AK/SK pair and security token) for access. In addition, you can configure permissions for the credential to specify what actions are allowed during the access with the credential used. To learn more, see [What Are Temporary Access Keys?](#)

Mobile apps can use temporary security credentials with specific permissions configured to directly upload data to OBS. This process does not expose users' permanent access keys, reducing security risks in the case of account leakage.

Figure 6-1 Process of using temporary security credentials to directly upload data to OBS



Role Analysis

- App client: End user's mobile app. It requests a temporary security credential from the server, and uploads data to or downloads data from OBS.
- App server: A backend provided by developers of Android or iOS apps. It manages user accounts and authorization.
- OBS: Huawei Cloud's object storage service. It processes requests from mobile apps.
- IAM: Huawei Cloud's Identity and Access Management. It generates temporary security credentials.

Workflow

1. An app client requests a temporary security credential from the app server.
2. The app server requests the temporary security credential from IAM.
3. IAM returns the credential to the app server.
4. The app server sends the credential to the app client.
5. The app client uses the security credential to upload data to and download data from OBS.

Prerequisites

You have created a bucket and set its access control to private read/write or public read and private write.

For details, see [Creating a Bucket](#) and [Creating a Custom Bucket Policy](#).

Resource and Cost Planning

The table below describes the resources that you need in this practice.

Table 6-1 Resource description

Resource	Description
App client	End user's mobile app. It requests a temporary security credential from the server, and uploads data to or downloads data from OBS.
App server	A backend provided by developers of Android or iOS apps. It manages user accounts and authorization.
OBS	Huawei Cloud's object storage service that processes requests from mobile apps.
IAM	Huawei Cloud's identity and access management service that generates temporary security credentials.

Procedure

Step 1 Obtain the OBS SDK and IAM SDK.

Step 2 Simulate the app server to request a temporary security credential from IAM.

The process is as follows:

1. Obtain the user's IAM token.
For details, see [Obtaining a User Token Through Password Authentication](#).
2. Use a token to obtain a temporary security credential (temporary AK/SK pair and security token). You need to use the **Policy** field to specify what actions are allowed by the security credential.
For details, see [Obtaining Temporary Access Keys and SecurityToken Through a Token](#).

Example: Obtain a temporary security credential whose validity period is 900 seconds. This credential allows you to upload data to only the **APPClient/APP-1/** directory of bucket **hi-company**.

```
{
  "auth":{
    "identity":{
      "policy":{
        "Version":"1.1",
        "Statement":[
          {
            "Action":[
              "obs:object:PutObject"
            ],
            "Resource":[
              "obs:*:object:hi-company/APPClient/APP-1/*"
            ],
            "Effect":"Allow"
          }
        ]
      }
    }
  }
}
```

```
    },
    "token":{
        "duration-seconds":900,
        "id":"MIIDKgYJKoZIhvcNAQcCollDgzCCA38CAQExDTALMEXXXXX..."
    },
    "methods":[
        "token"
    ]
}
}
```

Step 3 Initialize the ObsClient.

Initialization examples:

- **Android**

```
String endPoint = "https://your-endpoint";
// Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and
// store them in the configuration file or environment variables. In this example, the AK and SK are
// stored in environment variables for identity authentication. Before running the code in this example,
// configure environment variables ACCESS_KEY_ID and SECRET_ACCESS_KEY_ID.
// Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String token = System.getenv("Security-Token");

// Create an ObsConfiguration instance.
ObsConfiguration config = new ObsConfiguration();
config.setEndPoint(endPoint);
config.setSocketTimeout(30000);
config.setConnectionTimeout(10000);

// Create an ObsClient instance.
ObsClient obsClient = new ObsClient(ak, sk, token, config);

// Use the instance to access OBS.

// Close ObsClient.
obsClient.close();
```

NOTE

- **endPoint** indicates an endpoint.
- **ak** and **sk** indicate the temporary AK and SK, and **token** indicates the security token. For details about how to obtain them, see [Access Keys \(AK/SK\)](#).

- **iOS**

```
NSString *endPoint = @"your-endpoint";
// Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and
// store them in the configuration file or environment variables. In this example, the AK and SK are
// stored in environment variables for identity authentication. Before running the code in this example,
// configure environment variables AccessKeyID and SecretAccessKey.
// Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
NSString *SK = getenv("AccessKeyID");
NSString *AK = getenv("SecretAccessKey");
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
securityTokenCredentialProvider.securityToken = @""; // Provide your Security Token
// Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];
// Perform initialization.
clientOBSClient *client = [[OBSClient alloc] initWithConfiguration:conf];
```

 NOTE

- **endPoint** indicates an endpoint.
 - **ak** and **sk** indicate the temporary AK and SK, and **token** indicates the security token. For details about how to obtain them, see [Access Keys \(AK/SK\)](#).
- web.js

```
// AMD is not imported. Use the constructor to create an ObsClient instance.
var obsClient = new ObsClient({
    // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK
    and store them in the configuration file or environment variables. In this example, the AK and SK are
    stored in environment variables for identity authentication. Before running the code in this example,
    configure environment variables AccessKeyID and SecretAccessKey.
    // Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
    access_key_id: process.env.AccessKeyID,
    secret_access_key: process.env.SecretAccessKey,
    security_token: process.env.SecurityToken,
    server : 'https://your-endpoint'
});
// Use the instance to access OBS.

// AMD is imported. Use the injected constructor to create an ObsClient instance.
var obsClient;
define(['ObsClient'], function(ObsClient){
    obsClient = new ObsClient({
        // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK
        and store them in the configuration file or environment variables. In this example, the AK and SK are
        stored in environment variables for identity authentication. Before running the code in this example,
        configure environment variables AccessKeyID and SecretAccessKey.
        // Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
        access_key_id: process.env.AccessKeyID,
        secret_access_key: process.env.SecretAccessKey,
        security_token: process.env.SecurityToken,
        server : 'https://your-endpoint'
    });
    // Use the instance to access OBS.
});
```

 NOTE

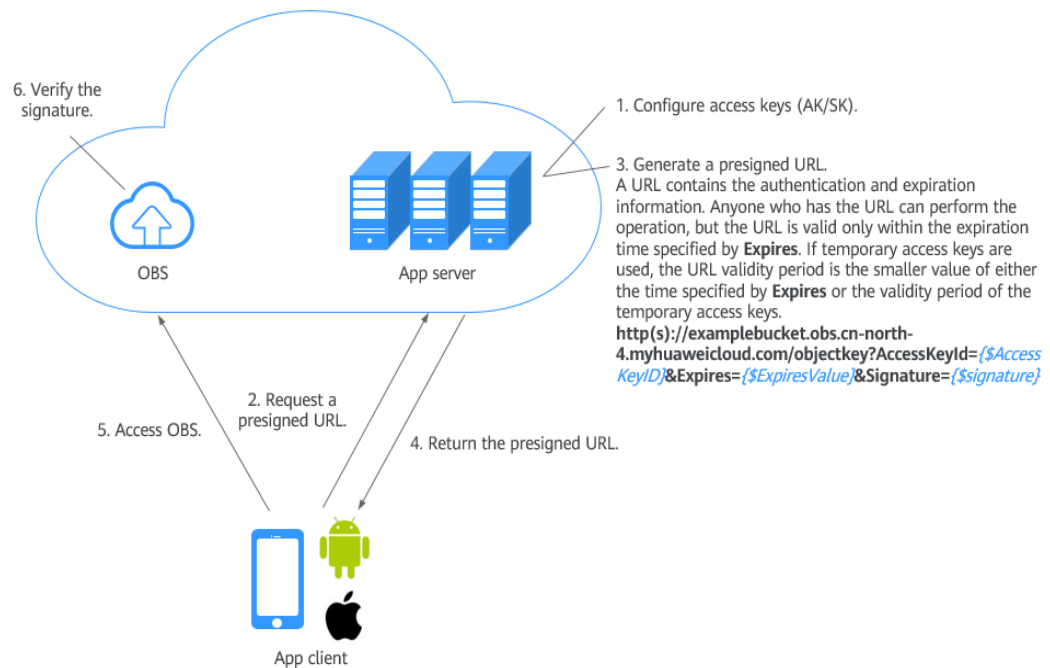
- **endPoint** indicates an endpoint.
 - **ak** and **sk** indicate the temporary AK and SK, and **token** indicates the security token. For details about how to obtain them, see [Access Keys \(AK/SK\)](#).

----End

6.3 Using a Presigned URL to Upload Data to OBS

Solution Architecture

Each request initiated by an app client applies for a presigned URL from the app server. The validity period of the presigned URL is determined by the app server. [Figure 6-2](#) describes the process.

Figure 6-2 Process for a mobile app to access data in OBS

Role Analysis

- App client: End user's mobile app. It requests a presigned URL from the app server, and uploads data to or downloads data from OBS.
- App server: A backend provided by developers of Android or iOS apps. It manages the credential information and issues presigned URLs.
- OBS: Huawei Cloud's object storage service. It processes requests from mobile apps.

Workflow

1. An app client requests a presigned URL from the app server.
Access keys (AK and SK) are not required for accessing OBS from Android or iOS apps. But a presigned URL must be obtained from the app server before accessing OBS, and required information must be carried in the URL, including the request type, resource path, and resource name. For example, an upload request needs to indicate that the URL is for uploading data. In the URL, the upload path and object name are specified. Similarly, a URL for downloading data should contain the name of the object to be downloaded.
2. As a trusted device, the application server stores access keys (AK and SK). After verifying that the client is valid, the app server generates a presigned URL using the stored access keys (AK and SK), in accordance with the operation type and resources to be accessed by the client.
3. Android/iOS mobile apps obtain the URL and use the URL to perform desired operations, such as uploading and downloading data.
The URL contains the access key ID (AK) of the user, signature, validity period, and resource information. Anyone who has the URL can perform the operation. After receiving the request and verifying the signature, OBS deems that the request is executed by the user who issues the URL. For example, you

can construct an object download URL with signature information, but the URL is valid only within the expiration time specified by **Expires**. If temporary access keys are used, the URL validity period is the smaller value of either the expiration time specified by **Expires** or the validity period of the temporary access keys. The URL that carries the signature is used to allow others to use the presigned URL for identity authentication when the SK is not provided, and perform the predefined operation.

Prerequisites

- A bucket has been created.
Create a bucket on OBS Console. Configure the bucket permissions, and allow it to be read/written privately, read publicly, or written privately.
For details, see [Creating a Bucket](#) and [Configuring a Custom Bucket Policy](#).
- Access keys (AK and SK) have been obtained.
The presigned URL is generated using the access keys. For details about how to obtain access keys, see [Access Keys \(AK/SK\)](#). The user who uses the access keys (AK/SK) needs to have the minimum required permissions. For details about how to authorize the permissions, see [Creating a User and Granting OBS Permissions](#).

Resource and Cost Planning

The table below describes the resources that you need in this practice.

Table 6-2 Resource description

Resource	Description
App client	End user's mobile app. It requests a presigned URL from the app server, and uploads data to or downloads data from OBS.
App server	A backend provided by developers of Android or iOS apps. It manages the credential information and issues presigned URLs.
OBS	Huawei Cloud's object storage service that processes requests from mobile apps.

Procedure

Step 1 Configure an app server.

1. Obtain the SDK.
2. Generate the code for issuing a presigned URL.
For details, see [Authentication of Signature in a URL](#).

The following example describes how to use Java for development on the app server.

 NOTE

The application server needs to identify the common request header and user-defined request header based on the operation type initiated by the app, and add the headers to the presigned URL for computing the signature.

- For details about common request headers, see [Constructing a Request](#).
- For details about user-defined request headers, see the corresponding operation in the *API Reference*. For example, for PUT uploads, see [Uploading Objects - PUT](#).

```
// Endpoint of the requested bucket
String endPoint = "http://your-endpoint";

// Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and
// store them in the configuration file or environment variables. In this example, the AK and SK are
// stored in environment variables for identity authentication. Before running the code in this example,
// configure environment variables ACCESS_KEY_ID and SECRET_ACCESS_KEY_ID.
// Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");

//Create an ObsClient.
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
//Define the expiration time, in seconds.
long expireSeconds = 3600L;

//Specify the requested operation.
TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.PUT,
    expireSeconds);

//Specify the bucket name and object name involved in this operation.
request.setBucketName("bucketname");
request.setObjectKey("objectname");

TemporarySignatureResponse response = obsClient.createTemporarySignature(request);

//If the following message is returned, the presigned URL is successfully issued, and you can print the
//URL information.
System.out.println(response.getSignedUrl());
```

Step 2 Use the presigned URL to initiate an OBS access request.

```
public class Demo extends Activity
{
    private static String bucketName = "my-obs-bucket-demo";
    private static String objectKey = "my-obs-object-key-demo";
    private static OkHttpClient httpClient;
    private static StringBuffer sb;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sb = new StringBuffer();
        /*
        * Constructs a client instance with your account for accessing OBS
        */
        httpClient = new OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
            .cache(null).build();
        final TextView tv = (TextView)findViewById(R.id.tv);
        tv.setText("Click to start test");
        tv.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                tv.setClickable(false);
                AsyncTask<Void, Void, String> task = new DownloadTask();
```

```
        task.execute();
    }
    });
}

class DownloadTask extends AsyncTask<Void, Void, String>
{
    @Override
    protected String doInBackground(Void... params)
    {
        try
        {
            /*
             * You need to construct an object upload request and send it to the application server to
generate a presigned URL for accessing OBS.
             * If the response result is stored in response, obtain the URL using the getSignedUrl() method.
             */
            sb.append("Uploading a new object to OBS from a file\n\n");
            Request.Builder builder = new Request.Builder();
//Make a PUT request to upload an object.
            Request httpRequest =
builder.url(response.getSignedUrl()).put(RequestBody.create(MediaType.parse(contentType), "Hello
OBS".getBytes("UTF-8"))).build();
            Call c = httpClient.newCall(httpRequest);
            Response res = c.execute();
            sb.append("\tStatus:" + res.code());
            if (res.body() != null) {
                sb.append("\tContent:" + res.body().string() + "\n");
            }
            res.close();

            /*
             * You need to construct an object download request and send it to the application server to
generate a presigned URL for accessing OBS.
             * If the response result is stored in response, obtain the URL using the getSignedUrl() method.
             */
            sb.append("Downloading an object\n\n");
            Request.Builder builder = new Request.Builder();
//Make a GET request to download an object.
            Request httpRequest = builder.url(response.getSignedUrl()).get().build();
            OkHttpClient httpClient = new
OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false).cache(null).build();
            Call c = httpClient.newCall(httpRequest);
            Response res = c.execute();
            System.out.println("\tStatus:" + res.code());
            if (res.body() != null) {
                sb.append("\tContent:" + res.body().string() + "\n");
            }
            res.close();

            return sb.toString();
        }
        catch (Exception e)
        {
            sb.append("\n\n");
            sb.append(e.getMessage());
            return sb.toString();
        }
        finally
        {
            if (httpClient != null)
            {
                try
                {
                    /*
                     * Close obs client
                     */
                    httpClient.close();
                }
            }
        }
    }
}
```

```
        catch (IOException e)
        {
        }
    }
}

@Override
protected void onPostExecute(String result)
{
    TextView tv = (TextView)findViewById(R.id.tv);
    tv.setText(result);
    tv.setOnClickListener(null);
    tv.setMovementMethod(ScrollingMovementMethod.getInstance());
}
}
```

----End

7 Uploading Data from Mini Programs to OBS

Context

Mini programs are now popular in a variety of scenarios. Uploading files to OBS through a mini program becomes a hot topic. In this section, an example is provided to demonstrate how to realize this.

Precautions

- Signature calculation of clients depends on open-source components **crypto-js** and **js-base64**, so an NPM module needs to be configured in the mini program.
- During the mini program compilation, if error message "Maximum call stack size exceed" is reported when the **crypto-js** package is imported, upgrade the program to the latest version.
- If 405 is returned during the upload, check whether the specified endpoint is the domain name of the bucket for storing uploaded files.

Procedure

Step 1 Enable CORS for a bucket.

Mini programs are developed based on BrowerJS. Due to the same-origin policy, CORS rules must be configured if website scripts and content in one origin need to interact with those in another one. OBS supports CORS that allows resources to be accessed across origins. For detailed configurations, see [Configuring CORS](#).

The following table describes the suggestions on configuring CORS rules.

Table 7-1 CORS rule parameters

Parameter	Description	Recommended Configurations
Allowed Origin	Mandatory. It specifies the origin from which the requests can access the bucket. You can enter multiple origins, with one separated from another using a line break. Each origin can contain one wildcard character (*) at most. An example is as follows: http://rds.example.com https://*.vbs.example.com	*
Allowed Method	Mandatory. It specifies the allowed cross-origin request methods, same as the operation types of buckets and objects. Request methods include Get, Post, Put, Delete, and Head.	Select all of them.
Allowed Header	Optional. It specifies the allowed headers for cross-origin requests. Only CORS requests matching the allowed headers are valid. You can enter multiple allowed headers, with one separated from another using a line break. Each header can contain one wildcard character (*) at most. Spaces, ampersands (&), colons (:), and less-than signs (<) are not allowed.	*

Parameter	Description	Recommended Configurations
Exposed Header	Optional. It specifies the supplemented header in CORS responses, providing additional information for clients. You can enter multiple exposed headers, with one separated from another using a line break. Spaces, wildcard characters (*), ampersands (&), colons (:), and less-than signs (<) are not allowed.	• ETag • x-obs-request-id • x-obs-api • Content-Type • Content-Length • Cache-Control • Content-Disposition • Content-Encoding • Content-Language • Expires • x-obs-id-2 • x-reserved-indicator • x-obs-version-id • x-obs-copy-source-version-id • x-obs-storage-class • x-obs-delete-marker • x-obs-expiration • x-obs-website-redirect-location • x-obs-restore • x-obs-version • x-obs-object-type • x-obs-next-append-position
Cache Duration (s)	Mandatory. It specifies the duration (in seconds) that your browser can cache CORS responses. The default value is 100 .	Configure it based on your service needs.

Step 2 Add the bucket domain name to the whitelist of the mini program.

Mini programs use a whitelist to manage cross-origin access. To implement data upload, you need to add the domain name for accessing the bucket to the whitelist of the mini program.

1. Obtain the bucket's access domain name.

In the bucket list, click the bucket you want to go to the **Objects** page. In the navigation pane, click **Overview**. In the Basic Information area, view the access domain name of the bucket.

Basic Information

Bucket Name	st
Storage Class	Standard
Bucket Version	3.0
Region	in
Used Capacity ?	0 byte
Objects ?	0
Account ID	e 2b
Created	Mar 17, 2023 15:14:02 GMT+08:00
Versioning ?	Enabled Edit
Endpoint ?	o u
Access Domain Name ?	ei eu
Enterprise Project	default
Storage Limitation	Unlimited Edit

2. Set the bucket domain name to be valid in the server domain name configuration of the mini program.

Step 3 Calculate the POST upload signature.

Before uploading a file using POST, calculate the signature based on the custom field **policy**. For details about signature calculation, see [Authentication of Signature Carried in the Table Uploaded Through a Browser](#). The following provides the related source code.

Base64 encoding on the policy (**GetPolicy.js**):

```
const Base64 = require('js-base64');

function getPolicyEncode(policy) {
  // Pass the policy field uploaded through a form and perform Base64 encoding on it.
  const encodedPolicy = Base64.encode(JSON.stringify(policy));
  return encodedPolicy;
}

module.exports = getPolicyEncode;
```

Source code for calculating the signature (**GetSignature.js**):

```
const Crypto = require('crypto-js');
const Base64 = require('js-base64');
```

```
function getSignature(policyEncoded, SecretKey){
  // Use the SK to perform HMAC-SHA1 signature calculation on the Base64-encoded policy.
  const bytes = Crypto.HmacSHA1(policyEncoded, SecretKey);
  // Perform Base64 encoding on the calculation result to obtain the final signature.
  const signature = Crypto.enc.Base64.stringify(bytes);
  return signature;
}

module.exports = getSignature;
```

Step 4 Use the mini program to transfer data to the OBS bucket.

Based on the encoded **policy** and **signature** obtained in [Step 3](#), call the upload API in the mini program to upload your local files. The sample code is as follows.

Configuration file (**Configuration.js**) for configuring the AK, SK, and access domain name:

- Use permanent access keys (AK/SK).

```
// Specify the AK, SK, and endpoint for OBS.
var Configuration = {
  // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and
  // store them in the configuration file or environment variables. In this example, the AK and SK are
  // stored in environment variables for identity authentication. Before running the code in this example,
  // configure environment variables AccessKeyId and SecretKey.
  // The front-end code does not have the process environment variable, so you need to use a module
  // bundler like webpack to define the process variable.
  // Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
  AccessKeyId: process.env.AccessKeyId,
  SecretKey: process.env.SecretAccessKey,
  EndPoint: 'https://your-test-bucket.obs.myhuaweicloud.eu', //Full bucket access domain name
};

module.exports = Configuration;
```

- Use temporary access keys (AK/SK and security token).

For details about how to obtain a temporary AK/SK pair and security token, see [Obtaining a Temporary AK/SK Pair and Security Token](#).

```
// Specify the AK, SK, security token, and endpoint for OBS.
var Configuration = {
  // Hard-coded or plaintext AK and SK are risky. For security purposes, encrypt your AK and SK and
  // store them in the configuration file or environment variables. In this example, the AK and SK are
  // stored in environment variables for identity authentication. Before running the code in this example,
  // configure environment variables AccessKeyId and SecretKey.
  // The front-end code does not have the process environment variable, so you need to use a module
  // bundler like webpack to define the process variable.
  // Obtain an AK and SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca\_01\_0003.html.
  AccessKeyId: process.env.AccessKeyId,
  SecretKey: process.env.SecretAccessKey,
  SecurityToken: process.env.SecurityToken, //securityToken
  EndPoint: 'https://your-test-bucket.obs.myhuaweicloud.eu', //Full bucket access domain name
};

module.exports = Configuration;
```

NOTE

The endpoint passed into the configuration file must be a full access domain name, for example, <https://bucketName.obs.myhuaweicloud.eu>, where *bucketName* indicates the name of the bucket which the data is uploaded to through the mini program.

```
// Import the configuration file.
const config = require('./Configuration.js');
```

```
// Import the calculation method for policy code.
const getPolicyEncode = require('./getPolicy.js');
// Import the signature calculation method.
const getSignature = require('./GetSignature.js');

const OBSupload = function (filePath){
  if(!filePath){
    wx.showToast({
      title: 'Invalid filePath',
      icon: 'Please re-select path',
    });
  }
  else{
    const fileName = 'testMiniprogram.jpg'; // Specify the name of the object to be uploaded to the bucket.

    const OBSPolicy = { // Configure the policy content. For details, see the hyperlink in Step 3.
      "expiration": "2021-12-31T12:00:00.000Z",
      "conditions": [
        { "bucket": "your-test-bucket"}, // Keep the bucket name same as that in the endpoint in the
        configuration file.
        // { "x-obs-security-token": config.SecurityToken } // This parameter is mandatory for authentication
        with temporary access keys.
        { 'key': fileName }
      ]
    }

    const policyEncoded = getPolicyEncode(OBSPolicy); // Calculate the Base64-encoded policy.
    const signature = getSignature(policyEncoded, config.SecretKey); // Calculate the signature.

    wx.uploadFile({
      url: config.EndPoint,
      filePath: filePath,
      name: 'file',
      header: {
        'content-type': 'multipart/form-data; boundary=-9431149156168',
      },
      formData: {
        // Obtain the AK and encoded policy and signature from the configuration file.
        'AccessKeyId': config.AccessKeyId,
        'policy': policyEncoded,
        'signature': signature,
        'key': fileName,
        // "x-obs-security-token": config.SecurityToken, // This parameter is mandatory for authentication with
        temporary access keys.
      },
      success: function(res){
        console.log(res.statusCode); //Print the response status code.
        if(res.statusCode==='204'){
          console.log('Uploaded successfully', res)
          wx.showToast({
            title: 'Uploaded successfully',
            icon: 'Success'
          });
        }
        else{
          console.log('Uploaded failed', res)
          wx.showToast({
            title: 'Uploaded failed',
            icon: 'Fail'
          });
        }
      },
      fail: function(e){
        console.log(e);
      }
    })
  }
}
```

```
}  
module.exports = OBSupload;
```

----End

Related Operations

After the upload is complete, obtain the URL for accessing the object. For details, see [How Do I Obtain the Access Path to an Object?](#)

8 Accessing OBS Through an NGINX Reverse Proxy

Application Scenario

Generally, you can access OBS using a bucket's access domain name (for example, <https://bucketname.obs.eu-west-101.myhuaweicloud.eu>) provided by OBS or using a user-defined domain name bound to an OBS bucket.

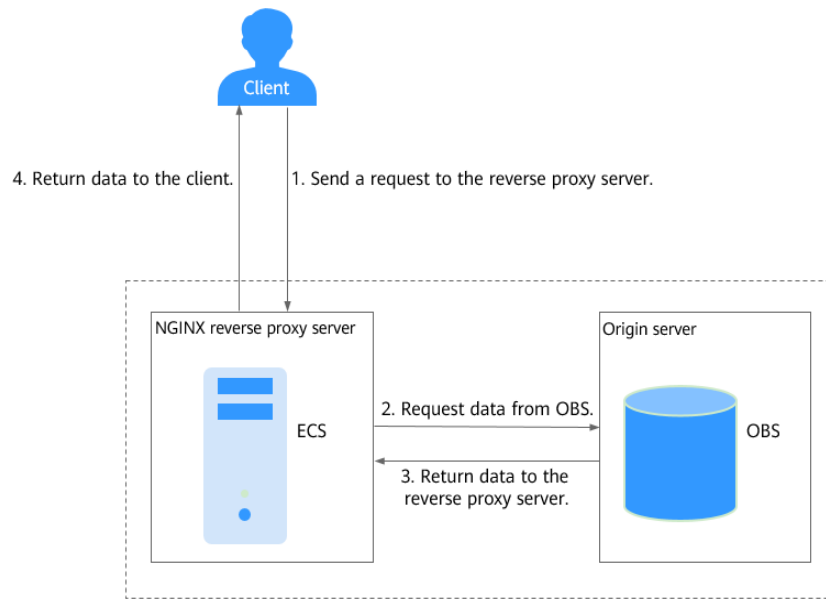
In some cases, you may need to use a fixed IP address to access OBS. For security purposes, some enterprises need to set a blacklist and a whitelist of external IP addresses. In this case, a fixed IP address is required. Also for security purposes, an OBS bucket does not have a fixed IP address, because the DNS service of Huawei Cloud resolves the bucket access domain name to different IP addresses.

In this case, you can set up an NGINX reverse proxy server on an ECS so that you can access OBS through a fixed IP address.

Solution Architecture

This part explains how to deploy NGINX on an ECS and set up an NGINX reverse proxy server. The proxy is imperceptible. Requests are sent to the reverse proxy server, which then obtains the required data from OBS and returns the data to users. The reverse proxy server and OBS work as a whole. Only the IP address of the proxy server is exposed, while the actual domain name or IP address of OBS is hidden.

Figure 8-1 Principles of accessing OBS through an NGINX reverse proxy



Prerequisites

- You have known the region and access domain name of the bucket. For example, the access domain name of a bucket in the EU-Dublin region is **nginx-obs.obs.eu-west-101.myhuaweicloud.eu**. To obtain the information, see [Querying Basic Information of a Bucket](#).
- You have a Linux ECS in the same region. CentOS is used here as an example. For details, see [Purchasing an ECS](#).
- The ECS is bound with an EIP, so that you can download the NGINX installation package over the public network.

Procedure

Step 1 Install NGINX on an ECS.

In this example, CentOS 7.6 is used as an example.

1. Log in to the ECS where you will set up the NGINX reverse proxy server.
2. Run the **wget** command to download the NGINX installation package for your operating system in use.
`wget http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7ngx.noarch.rpm`
3. Run the following command to create the NGINX yum repository:
`rpm -ivh nginx-release-centos-7-0.el7ngx.noarch.rpm`
4. Run the following command to install NGINX:
`yum -y install nginx`
5. Run the following commands to start NGINX and configure NGINX to start upon system startup:
`systemctl start nginx`
`systemctl enable nginx`
6. Use a browser on any device to access **http://ECS EIP**. If the following information is displayed, NGINX is successfully installed.

Figure 8-2 NGINX installed successfully

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Step 2 Modify the NGINX configuration file to configure the reverse proxy for your OBS bucket.

1. Run the following command to open the **default.conf** file:
vim /etc/nginx/conf.d/default.conf
2. Press the **i** key to go to the edit mode and modify the **default.conf** file.

```
server {  
    listen      80;  
    server_name *.*.*.*, # Enter the EIP of the ECS.  
    proxy_buffering off; # Disable the proxy buffer (memory).  
  
    location / {  
        proxy_pass https://nginx-obs.obs.eu-west-101.myhuaweicloud.eu; #Enter the OBS bucket  
domain name that starts with http:// or https://.  
        index index.html index.htm ; #Specify the homepage of the website. If there are multiple  
files, Nginx checks the files based on their enumeration sequence.  
    }  
}
```

Table 8-1 Parameters in the configuration file

Parameter	Description
server_name	IP address that provides the reverse proxy service. It is the fixed IP address that is exposed to end users for access. Enter the EIP of the ECS where the NGINX reverse proxy service is deployed.
proxy_pass	IP address of the proxied server. Enter the OBS bucket access domain name required in Prerequisites . The domain name must start with http:// or https:// . Example: https://nginx-obs.obs.eu-west-101.myhuaweicloud.eu Note: When you use an API, SDK, or obsutil for calling, set this parameter to the region domain name. The following is an example: obs.eu-west-101.myhuaweicloud.eu

Parameter	Description
proxy_buffering	Whether to enable the proxy buffer. The value can be on or off. If this parameter is set to on, Nginx stores the response returned by the backend in a buffer and then sends the data to the client. If this parameter is set to off, Nginx sends the response to the client as soon as it receives the data from the backend. Default value: on Example: proxy_buffering off

3. Press the **Esc** key and enter **:wq** to save the configuration and exit.
4. Run the following command to check the status of the NGINX configuration file:

```
nginx -t
```
5. Run the following command to restart the NGINX service for the configuration to take effect:

```
systemctl stop nginx  
systemctl start nginx
```

Step 3 (Optional) Configure an OBS bucket policy to allow the IP address of the NGINX proxy server to access OBS.

If your bucket is publicly read or the URL needs to have a signature contained when accessing objects in a private bucket, skip this step. For details, see [Authentication of Signature in a URL](#).

If you do not want URLs containing a signature to access resources in your private bucket, configure the following bucket policy that allows only the IP address of the NGINX proxy server to access your bucket.

1. In the navigation pane of OBS Console, choose **Object Storage**.
2. In the bucket list, click the bucket you want to go to the **Objects** page.
3. In the navigation pane, choose **Permissions > Bucket Policies**.
4. Click **Create**.
5. In the first row of the bucket policy template, click **Create Custom Policy**.
6. Configure the following parameters.

Table 8-2 Bucket policy parameters

Parameter		Description
Policy View		Select Visual editor .
Policy Name		Custom
Policy Content	Effect	Select Allow .
	Principal	– Principal: Select Anonymous user. – User Policy: Select Include specified users.

Parameter		Description
	Resource	– Resource: Select both Current bucket and Objects in bucket. – Resource Policy: Select Include specified resources.
	Actions	– Select Get* and List*. – Operation Strategy: Select Include selected.
	Conditions	– Conditional Operator: Select IpAddress. – Key: Select SourceIp. – Value: ▪ If the ECS uses a public DNS, the value is as follows: <i>Elastic IP address of the ECS</i> ▪ If the ECS uses a Huawei Cloud private DNS, the value is as follows: 100.64.0.0/10,214.0.0.0/7, <i>Private IP address of the ECS</i> NOTE The value must contain three IP addresses (CIDR blocks) that are separated with commas (,). IP addresses in the range starting with 100 or 214 are for ECSs to access OBS through an internal network.

7. Click **Next** in the lower right corner.

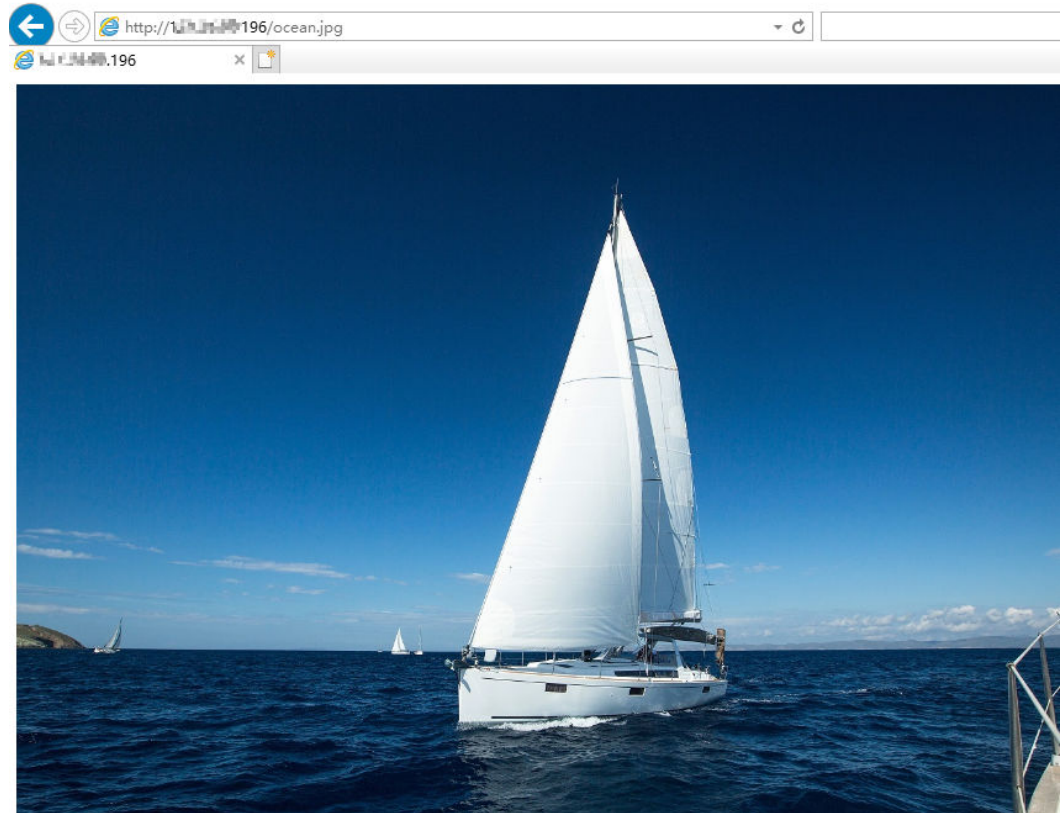
8. Click **Create**.

Step 4 Verify the reverse proxy configuration.

On any device, use the ECS EIP and object name to access specified OBS resources. If the resources are properly accessed, the configuration is successful.

For example, visit **<http://ECS EIP/ocean.jpg>**.

Figure 8-3 Using a fixed IP address to access OBS resources



----End

9 Using OBS to Decouple Storage from Compute in Big Data Scenarios

9.1 Overview

Application Scenario

As big data technologies burgeon, people are deepening their understanding of data values. Big data is everywhere in a variety of industries. According to a report, of all enterprises around the world, over 39.6% have applied big data to their businesses and earned benefits, more than 89.6% already have or plan to set up departments for big data analysis, and over 60% are investing more in big data. The capability of leveraging big data is crucial to each industry's success in the future.

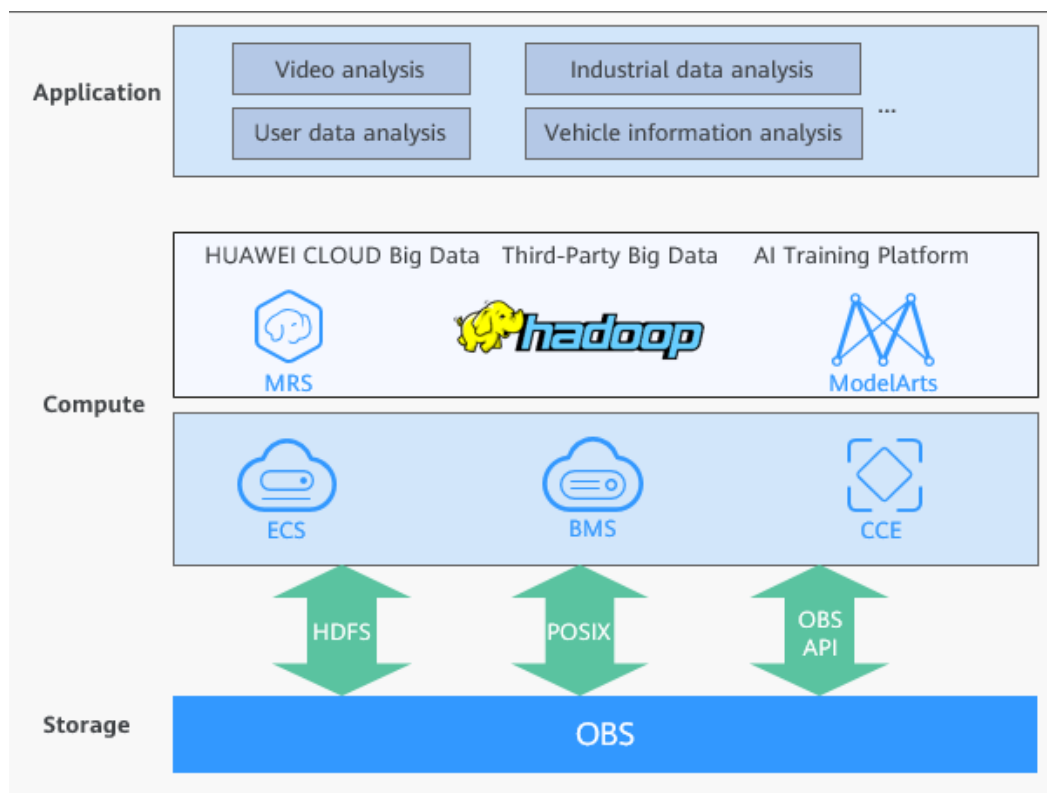
In big data scenarios, data is a new asset, and intelligence has become a new productivity. Enterprises are in urgent need of digital transformation to improve productivity and to maximize the data value. Before services are migrated to the cloud, traditional enterprises deploy their services and store data in multiple clusters in the on-premises IDC, and one server provides both compute and storage capabilities. This causes key problems shown in [Table 9-1](#), and these problems have hindered the enterprise's digital transformation.

Table 9-1 Key concerns faced by traditional enterprises in big data scenarios

No.	Key Concern	Description
1	Hard to share data among multiple clusters	Enterprise's data is stored in multiple clusters, resulting in the following problems: • There is no global view. Data in one cluster cannot be used in another, unless data is copied. • Copying data is the only way to share data across clusters, which takes a long time. • Public data set copies are stored in multiple clusters, leaving data redundant.
2	Resource waste due to coupled compute and storage resources	Compute and storage resources must be expanded proportionally even if their demands are inconsistent, which causes a waste of resources.
3	Low utilization and high cost due to three copies of data	The Hadoop Distributed File System (HDFS) stores data in three copies. The disk space utilization is only 33%, and the utilization of a single disk is lower than 70%.

Solution Architecture

To address the problems in the table above, Huawei Cloud provides a solution with decoupled storage and compute, where OBS is used as the unified data lake storage.

Figure 9-1 OBS-based big data solution with decoupled storage and compute

Relying on the large capacity and high bandwidth of OBS and shared access based on multiple protocols (HDFS, POSIX, and OBS API), this solution enables Hadoop compute engines (such as Hive and Spark) compatible with each other.

Solution Advantages

Compared with traditional solutions, this solution has the advantages described in [Table 9-2](#).

Table 9-2 Advantages

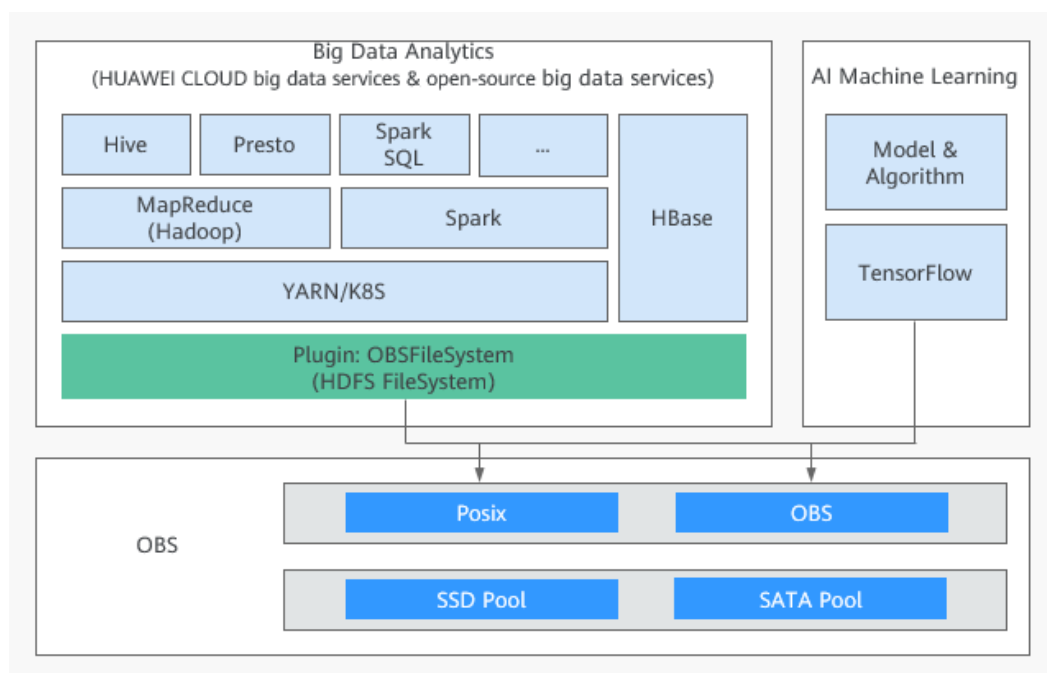
No.	Advantage	Description
1	Converged, efficient, and collaborative analysis	• Data can be shared among multiple clusters through unified permission control. • No data copy is required. • Integration of big data and AI reduces the operation time.
2	High resource utilization thanks to decoupled storage and compute	Compute and storage resources can be separately scaled. This improves the resource utilization.

No.	Advantage	Description
3	High utilization and low cost with EC storage	OBS supports Erasure Code (EC), the most utilized distributed fault tolerance technology. EC greatly increases the disk space utilization and requires much less storage space than the three copies of data mechanism.

In addition, OBS provides the OBSFileSystem plug-in (OBSA-HDFS) to seamlessly connect to the upper-layer big data platform, requiring no modifications.

OBSFileSystem provides HDFS-related APIs so that big data compute engines (such as Hive and Spark) can use OBS as the underlying storage.

Figure 9-2 OBSFileSystem in the solution with decoupled storage and compute



NOTE

OBS offers object storage buckets (object semantics) and parallel file systems (POSIX). In big data scenarios, parallel file systems are recommended. Parallel file systems support POSIX and are encapsulated through OBSFileSystem. Compared with object semantics, parallel file systems have additional APIs (including Rename, Append, hflush, and hsync). These APIs supplement HDFS semantics and provide better performance for big data computing.

Based on the preceding advantages, compared with traditional big data solutions, the Huawei Cloud big data solution with decoupled storage and compute requires significantly fewer compute resources, storage resources, and servers for the same service scale. This greatly increases resource utilization and reduces the total cost of ownership (TCO).

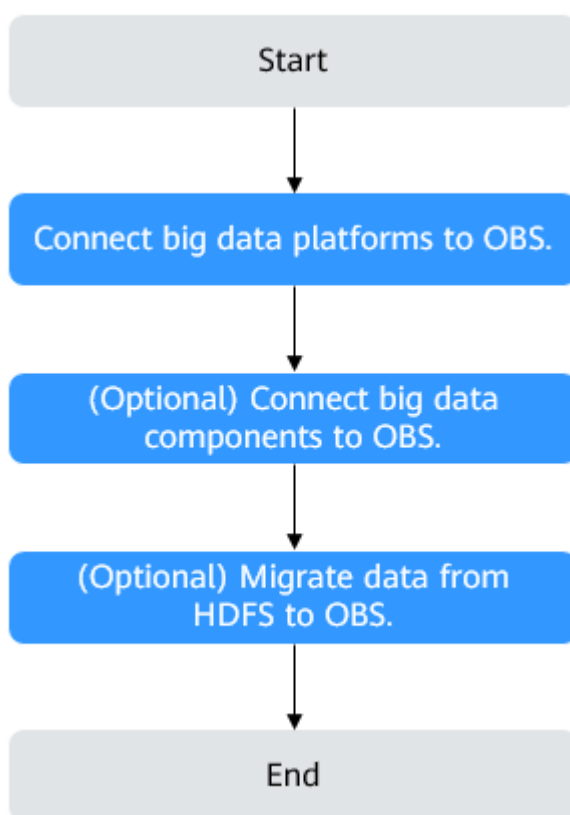
Application Scope

This practice explains how to connect different big data platforms and components to OBS in the big data solutions with decoupled storage and compute, and how to migrate data from HDFS to OBS.

9.2 Process

Figure 9-3 shows the process of using OBS to decouple storage from compute in big data scenarios.

Figure 9-3 Process of using OBS to enable decoupled storage and compute



1. Connect big data platforms to OBS. This step is the core, which makes OBS the unified data lake storage. This connection guide covers three mainstream big data platforms. For details, see [Supported Big Data Platforms](#).
2. (Optional) Connect open-source big data components to OBS. For details, see [Supported Big Data Components](#).
3. (Optional) If your data is still stored in local HDFS, migrate it to OBS on Huawei Cloud. For details, see [Migrating HDFS Data to OBS](#).

9.3 Connecting Big Data Platforms to OBS

9.3.1 Supported Big Data Platforms

In the Huawei Cloud big data solution with decoupled storage and compute, OBS can be connected to Huawei Cloud MapReduce Service (MRS), Cloudera CDH, or Hortonworks HDP, meeting users' different needs.

Huawei Cloud MRS

MRS is a big data service that allows you to deploy Hadoop clusters with one click and manage the deployed clusters with ease.

MRS provides enterprise-class big data clusters on the cloud. Users have full control over their own clusters and can easily run big data components such as Hadoop, Spark, HBase, Kafka, and Storm. Fully compatible with open source APIs, MRS leverages Huawei Cloud's best-in-class compute, storage, and big data services to provide customers with a full-stack big data platform featuring high performance, low cost, flexibility, and ease-of-use. In addition, the platform can be customized based on customer needs to help enterprises quickly build a hyperscale data processing system and discover new value and business opportunities by analyzing and mining massive amounts of data, either in real time or in non-real time.

For details about connecting MRS to OBS, see [Connecting MRS to OBS](#).

Cloudera CDH

CDH is a big data analysis and management platform distribution built on Apache Hadoop.

For details about connecting Cloudera CDH to OBS, see [Connecting Cloudera CDH to OBS](#).

Hortonworks HDP

HDP is a big data analysis and management platform based on open-source Apache Hadoop components.

For details about connecting Hortonworks HDP to OBS, see [Connecting Hortonworks HDP to OBS](#).

9.3.2 Connecting MRS to OBS

Procedure

Step 1 Configure a cluster with decoupled storage and compute.

For details, see [Configuring a Storage-Compute Decoupled Cluster \(Agency\)](#).

Step 2 Use the cluster.

For details, see [Using a Storage-Compute Decoupled Cluster](#).

----End

9.3.3 Connecting Cloudera CDH to OBS

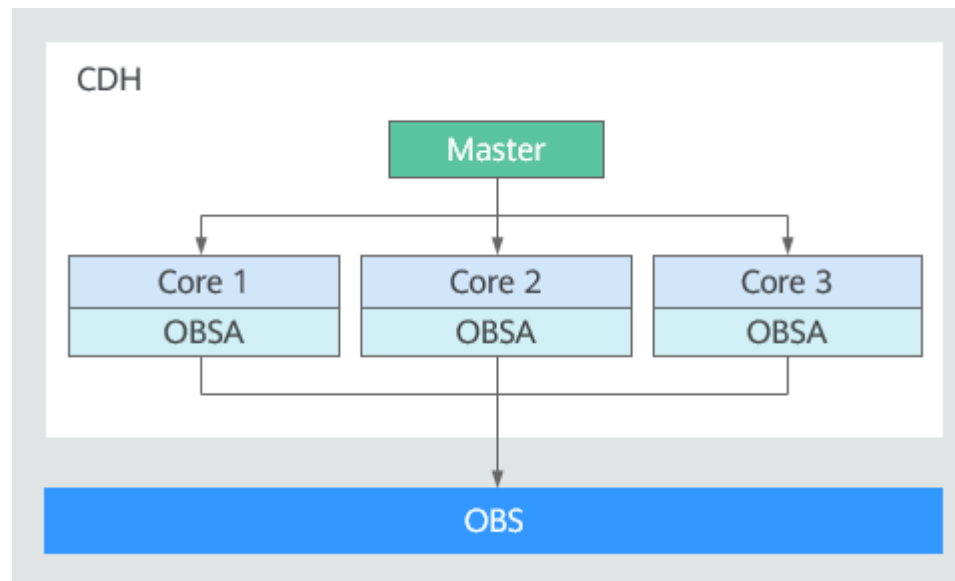
Deployment View

Version Information

Hardware: 1 Master + 3 Cores (flavor: 8U32G; OS: CentOS 7.5)

Software: CDH 6.0.1

Deployment View



Updating OBSA-HDFS

Step 1 Download the **OBSA-HDFS** that matches the Hadoop version.

Upload the OBSA-HDFS JAR package (for example, **hadoop-huaweicloud-3.1.1-hw-53.8.jar**) to the **/opt/obsa-hdfs** directory of each CDH node.

NOTE

- In a **hadoop-huaweicloud-*x.x.x*-hw-*y*.jar** package name, *x.x.x* indicates the Hadoop version number, and *y* indicates the OBSA version number. For example, in **hadoop-huaweicloud-3.1.1-hw-53.8.jar**, **3.1.1** is the Hadoop version number, and **53.8** is the OBSA version number.
- If the Hadoop version is **3.1.x**, select **hadoop-huaweicloud-3.1.1-hw-53.8.jar**.

Step 2 Add the downloaded JAR package of **hadoop-huaweicloud**.

Perform the following operations on each CDH cluster node (replace the JAR package name and CDH version number with the ones actually used).

1. Save the OBSA-HDFS JAR package in the **/opt/cloudera/parcels/CDH-6.0.1-1.cd6.0.1.p0.590678/jars/** directory:
cp /opt/obsa-hdfs/hadoop-huaweicloud-3.1.1-hw-53.8.jar /opt/cloudera/parcels/CDH-6.0.1-1.cd6.0.1.p0.590678/jars/
2. Create a soft link for each directory and save the JAR package to the following directories:

```
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud-3.1.1-hw-53.8.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/cloudera-navigator-server/libs/cdh6/  
hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/common_jars/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/lib/cdh6/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/cloudera-scm-telepub/libs/cdh6/  
hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/lib/hadoop/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/lib/hadoop/client/hadoop-  
huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/lib/spark/jars/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/lib/impala/lib/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/parcels/  
CDH-6.0.1-1.cdh6.0.1.p0.590678/lib/hadoop-mapreduce/hadoop-  
huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/lib/cdh5/hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/cloudera-scm-telepub/libs/cdh5/  
hadoop-huaweicloud.jar  
ln -s /opt/cloudera/parcels/CDH-6.0.1-1.cdh6.0.1.p0.590678/jars/hadoop-  
huaweicloud.jar /opt/cloudera/cm/cloudera-navigator-server/libs/cdh5/  
hadoop-huaweicloud.jar
```

----End

Connecting OBS to HDFS and Yarn Clusters

- Step 1** In the advanced configuration area of the HDFS cluster, configure **fs.obs.access.key**, **fs.obs.secret.key**, **fs.obs.endpoint**, and **fs.obs.impl**, corresponding to the OBS AK, SK, endpoint, and IMPL, in the **core-site.xml**.

 NOTE

1. Enter the actually used AK/SK pair and endpoint. To obtain them, see [Access Keys \(AK/SK\)](#) and [Endpoints and Domain Names](#), respectively.
2. Set `fs.obs.impl` to `org.apache.hadoop.fs.obs.OBSFileSystem`.

Step 2 Restart or roll restart the HDFS cluster, and then restart the client.

Step 3 Go to the YARN cluster and restart the client.

Step 4 Check whether the AK, SK, endpoint, and impl have been configured in file `/etc/hadoop/conf/core-site.xml` on the node.

```
<property>
<name>fs.obs.access.key</name>
<value>****</value>
</property>
<property>
<name>fs.obs.secret.key</name>
<value>*****</value>
</property>
<property>
<name>fs.obs.endpoint</name>
<value>{Target Endpoint}</value>
</property>
<property>
<name>fs.obs.impl</name>
<value>org.apache.hadoop.fs.obs.OBSFileSystem</value>
</property>
```

----End

Connecting OBS to a Spark Cluster

Step 1 Configure related items (including AK, SK, endpoint, and impl) in file `core-site.xml` in the YARN cluster.

Step 2 Restart the YARN cluster and then the Spark cluster client.

----End

Connecting OBS to a Hive Cluster

Step 1 Configure related items (including AK, SK, endpoint, and impl) in file `core-site.xml` in the Hive cluster.

Step 2 Restart the Hive cluster and then the client.

----End

9.3.4 Connecting Hortonworks HDP to OBS

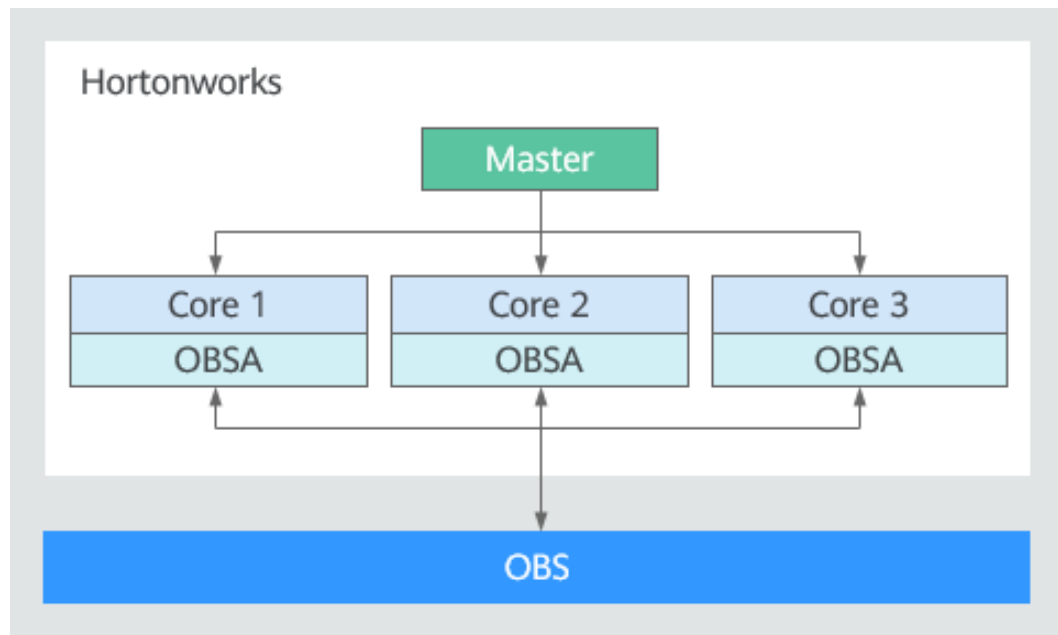
Deployment View

Version Information

Hardware: 1 Master + 3 Cores (flavor: 8U32G; OS: CentOS 7.5)

Software: Ambari 2.7.1.0 and HDP 3.0.1.0

Deployment View



Updating OBSA-HDFS

Step 1 Download the [OBSA-HDFS](#) that matches the Hadoop version.

Download the OBSA-HDFS JAR package (for example, **hadoop-huaweicloud-3.1.1-hw-53.8.jar**) to the **/mnt/obsjar** directory.

NOTE

- In a **hadoop-huaweicloud-*x.x.x*-hw-*y*.jar** package name, *x.x.x* indicates the Hadoop version number, and *y* indicates the OBSA version number. For example, in **hadoop-huaweicloud-3.1.1-hw-53.8.jar**, **3.1.1** is the Hadoop version number, and **53.8** is the OBSA version number.
- If the Hadoop version is **3.1.x**, select **hadoop-huaweicloud-3.1.1-hw-53.8.jar**.

Step 2 Copy the downloaded OBSA-HDFS JAR package to the following directories:

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /usr/hdp/share/hst/activity-explorer/lib/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /usr/hdp/3.0.1.0-187/hadoop-mapreduce/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /usr/hdp/3.0.1.0-187/spark2/jars/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /usr/hdp/3.0.1.0-187/tez/lib/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /var/lib/ambari-server/resources/views/work/CAPACITY-SCHEDULER{1.0.0}/WEB-INF/lib/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /var/lib/ambari-server/resources/views/work/FILES{1.0.0}/WEB-INF/lib/
```

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /var/lib/ambari-server/resources/views/work/WORKFLOW_MANAGER{1.0.0}/WEB-INF/lib/
```

```
ln -s /usr/hdp/3.0.1.0-187/hadoop-mapreduce/hadoop-huaweicloud-3.1.1-  
hw-53.8.jar /usr/hdp/3.0.1.0-187/hadoop-mapreduce/hadoop-huaweicloud.jar  
----End
```

Adding Configuration Items to the HDFS Cluster

Step 1 Add configuration items in file **Custom core-site.xml** to the **ADVANCED** in the HDFS cluster's **CONFIGS**. These items include **fs.obs.access.key**, **fs.obs.secret.key**, **fs.obs.endpoint**, and **fs.obs.impl**.

NOTE

1. **fs.obs.access.key**, **fs.obs.secret.key**, and **fs.obs.endpoint** indicate the AK, SK, and endpoint respectively. Enter the actually used AK/SK pair and endpoint. To obtain them, see [Access Keys \(AK/SK\)](#) and [Endpoints and Domain Names](#), respectively.
2. Set **fs.obs.impl** to **org.apache.hadoop.fs.obs.OBSFileSystem**.

Step 2 Restart the HDFS cluster.

----End

Adding Configuration Items to the MapReduce2 Cluster

Step 1 In the **mapred-site.xml** file under **ADVANCED** in **CONFIGS** of the MapReduce2 cluster, change the value of **mapreduce.application.classpath** to **/usr/hdp/3.0.1.0-187/hadoop-mapreduce/***.

Step 2 Restart the MapReduce2 cluster.

----End

Adding a JAR Package for Connecting Hive to OBS

Step 1 Create the **auxlib** folder on the Hive Server node:

```
mkdir /usr/hdp/3.0.1.0-187/hive/auxlib
```

Step 2 Save the OBSA-HDFS JAR package to the **auxlib** folder:

```
cp /mnt/obsjar/hadoop-huaweicloud-3.1.1-hw-53.8.jar /usr/hdp/3.0.1.0-187/  
hive/auxlib
```

Step 3 Restart the Hive cluster.

----End

9.4 Connecting OBS to Big Data Components

9.4.1 Supported Big Data Components

In the Huawei Cloud big data solution with decoupled storage and compute, OBS can also be directly connected to open-source big data components.

Currently, the big data components that can connect to OBS include:

- [Hadoop](#)
- [Hive](#)
- [Spark](#)
- [Flume](#)
- [DataX](#)
- [Druid](#)
- [Flink](#)
- [Logstash](#)

9.4.2 Connecting Hadoop to OBS

Overview

Hadoop provides a distributed resource scheduling engine for processing and analyzing large-scale data sets. OBS effects the Hadoop HDFS protocol. It can replace HDFS in the Hadoop system to connect to big data components such as Spark, MapReduce, and Hive, serving as the data lake storage for big data computing.

NOTE

The HDFS protocol is defined through the FileSystem abstract class in Hadoop, which can be effected by different storage systems, such as the HDFS service built in Hadoop and Huawei Cloud OBS.

Constraints

The following HDFS semantics are not supported:

- Lease
- Symbolic link operations
- Proxy users
- File concat
- File checksum
- File replication factor
- Extended attributes (Xattrs) operations
- Snapshot operations
- Storage policy
- Quota
- POSIX ACL
- Delegation token operations

Precautions

To reduce output logs, add the following configuration to the **/opt/hadoop-3.1.1/etc/hadoop/log4j.properties** file:

```
log4j.logger.com.obs=ERROR
```

Procedure

Hadoop 3.1.1 is used as an example in the steps below. You are advised to use the latest Hadoop version.

Step 1 Download **hadoop-3.1.1.tar.gz** and decompress it to the **/opt/hadoop-3.1.1** directory.

Step 2 Add the following content to the **/etc/profile** file:

```
export HADOOP_HOME=/opt/hadoop-3.1.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

Step 3 Install **hadoop-huaweicloud**.

1. Download it from [GitHub](#).

NOTE

If no JAR package of the required version is available, modify the Hadoop version in the POM file under the **hadoop-huaweicloud** directory and then compile the file again.

2. Copy the **hadoop-huaweicloud-x.x.x-hw-y.jar** package to the **/opt/hadoop-3.1.1/share/hadoop/tools/lib** and **/opt/hadoop-3.1.1/share/hadoop/common/lib** directories.

NOTE

In a **hadoop-huaweicloud-x.x.x-hw-y.jar** package name, *x.x.x* indicates the Hadoop version number, and *y* indicates the OBSA version number. For example, in the **hadoop-huaweicloud-3.1.1-hw-40.jar** package name, **3.1.1** is the Hadoop version number, and **40** is the OBSA version number.

Step 4 Configure Hadoop.

Add OBS configurations to the **/opt/hadoop-3.1.1/etc/hadoop/core-site.xml** file:

```
<property>
<name>fs.obs.impl</name>
<value>org.apache.hadoop.fs.obs.OBSFileSystem</value>
</property>
<property>
<name>fs.AbstractFileSystem.obs.impl</name>
<value>org.apache.hadoop.fs.obs.OBS</value>
</property>
<property>
<name>fs.obs.access.key</name>
<value>xxx</value>
<description>HuaweiCloud Access Key Id</description>
</property>
<property>
<name>fs.obs.secret.key</name>
<value>xxx</value>
<description>HuaweiCloud Secret Access Key</description>
</property>
<property>
<name>fs.obs.endpoint</name>
<value>xxx</value>
<description>HuaweiCloud Endpoint</description>
</property>
```

Step 5 Check whether the connection is successful.

You can use a CLI or MapReduce program for verification. Examples are provided as follows:

- CLI
hadoop fs -ls obs://obs-bucket/
Command output:

```
-rw-rw-rw- 1 root root 1087 2018-06-11 07:49 obs://obs-bucket/test1  
-rw-rw-rw- 1 root root 1087 2018-06-11 07:49 obs://obs-bucket/test2
```
- MapReduce program
hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar wordcount obs://example-bucket/input/test.txt obs://obs-bucket/output

----End

[Appendix] hadoop-huaweicloud Configurations

Configuration Item	Default Value	Mandatory	Description
fs.obs.impl	org.apache.hadoop.fs.obs.OBSFileSystem	Yes	-
fs.AbstractFileSystem.obs.impl	org.apache.hadoop.fs.obs.OBS	Yes	-
fs.obs.endpoint	N/A	Yes	Endpoint of Huawei Cloud OBS
fs.obs.access.key	N/A	Yes	Huawei Cloud's access key ID (AK) for accessing the corresponding OBS bucket.
fs.obs.secret.key	N/A	Yes	Huawei Cloud's secret access key (SK) for accessing the corresponding OBS bucket.
fs.obs.session.token	N/A	No	Huawei Cloud's security token for accessing the corresponding OBS bucket. This token is required when a temporary AK/SK pair is used.
fs.obs.security.provider	N/A	No	Class for calling the com.obs.services.IObsCredentialsProvider API. This API is used to obtain the credentials for accessing OBS.
fs.obs.connection.ssl.enabled	FALSE	No	Specifies whether to access OBS through HTTPS.
fs.obs.threads.keepalivetime	60	No	Parameter keepAliveTime is used to control the read and write thread pool.

Configuration Item	Default Value	Mandatory	Description
fs.obs.threads.max	20	No	Parameters corePoolSize and maximumPoolSize are used to control the read and write thread pool.
fs.obs.max.total.tasks	20	No	Parameter BlockingQueue is used to control the capacity of the read and write thread pool. Its value is the sum of the values of fs.obs.threads.max and fs.obs.max.total.tasks .
fs.obs.multipart.size	104857600	No	Size of a multipart upload.
fs.obs.fast.upload.buffer	disk	No	Specifies a cache method. All written data is cached and then uploaded to OBS. The options are as follows: • disk: Data is cached on a disk. • array: Data is cached in the JVM on-heap memory. • bytebuffer: Data is cached in the JVM off-heap memory.
fs.obs.buffer.dir	\${hadoop.tmp.dir}	No	Specifies the cache directory when fs.obs.fast.upload.buffer is set to disk . In such cases, multiple directories are supported and separated by commas (,).
fs.obs.bufferdir.verify.enable	FALSE	No	Specifies whether to verify the existence of the cache directory and whether the directory has the write permissions, when fs.obs.fast.upload.buffer is set to disk .
fs.obs.fast.upload.active.blocks	4	No	Specifies the maximum number of caches allowed by each stream operation (the maximum number of thread tasks that can be submitted through a multipart upload thread pool). This limits the maximum cache space (calculated from fs.obs.fast.upload.active.blocks x fs.obs.multipart.size) that can be used by each stream operation.

Configuration Item	Default Value	Mandatory	Description
fs.obs.fast.upload.array.first.buffer	1048576	No	When fs.obs.fast.upload.buffer is set to array , this parameter is used to control the initial size of JVM on-heap memory.
fs.obs.readahead.range	1048576	No	Size of the part that will be read ahead.
fs.obs.multiobject.delete.enable	TRUE	No	Specifies whether to enable a batch deletion when directories are deleted.
fs.obs.delete.threads.max	20	No	Parameters maximumPoolSize and corePoolSize are used to control the thread pool.
fs.obs.multiobject.delete.maximum	1000	No	Specifies the maximum number of objects that can be deleted in a batch deletion request. The maximum value is 1000 .
fs.obs.multiobject.delete.threshold	3	No	Specifies the minimum number of objects in a batch deletion. If the number of objects to be batch deleted is less than this parameter value, batch deletion will not be started.
fs.obs.list.threads.core	30	No	Parameter corePoolSize is used to control the thread pool.
fs.obs.list.threads.max	60	No	Parameter maximumPoolSize is used to control the thread pool.
fs.obs.list.workqueue.capacity	1024	No	Capacity of the parameter BlockingQueue that is used to control the thread pool.
fs.obs.list.parallel.factor	30	No	This parameter is used to control concurrency factors.
fs.obs.paging.maximum	1000	No	Specifies the maximum number of objects that can be returned in a list request. The maximum value is 1000 .

Configuration Item	Default Value	Mandatory	Description
fs.obs.copy.thread.s.max	40	No	When a bucket renames a directory, parameters maximumPoolSize and corePoolSize are used to control the thread copy. Their value is half of the value of this parameter. The capacity of BlockingQueue is 1024.
fs.obs.copypart.size	104857600	No	Specifies the size of a single part in a multipart copy. If the size of an object to be copied exceeds this parameter value, multipart copy is performed, and the size of a single part is set to this parameter value. Otherwise, simple copy is performed.
fs.obs.copypart.threads.max	5368709120	No	If multipart copy is performed during the copy of a single object, maximumPoolSize and corePoolSize are used to configure the multipart copy thread pool. Their value is half of the value of this parameter. The capacity of BlockingQueue is 1024.
fs.obs.getcanonicalservice.enable	FALSE	No	Controls the return value of API getCanonicalServiceName() . • TRUE: obs://bucketname • FALSE: null
fs.obs.multipart.purge	FALSE	No	Specifies whether to clear multipart upload tasks in a bucket when OBSFilesystem is initialized.
fs.obs.multipart.purge.age	86400	No	Time before which multipart upload tasks in a bucket will be cleared when OBSFilesystem is initialized.
fs.obs.trash.enable	FALSE	No	Specifies whether to enable the trash feature.
fs.obs.trash.dir	N/A	No	Directory for storing deleted files.
fs.obs.block.size	134217728	No	Block size.

9.4.3 Connecting Hive to OBS

Overview

Hive is a data warehouse tool that can extract, transform, and load large-scale data sets stored distributedly. It provides various SQL query methods for data analysis.

Prerequisites

Hadoop has been installed. For details, see [Connecting Hadoop to OBS](#).

Procedure

The following uses Hive 2.3.3 as an example.

Step 1 Download **apache-hive-2.3.3-bin.tar.gz** and decompress it to the **/opt/hive-2.3.3** directory.

Step 2 Add the following content to the **/etc/profile** file:

```
export HIVE_HOME=/opt/hive-2.3.3
export PATH=$HIVE_HOME/bin:$PATH
```

Step 3 Configure Hive.

1. Rename **hive-env.sh.template** under **/opt/hive-2.3.3/conf/** as **hive-env.sh**.
2. Rename **hive-log4j2.properties.template** under **opt/hive-2.3.3/conf/** as **hive-log4j2.properties**.
3. Create the **hive-site.xml** file and add the following configurations:

```
<property>
<name>hive.metastore.warehouse.dir</name>
<value>obs://obs-bucket/warehouse/hive</value>
</property>
```

NOTE

Adding these configurations is optional. After they are added, you do not need to explicitly specify the location when you create a Hive table, and the created Hive table will be automatically stored in OBS.

4. Initialize the metadata:

```
/opt/hive-2.3.3/bin/schematool -dbType derby -initSchema
```

Step 4 Check whether the connection is successful.

In the following example, the location is **obs://obs-bucket/warehouse/hive/student**.

```
hive>
create table student(id int comment "Student ID",name string comment "Student name",age int comment
"Student age")
comment "Student information table"
row format delimited fields terminated by ",";

insert into table student select 6,"yangdong",29;
```

----End

9.4.4 Connecting Spark to OBS

Overview

Apache Spark is a fast and general compute engine for processing large-scale data sets.

Prerequisites

Hadoop has been installed. For details, see [Connecting Hadoop to OBS](#).

Precautions

To reduce output logs, add the following configuration to the `/opt/spark-2.3.3/conf/log4j.properties` file:

```
log4j.logger.com.obs= ERROR
```

Procedure

The following uses Spark 2.3.3 as an example.

Step 1 Download **spark-2.3.3-bin-without-hadoop.tgz** and decompress it to the `/opt/spark-2.3.3` directory.

Step 2 Add the following content to the `/etc/profile` file:

```
export SPARK_HOME=/opt/spark-2.3.3
export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
```

Step 3 Configure Spark.

1. Rename **spark-env.sh.template** under `/opt/spark-2.3.3/conf/` as **spark-env.sh** and add the following configuration:

```
export SPARK_DIST_CLASSPATH=$(hadoop classpath)
```

For more configurations, see [Apache Hadoop](#).

2. Rename **log4j.properties.template** under `/opt/spark-2.3.3/conf/` as **log4j.properties**.

Step 4 Check whether the connection is successful:

```
$$SPARK_HOME/bin/run-example org.apache.spark.examples.JavaWordCount
obs://obs-bucket/input/test.txt
```

----End

9.4.5 Connecting Presto to OBS

Overview

There are PrestoSQL (renamed to Trino) and PrestoDB available.

Only PrestoSQL (Trino) can connect to OBS. The following example describes how to connect PrestoSQL 333 to OBS. PrestoSQL 332 and later must use JDK 11.

 NOTE

Presto in this section refers to PrestoSQL (Trino).

Prerequisites

- Hadoop has been installed. For details, see [Connecting Hadoop to OBS](#).
- Hive has been installed. For details, see [Connecting Hive to OBS](#).

Installing the Presto Server

Version: PrestoSQL 333

Step 1 Download the Presto client and server.

[Presto client](#)

[Presto server](#)

Step 2 Download [the hadoop-huaweicloud pug-in](#).

Step 3 Decompress the Presto server package:

```
tar -zxvf presto-server-333.tar.gz
```

Place the following JAR packages in the Presto root directory `/plugin/hive-hadoop2`:

- [hadoop-huaweicloud-\\${hadoop.version}-hw-\\${version}.jar](#)
- **Apache commons-lang-xxx.jar**

You can download them from the Maven central repository or copy them from the **hadoop** directory.

----End

Configuring Presto

Create an **etc** directory inside the installation directory. Under **etc**, create the following configuration files:

- Node configuration file: environment configurations of each node
- JVM configuration file: command line options for Java virtual machines (JVMs)
- Server configuration file: configurations of the Presto server
- Catalog configuration file: configurations of different Presto connectors (data sources)
- Log configuration file: Presto log configurations

Node Configuration File

etc/node.properties is the node property file that contains configurations of each node. A node is a Presto instance. This file is typically created when Presto is first installed. The minimum configuration is as follows:

```
node.environment=production
node.id=ffffffff-ffff-ffff-ffffffffffff
node.data-dir=/var/presto/data
```

Explanations:

node.environment: environment name. All nodes in a Presto cluster must have the same environment name.

node.id: the unique identifier for a node. A node ID must keep unchanged across reboots or upgrades of the Presto cluster.

node.data-dir: data directory. It is used by Presto to store logs and other data.

Example:

node.environment=presto_cluster

node.id=bigdata00

node.data-dir=/home/modules/presto-server-0.215/data #data needs to be manually created.

JVM Configuration File

etc/jvm.config is the JVM configuration file that contains command line options for starting JVMs. Each command line option is on a separate line. This file is interpreted by the shell, so options containing spaces or special characters will be ignored.

Reference configurations:

```
-server
-Xmx16G
-XX:-UseBiasedLocking
-XX:+UseG1GC
-XX:G1HeapRegionSize=32M
-XX:+ExplicitGCInvokesConcurrent
-XX:+ExitOnOutOfMemoryError
-XX:+UseGCOverheadLimit
-XX:+HeapDumpOnOutOfMemoryError
-XX:ReservedCodeCacheSize=512M
-Djdk.attach.allowAttachSelf=true
-Djdk.nio.maxCachedBufferSize=2000000
```

The parameters above are from the Presto official website and must be adjusted in an actual environment.

Server Configuration File

etc/config.properties is a configuration property file that contains the configurations for the Presto server. A Presto server can serve as both a coordinator and a worker. In large clusters, you are advised to specify only one machine as the coordinator.

1. Configuration file of the coordinator node

```
coordinator=true
node-scheduler.include-coordinator=true
http-server.http.port=5050
discovery-server.enabled=true
discovery.uri=http://192.168.XX.XX:5050
query.max-memory=20GB
```

```
query.max-memory-per-node=1GB  
query.max-total-memory-per-node=2GB
```

2. Configuration file of the worker node

```
coordinator=false  
http-server.http.port=5050  
discovery.uri=http://192.168.XX.XX:5050  
query.max-memory=20GB  
query.max-memory-per-node=1GB  
query.max-total-memory-per-node=2GB
```

Explanations:

coordinator: whether to run the instance as a coordinator, to receive queries from clients and manage query executions.

node-scheduler.include-coordinator: whether the coordinator also serves as a worker. For larger clusters, processing work on the coordinator can impact query performance.

http-server.http.port: HTTP port. Presto uses HTTP for all external and internal communications.

query.max-memory: the total maximum memory that can be allocated for queries

query.max-memory-per-node: the maximum single-node memory that can be allowed for queries

discovery-server.enabled: Presto uses the Discovery service to find all nodes in the cluster. The Presto coordinator has a built-in Discovery service, and each Presto instance will be registered with the Discovery service on startup. This way, the deployment can be simplified and no additional service is required.

discovery.uri: URI of the Discovery service. In the URI, replace **example.net:8080** with the host and port of the coordinator. The URI cannot end with a slash, or error 404 will be reported.

Additional properties:

jmx.rmiregistry.port: registry of the JMX RMI. The JMX client can connect to the port specified here.

jmx.rmiserver.port: server of the JMX RMI. The JMX can be used for listening.

Catalog Configuration File (Key)

Configure a Hive connector as follows:

1. Create a **catalog** directory under **etc**.
2. Create the configuration file **hive.properties** for the Hive connector.

```
# hive.properties  
#Connector name  
connector.name=hive-hadoop2  
#Configure the Hive metastore connection.  
hive.metastore.uri=thrift://192.168.XX.XX:9083  
#Specify the Hadoop configuration file.  
hive.config.resources=/home/modules/hadoop-2.8.3/etc/hadoop/core-site.xml,/home/modules/  
hadoop-2.8.3/etc/hadoop/hdfs-site.xml,/home/modules/hadoop-2.8.3/etc/hadoop/mapred-site.xml  
# Grant the permission to drop tables.  
hive.allow-drop-table=true
```

Log Configuration File

1. Create a **log.properties** file.
2. Write content: **com.facebook.presto=INFO**.

There are four log levels: **DEBUG**, **INFO**, **WARN**, and **ERROR**.

Starting Presto

The procedure is as follows:

Step 1 Run **hive --service metastore &** to start the Hive metastore.

Step 2 Run **bin/launcher start** to start the Presto server. To stop the Presto server, run **bin/launcher stop**.

Step 3 Start the Presto client.

1. Rename **presto-cli-333-executable.jar** to **presto**, place it in the **bin** directory, and run the **chmod +x presto** command to make it executable.
2. Run **./presto --server XX.XX.XX.XX:5050 --catalog hive --schema default** to start the client.

----End

Using Presto to Query OBS

Creating a Hive table

```
hive>
CREATE TABLE sample01(id int,name string,address string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'obs://obs-east-bkt001/sample01';

insert into sample01 values(1,'xiaoming','cd');
insert into sample01 values(2,'daming','sh');
```

Using Presto to query the Hive table

```
./presto --server XX.XX.XX.XX:5050 --catalog hive --schema default
```

```
presto:default>
select * from sample01;
```

9.4.6 Connecting Flume to OBS

Overview

Flume is a distributed, reliable, and highly available service for collecting, aggregating, and moving a large amount of log data. For details, see [Apache Flume](#). In big data scenarios, OBS can replace HDFS in the Hadoop system.

Precautions

- Multiple sinks write the same file.
OBS and HDFS differ in consistency assurance. The HDFS lease mechanism keeps data consistent when the same file is concurrently written, but the

HDFS protocol effected by OBS does not support the lease mechanism, that is, something uncertain will happen when the same file is concurrently written. To address this issue, the file naming rules can be used in Flume scenarios.

For example, **hostname-sinkname** is used as the prefix of a sink file name. If a host has multiple Flume agents deployed, each agent must have a different sink name.

- Flume log configuration

To reduce output logs, add the following configuration to the **/opt/apache-flume-1.9.0-bin/conf/log4j.properties** file:

```
log4j.logger.com.obs=ERROR
```

- Configuration for the directory of temporary files that OBSA writes data to.

When Flume writes data to OBS, the data is first written to the local disk buffer and then uploaded to OBS. If you require better performance for data write, select a high-performance disk as the buffer. Specifically, add the following configuration to the **core-site.xml** file:

```
<property>
<name>fs.obs.buffer.dir</name>
<value>xxx</value>
</property>
```

Procedure

The following uses Flume 1.9 as an example.

Step 1 Download **apache-flume-1.9.0-bin.tar.gz**.

Step 2 Install Flume.

Decompress **apache-flume-1.9.0-bin.tar.gz** to the **/opt/apache-flume-1.9.0-bin** directory.

- If Hadoop has been deployed, no additional operation is required. For details about the deployment, see [Connecting Hadoop to OBS](#).
- If Hadoop is not deployed:
 - a. Copy the Hadoop JAR packages, including **hadoop-huaweicloud-xxx.jar**, to the **/opt/apache-flume-1.9.0-bin/lib** directory.
 - b. Copy the **core-site.xml** file containing the OBS configurations to the **/opt/apache-flume-1.9.0-bin/conf** directory.

Step 3 Check whether the connection is successful.

Example: The built-in **StressSource** is used as the source, the **file** is used as the channel, and the **obs** is used as the sink.

1. Create a Flume configuration file **sink2obs.properties**.

```
agent.sources = r1
agent.channels = c1
agent.sinks = k1

agent.sources.r1.type = org.apache.flume.source.StressSource
agent.sources.r1.channels = c1
agent.sources.r1.size = 1024
agent.sources.r1.maxTotalEvents = 100000
agent.sources.r1.maxEventsPerSecond = 10000
agent.sources.r1.batchSize=1000
```

```
agent.sources.r1.interceptors = i1
agent.sources.r1.interceptors.i1.type = host
agent.sources.r1.interceptors.i1.useIP = false

agent.channels.c1.type = file
agent.channels.c1.dataDirs = /data/agent/flume-data
agent.channels.c1.checkpointDir = /data/agent/flume-checkpoint
agent.channels.c1.capacity = 500000
agent.channels.c1.transactionCapacity = 50000

agent.sinks.k1.channel = c1
agent.sinks.k1.type = hdfs
agent.sinks.k1.hdfs.useLocalTimeStamp = true
agent.sinks.k1.hdfs.filePrefix = %{host}_k1
agent.sinks.k1.hdfs.path = obs://obs-bucket/flume/create_time=%Y-%m-%d-%H-%M
agent.sinks.k1.hdfs.fileType = DataStream
agent.sinks.k1.hdfs.writeFormat = Text
agent.sinks.k1.hdfs.rollSize = 0
agent.sinks.k1.hdfs.rollCount = 1000
agent.sinks.k1.hdfs.rollInterval = 0
agent.sinks.k1.hdfs.batchSize = 1000
agent.sinks.k1.hdfs.round = true
agent.sinks.k1.hdfs.roundValue = 10
agent.sinks.k1.hdfs.roundUnit = minute
```

2. Start the Flume agent:

```
./bin/flume-ng agent -n agent -c conf/ -f conf/sink2obs.properties
```

----End

9.4.7 Connecting DataX to OBS

Overview

DataX is a data synchronization framework. It can efficiently synchronize data among heterogeneous data sources such as MySQL, SQL Server, Oracle, PostgreSQL, HDFS, Hive, HBase, OTS and ODPS. In big data scenarios, OBS can replace HDFS in the Hadoop system. This section describes how to connect DataX to OBS.

Procedure

Step 1 Download the DataX source code ([version datax_v202308](#) as an example).

Step 2 Modify and compile DataX.

1. Upgrade the Hadoop version which HdfsReader and HdfsWriter depend on. In this example, the Hadoop will be upgraded to version 2.8.3.

Modify the **pom.xml** files under **datax\hdfswriter** and **datax\hdfsreader**.

```
<properties>
<!--Upgrade from 2.7.1 to 2.8.3-->
<hadoop.version>2.8.3</hadoop.version>
</properties>
```

2. Compile DataX.
3. Generate the **datax.tar.gz** file in the **/target** directory, the root directory of the datax source code:

```
mvn -U clean package assembly:assembly -Dmaven.test.skip=true
```

Step 3 Install DataX.

1. Decompress **datax.tar.gz** to the **/opt/datax** directory.
2. Download **hadoop-huaweicloud** from [GitHub](#). You are advised to download the latest hadoop-huaweicloud version under Hadoop 2.8.3, for example, **hadoop-huaweicloud-2.8.3-hw-53.8**.
3. Save the downloaded JAR package to **/opt/datax/plugin/writer/hdfswriter/libs** and **/opt/datax/plugin/reader/hdfsreader/libs** directories.

Step 4 Check whether the connection is successful.

Example: **txtfilereader** is the source, and OBS is the destination.

1. Create a job configuration file **file2obs.json**.

```
{
  "setting":{
  },
  "job":{
    "setting":{
      "speed":{
        "channel":2
      }
    },
    "content":[
      {
        "reader":{
          "name":"txtfilereader",
          "parameter":{
            "path":[
              "/opt/test.txt"
            ],
            "encoding":"UTF-8",
            "column":[
              {
                "index":0,
                "type":"STRING"
              },
              {
                "index":1,
                "type":"STRING"
              }
            ]
          },
          "fieldDelimiter":"\t"
        },
        "writer":{
          "name":"hdfswriter",
          "parameter":{
            "defaultFS":"obs://obs-bucket",##OBS bucket
            "fileType":"text",
            "path":"/test",##Path in the OBS bucket
            "fileName":"test",
            "column":[
              {
                "name":"col1",
                "type":"STRING"
              },
              {
                "name":"col2",
                "type":"STRING"
              }
            ]
          },
          "writeMode":"append",
          "fieldDelimiter":"\t",
          "hadoopConfig": {##Hadoop configurations must be added.
            "fs.obs.impl":"org.apache.hadoop.fs.obs.OBSFileSystem",
            "fs.obs.access.key":"AK that can access OBS",
            "fs.obs.secret.key":"SK that can access OBS",
```

```
        "fs.obs.endpoint": "Region where the OBS bucket is located"
    }
}
]
```

2. Start DataX:

```
python /opt/datax/bin/datax.py file2obs.json
```

----End

9.4.8 Connecting Druid to OBS

Overview

Druid is specially designed for workflows where fast data query and ingestion are required. It performs well in instant data visibility, ad hoc query, operations analytics, and high concurrency.

You can use OBSA-HDFS to connect OBS to Druid. In this way, you do not need to recompile Druid. OBS should be configured as deep storage.

Procedure

Step 1 Configure Druid.

1. Modify the configurations:

```
conf/druid/single-server/micro-quickstart/_common/common.runtime.properties
```

Add **druid-hdfs-storage** to **druid.extensions.loadList**.

```
# If you specify 'druid.extensions.loadList=[]', Druid won't load any extension from file system.
# If you don't specify 'druid.extensions.loadList', Druid will load all the extensions under root extension directory.
# More info: https://druid.apache.org/docs/latest/operations/including-extensions.html
druid.extensions.loadList=["druid-hdfs-storage", "druid-kafka-indexing-service", "druid-datasketches"]
```

2. Configure the Deep Storage path in OBS.

```
#
# Deep storage
#

# For local disk (only viable in a cluster if this is a network mount):
#druid.storage.type=local
#druid.storage.storageDirectory=var/druid/segments

# For HDFS:
druid.storage.type=hdfs
druid.storage.storageDirectory=obs://wxg-sg-oms-test/druidhdfs/segments
```

```
#  
# Indexing service logs  
#  
# For local disk (only viable in a cluster if this is a network mount):  
#druid.indexer.logs.type=file  
#druid.indexer.logs.directory=var/druid/indexing-logs  
  
# For HDFS:  
druid.indexer.logs.type=hdfs  
druid.indexer.logs.directory=obs://wxg-sg-oms-test/druidhdfs/indexing-logs
```

Step 2 Configure OBSA-HDFS.

1. [Download OBSA-HDFS](#) from GitHub and copy it to the **extensions/druid-hdfs-storage/** directory.
2. Add **hdfs-site.xml** to the **conf/druid/single-server/micro-quickstart/_common/** directory and configure it as follows (replace the endpoint with the one actually used).

```
<configuration>  
  <property>  
    <name>fs.obs.access.key</name>  
    <value>[REDACTED]</value>  
  </property>  
  <property>  
    <name>fs.obs.secret.key</name>  
    <value>[REDACTED]</value>  
  </property>  
  <property>  
    <name>fs.obs.endpoint</name>  
    <value>obs.ap-southeast-3.myhuaweicloud.com</value>  
  </property>  
  <property>  
    <name>fs.obs.buffer.dir</name>  
    <value>/home/modules/data/buf</value>  
  </property>  
  <property>  
    <name>fs.obs.impl</name>  
    <value>org.apache.hadoop.fs.obs.OBSFileSystem</value>  
  </property>  
</configuration>
```

Step 3 Start Druid.

----End

9.4.9 Connecting Flink to OBS

Overview

Flink is a distributed data processing engine for processing bounded and unbounded data streams. Flink defines the file system APIs and OBS implements the defined APIs, so that OBS can be used as the Flink StateBackend and the carrier of data read/write.

Precautions

- **flink-obs-fs-hadoop** currently supports only OBS parallel file systems.
- You are advised not to store stateful data on OBS.
- To reduce output logs, add the following configurations to the **/opt/flink-1.12.1/conf/log4j.properties** file:

```
logger.obs.name=com.obs  
logger.obs.level=ERROR
```
- **flink-obs-fs-hadoop** is implemented based on the plug-in loading mechanism of Flink (introduced from Flink 1.9). It must be loaded using this mechanism, that is, placing **flink-obs-fs-hadoop** in the **/opt/flink-1.12.1/plugins/obs-fs-hadoop** directory.

Procedure

The following uses **flink-1.12.1** as an example.

Step 1 Download **flink-1.12.1-bin-scala_2.11.tgz** and decompress it to the **/opt/flink-1.12.1** directory.

Step 2 Add the following content to the **/etc/profile** file:

```
export FLINK_HOME=/opt/flink-1.12.1  
export PATH=$FLINK_HOME/bin:$PATH
```

Step 3 Install **flink-obs-fs-hadoop**.

1. Download it from [GitHub](#).

NOTE

- In **flink-obs-fs-hadoop-*{flinkversion}*-hw-*{version}*.jar**, *flinkversion* indicates the Flink version number, and *version* indicates the version number of **flink-obs-fs-hadoop**.
 - If no JAR package of a required version is available, modify the Flink version in the POM file under the **flink-obs-fs-hadoop** directory and recompile the file.
 - When compiling **flink-obs-fs-hadoop**, you are advised to use **hadoop-huaweicloud** in version 53.8 or later.
2. Create the **obs-fs-hadoop** directory under the **/opt/flink-1.12.1/plugins** directory and save the JAR package above to **obs-fs-hadoop**.

Step 4 Configure Flink.

Configure the following parameters in the **/opt/flink-1.12.1/conf/flink-conf.yaml** file or in the code:

```
fs.obs.impl: org.apache.hadoop.fs.obs.OBSFileSystem  
fs.obs.access.key: xxx  
fs.obs.secret.key: xxx
```

```
fs.obs.endpoint: xxx  
fs.obs.buffer.dir: /data/buf # Local temporary directory for you to write data to OBS. The Flink must have  
the read and write permissions for this directory.
```

Step 5 Compile the Flink application.

1. Set **StateBackend** to a path in OBS.

Example:

```
env.setStateBackend(new FsStateBackend("obs://obs-bucket/test/checkpoint"));
```

2. Set **StreamingFileSink** to a path in OBS.

Example:

```
final StreamingFileSink<String> sink = StreamingFileSink  
.forRowFormat(new Path("obs://obs-bucket/test/data"),  
new SimpleStringEncoder<String>("UTF-8"))  
.withBucketAssigner(new BasePathBucketAssigner())  
.withRollingPolicy(rollingPolicy)  
.withBucketCheckInterval(1000L)  
.build();
```

----End

9.4.10 Connecting Logstash to OBS

Overview

Logstash collects data from a multitude of sources, transforms it, and then ships it to the storage system. This section describes how to connect Logstash to OBS.

Precautions

Logstash 7.10.2 or later is recommended.

Procedure

The following uses **logstash-7.10.2** as an example.

- Step 1** Download **logstash-7.10.2-linux-x86_64.tar.gz** and decompress it to the **/opt/logstash-7.10.2-linux-x86_64** directory.

- Step 2** Check whether the connection is successful.

Example: Use **file** as the source and OBS as the destination.

1. Create a configuration file **file2obs.conf**. [Table 9-3](#) describes the parameters. For more information, see [Logstash Reference](#).

```
input {  
  file {  
    path => "/opt/nginx/logs/access.log"  
    start_position => "beginning"  
  }  
}  
  
output {  
  s3 {  
    endpoint => "obs endpoint" # The endpoint should be an HTTP or HTTPS URL  
    access_key_id => "ak"  
    secret_access_key => "sk"  
    bucket => "obs bucket name"  
    size_file => 1048576  
    time_file => 1  
    prefix => "logstash/"  
  }  
}
```

```
enable_metric => true
}
}
```

Table 9-3 Parameters

Parameter	Description
endpoint	An endpoint of OBS. Examples are as follows: - https://obs.eu-west-101.myhuaweicloud.eu - http://obs.eu-west-101.myhuaweicloud.eu
access_key_id	AK for accessing OBS.
secret_access_key	SK for accessing OBS.
bucket	OBS bucket name.
size_file	Specifies the file size (in bytes). When the size exceeds this parameter value, a new file is created.
time_file	Sets the time (in minutes). When the data write period exceeds this parameter value, a new file is created.
prefix	File storage directory, for example, logstash/ . In this case, files will be written to the logstash/ directory of the bucket. The directory cannot start with a slash (/).

- Run Logstash:
bin/logstash -f ../conf/file2obs.conf
----End

9.5 Migrating HDFS Data to OBS

Scenarios

In the Huawei Cloud big data solution with decoupled storage and compute, OBS serves as a unified data lake to provide storage. If your data is still stored in local HDFS, migrate HDFS data to OBS first.

You can use any of the following methods to migrate data: [DistCp](#) or [CDM](#).

Migration Using DistCp

Hadoop DistCp (abbreviation of distributed copy) is a tool used for large inter- or intra-Hadoop cluster copying. It uses MapReduce to implement file distribution, error handling and recovery, and reporting. It puts a list of files and directories as the input of map tasks, and each task will copy some files specified in the source list.

Configuration

Configure OBS by referring to the **hadoop-huaweicloud** installation and configuration in [Connecting Hadoop to OBS](#).

Example

- Step 1** View the files and directories in an HDFS directory (**/data/sample** as an example) to migrate:

```
hadoop fs -ls hdfs:///data/sample
```

- Step 2** Migrate all files and directories inside **/data/sample** to the **data/sample** directory in OBS bucket **obs-bigdata-posix-bucket**:

```
hadoop distcp hdfs:///data/sample obs://obs-bigdata-posix-bucket/data/sample
```

- Step 3** View the file copies:

```
hadoop fs -ls obs://obs-bigdata-posix-bucket/data/sample
```

----End

Migration Using CDM

Cloud Data Migration (CDM) enables batch data migration among homogeneous and heterogeneous data sources, to realize flexible data flow. The data sources supported include relational databases, data warehouses, NoSQL, and big data cloud services.

For details, see [What Is CDM?](#)

A Change History

Release Date	What's New
2023-12-15	This issue is the second official release. This issue incorporates the following change: Added Using OBS to Decouple Storage from Compute in Big Data Scenarios.
2022-09-30	This issue is the first official release.