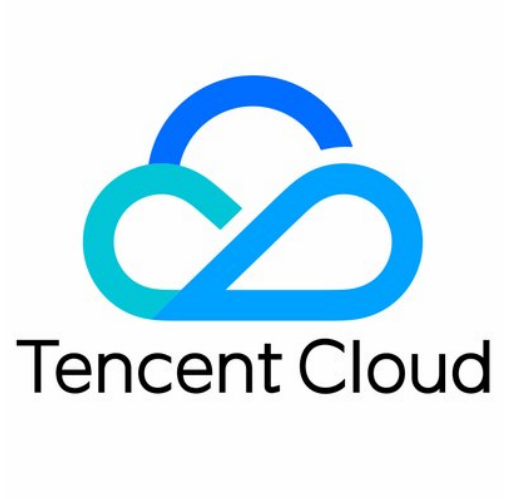
# Cloud HDFS

# Operation Guide

# Product Documentation



# Contents

# Operation Guide
# Creating CHDFS Instance

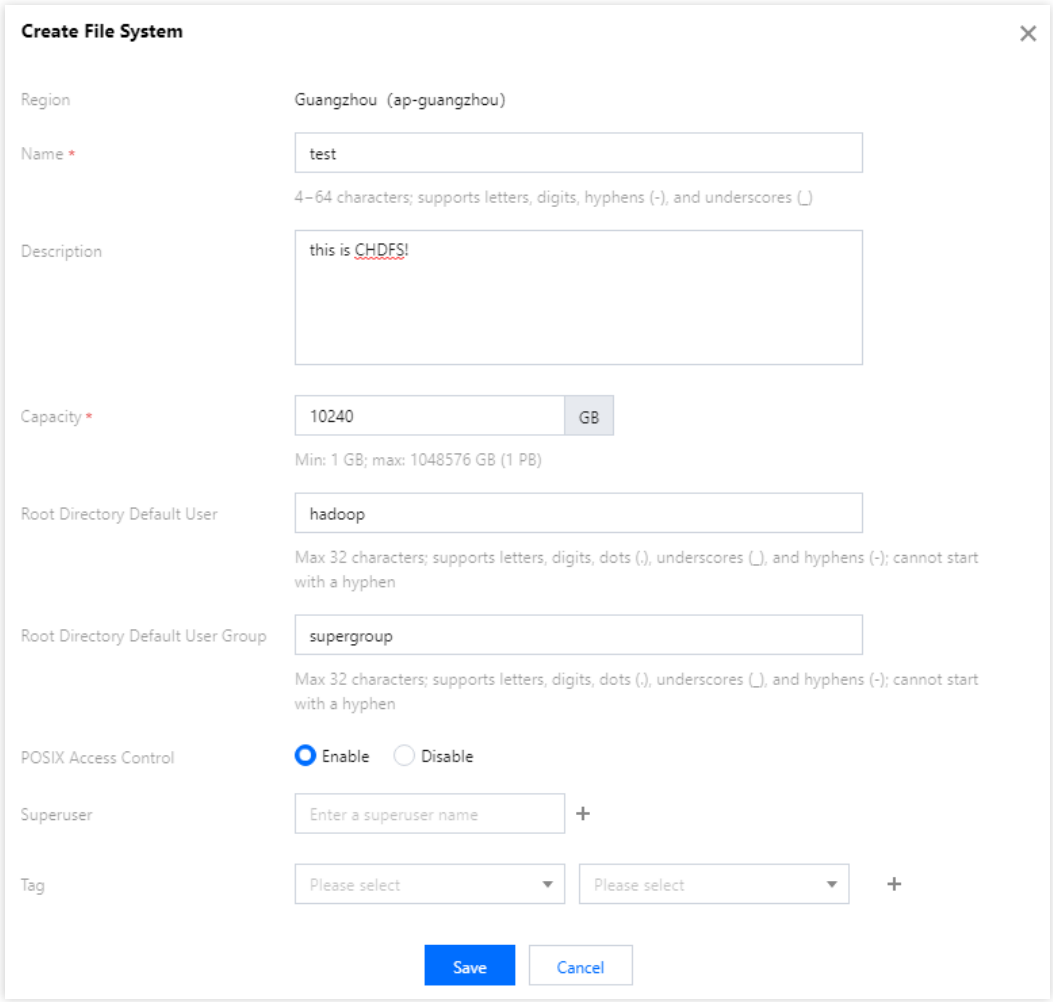
Last updated：2022-03-30 09:30:25

This document describes how to create a file system in the CHDFS console.

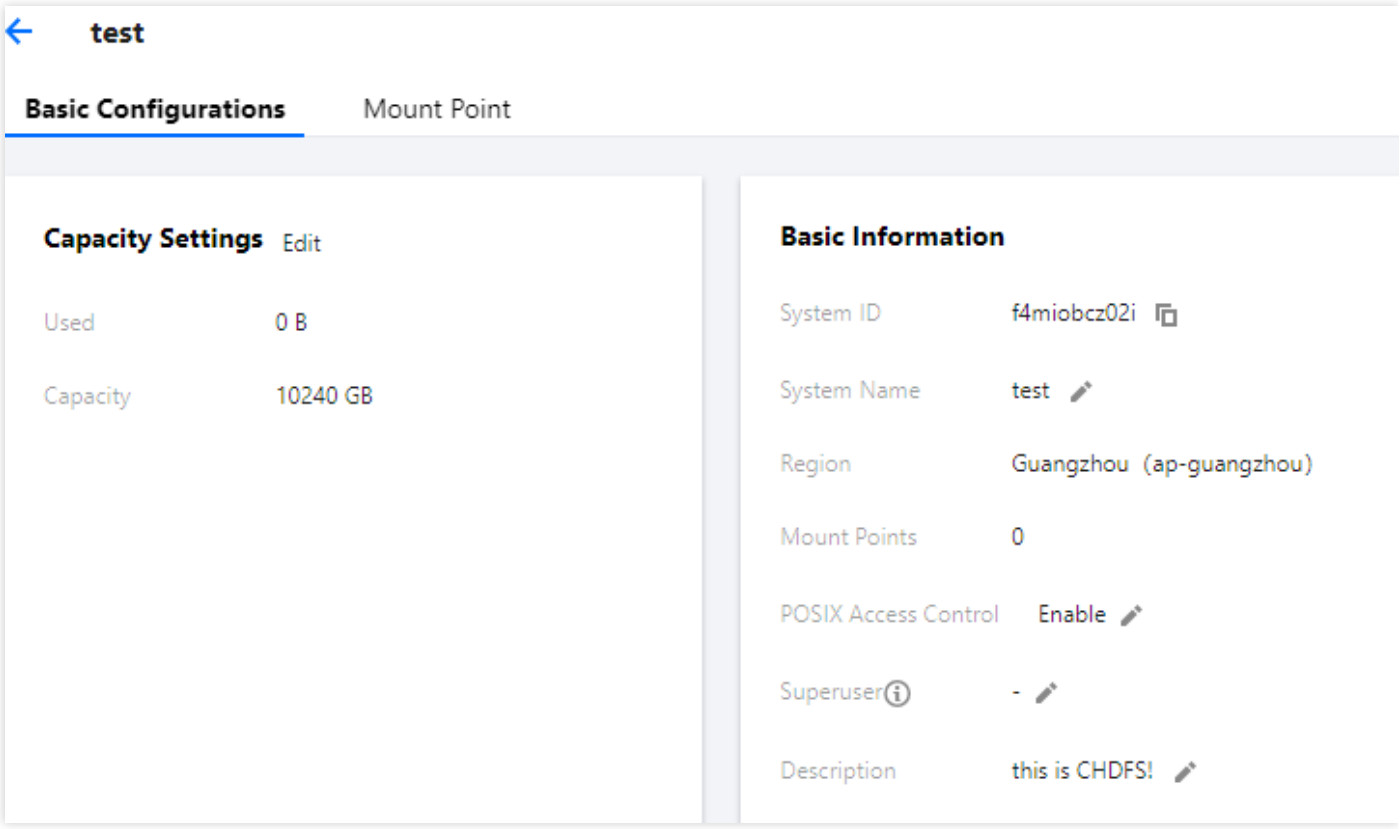
## Directions

1. Log in to the CHDFS console, click **File Systems** on the left sidebar, and select the target region, such as Guangzhou.
2. Click **Create** and enter the name and description in the pop-up window.

- **Name**: file system name, which can contain 4 to 64 letters, digits, hyphens, and underscores.
- **Description**: file system description.



3. Click **Save**.
4. Click **Configure** for the CHDFS instance to view its basic configuration and mount information as shown below:



# Creating Permission Group

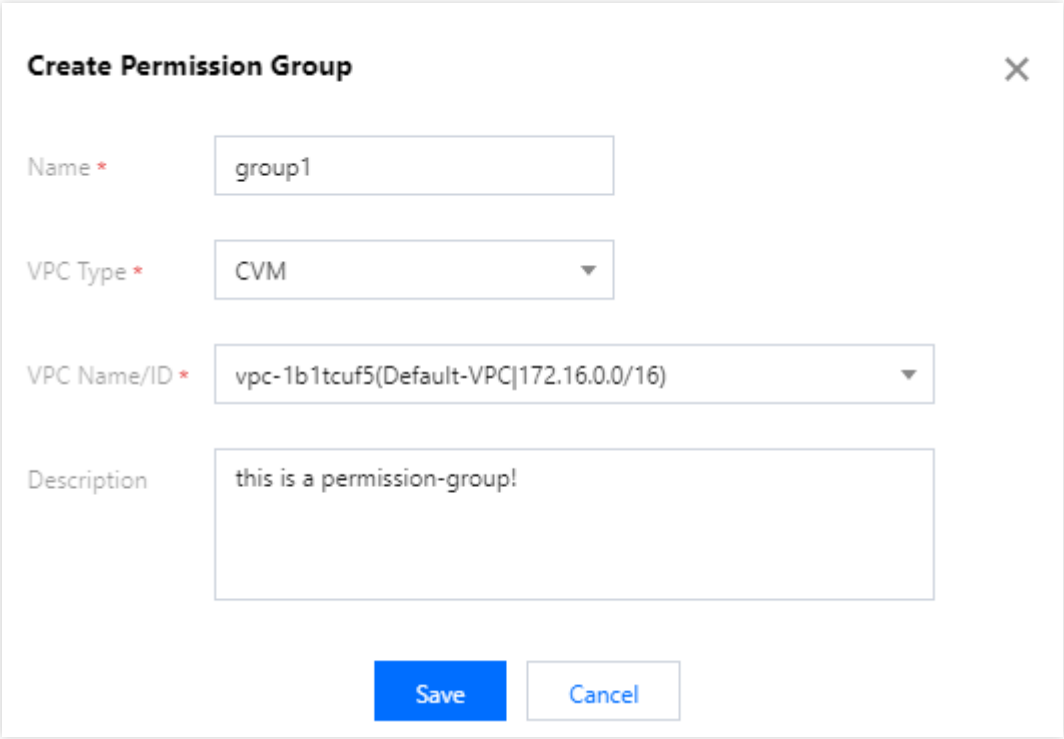
Last updated：2022-03-30 09:30:25

This document describes how to create a permission group, which is used to manage CHDFS permissions. Before using CHDFS, you need to create a permission group first.

## Directions

1. Log in to the CHDFS console, click **Permission Groups** on the left sidebar, and select the target region, such as Guangzhou.
2. Click **Create** and enter the name and description in the pop-up window.

- Name: permission group name, which can contain 4 to 64 letters, digits, hyphens, and underscores.
- Description: permission group description.




3. Click **Save**.

# Creating Permission Rule

Last updated：2022-03-30 09:30:25

This document describes how to create a permission rule. A permission group contains various permission rules for management of CHDFS permissions. Before using CHDFS, you need to create a permission rule in a created permission group first.

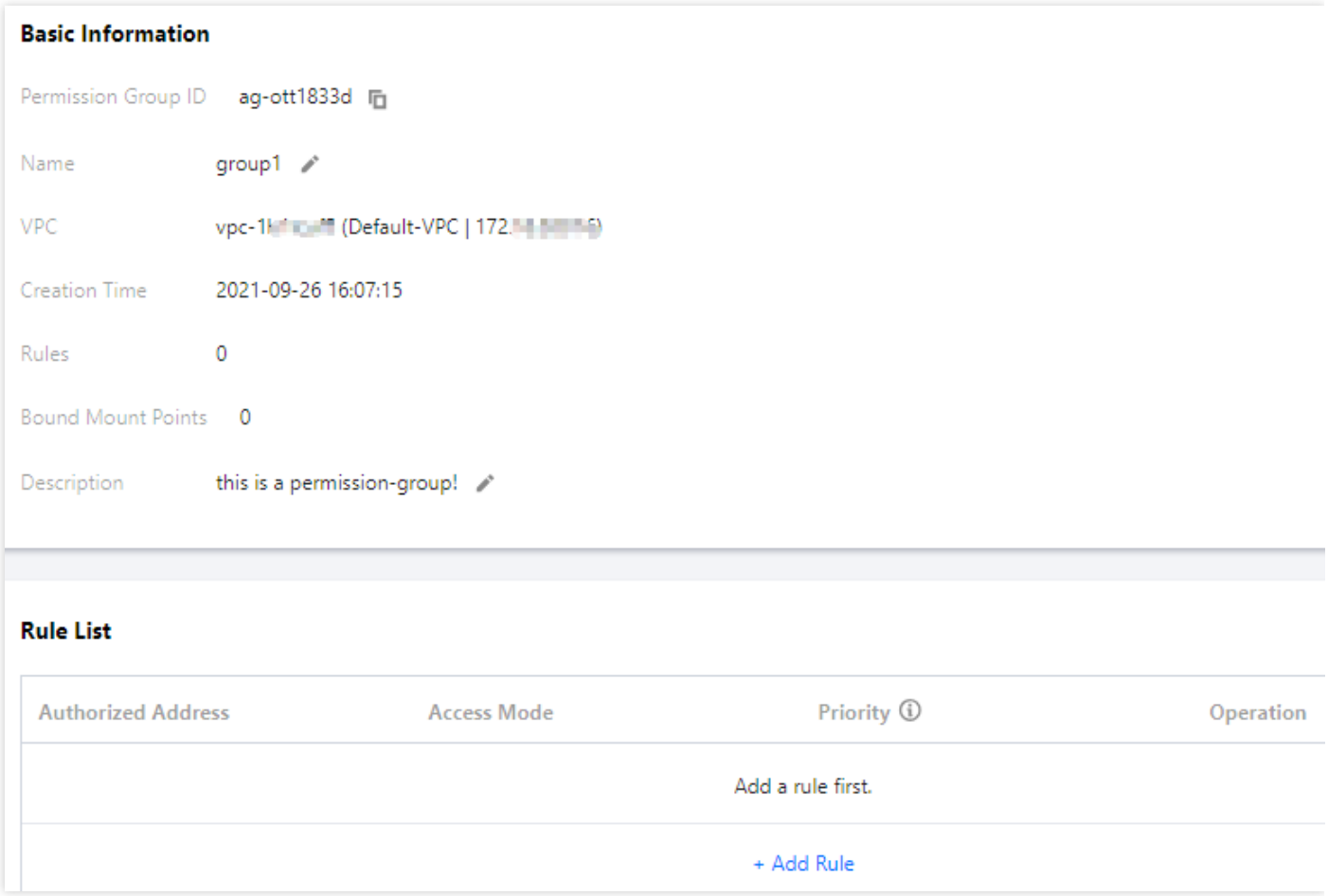
## Prerequisites

You have created a permission group. For more information on how to create one, please see Creating Permission Group.

## Directions

1. Log in to the CHDFS console, click **Permission Groups** on the left sidebar, and select the target region, such as Guangzhou.
2. Find the target permission group and click **Add Rule** to enter the rule configuration page, where you can view the basic information of the permission group and the permission rule list. In the rule list, enter **Authorized Address**,

**Access Mode**, and **Priority**.




- Authorized Address: IP address or IP range, which is authorized to access CHDFS, such as `10.10.1.2` or `10.10.1.2/20`.
- Access Mode: "Read-write" or "Read-only" access to CHDFS.
- Priority: 1–100, where "1" represents the highest priority. When a CHDFS instance matches multiple permission rules, high-priority rules override low-priority ones.

3. Click **Save**.

# Creating Mount Point

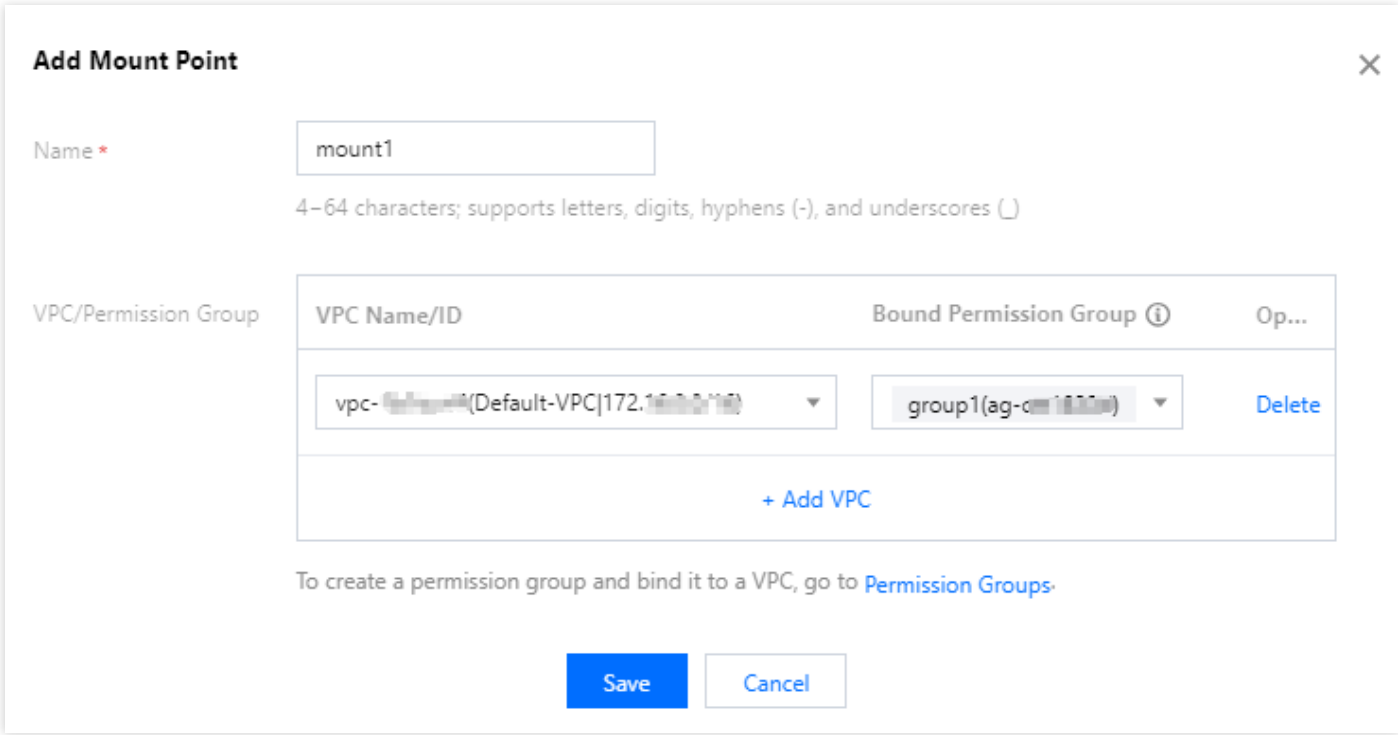
Last updated：2022-03-30 09:30:26

This document describes how to create a mount point. CVM, CPM 2.0, or TKE accesses data in CHDFS through a mount point, which is the destination address of CHDFS for access in a VPC and corresponds to a domain name.

## Directions

1. Log in to the CHDFS console, click **File Systems** on the left sidebar, and select the target region, such as Guangzhou.
2. Find the target CHDFS instance and click **Configure** to view its basic configuration and mount information.
3. Select **Mount Points** > **Add Mount Point** and enter the name and specify the VPC and permission group on the mount point addition page.

- Name: mount point name, which can contain 4–64 letters, digits, hyphens, and underscores.
- VPC Name/ID: select a VPC.
- Permission Group: select a permission group. If there are no permission groups, please create one first.




3. Click **Save**.

# Mounting CHDFS Instance

Last updated：2022-03-30 09:30:26

This document describes how to mount a CHDFS instance. After creating a CHDFS instance and a mount point, you can mount the instance to the mount point.

## Prerequisites

- The target server or container has Java 1.8 installed.
- The target server or container and the mount point are in the same VPC.
- The VPC IP of the target server or container matches the address authorized by a permission rule in the specified permission group of the mount point.

## Directions

1. Download the CHDFS for Hadoop JAR package.

2. Place the JAR package in the corresponding directory. For an EMR cluster, it can be synced to the `/usr/local/service/hadoop/share/hadoop/common/lib/` directory of all nodes.

3. Edit the `core-site.xml` file to add the following basic configuration:

```
<!--Implementation class of CHDFS-->
<property>
<name>fs.AbstractFileSystem.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter</value>
</property>
<property>
<name>fs.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter</value>
</property>
<!--Temporary directory of the local cache. For data read/write, data will be w
ritten to the local disk when the memory cache is insufficient. This path will
be created automatically if it does not exist-->
<property>
<name>fs.ofs.tmp.cache.dir</name>
<value>/data/chdfs_tmp_cache</value>
</property>
<!--appId-->
```

```
<property>
<name>fs.ofs.user.appid</name>
<value>1250000000</value>
</property>
```

4. Sync `core-site.xml` to all Hadoop nodes.

> Note：
> For an EMR cluster, you only need to modify the HDFS configuration in component management in the EMR console for the above steps 3 and 4.

5. Use the `hadoop fs` command line tool to run the `hadoop fs -ls ofs://${mountpoint}/` command. Here, `mountpoint` is the mount address. If the file list is output properly, the CHDFS instance has been mounted successfully.

6. You can also use other configuration items of Hadoop or MR tasks to run data tasks on CHDFS. For an MR task, you can change the default input and output file systems of the task to `CHDFS` through `-Dfs.defaultFS=ofs://${mountpoint}/` .

## Other Configuration Items

| Configuration Item | Description | Default Value | Required |
| --- | --- | --- | --- |
| fs.ofs.tmp.cache.dir | Stores temporary data | None | Yes |
| fs.ofs.map.block.size | Block size of the CHDFS file system in bytes. The default value is 128 MB (this item only affects map segmentation and has nothing to do with the size of the underlying storage block of CHDFS) | 134217728 | No |
| fs.ofs.data.transfer.thread.count | Number of parallel threads when CHDFS transfers data | 32 | No |
| fs.ofs.block.max.memory.cache.mb | Size of the memory buffer used by the CHDFS plugin in MB (which accelerates both reads and writes) | 16 | No |

| Configuration Item | Description | Default Value | Required |
| --- | --- | --- | --- |
| fs.ofs.block.max.file.cache.mb | Size of the disk buffer used by the CHDFS plugin in MB (which accelerates writes) | 256 | No |
| fs.ofs.prev.read.block.count | Number of CHDFS blocks read ahead during reads (the size of the underlying block of CHDFS is generally 4 MB) | 4 | No |
| fs.ofs.plugin.info.log | Specifies whether to print plugin debugging logs. Logs are printed at the info level. Valid values: true, false | false | No |

# Authorizing Access with CAM

Last updated：2022-03-30 09:30:26

## Preset Policies for CHDFS

The preset authorization policies for CHDFS are as follows:

| Policy | Description |
| --- | --- |
| QcloudCHDFSReadOnlyAccess | Read-only access to CHDFS |
| QcloudCHDFSFullAccess | Permission to manage CHDFS |

**Authorized operations in CHDFS**

| Action | Resource | Description |
| --- | --- | --- |
| chdfs:CreateFileSystem | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/* | Creates CHDFS instance |
| chdfs:DeleteFileSystem | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Deletes CHDFS instance |
| chdfs:ModifyFileSystem | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Modifies CHDFS instance attribute |
| chdfs:DescribeFileSystem | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Views CHDFS instance details |
| chdfs:DescribeFileSystems | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Views CHDFS instance list |
| chdfs:CreateMountPoint | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Creates mount point |
| chdfs:DeleteMountPoint | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Deletes mount point |
| chdfs:ModifyMountPoint | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Modifies mount point attribute |
| chdfs:DescribeMountPoint | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Views mount point details |

| Action | Resource | Description |
| --- | --- | --- |
| chdfs:DescribeMountPoints | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Views mount point list |
| chdfs:AssociateAccessGroups | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Associates permission group list |
| chdfs:DisassociateAccessGroups | qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id} | Disassociates permission group list |
| chdfs:CreateAccessGroup | qcs::chdfs:${region-id}:uin/${account-uin}:vpc/${vpc-id}<br>qcs::chdfs:${region-id}:uin/${account-uin}:unVpcId/${unVpcId} | Creates permission group |
| chdfs:DeleteAccessGroup | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Deletes permission group |
| chdfs:ModifyAccessGroup | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Modifies permission group attribute |
| chdfs:DescribeAccessGroup | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Views permission group details |
| chdfs:DescribeAccessGroups | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Views permission group list |
| chdfs:CreateAccessRules | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Batch creates permission rules |
| chdfs:DeleteAccessRules | qcs::chdfs:${region-id}:uin/${account-uin}:accessrule/${access-rule-id} | Batch deletes permission rules |
| chdfs:ModifyAccessRules | qcs::chdfs:${region-id}:uin/${account-uin}:accessrule/${access-rule-id} | Batch modifies the attribute of permission rules |
| chdfs:DescribeAccessRules | qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id} | Views permission rule list |
| chdfs:CreateLifeCycleRules | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Batch creates lifecycle rules |
| chdfs:DeleteLifeCycleRules | qcs::chdfs:${region-id}:uin/${account-uin}:lifecyclerule/${life-cycle-rule-id} | Batch deletes lifecycle rules |

| Action | Resource | Description |
| --- | --- | --- |
| chdfs:ModifyLifeCycleRules | qcs::chdfs:${region-id}:uin/${account-uin}:lifecyclerule/${life-cycle-rule-id} | Batch modifies the attribute of lifecycle rules |
| chdfs:DescribeLifeCycleRules | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Views lifecycle rule list |
| chdfs:CreateRestoreTasks | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Batch creates restoration tasks |
| chdfs:DescribeRestoreTasks | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Views restoration task list |
| chdfs:ModifyResourceTags | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Modifies resource tag list |
| chdfs:DescribeResourceTags | qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id} | Views resource tag list |

## Sample CHDFS Authorization Policies

Below is a sample policy for granting a sub-account the read-only access to the CHDFS control system:

```
{
"version": "2.0",
"statement": [{
"effect": "allow",
"action": [
"name/chdfs:Describe*"
],
"resource": [
"*"
]
}]
}
```

Below is a sample policy for granting a sub-account the permission to view CHDFS:

```
{
"version": "2.0",
"statement": [{
"effect": "allow",
"action": [
```

```
"name/chdfs:DescribeFileSystem"
],
"resource": [
"qcs::chdfs::uin/ownerUin:filesystem/fileSystemId"
]
}]
}
```

# Accessing CHDFS Through Java Code

Last updated：2022-03-30 09:30:26

## Overview

This document describes how to access CHDFS through Java code. After deploying the JAR package of CHDFS, in addition to manipulating CHDFS by using command lines, big data components, and other methods, you can also access CHDFS through Java code.

## Prerequisites

- The relevant JAR package of CHDFS has been deployed. For more information on the deployment, please see Mounting CHDFS Instance.
- The server that runs the Java program is in a VPC allowed to be accessed by the permission group of the mount point.

## Directions

1. Create a Maven project and add the following dependencies to `pom.xml` in Maven (please set the version of the `hadoop-common` package based on your actual Hadoop environment).

```
<dependencies>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
<version>2.8.5</version>
<scope>provided</scope>
</dependency>
</dependencies>
```

2. Run the Hadoop code for modification as detailed below. For the configuration items and their descriptions, please see Mounting CHDFS Instance.
   Only certain common file system operation APIs are listed below. For other APIs, please see Hadoop FileSystem APIs.

```
package com.qcloud.chdfs.demo;
import org.apache.commons.io.IOUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileChecksum;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import java.io.IOException;
import java.net.URI;
import java.nio.ByteBuffer;
public class Demo {
private static FileSystem initFS() throws IOException {
Configuration conf = new Configuration();
// For the CHDFS configuration items, please visit https://cloud.tencent.com/do
cument/product/1105/36368
// The following configuration items are required
conf.set("fs.ofs.impl", "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter");
conf.set("fs.AbstractFileSystem.ofs.impl", "com.qcloud.chdfs.fs.CHDFSDelegateFS
Adapter");
conf.set("fs.ofs.tmp.cache.dir", "/data/chdfs_tmp_cache");
conf.set("fs.ofs.user.appid", "1250000000");
// For other optional configuration items, please visit https://cloud.tencent.c
om/document/product/1105/36368
String chdfsUrl = "ofs://f4maaabbb-ccdd.chdfs.ap-guangzhou.myqcloud.com/";
return FileSystem.get(URI.create(chdfsUrl), conf);
}
private static void mkdir(FileSystem fs, Path filePath) throws IOException {
fs.mkdirs(filePath);
}
private static void createFile(FileSystem fs, Path filePath) throws IOException
{
// Create a file (if it already exists, it will be overwritten)
// if the parent dir does not exist, fs will create it!
FSDataOutputStream out = fs.create(filePath, true);
try {
// Write a file
String content = "test write file";
out.write(content.getBytes());
} finally {
IOUtils.closeQuietly(out);
}
}
private static void readFile(FileSystem fs, Path filePath) throws IOException {
FSDataInputStream in = fs.open(filePath);
```

```
try {
byte[] buf = new byte[4096];
int readLen = -1;
do {
readLen = in.read(buf);
} while (readLen >= 0);
} finally {
IOUtils.closeQuietly(in);
}
}

private static void queryFileOrDirStatus(FileSystem fs, Path path) throws IOExc
eption {
FileStatus fileStatus = fs.getFileStatus(path);
if (fileStatus.isDirectory()) {
System.out.printf("path %s is dir\n", path);
return;
}
long fileLen = fileStatus.getLen();
long accessTime = fileStatus.getAccessTime();
long modifyTime = fileStatus.getModificationTime();
String owner = fileStatus.getOwner();
String group = fileStatus.getGroup();

System.out.printf("path %s is file, fileLen: %d, accessTime: %d, modifyTime: %
d, owner: %s, group: %s\n",
path, fileLen, accessTime, modifyTime, owner, group);
}

// The default verification type is `COMPOSITE-CRC32C`
private static void getFileCheckSum(FileSystem fs, Path path) throws IOExceptio
n {
FileChecksum checksum = fs.getFileChecksum(path);
System.out.printf("path %s, checkSumType: %s, checkSumCrcVal: %d\n",
path, checksum.getAlgorithmName(), ByteBuffer.wrap(checksum.getBytes()).getInt
());
}

private static void copyFileFromLocal(FileSystem fs, Path chdfsPath, Path local
Path) throws IOException {
fs.copyFromLocalFile(localPath, chdfsPath);
}

private static void copyFileToLocal(FileSystem fs, Path chdfsPath, Path localPa
th) throws IOException {
fs.copyToLocalFile(chdfsPath, localPath);
}
```

```
private static void renamePath(FileSystem fs, Path oldPath, Path newPath) throw
s IOException {
fs.rename(oldPath, newPath);
}

private static void listDirPath(FileSystem fs, Path dirPath) throws IOException
{
FileStatus[] dirMemberArray = fs.listStatus(dirPath);

for (FileStatus dirMember : dirMemberArray) {
System.out.printf("dirMember path %s, fileLen: %d\n", dirMember.getPath(), dirM
ember.getLen());
}
}

// The recursive deletion flag is used to delete directories
// If recursion is `false` and `dir` is not empty, the operation will fail
private static void deleteFileOrDir(FileSystem fs, Path path, boolean recursiv
e) throws IOException {
fs.delete(path, recursive);
}

private static void closeFileSystem(FileSystem fs) throws IOException {
fs.close();
}

public static void main(String[] args) throws IOException {
// Initialize a file
FileSystem fs = initFS();

// Create a file
Path chdfsFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, chdfsFilePath);

// Read a file
readFile(fs, chdfsFilePath);

// Query a file or directory
queryFileOrDirStatus(fs, chdfsFilePath);

// Get a file checksum
getFileCheckSum(fs, chdfsFilePath);

// Copy a file from the local system
Path localFilePath = new Path("file:///home/hadoop/ofs_demo/data/exampleobject.
txt");
```

```
copyFileFromLocal(fs, chdfsFilePath, localFilePath);

// Get a file to the local system
Path localDownFilePath = new Path("file:///home/hadoop/ofs_demo/data/exampleobj
ect.txt");
copyFileToLocal(fs, chdfsFilePath, localDownFilePath);

// Rename
Path newPath = new Path("/doc/example.txt");
renamePath(fs, chdfsFilePath, newPath);

// Delete a file
deleteFileOrDir(fs, newPath, false);

// Create a directory
Path dirPath = new Path("/folder");
mkdir(fs, dirPath);

// Create a file in a directory
Path subFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, subFilePath);

// List directories
listDirPath(fs, dirPath);

// Deletes a directory
deleteFileOrDir(fs, dirPath, true);

// Close a file system
closeFileSystem(fs);
}
}
```

3. Compile and run.

Note：

- Before running the code, please be sure to correctly set `classpath`, which must contain the paths of the Hadoop `common` package and the CHDFS package.
- For an EMR environment, if you follow the steps as detailed in Mounting CHDFS Instance, the Hadoop `common` package is generally in the `/usr/local/service/hadoop/share/hadoop/common/` directory, and the CHDFS package is generally in the `/usr/local/service/hadoop/share/hadoop/common/lib/` directory.

# Deleting File System

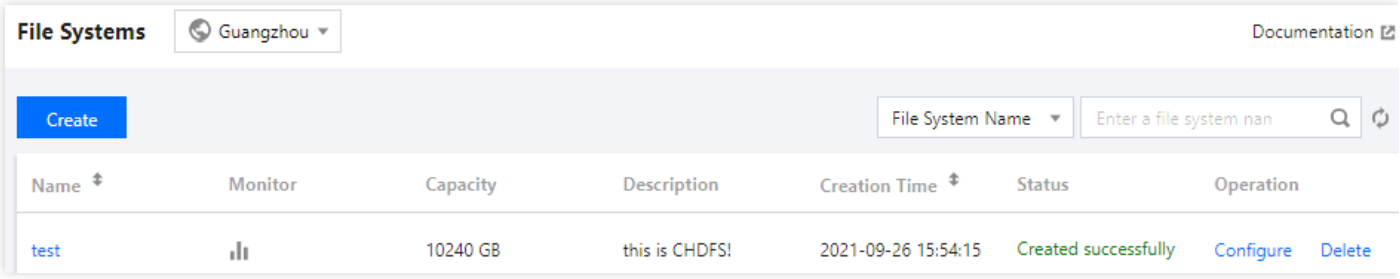
Last updated：2022-03-30 09:30:26

This document describes how to delete a file system in the CHDFS console.

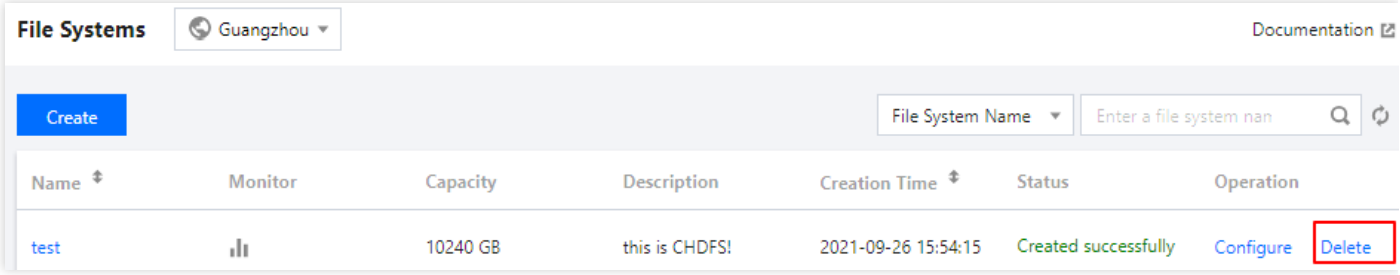
Note：
After a file system is deleted, all the configuration under it will be cleared and cannot be restored; therefore, please do so with caution.

## Directions

1. Log in to the CHDFS console.
2. On the left sidebar, click **File Systems**.
3. On the file system page, select the region of the target file system, such as Guangzhou.




4. Find the file system to be deleted in the file system list and click **Delete** on the right.



5. In the pop-up window, click **Delete**.