Alibaba Cloud E-MapReduce

Developer Guide

Issue: 20181211

Legal disclaimer

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

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

secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).

6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

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

Contents

1 Preparations

1.1 Development preparations

This topic describes the preparations required for E-MapReduce development.

- You have activated the Alibaba Cloud service and created an AccessKey ID and AccessKey Secret. ID represents the AccessKey ID and KEY represents the AccessKey Secret. If you have not activated or do not know about the OSS service, log on to the *OSS* product page for more help.
- You have a basic understanding of Spark, Hadoop, Hive, and Pig. This topic does not describe the development practices of Spark, Hadoop, Hive, and Pig. For more information, see the *Apache website*.
- You have a basic understanding of Alibaba Cloud *Elastic MapReduce (E-MapReduce) development components*. The open source community is welcome to contribute to the project. For example, we appreciate your problem feedback, bug fixes, and new components.

1.2 Configure the OSS URI to use E-MapReduce

This topic describes how to configure the OSS URI to use E-MapReduce.

OSS URI

When using E-MapReduce, you can use two types of OSS URIs:

- native URI： oss://[accessKeyId:accessKeySecret@]bucket[.endpoint]/object/path

  This URI is used for specifying input and output data sources for a job, which is similar to hdfs ://. When you operate OSS data, you can configure the accessKey ID, accessKey Secret, and the endpoint. Alternatively, you can specify the accessKey ID, accessKey Secret, and the endpoint in the URI.
- ref URI: ossref://bucket/object/path

  It is only valid in the configuration of an E-MapReduce job and is used to specify the resources needed for running the job. In the following job configuration example:

We call prefixes, such as oss and ossref, as schemes. Pay special attention to the scheme difference in URI.

Note:

Currently, operations only support OSS for standard storage types.

- E-MapReduce uses the multipart mode to upload large files to OSS. Note that if your job is interrupted, some of the result data remains in OSS. You must remove it manually. The procedure here is the same with the use of HDFS. One difference, however, is that E-MapReduce uses the multipart mode to upload large files. The file fragments are uploaded to the OSS fragment management. Therefore, you must delete the remaining job files in OSS file management and clean the file fragments in OSS fragment management. Otherwise, you are charged for data storage.
- Except for the preceding manual cleanup, you can also configure the lifecycle of the fragments, so that the expired fragments are automatically cleaned. For more information, see *Lifecycle management of OSS files*.

1.3 Use sample projects

This sample project is a complete, compilable, and runnable project, including the sample code of MapReduce, Pig, Hive, and Spark.

Sample project

See the open source project. The details are as follows:

- MapReduce

  WordCount: Counts the number of words.
- Hive

  sample.hive: A simple search for the tables.
- Pig

  sample.pig: OSS data instances processed by Pig.
- Spark
  - SparkPi: Calculates Pi.
  - SparkWordCount: Counts the number of words.
  - LinearRegression: Linear regression.
  - OSSSample: OSS sample.
  - ODPSSample: ODPS sample.
  - MNSSample: MNS sample.
  - LoghubSample: Loghub sample.

Dependencies

- Test data (under the data directory)
  - The_Sorrows_of_Young_Werther.txt: Used as the input data of WordCount (MapReduce/Spark).
  - patterns.txt: Filter characters of the WordCount (MapReduce) job.
  - u.data: The data of the test table of the sample.hive script.
  - abalone: Test data for linear regression algorithms.
- The JAR dependencies (under the lib directory)

  tutorial.jar: The JAR dependencies required for the sample.pig job.

Preparations

This project provides some test data. You can simply upload it to OSS to use it. For other samples, such as MaxCompute, MNS, ONS, and Log Service, you need to prepare the data as follows:

- (Optional) Enable Log Service. See the *User Guide for Log Service*.
- (Optional) Create MaxCompute projects and tables. See *Create a MaxCompute project* and *Quick Start for MaxCompute*.
- (Optional) Use ONS. See *Quick Start for Message Queue*.
- (Optional) Use MNS. See *topics for using the Message Service console*.

Concepts

- OSSURI: oss://accessKeyId:accessKeySecret@bucket.endpoint/a/b/c.txt. Specify the input and output data sources. OSSURI is similar to URLs like hdfs://.
- The combination of AccessKey ID and AccessKey Secret is the key for you to access an Alibaba Cloud API. Click *here* to obtain it.

Run jobs in clusters

- Spark
  - SparkWordCount：

```
spark-submit --class SparkWordCount examples-1.0-SNAPSHOT-shaded.
jar <inputPath>
                <outputPath> <numPartition>
```

  Parameters are described as follows:

- inputPath: Data input path.
- outputPath: Data output path.
- numPartition: The number of RDD partitions of the input data.

— SparkPi: `spark-submit --class SparkPi examples-1.0-SNAPSHOT-shaded.jar`

— OSSSample:

```
spark-submit --class OSSSample examples-1.0-SNAPSHOT-shaded.jar <
inputPath>
                <numPartition>
```

Parameters are described as follows:

- inputPath: Data input path.
- numPartition: Input the number of data RDD partitions.

— ONSSample:

```
spark-submit --class ONSSample examples-1.0-SNAPSHOT-shaded.jar <
accessKeyId>
                <accessKeySecret> <consumerId> <topic> <subExpress
ion> <parallelism>
```

Parameters are described as follows:

- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- consumerId: See *Consumer ID description*.
- topic: Each message queue has a topic.
- subExpression: See *Message filtering*.
- parallelism: Specifies the number of receivers to consume messages in the queue.

— ODPSSample:

```
spark-submit --class ODPSSample examples-1.0-SNAPSHOT-shaded.jar <
accessKeyId>
                <accessKeySecret> <envType> <project> <table> <
numPartitions>
```

Parameters are described as follows:

- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- envType: 0 indicates the public network, and 1 indicates the private network. Select 0 for local debugging, and select 1 for execution on E-MapReduce.

- project: see *ODPS Quick Start*.
- numPartition: The number of RDD partitions of the input data.

— MNSSample:

```
spark-submit --class MNSSample examples-1.0-SNAPSHOT-shaded.jar <
queueName>
                <accessKeyId> <accessKeySecret> <endpoint>
```

Parameters are described as follows:

- queueName: Queue name. See *MNS glossary*.
- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- endpoint: Address for accessing the queue data.

— LoghubSample:

```
spark-submit --class LoghubSample examples-1.0-SNAPSHOT-shaded.jar
 <sls project> <sls
                logstore> <loghub group name> <sls endpoint> <
access key id> <access key secret> <batch
                interval seconds>
```

Parameters are described as follows:

- sls project: The name of the Log Service project.
- sls logstore: Logstore name.
- loghub group name: The name of the group that consumes Logstore data in jobs. You can specify a name as needed. When the values of sls project and sls store are the same, jobs with the same group name consume data in sls store collaboratively. Jobs with different group names consume data in sls store separately.
- sls endpoint: See *Service endpoint*.
- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- batch interval seconds: The interval between Spark Streaming jobs (unit: seconds).

— LinearRegression:

```
spark-submit --class LinearRegression examples-1.0-SNAPSHOT-shaded
.jar <inputPath>
                <numPartitions>
```

Parameters are described as follows:

- inputPath: Data input path.

- numPartition: The number of RDD partitions of the input data.

- MapReduce

  — WordCount:

  ```
  hadoop jar examples-1.0-SNAPSHOT-shaded.jar WordCount
                  -Dwordcount.case.sensitive=true <inputPath> <
  outputPath> -skip <patternPath>
  ```

  Parameters are described as follows:

  - inputPath: Data input path.
  - outputPath: Data output path.
  - patternPath: The file that contains characters to be filtered. You can use data/patterns.txt .

- Hive

  — `hive -f sample.hive -hiveconf inputPath=<inputPath>`

  Parameters are described as follows:

  - inputPath: Data input path.

- Pig

  —
  ```
  pig -x mapreduce -f sample.pig -param tutorial=<tutorialJarPath> -
  param
                  input=<inputPath> -param result=<resultPath>
  ```

  Parameters are described as follows:

  - tutorialJarPath: The JAR dependencies. You can use lib/tutorial.jar.
  - inputPath: Data input path.
  - resultPath: Data output path.

Note:

- If you are working on E-MapReduce, upload the testing data and the JAR dependencies to OSS. The path rule should follow the definition of OSSURI, as described above.
- If used in a cluster, it can be stored locally.

Run Locally

Here we describe how to run a Spark program locally to visit Alibaba Cloud's data sources, such as OSS. If you want to debug and run the program locally, we recommend that you use some

development tools, such as IntelliJ IDEA or Eclipse, especially when using Windows. Otherwise, you need to configure Hadoop and Spark runtime environments on Windows machines.

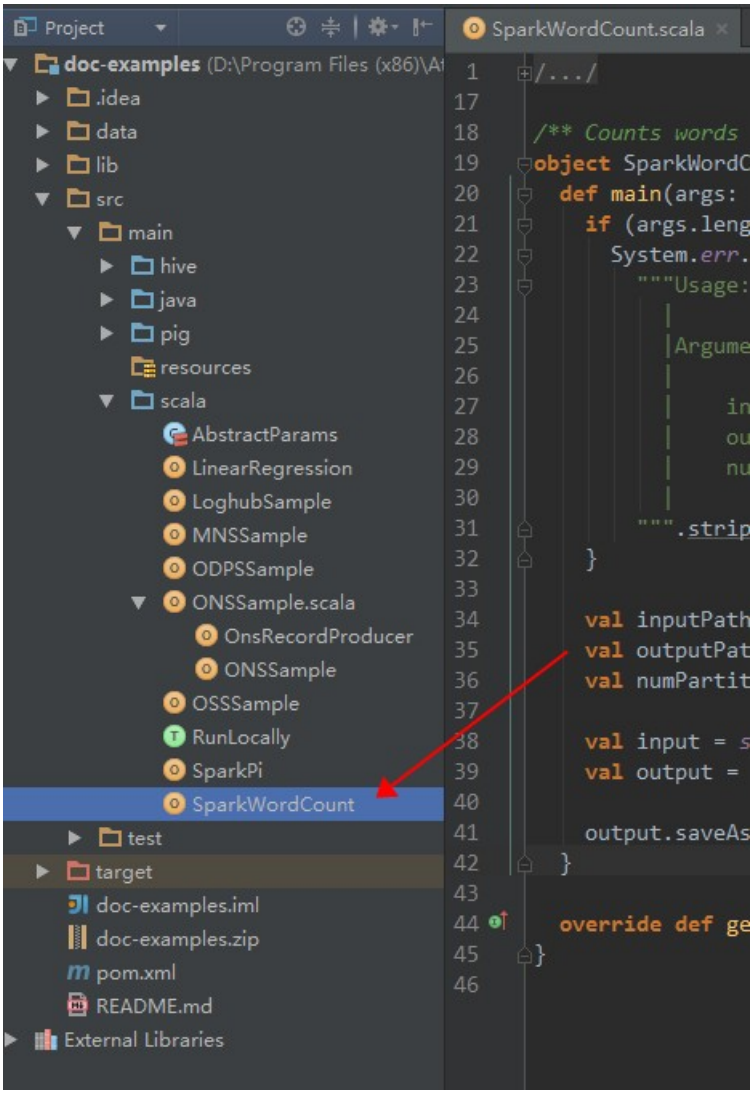
- IntelliJ IDEA

  — Preparations

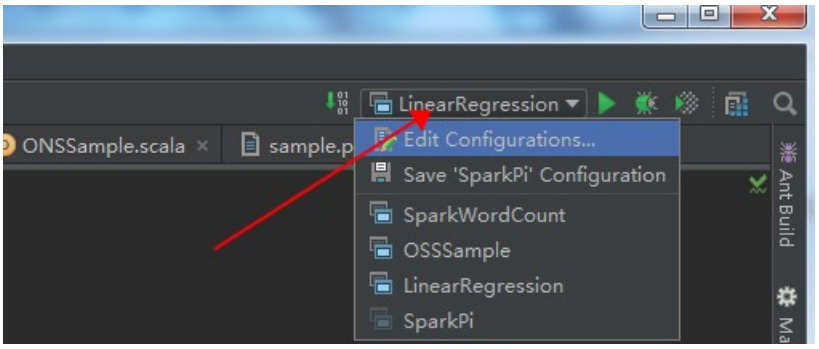
  Install IntelliJ IDEA, Maven, Maven plugin for IntelliJ IDEA, Scala and Scala plugin for IntelliJ IDEA.

  — Development process

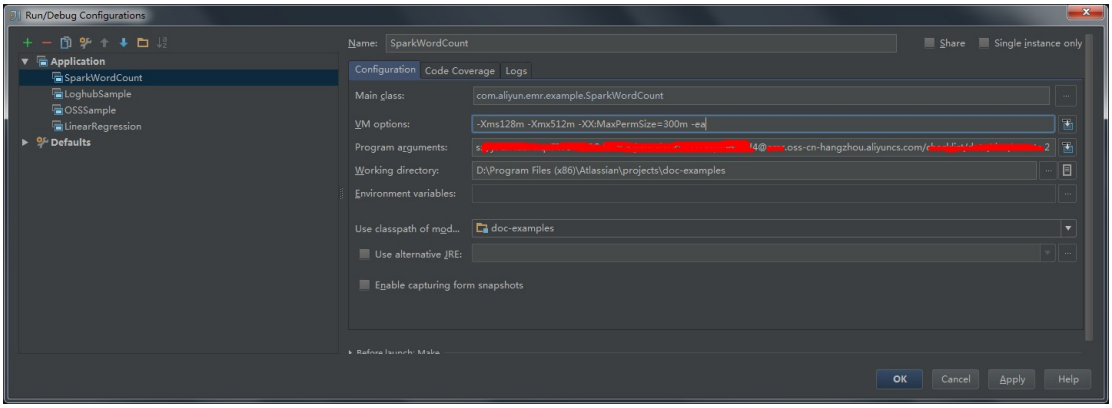
  1. Double-click to enter SparkWordCount.scala.



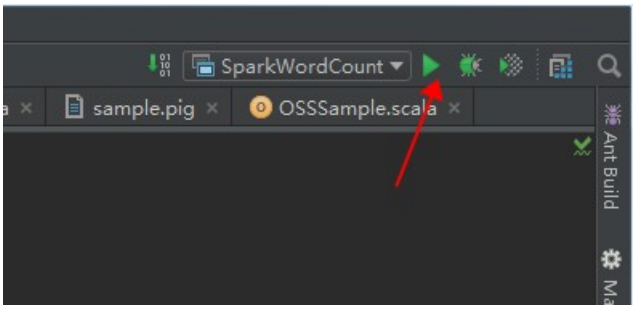
  2. Enter the job configuration page from the arrow as shown in the following figure.



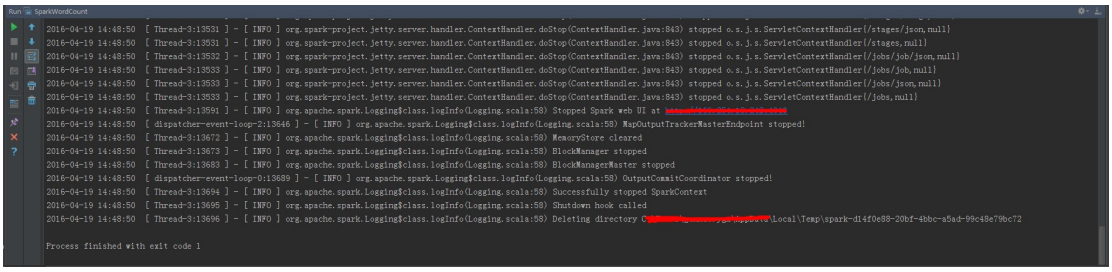
3. Select SparkWordCount and enter the required job parameters in the job parameter box.



4. Click **OK**.
5. Click Run to run the job.



6. View job logs



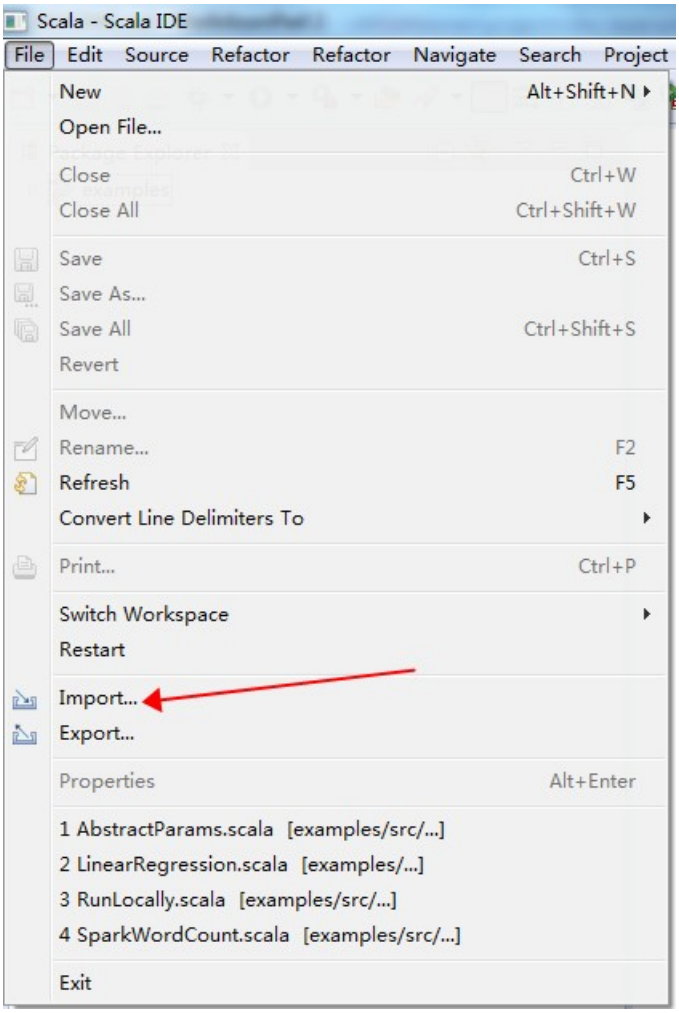
- Scala IDE for Eclipse

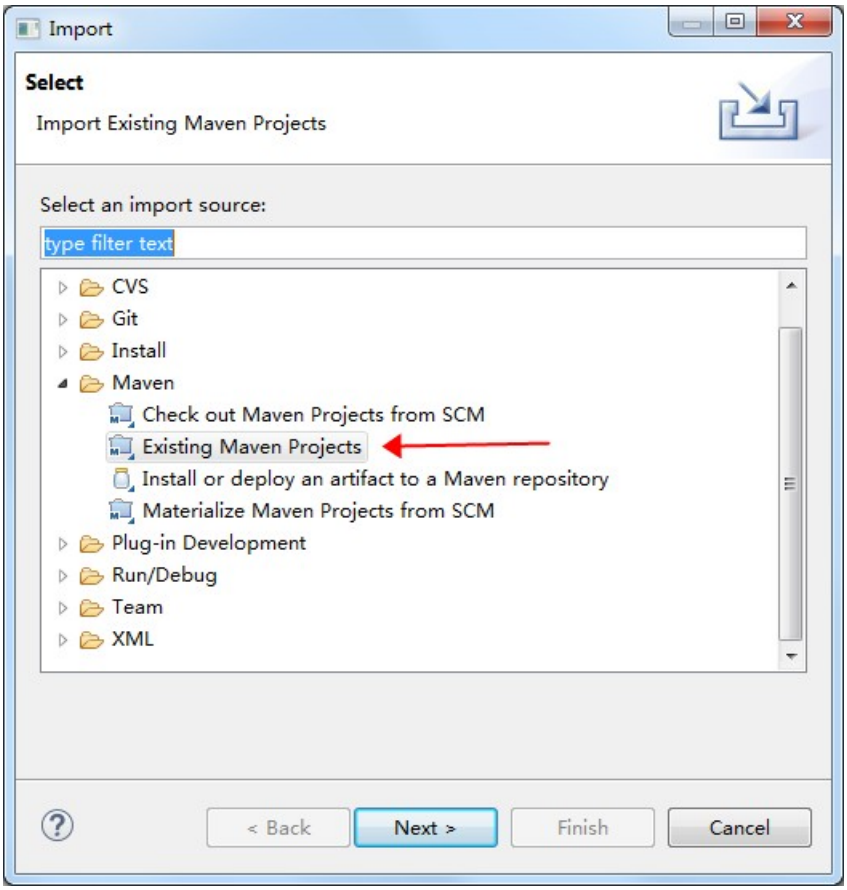
  — Preparations

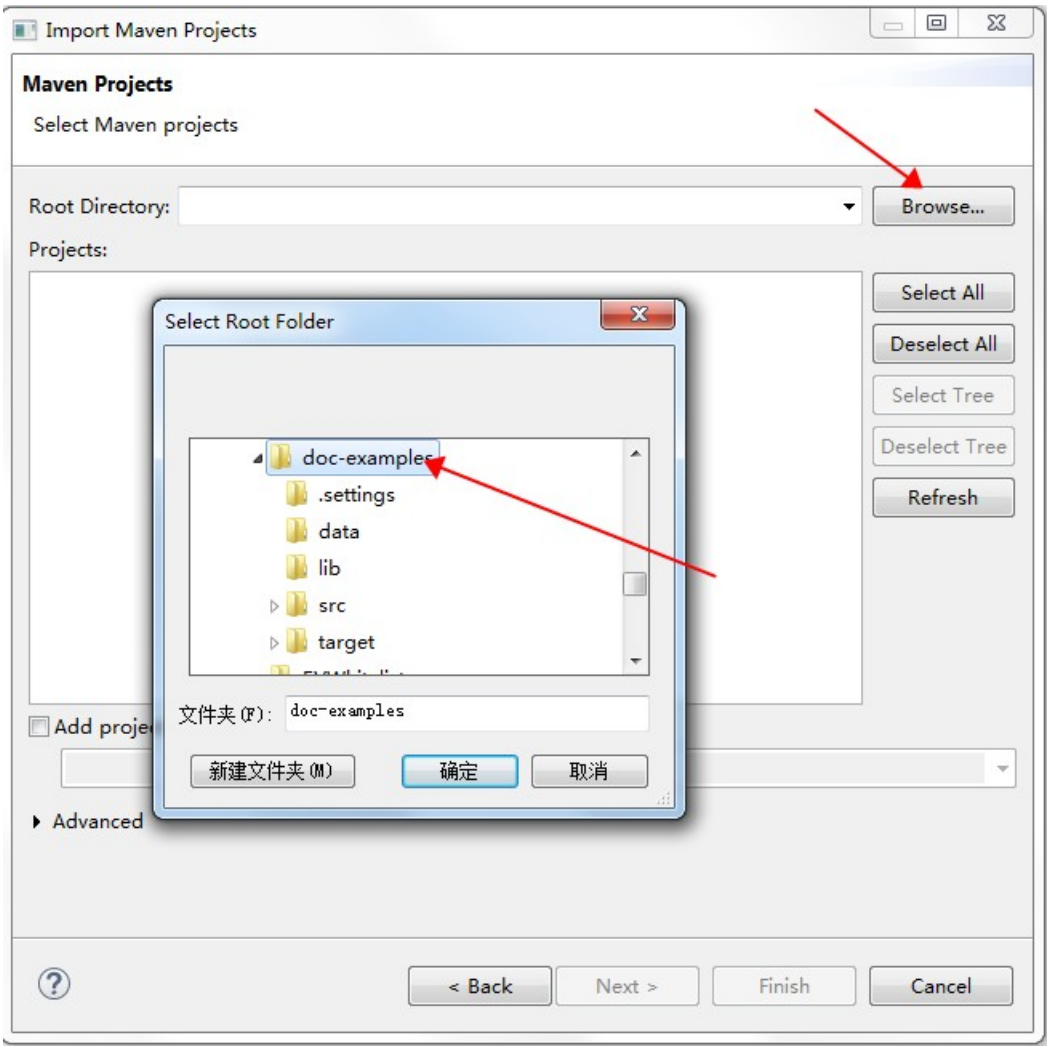
  Install the Scala IDE for Eclipse, Maven and the Maven plugin for Eclipse.

— Development process

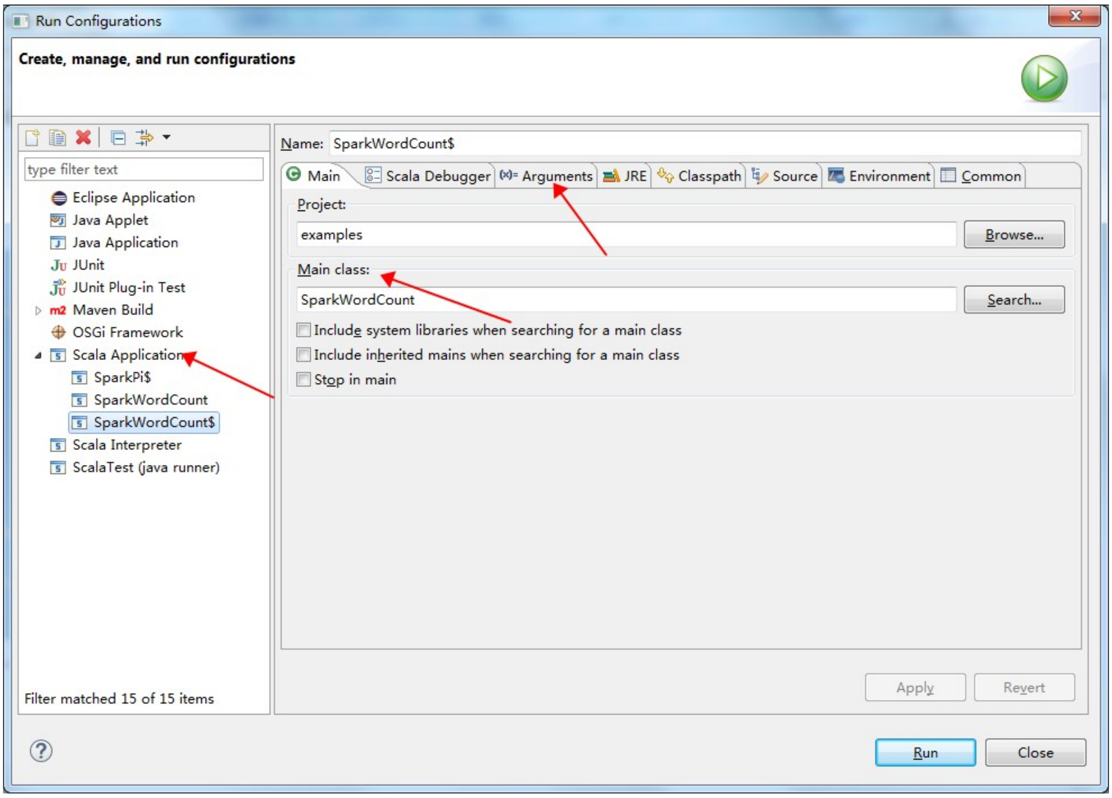
1. Import a project as described in the figure below:




Import
Select
Import Existing Maven Projects
Select an import source:
type filter text
CVS
Git
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
Plug-in Development
Run/Debug
Team
XML
< Back
Next >
Finish
Cancel

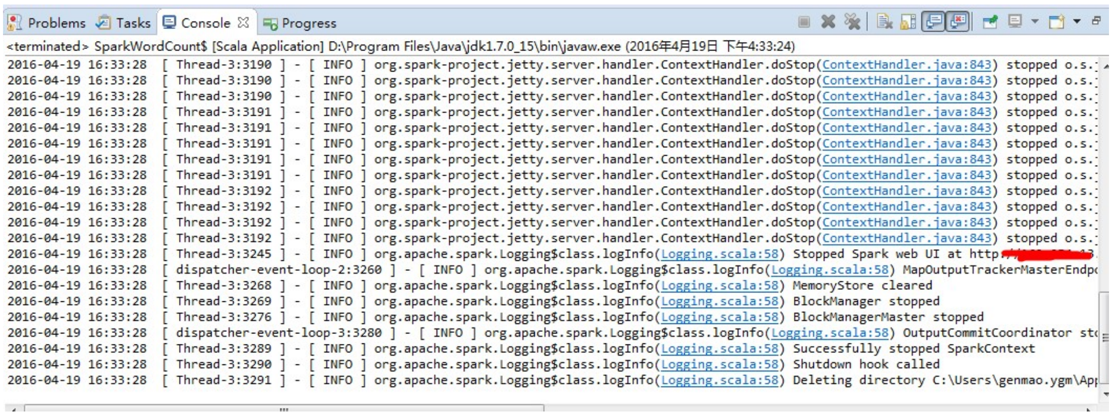


2. The shortcut for Run as Maven build is Alt + Shift + X, M. You can also right-click the project name and choose **Run As** > **Maven build**.
3. Right-click on the job to run after it has been compiled, select **Run Configuration** to enter the configuration page.
4. In the configuration page, select **Scala Application** and configure the Main Class and parameters of the job, As shown in the following figure:



5. Click **Run**.

6. View the output log of the console, as shown in the following figure:



2 Spark

2.1 Preparations

This topic describes how to prepare for Spark development.

Install E-MapReduce SDK

You can install E-MapReduce SDK using either of the following methods:

- Method 1: Directly use the JAR package in Eclipse. Follow these steps:
  1. Download the *dependencies* required by E-MapReduce from the Maven repository.
  2. Copy the required JAR files to your project directory. (Choose SDK version 2.10 or 2.11 based on the Spark version you use in the cluster in which your jobs run. Version 2.10 is recommended for Spark1.x, and 2.11 for Spark2.x.)
  3. Right-click the project name in Eclipse, and then choose **Properties** > **Java Build Path** > **Add JARs**.
  4. Select the downloaded SDK.
  5. After you have completed the preceding steps, you can read and write data in OSS, Log Service, MNS, ONS, Table Store, and MaxCompute.
- Method 2: Edit the Maven project to add the dependencies as follows:

```
<! -- Support for OSS data sources -->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-core</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! -- Support for Table Store data sources -->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-tablestore</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! -- Support for Message Service (MNS), Open Notification
Service (ONS), Log Service, MaxCompute data sources (Spark 1.x
environment) -->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-mns_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-logservice_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
```

```
        <artifactId>emr-maxcompute_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-ons_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! -- Support for MNS, ONS, Log Service, MaxCompute data sources
 (spark 2.x environment) -->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-mns_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-logservice_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-maxcompute_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-ons_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
```

Debug Spark code locally

The *spark.hadoop.mapreduce.job.run-local* property is for scenarios where you need to debug for Spark reading and writing OSS. Use the default settings in other scenarios.

If you need to debug for Spark reading and writing OSS, configure the SparkConf object by setting *spark.hadoop.mapreduce.job.run-local* to `true`. See the following example code:

```
val conf = new SparkConf().setAppName(getAppName).setMaster("local[4
]")
   conf.set("spark.hadoop.fs.oss.impl", "com.aliyun.fs.oss.nat.
NativeOssFileSystem")
   conf.set("spark.hadoop.mapreduce.job.run-local", "true")
   val sc = new SparkContext(conf)
   val data = sc.textFile("oss://...")
   println(s"count: ${data.count()}")
```

Third-party dependency description

To allow E-MapReduce to access Alibaba Cloud data sources (such as OSS and MaxCompute), you need to install the required third-party dependencies.

See the *pom* file to add or delete required third-party dependencies.

OSS output directory settings

Configure the parameter **`fs.oss.buffer.dirs`** to the local hadoop configuration file. The parameter value is a local directory path. If the parameter value is not set, a null pointer exception will occur when OSS data is written during the process of running Spark.

Clean up data

When a Spark job fails, the data generated is not deleted automatically. If a job fails, check the OSS output directory to see if any file exists. You must also check OSS fragment management for any fragments that have not been submitted. Remove fragments that are found.

Use PySpark

If you use Python to create a Spark job, see the *instructions* on configuring the environment.

2.2 Parameter description

The description of the parameters in Spark code

The following parameters can be configured in Spark code:

Property	Default	Description
spark.hadoop.fs.oss.accessKeyId	None.	(Optional) The AccessKey ID required for accessing OSS.
spark.hadoop.fs.oss.accessKeySecret	None.	(Optional) The AccessKey Secret required for accessing OSS.
spark.hadoop.fs.oss.securityToken	None.	(Optional) The STS token required for accessing OSS.
spark.hadoop.fs.oss.endpoint	None.	(Optional) The endpoint used for accessing OSS.
spark.hadoop.fs.oss.multipart.thread.number	5	The number of threads (concurrency) used by OSS for the upload part - copy operation.
spark.hadoop.fs.oss.copy.simple.max.byte	134217728	The file size limit for copying files between buckets in OSS using common APIs.
spark.hadoop.fs.oss.multipart.split.max.byte	67108864	The file multipart limit for copying files between buckets in OSS using common APIs.

Property	Default	Description
spark.hadoop.fs.oss.multipart.split.number	5	The number of multipart files for copying files between buckets in OSS using common APIs. By default, the number is the same as the number of threads (concurrency) used.
spark.hadoop.fs.oss.impl	com.aliyun.fs.oss.nat.NativeOssFileSystem	The implementation class for the native file system of OSS.
spark.hadoop.fs.oss.buffer.dirs	/mnt/disk1,/mnt/disk2,…	The OSS local temporary directory. It uses a data disk in the cluster by default.
spark.hadoop.fs.oss.buffer.dirs.exists	false	Whether the OSS temporary directory exists.
spark.hadoop.fs.oss.client.connection.timeout	50000	Timeout period for OSS client connections (unit: milliseconds).
spark.hadoop.fs.oss.client.socket.timeout	50000	Timeout period for OSS Client sockets (unit: milliseconds).
spark.hadoop.fs.oss.client.connection.ttl	-1	The value of time-to-live for clients
spark.hadoop.fs.oss.connection.max	1024	The maximum connections allowed.
spark.hadoop.job.runlocal	false	If the data source is OSS and you need to run and debug Spark code locally, set this parameter to true. Otherwise, set this parameter to false.
spark.logservice.fetch.interval.millis	200	The interval for receivers to retrieve data from LogHub.
spark.logservice.fetch.inOrder	true	Whether to consume the Shard data in order after the data has been parted.
spark.logservice.heartbeat.interval.millis	30000	The heartbeat interval for the data consumption processes (unit: milliseconds).

Property	Default	Description
spark.mns.batchMsg.size	16	The number of MNS messages to fetch in bulk, with a maximum of 16.
spark.mns.pollingWait.seconds	30	The polling wait time if the MNS queue is empty.
spark.hadoop.io.compression.codec.snappy.native	false	Whether the Snappy files are in the standard Snappy format. By default, Hadoop recognizes Snappy files edited in Hadoop.

2.3 Operate OSS files

This topic describes the possible issues of using OSS SDK and the recommended practices.

Issues in the use of OSS SDK

If you cannot directly use OSS SDK to operate OSS files in Spark or Hadoop jobs, it is because conflicts exist between the http-client-4.4.x version on which the OSS SDK is dependent and the http-client version in the Spark or Hadoop running environments. If such operations are required, you must resolve the dependency conflicts first. In fact, Spark and Hadoop are already seamlessly compatible with OSS in E-MapReduce, and you can operate OSS files as you are using HDFS.

- The current E-MapReduce environment supports MetaService, which allows you to access OSS data in the E-MapReduce environment without an AccessKey. The previous way of using an AccessKey is still supported. Internal network endpoints are preferred when using OSS.
- When you test locally, you need to use a public network endpoint of OSS so that you can access OSS data from your local machine.

For more information about a complete endpoint list, see *OSS endpoints*.

Recommended practices (without using an AccessKey)

We recommend that you use the following methods to query files under OSS directories:

```
[Scala]
   import org.apache.hadoop.conf.Configuration
   import org.apache.hadoop.fs.{ Path, FileSystem}
   val dir = "oss://bucket/dir"
   val path = new Path(dir)
   val conf = new Configuration()
   conf.set("fs.oss.impl", "com.aliyun.fs.oss.nat.NativeOssFileSystem
")
   val fs = FileSystem.get(path.toUri, conf)
```

```
   val fileList = fs.listStatus(path)
   ...
   [Java]
   import org.apache.hadoop.conf.Configuration;
   import org.apache.hadoop.fs.Path;
   import org.apache.hadoop.fs.FileStatus;
   import org.apache.hadoop.fs.FileSystem;
   String dir = "oss://bucket/dir";
   Path path = new Path(dir);
   Configuration conf = new Configuration();
   conf.set("fs.oss.impl", "com.aliyun.fs.oss.nat.NativeOssFileSystem
");
   FileSystem fs = FileSystem.get(path.toUri(), conf);
   FileStatus[] fileList = fs.listStatus(path);
   ...
```

2.4 Use Spark to access OSS

E-MapReduce supports *MetaService*, which allows you to access OSS data in the E-MapReduce environment without an AccessKey. The previous way of using an AccessKey and endpoint is also supported. Make sure that you use an internal IP address for the OSS endpoint. For more information about a complete list of endpoints, see *OSS endpoints*.

Allow Spark to access OSS

The following example shows how Spark reads data from OSS without an AccessKey and writes the processed data back to OSS.

```
val conf = new SparkConf().setAppName("Test OSS")
    val sc = new SparkContext(conf)
    val pathIn = "oss://bucket/path/to/read"
    val inputData = sc.textFile(pathIn)
    val cnt = inputData.count
    println(s"count: $cnt")
    val outputPath = "oss://bucket/path/to/write"
    val outpuData = inputData.map(e => s"$e has been processed.")
    outpuData.saveAsTextFile(outputPath)
```

Appendix

For the complete sample code, see:

- *Allow Spark to access OSS*

2.5 Use MaxCompute in Spark

This topic describes how to use the E-MapReduce SDK to read and write MaxCompute data in Spark.

Allow Spark to access MaxCompute

1. Initialize an OdpsOps object. In Spark, data operations of MaxCompute are performed using the OdpsOps class. Follow these steps to create an OdpsOps object:

```
import com.aliyun.odps.TableSchema
     import com.aliyun.odps.data.Record
     import org.apache.spark.aliyun.odps.OdpsOps
     import org.apache.spark.{ SparkContext, SparkConf}
     object Sample {
       def main(args: Array[String]): Unit = {
         // == Step-1 ==
         val accessKeyId = "<accessKeyId>"
         val accessKeySecret = "<accessKeySecret>"
         // Take the internal network address as an example
         val urls = Seq("http://odps-ext.aliyun-inc.com/api", "http
://dt-ext.odps.aliyun-inc.com")
         val conf = new SparkConf().setAppName("Test Odps")
         val sc = new SparkContext(conf)
         val odpsOps = OdpsOps(sc, accessKeyId, accessKeySecret,
urls(0), urls(1))
         // A part of the calling code is shown as follows
         // == Step-2 ==
         ...
         // == Step-3 ==
         ...
       }
       // == Step-2 ==
       // Method definition 1
       // == Step-3 ==
       // Method definition 2
     }
```

2. Load the table data from MaxCompute to Spark. By using the readTable method of the OdpsOps object, you can load a MaxCompute table to Spark and create an RDD as follows:

```
// == Step-2 ==
         val project = <odps-project>
         val table = <odps-table>
         val numPartitions = 2
         val inputData = odpsOps.readTable(project, table, read,
numPartitions)
         inputData.top(10).foreach(println)
         // == Step-3 ==
         ...
```

In the preceding code, you must define a read function to parse and pre-process the MaxCompute table data, as follows:

```
def read(record: Record, schema: TableSchema): String = {
           record.getString(0)
```

```
        }
```

This function loads the first column of the MaxCompute table into the Spark runtime environment.

3. Save the result data in Spark to the MaxCompute table. Using the saveToTable method of the OdpsOps object, you can save Spark RDD to MaxCompute.

```
val resultData = inputData.map(e => s"$e has been processed.")
        odpsOps.saveToTable(project, table, dataRDD, write)
```

In the preceding code, you must define a write function for data pre-processing before writing to the MaxCompute table as follows:

```
def write(s: String, emptyReord: Record, schema: TableSchema): Unit
 = {
          val r = emptyReord
          r.set(0, s)
        }
```

This function writes RDD data in each row to the first column of the corresponding MaxCompute table.

4. Notation of the partition table parameters.

The SDK supports reading and writing MaxCompute partition tables. The standard naming of the tables is partition_column_name=partition_name (with multiple partitions separated by commas). Assume that we have partition columns pt and ps:

- Read the table data of the partition where pt is 1.
- Read the table data of the partition where pt is 1 and ps is 2.

Appendix

For the complete sample code, see:

- *Allow Spark to access MaxCompute*

2.6 Use Spark Streaming to consume MQ data

This topic describes how to use Spark Streaming to consume MQ data and count the words in each batch.

Use Spark to access MQ

The following example shows how to use Spark Streaming to consume MQ data and count the words in each batch.

```
val Array(cId, topic, subExpression, parallelism, interval) = args
```

```
    val accessKeyId = "<accessKeyId>"
    val accessKeySecret = "<accessKeySecret>"
    val numStreams = parallelism.toInt
    val batchInterval = Milliseconds(interval.toInt)
    val conf = new SparkConf().setAppName("Test ONS Streaming")
    val ssc = new StreamingContext(conf, batchInterval)
    def func: Message => Array[Byte] = msg => msg.getBody
    val onsStreams = (0 until numStreams).map { i =>
      println(s"starting stream $i")
      OnsUtils.createStream(
        ssc,
        cId,
        topic,
        subExpression,
        accessKeyId,
        accessKeySecret,
        StorageLevel.MEMORY_AND_DISK_2,
        func)
    }
    val unionStreams = ssc.union(onsStreams)
    unionStreams.foreachRDD(rdd => {
      rdd.map(bytes => new String(bytes)).flatMap(line => line.split("
 "))
        .map(word => (word, 1))
        .reduceByKey(_ + _).collect().foreach(e => println(s"word: ${e
._1}, cnt: ${e._2}"))
    })
    ssc.start()
    ssc.awaitTermination()
```

Appendix

For the complete sample code, see:

- *Allow Spark to access ONS*

2.7 Consume Table Store data in Spark

This topic describes how to consume Table Store data in Spark.

Allow Spark to access Table Store

- Prepare a table

Create a table named pet. Set the name field as the primary key field.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29

name	owner	species	sex	birth	death
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

- The following example shows how Spark consumes Table Store data.

```
private static RangeRowQueryCriteria fetchCriteria() {
    RangeRowQueryCriteria res = new RangeRowQueryCriteria("pet");
    res.setMaxVersions(1);
    List<PrimaryKeyColumn> lower = new ArrayList<PrimaryKeyColumn
>();
    List<PrimaryKeyColumn> upper = new ArrayList<PrimaryKeyColumn
>();
    lower.add(new PrimaryKeyColumn("name", PrimaryKeyValue.INF_MIN
));
    upper.add(new PrimaryKeyColumn("name", PrimaryKeyValue.INF_MAX
));
    res.setInclusiveStartPrimaryKey(new PrimaryKey(lower));
    res.setExclusiveEndPrimaryKey(new PrimaryKey(upper));
    return res;
}
public static void main(String[] args) {
    SparkConf sparkConf = new SparkConf().setAppName("RowCounter");
    JavaSparkContext sc = new JavaSparkContext(sparkConf);
    Configuration hadoopConf = new Configuration();
    JavaSparkContext sc = null;
    try {
        sc = new JavaSparkContext(sparkConf);
        Configuration hadoopConf = new Configuration();
        TableStore.setCredential(
                hadoopConf,
                new Credential(accessKeyId, accessKeySecret,
securityToken));
        Endpoint ep = new Endpoint(endpoint, instance);
        TableStore.setEndpoint(hadoopConf, ep);
        TableStoreInputFormat.addCriteria(hadoopConf, fetchCriteria
());
        JavaPairRDD<PrimaryKeyWritable, RowWritable> rdd = sc.
newAPIHadoopRDD(
                hadoopConf, TableStoreInputFormat.class,
                PrimaryKeyWritable.class, RowWritable.class);
        System.out.println(
            new Formatter().format("TOTAL: %d", rdd.count()).
toString());
    } finally {
        if (sc ! = null) {
            sc.close();
        }
    }
```

```
}
```

Appendix

For the complete sample code, see:

- *Allow Spark to access Table Store*

2.8 Use Spark Streaming to consume data in Log Service

This topic describes how to use Spark Streaming to consume the log data of Log Service and count the log entries.

Allow Spark to access Log Service

The following example shows how Spark Streaming consumes the log data of Log Service and counts log entries.

- Method 1: Receiver-based DStream

```
val logServiceProject = args(0)    // project name in Log Service
    val logStoreName = args(1)     // logstore name in Log Service
    val loghubConsumerGroupName = args(2)  // loghubGroupName (The
same jobs consume the Logstore data together)
    val loghubEndpoint = args(3)  // API Endpoint (Alibaba Cloud Log
 Service)
    val accessKeyId = "<accessKeyId>"     // AccessKey ID for
accessing Log Service
    val accessKeySecret = "<accessKeySecret>" // AccessKey Secret
for accessing Log Service
    val numReceivers = args(4).toInt  // The number of receivers
started to read data in Logstore
    val batchInterval = Milliseconds(args(5).toInt * 1000) // The
interval of the Spark Streaming proccessing
    val conf = new SparkConf().setAppName("Test Loghub Streaming")
    val ssc = new StreamingContext(conf, batchInterval)
    val loghubStream = LoghubUtils.createStream(
      ssc,
      logServiceProject,
      logStoreName,
      loghubConsumerGroupName,
      loghubEndpoint,
      numReceivers,
      accessKeyId,
      accessKeySecret,
      StorageLevel.MEMORY_AND_DISK)
    loghubStream.foreachRDD(rdd => println(rdd.count()))
    ssc.start()
    ssc.awaitTermination()
```

- Method 2: Direct API-based DStream

```
val logServiceProject = args(0)
    val logStoreName = args(1)
    val loghubConsumerGroupName = args(2)
    val loghubEndpoint = args(3)
```

```
    val accessKeyId = args(4)
    val accessKeySecret = args(5)
    val batchInterval = Milliseconds(args(6).toInt * 1000)
    val zkConnect = args(7)
    val checkpointPath = args(8)
    def functionToCreateContext(): StreamingContext = {
      val conf = new SparkConf().setAppName("Test Direct Loghub
Streaming")
      val ssc = new StreamingContext(conf, batchInterval)
      val zkParas = Map("zookeeper.connect" -> zkConnect, "enable.
auto.commit" -> "false")
      val loghubStream = LoghubUtils.createDirectStream(
        ssc,
        logServiceProject,
        logStoreName,
        loghubConsumerGroupName,
        accessKeyId,
        accessKeySecret,
        loghubEndpoint,
        zkParas,
        LogHubCursorPosition.END_CURSOR)
      ssc.checkpoint(checkpointPath)
      val stream = loghubStream.checkpoint(batchInterval)
      stream.foreachRDD(rdd => {
        println(rdd.count())
        loghubStream.asInstanceOf[CanCommitOffsets].commitAsync()
      })
      ssc
    }
    val ssc = StreamingContext.getOrCreate(checkpointPath,
functionToCreateContext _)
    ssc.start()
    ssc.awaitTermination()
```

The E-MapReduce SDK with a version of 1.4.0 or higher supports Direct API. It avoids writing duplicate Loghub data to the Write Ahead Log, which means you can make sure that the data is at least once without enabling the WAL feature of Spark Streaming. The Direct API is experimental. Note that:

- For DStream actions, a commit action is required.
- For Spark Streaming, multiple actions on Logstore data sources are not supported.
- Direct API requires the support of the Zookeeper service.

Enable MetaService

In the preceding example, we explicitly pass the AccessKey to the API. With E-MapReduce SDK 1.3.2 or a later version, Spark Streaming can process Log Service data using MetaService without an AccessKey. For more information, see the description of the LoghubUtils class in E-MapReduce SDK:

```
LoghubUtils.createStream(ssc, logServiceProject, logStoreName,
loghubConsumerGroupName, storageLevel)
LoghubUtils.createStream(ssc, logServiceProject, logStoreName,
loghubConsumerGroupName, numReceivers, storageLevel)
```

```
LoghubUtils.createStream(ssc, logServiceProject, logStoreName,
loghubConsumerGroupName, storageLevel, cursorPosition, mLoghubCur
sorStartTime, forceSpecial)
LoghubUtils.createStream(ssc, logServiceProject, logStoreName,
loghubConsumerGroupName, numReceivers, storageLevel, cursorPosition,
mLoghubCursorStartTime, forceSpecial)
```

Note:

- E-MapReduce SDK supports three consumption modes for Log Service, which are BEGIN_CURSOR, END_CURSOR, and SPECIAL_TIMER_CURSOR. By default, the mode is END_CURSOR.
  - BEGIN_CURSOR: Consumes from the log header. If a checkpoint record exists, the consumption starts from the checkpoint.
  - END_CURSOR: Consumes from the end of the log. If a checkpoint record exists, the consumption starts from the checkpoint.
  - SPECIAL_TIMER_CURSOR: Consumes from a specified time. If a checkpoint record exists, the consumption starts from the checkpoint. Unit: second.
  - The three consumption modes are affected by checkpoint records. If a checkpoint record exists, the consumption always starts from the checkpoint. E-MapReduce SDK is based on the SPECIAL_TIMER_CURSOR mode. It allows you to start consumption from a specified time. In the LoghubUtils#createStream interface, the following parameters must be used in combination:
    - cursorPosition: LogHubCursorPosition.SPECIAL_TIMER_CURSOR
    - forceSpecial: true
- E-MapReduce machines cannot connect to the public network (except the master node). When configuring the Log Service endpoint, make sure that you use the internal endpoints provided by Log Service. Otherwise, requests to the Log Service fail.
- For more information about Log Service, see *Log Service*.

Appendix

For the complete sample code, see:

- *Allow Spark to access Log Service*

2.9 Use Spark Streaming to consume MNS data

This topic describes how to use Spark Streaming to consume data in MNS and calculate the word count of every batch.

Allow Spark to access MNS

The following example describes how to use Spark Streaming to consume data in MNS and calculate the word count of every batch.

```
val conf = new SparkConf().setAppName("Test MNS Streaming")
    val batchInterval = Seconds(10)
    val ssc = new StreamingContext(conf, batchInterval)
    val queuename = "queuename"
    val accessKeyId = "<accessKeyId>"
    val accessKeySecret = "<accessKeySecret>"
    val endpoint = "http://xxx.yyy.zzzz/abc"
    val mnsStream = MnsUtils.createPullingStreamAsRawBytes(ssc,
queuename, accessKeyId, accessKeySecret, endpoint,
      StorageLevel.MEMORY_ONLY)
    mnsStream.foreachRDD( rdd => {
      rdd.map(bytes => new String(bytes)).flatMap(line => line.split("
 "))
        .map(word => (word, 1))
        .reduceByKey(_ + _).collect().foreach(e => println(s"word: ${e
._1}, cnt: ${e._2}"))
    })
    ssc.start()
    ssc.awaitTermination()
```

Use Spark Streaming with MetaService

In the preceding example, we explicitly pass the AccessKey to the API. However, since E-MapReduce SDK 1.3.2, Spark Streaming can process MNS data using MetaService without an AccessKey. For more information, see the description of the MnsUtils class in E-MapReduce SDK:

```
MnsUtils.createPullingStreamAsBytes(ssc, queueName, endpoint,
storageLevel)
MnsUtils.createPullingStreamAsRawBytes(ssc, queueName, endpoint,
storageLevel)
```

Appendix

For the complete sample code, see:

- *Allow Spark to access MNS*

2.10 Use Spark to write data to HBase

This topic describes how Spark writes data to HBase. Note that computing clusters must be in the same security group as HBase clusters. Otherwise, the network cannot be connected. When you

create a cluster in E-MapReduce, make sure that you select the security group where the HBase cluster is located.

Allow Spark to access HBase

Use the following code:

```
object ConnectionUtil extends Serializable {
      private val conf = HBaseConfiguration.create()
      conf.set(HConstants.ZOOKEEPER_QUORUM,"ecs1,ecs1,ecs3")
      conf.set(HConstants.ZOOKEEPER_ZNODE_PARENT, "/hbase")
      private val connection = ConnectionFactory.createConnection(conf
)
      def getDefaultConn: Connection = connection
    }
    //Create data streaming unionStreams
    unionStreams.foreachRDD(rdd => {
      rdd.map(bytes => new String(bytes))
        .flatMap(line => line.split(" "))
        .map(word => (word, 1))
        .reduceByKey(_ + _)
        .mapPartitions {words => {
          val conn = ConnectionUtil.getDefaultConn
          val tableName = TableName.valueOf(tname)
          val t = conn.getTable(tableName)
          try {
            words.sliding(100, 100).foreach(slice => {
              val puts = slice.map(word => {
                println(s"word: $word")
                val put = new Put(Bytes.toBytes(word. _1 + System.
currentTimeMillis()))
                put.addColumn(COLUMN_FAMILY_BYTES, COLUMN_QUA
LIFIER_BYTES,
                  System.currentTimeMillis(), Bytes.toBytes(word._2))
                put
              }).toList
              t.put(puts)
            })
          } finally {
            t.close()
          }
          Iterator.empty
        }}.count()
    })
    ssc.start()
    ssc.awaitTermination()
```

Appendix

For the complete sample code, see:

- *Allow Spark to access HBase*

2.11 Use Spark Streaming to process Kafka data

This topic describes how to run Spark Streaming jobs in Hadoop clusters of E-MapReduce to process data in Kafka clusters.

Programming reference

The Hadoop and Kafka clusters in E-MapReduce are based on completely open source software. Therefore, you can refer to the corresponding official documentation during your development.

- Spark official documentation: *streaming-kafka-integration*
- Spark official documentation: *structured-streaming-kafka-integration*
- E-MapReduce-demo: *Github*

Allow Spark Streaming jobs to access Kerberos Kafka clusters

E-MapReduce allows you to create Kafka clusters based on Kerberos authentication. Jobs in Hadoop clusters can access Kerberos Kafka clusters in two ways:

- Non-Kerberos Hadoop cluster: Provide the kafka_client_jaas.conf file for Kerberos authentica tion of Kafka clusters.
- Kerberos Hadoop cluster: Cross-domain communication based on Kerberos clusters. Provide the kafka_client_jaas.conf file for Kerberos authentication of Hadoop clusters.

Both methods require that you provide the kafka_client_jaas.conf file for Kerberos authentication when you run a job.

The file format of kafka_client_jaas.conf is as follows:

```
KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    serviceName="kafka"
    keyTab="/path/to/kafka.keytab"
    principal="kafka/emr-header-1.cluster-12345@EMR. 12345. COM";
};
```

For more information about how to obtain the keytab file, see **Kerberos authentication** in the *Cite LeftUser GuideCite Right*.

Allow Spark Streaming jobs to access Kerberos Kafka clusters

When you run Spark Streaming jobs to access Kerberos Kafka clusters, you can provide the kafka_client_jaas.conf file and the kafka.keytab file as needed in the spark-submit command line parameters.

```
spark-submit --conf spark.driver.extraJavaOptions=-Djava.security.auth
.login.config={{PWD}}/kafka_client_jaas.conf --conf spark.executor.
extraJavaOptions=-Djava.security.auth.login.config={{PWD}}/kafka_clie
nt_jaas.conf --files /local/path/to/kafka_client_jaas.conf,/local/
path/to/kafka.keytab --class xx.xx.xx.KafkaSample --num-executors 2 --
executor-cores 2 --executor-memory 1g --master yarn-cluster xxx.jar
arg1 arg2 arg3
```

Note that in the kafka_client_jaas.conf file, the keytab file path must be a relative path. Make sure that you configure the path in the following format:

```
KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    serviceName="kafka"
    keyTab="kafka.keytab"
    principal="kafka/emr-header-1.cluster-12345@EMR. 12345. COM";
};
```

2.12 Use Spark and write data to MySQL

This topic describes how to run Spark jobs in Hadoop clusters of E-MapReduce to calculate the word count and write the result to MySQL. A Spark Streaming job writes data to MySQL in a similar way as a Spark job does.

Allow Spark to access MySQL

Use the following code:

```
val input = getSparkContext.textFile(inputPath, numPartitions)
    input.flatMap(_.split(" ")).map(x => (x, 1)).reduceByKey(_ + _)
      .mapPartitions(e => {
        var conn: Connection = null
        var ps: PreparedStatement = null
        val sql = s"insert into $tbName(word, count) values (?, ?)"
        try {
          conn = DriverManager.getConnection(s"jdbc:mysql://$dbUrl:$
dbPort/$dbName", dbUser, dbPwd)
          ps = conn.prepareStatement(sql)
          e.foreach(pair => {
            ps.setString(1, pair._1)
            ps.setLong(2, pair._2)
            ps.executeUpdate()
          })

          ps.close()
          conn.close()
```

```
        } catch {
          case e: Exception => e.printStackTrace()
        } finally {
          if (ps ! = null) {
            ps.close()
          }
          if (conn ! = null) {
            conn.close()
          }
        }
      Iterator.empty
    }).count()
```

Appendix

For the complete sample code, see:

- *Allow Spark to write data to RDS*

2.13 Configure Spark-Submit parameters

This topic describes how to configure spark-submit parameters in E-MapReduce.

Cluster configuration

- Software configuration

  E-MapReduce version 1.1.0

  — Hadoop 2.6.0

  — Spark 1.6.0

- Hardware configuration

  — Master node

    - 8-core, 16 GB RAM, 500 GB Ultra Disk
    - 1 unit

  — Worker node (x 10)

    ■ 8-core, 16 GB RAM, 500 GB Ultra Disk

    ■ 10 units

  — Total: 8-core 16GB (Worker) x 10 + 8-core 16GB (Master)

  Note:
  Note: Only CPU and memory resources are calculated when a job is submitted. Therefore, the disk size is not included in the total resources.

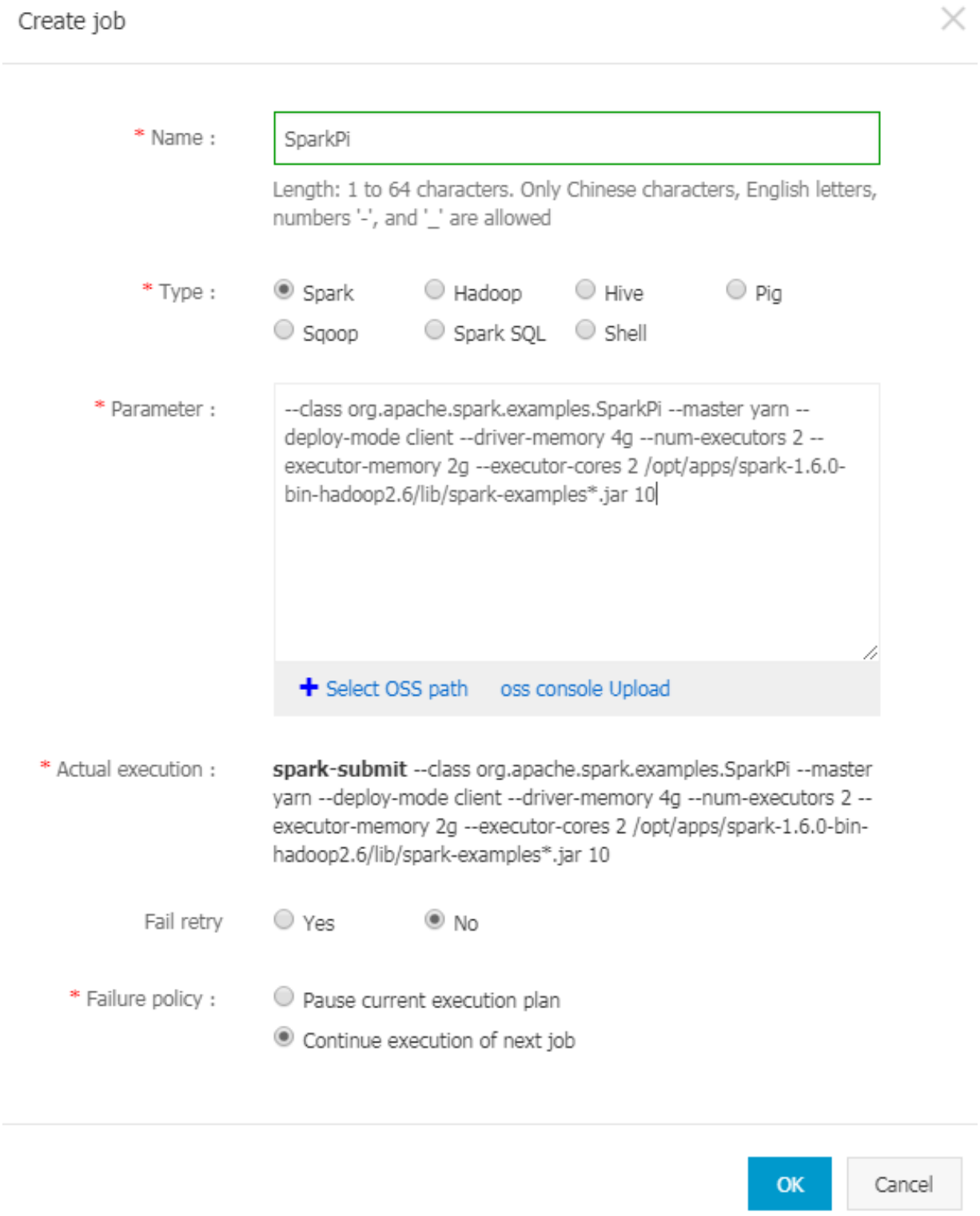
  — Yarn total available resources: 12-core, 12.8 GB (worker) x 10

Note:

By default, cores available for Yarn = number of cores x 1.5, memory available for Yarn = machine memory x 0.8.

Submit a job

After a cluster has been created, you can submit a job. First, you need to create a job in E-MapReduce with the following configuration:



The job in the preceding figure directly uses the official Spark example package, so you do not need to upload your own JAR package.

The parameters are listed as follows:

```
--class org.apache.spark.examples.SparkPi --master yarn --deploy-
mode client --driver-memory 4g --num-executors 2 --executor-memory
2g --executor-cores 2 /opt/apps/spark-1.6.0-bin-hadoop2.6/lib/spark-
examples*.jar 10
```

The details of the parameters are described as follows:

Parameter	Reference value	Description
class	org.apache.spark.examples.SparkPi	The main class of the job.
master	yarn	E-MapReduce uses the Yarn mode. Therefore, the mode must be in the Yarn mode.
	yarn-client	Equivalent to setting the master parameter to yarn and the deploy-mode parameter to client. You do not need to set the deploy-mode parameter separately here.
	yarn-cluster	Equivalent to setting the master parameter to yarn and the deploy-mode parameter to cluster. You do not need to set the deploy-mode parameter separately here.
deploy-mode	client	The client mode indicates that the AM runs on the master node. You must set the preceding master parameter to yarn at the same time when setting this parameter.
	cluster	The cluster mode indicates that the AM runs randomly on one of the worker nodes . However, if you set this parameter, you must also set the preceding master parameter to yarn.
driver-memory	4g	The memory used by the driver cannot exceed the total

Parameter	Reference value	Description
		number of cores of a single machine.
num-executors	2	The number of executors to be created.
executor-memory	2g	The maximum memory used by each executor cannot exceed the maximum available memory of a single machine.
executor-cores	2	The number of threads used by each executor, which is also the maximum number of tasks that can be executed concurrently by each executor.

Resource calculations

The resources used by jobs running in different modes and settings are shown in the following table:

- **Resource calculation for the yarn-client mode**

Node	Resource type	Resource amount (the result is calculated based on the preceding examples)
master	core	1
	mem	driver-memory = 4 GB
worker	core	num-executors * executor-cores = 4
	mem	num-executors * executor-memory = 4 GB

  - The primary program of the job (the driver program) runs on the master node. 4 GB memory (specified by --driver-memory) is allocated to the primary program based on the job settings (the memory may not be fully used).
  - Two executors (specified by --num-executors) are initiated on the worker node, with each executor allocated with 2 GB (specified by --executor-memory) memory, and each executor supports a maximum of 2 (specified by - -executor-cores) concurrent tasks.

- Resource calculation for the yarn-cluster mode

Node	Resource type	Resource amount (the result is calculated using the preceding examples)
master		A tiny client program, which is responsible for synchroniz ing job information and consumes significantly less space.
worker	core	num-executors * executor-cores + spark.driver.cores = 5
	mem	num-executors * executor-memory + driver-memory = 8g

Note:

By default, the value of spark.driver.cores is 1. You can set it to a value greater than 1.

Resource optimization

- yarn-client mode

  If you have a large job in the yarn-client mode and want to use more resources of the cluster, see the following configurations:

```
--master yarn-client --driver-memory 5g –-num-executors 20 --
executor-memory 4g --executor-cores 4
```

Note:

- When allocating memory, Spark allocates 375 MB or 7% (whichever is higher) extra memory in addition to the memory value you have set.
- When allocating memory to containers, Yarn rounds up to the nearest integer gigabyte. Here the memory must be the integral multiple of 1 GB.

Based on the preceding resource formula:

— The resource amount for the master is:

  - cores: 1
  - mem: 6 GB (5 GB + 375 MB, rounded up to 6 GB)

— The resource amount for the workers is:

- core: 20*4 = 80
- mem: 20*5 GB (4 GB + 375 MB, rounded up to 5 GB) = 100 GB

We can see that the total resources allocated to the job have not exceeded the total resources for the cluster. Following this rule, you have multiple available configurations, such as:

```
--master yarn-client --driver-memory 5g --num-executors 40 --
executor-memory 1g --executor-cores 2
```

```
--master yarn-client --driver-memory 5g --num-executors 15 --
executor-memory 4g --executor-cores 4
```

```
--master yarn-client --driver-memory 5g --num-executors 10 --
executor-memory 9g --executor-cores 6
```

In principle, you only have to make sure that the total resources calculated using the preceding formula do not exceed the total resources of the cluster. However, in production scenarios , the services of the system, HDFS, and E-MapReduce occupy certain core resources and memory resources. Therefore, if you use up the core resources and memory resources, the job performance declines or the job fails to run.

Usually, the number of executor-cores is set to the same value as the core number of the cluster. If you set the value too large, the CPU switches frequently without benefiting the performance as expected.

- yarn-cluster mode

  In the yarn-cluster mode, the driver program runs on worker nodes. Resources in the resource pool of worker nodes are used. If you want to use more resources of this cluster, use the following configurations:

  ```
  --master yarn-cluster --driver-memory 5g --num-executors 15 --
  executor-memory 4g --executor-cores 4
  ```

Recommendations on configuration

- If you set the memory to a very large value, you should pay attention to the overhead caused by garbage collection. In general, we recommend that you set an executor with memory less than or equal to 64GB.
- If you are executing an HDFS read/write job, we recommend that you set the number of concurrent jobs for each executor to be less than or equal to 5 for reading and writing.

- If you are executing an OSS read/write job, we recommend that you distribute executors to different ECS instances so that the bandwidth of every ECS instance can be used. For example, if you have 10 ECS instances, you can set num-executors to 10, and set the appropriate memory and number of concurrent jobs.
- If the code you use in the job is not thread-safe, you need to monitor whether the concurrency causes job errors when setting the executor-cores. If yes, we recommend that you set executor -cores to 1.

3 Hadoop

3.1 Parameter description

Description of parameters in Hadoop code

The following parameters can be used in Hadoop code:

Property	Default	Description
fs.oss.accessKeyId	None	(Optional) The required AccessKey ID used to access OSS.
fs.oss.accessKeySecret	None	(Optional) The required AccessKey Secret used to access OSS.
fs.oss.securityToken	None	(Optional) The required STS token used to access OSS.
fs.oss.endpoint	None	(Optional) The endpoint used to access OSS.
fs.oss.multipart.thread.number	5	The number of threads (concurrency) used by OSS for upload part copy.
fs.oss.copy.simple.max.byte	134217728	The file limit of the internal copy in OSS using common APIs.
fs.oss.multipart.split.max.byte	67108864	The file multipart limit of copying files between buckets in OSS using common APIs.
fs.oss.multipart.split.number	5	The file multipart limit of copying files between buckets in OSS that is using common APIs. By default, the limit is equal to the number of threads used in the copy.
fs.oss.impl	com.aliyun.fs.oss.nat. NativeOssFileSystem	The implementation class for the native OSS file system.
fs.oss.buffer.dirs	/mnt/disk1,/mnt/disk2,…	OSS local temporary directory . It uses a cluster data disk by default.

Property	Default	Description
fs.oss.buffer.dirs.exists	false	Indicates whether the OSS temporary directory exists.
fs.oss.client.connection.timeout	50000	The connection timeout for the OSS client (ms).
fs.oss.client.socket.timeout	50,000	The socket timeout for the OSS client (ms).
fs.oss.client.connection.ttl	-1	The connection alive time
fs.oss.connection.max	1024	The maximum connections allowed.
io.compression.codec.snappy.native	false	Indicates whether to mark Snappy files as standard. By default, Hadoop recognizes Snappy files modified by Hadoop.

3.2 Create and run MapReduce jobs

This topic describes how to create and run a MapReduce job.

Use OSS in MapReduce

To read and write data to OSS in MapReduce, configure the following parameters:

```
conf.set("fs.oss.accessKeyId", "${accessKeyId}");
    conf.set("fs.oss.accessKeySecret", "${accessKeySecret}");
    conf.set("fs.oss.endpoint","${endpoint}");
```

Parameter descriptions:

- ${accessKeyId}: The AccessKey ID of your account.
- ${accessKeySecret}: The AccessKey Secret corresponding to the AccessKey ID.
- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists, and the corresponding OSS must also be in the region in which your cluster exists.

  For more information about specific values, see *OSS endpoints*.

Word count

The following example describes how to read text from OSS and calculate the word count. The procedure is as follows:

1. Write the program

   Take the Java code as an example. Modify the WordCount sample on the official website of Hadoop as follows: Modify the instance by adding the configuration of the AccessKey ID and AccessKey Secret in the code so that the job has the permission to access OSS files.

```
package org.apache.hadoop.examples;
 import java.io.IOException;
 import java.util.StringTokenizer;
 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.IntWritable;
 import org.apache.hadoop.io.Text;
 import org.apache.hadoop.mapreduce.Job;
 import org.apache.hadoop.mapreduce.Mapper;
 import org.apache.hadoop.mapreduce.Reducer;
 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
 import org.apache.hadoop.util.GenericOptionsParser;
 public class EmrWordCount {
  public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{
     private final static IntWritable one = new IntWritable(1);
     private Text word = new Text();
     public void map(Object key, Text value, Context context
                     ) throws IOException, InterruptedException {
       StringTokenizer itr = new StringTokenizer(value.toString());
       while (itr.hasMoreTokens()) {
         word.set(itr.nextToken());
         context.write(word, one);
       }
     }
   }
   public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
     private IntWritable result = new IntWritable();
     public void reduce(Text key, Iterable<IntWritable> values,
                        Context context
                        ) throws IOException, InterruptedException {
       int sum = 0;
       for (IntWritable val : values) {
         sum += val.get();
       }
       result.set(sum);
       context.write(key, result);
     }
   }
   public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();
     String[] otherArgs = new GenericOptionsParser(conf, args).
getRemainingArgs();
     if (otherArgs.length < 2) {
       System.err.println("Usage: wordcount <in> [<in>...] <out>");
       System.exit(2);
     }
     conf.set("fs.oss.accessKeyId", "${accessKeyId}");
     conf.set("fs.oss.accessKeySecret", "${accessKeySecret}");
     conf.set("fs.oss.endpoint","${endpoint}");
     Job job = Job.getInstance(conf, "word count");
     job.setJarByClass(EmrWordCount.class);
```

```
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

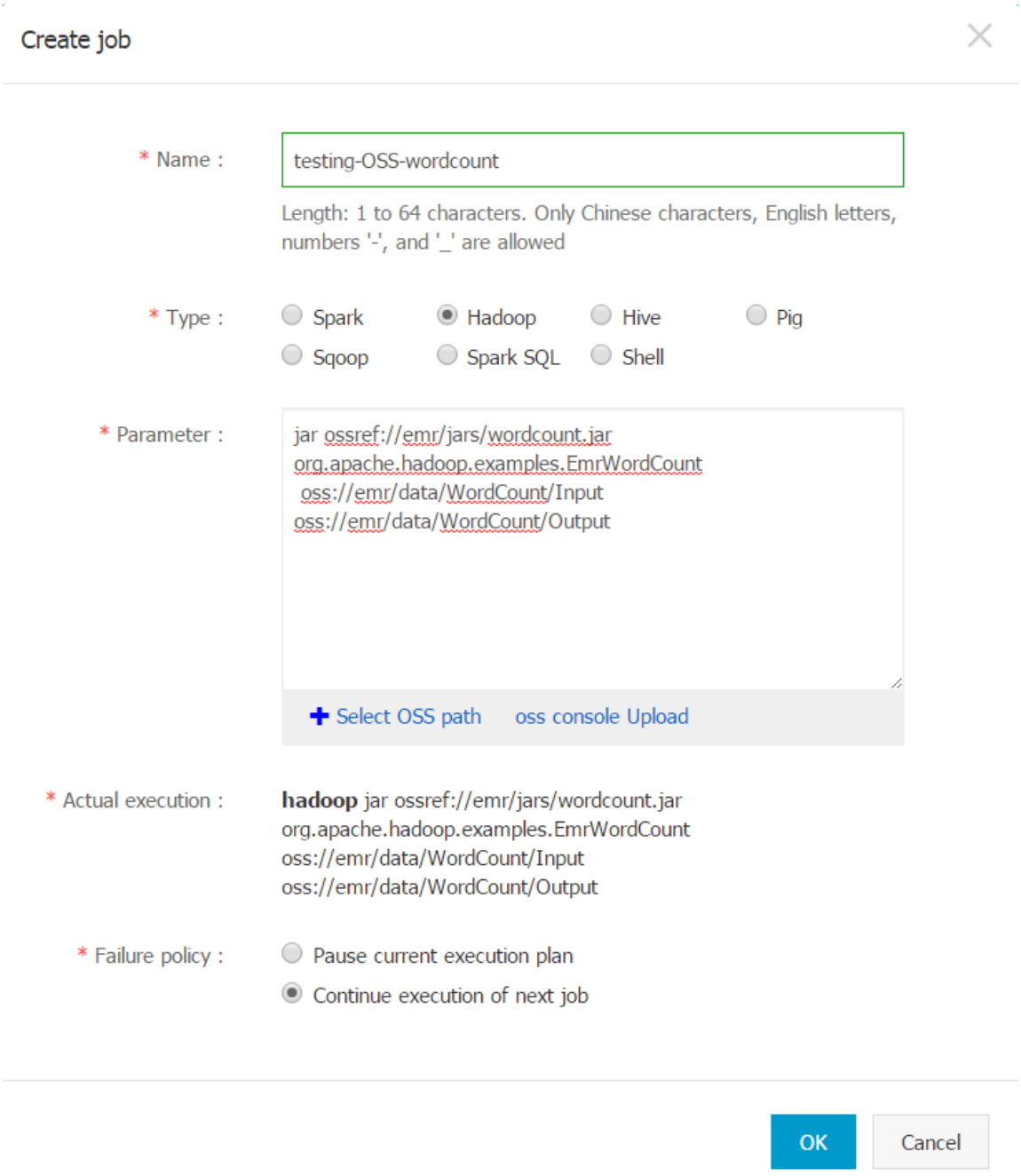
2. Compile the program

   First, you need to configure the JDK and Hadoop environments and then perform the following operations:

```
mkdir wordcount_classes
 javac -classpath ${HADOOP_HOME}/share/hadoop/common/hadoop-common
-2.6.0.jar:${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-
client-core-2.6.0.jar:${HADOOP_HOME}/share/hadoop/common/lib/commons
-cli-1.2.jar -d wordcount_classes EmrWordCount.java
 jar cvf wordcount.jar -C wordcount_classes .
```

3. Create a job

   - Upload the JAR file prepared in the previous step to OSS. Log on to the OSS website for detailed operations. Assume that the path of the JAR file on OSS is *oss://emr/jars/wordcount.jar* and the input and output paths are *oss://emr/data/WordCount/Input* and *oss://emr/data/WordCount/Output*.
   - Create *E-MapReduce jobs* as follows:



4. Create an execution plan

   Create an execution plan in E-MapReduce and add the created job to the execution plan. Select **Run Now** as the policy so that the WordCount job runs in the selected cluster.

Manage MR jobs using Maven

When your project grows larger in size, it becomes considerably complicated to manage. We recommend that you use Maven or similar software management tools to manage projects. The procedure is as follows:

1. Install Maven

   First, make sure that you have installed *Maven*.

2. Generate the project framework.

   At the root directory of your project (suppose the root directory of your project is D:/workspace), execute the following commands:

```
mvn archetype:generate -DgroupId=com.aliyun.emr.hadoop.examples
 -DartifactId=wordcountv2 -DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

   Maven automatically generates an empty sample project at *D:/workspace/wordcountv2* (same with the artifactId you specified) containing a basic pom.xml file and an App class (the class package path is the same as the groupId you specified).

3. Add Hadoop dependencies

   Open the project with your favorite IDE and edit the pom.xml file. Add the following content to the dependencies:

```
<dependency>
         <groupId>org.apache.hadoop</groupId>
         <artifactId>hadoop-mapreduce-client-common</artifactId>
         <version>2.6.0</version>
     </dependency>
     <dependency>
         <groupId>org.apache.hadoop</groupId>
         <artifactId>hadoop-common</artifactId>
         <version>2.6.0</version>
     </dependency>
```

4. Write the program

   Add a new class named *WordCount2.java* at the same directory level as that of the App class under the *com.aliyun.emr.hadoop.examples* package. The content is as follows:

```
package com.aliyun.emr.hadoop.examples;
 import java.io.BufferedReader;
 import java.io.FileReader;
 import java.io.IOException;
 import java.net.URI;
 import java.util.ArrayList;
 import java.util.HashSet;
 import java.util.List;
 import java.util.Set;
 import java.util.StringTokenizer;
 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.IntWritable;
 import org.apache.hadoop.io.Text;
 import org.apache.hadoop.mapreduce.Job;
 import org.apache.hadoop.mapreduce.Mapper;
 import org.apache.hadoop.mapreduce.Reducer;
```

```
 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
 import org.apache.hadoop.mapreduce.Counter;
 import org.apache.hadoop.util.GenericOptionsParser;
 import org.apache.hadoop.util.StringUtils;
 public class WordCount2 {
     public static class TokenizerMapper
             extends Mapper<Object, Text, Text, IntWritable>{
         static enum CountersEnum { INPUT_WORDS }
         private final static IntWritable one = new IntWritable(1);
         private Text word = new Text();
         private boolean caseSensitive;
         private Set<String> patternsToSkip = new HashSet<String>();
         private Configuration conf;
         private BufferedReader fis;
         @Override
         public void setup(Context context) throws IOException,
                 InterruptedException {
             conf = context.getConfiguration();
             caseSensitive = conf.getBoolean("wordcount.case.
sensitive", true);
             if (conf.getBoolean("wordcount.skip.patterns", true)) {
                 URI[] patternsURIs = Job.getInstance(conf).
getCacheFiles();
                 for (URI patternsURI : patternsURIs) {
                     Path patternsPath = new Path(patternsURI.
getPath());
                     String patternsFileName = patternsPath.getName
().toString();
                     parseSkipFile(patternsFileName);
                 }
             }
         }
         private void parseSkipFile(String fileName) {
             try {
                 fis = new BufferedReader(new FileReader(fileName));
                 String pattern = null;
                 while ((pattern = fis.readLine()) ! = null) {
                     patternsToSkip.add(pattern);
                 }
             } catch (IOException ioe) {
                 System.err.println("Caught exception while parsing
the cached file '"
                         + StringUtils.stringifyException(ioe));
             }
         }
         @Override
         public void map(Object key, Text value, Context context
         ) throws IOException, InterruptedException {
             String line = (caseSensitive) ?
                     value.toString() : value.toString().toLowerCase
();
             for (String pattern : patternsToSkip) {
                 line = line.replaceAll(pattern, "");
             }
             StringTokenizer itr = new StringTokenizer(line);
             while (itr.hasMoreTokens()) {
                 word.set(itr.nextToken());
                 context.write(word, one);
                 Counter counter = context.getCounter(CountersEnum.
class.getName(),
                         CountersEnum.INPUT_WORDS.toString());
```

```
                counter.increment(1);
            }
        }
    }
    public static class IntSumReducer
            extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        conf.set("fs.oss.accessKeyId", "${accessKeyId}");
       conf.set("fs.oss.accessKeySecret", "${accessKeySecret}");
        conf.set("fs.oss.endpoint","${endpoint}");
        GenericOptionsParser optionParser = new GenericOpt
ionsParser(conf, args);
        String[] remainingArgs = optionParser.getRemainingArgs();
        if (!( remainingArgs.length ! = 2 || remainingArgs.length !
 = 4)) {
            System.err.println("Usage: wordcount <in> <out> [-skip
skipPatternFile]");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount2.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        List<String> otherArgs = new ArrayList<String>();
        for (int i=0; i < remainingArgs.length; ++i) {
            if ("-skip".equals(remainingArgs[i])) {
                job.addCacheFile(new Path(EMapReduceOSSUtil.
buildOSSCompleteUri(remainingArgs[++i], conf)).toUri());
                job.getConfiguration().setBoolean("wordcount.skip.
patterns", true);
            } else {
                otherArgs.add(remainingArgs[i]);
            }
        }
        FileInputFormat.addInputPath(job, new Path(EMapReduce
OSSUtil.buildOSSCompleteUri(otherArgs.get(0), conf)));
        FileOutputFormat.setOutputPath(job, new Path(EMapReduce
OSSUtil.buildOSSCompleteUri(otherArgs.get(1), conf)));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

```
 }
```

See the following sample code for the EMapReduceOSSUtil class, which is located in the same directory of WordCount2:

```
package com.aliyun.emr.hadoop.examples;
 import org.apache.hadoop.conf.Configuration;
 public class EMapReduceOSSUtil {
     private static String SCHEMA = "oss://";
     private static String AKSEP = ":";
     private static String BKTSEP = "@";
     private static String EPSEP = ".";
     private static String HTTP_HEADER = "http://";
     /**
      * complete OSS uri
      * convert uri like: oss://bucket/path  to  oss://accessKeyId:
accessKeySecret@bucket.endpoint/path
      * ossref do not need this
      *
      * @param oriUri original OSS uri
      */
     public static String buildOSSCompleteUri(String oriUri, String
akId, String akSecret, String endpoint) {
         if (akId == null) {
             System.err.println("miss accessKeyId");
             return oriUri;
         }
         if (akSecret == null) {
             System.err.println("miss accessKeySecret");
             return oriUri;
         }
         if (endpoint == null) {
             System.err.println("miss endpoint");
             return oriUri;
         }
         int index = oriUri.indexOf(SCHEMA);
         if (index == -1 || index ! = 0) {
             return oriUri;
         }
         int bucketIndex = index + SCHEMA.length();
         int pathIndex = oriUri.indexOf("/", bucketIndex);
         String bucket = null;
         if (pathIndex == -1) {
             bucket = oriUri.substring(bucketIndex);
         } else {
             bucket = oriUri.substring(bucketIndex, pathIndex);
         }
         StringBuilder retUri = new StringBuilder();
         retUri.append(SCHEMA)
                 .append(akId)
                 .append(AKSEP)
                 .append(akSecret)
                 .append(BKTSEP)
                 .append(bucket)
                 .append(EPSEP)
                 .append(stripHttp(endpoint));
         if (pathIndex > 0) {
             retUri.append(oriUri.substring(pathIndex));
         }
         return retUri.toString();
     }
```

```
    public static String buildOSSCompleteUri(String oriUri,
Configuration conf) {
        return buildOSSCompleteUri(oriUri, conf.get("fs.oss.
accessKeyId"), conf.get("fs.oss.accessKeySecret"), conf.get("fs.oss.
endpoint"));
    }
    private static String stripHttp(String endpoint) {
        if (endpoint.startsWith(HTTP_HEADER)) {
            return endpoint.substring(HTTP_HEADER.length());
        }
        return endpoint;
    }
 }
```

5. Compile, package, and upload the code

   In the project directory, run the following commands:

   ```
   mvn clean package -DskipTests
   ```

   You can see the *wordcountv2-1.0-SNAPSHOT.jar* file in the target directory of your project directory, which is the JAR package of the job. Upload the JAR package to OSS.

6. Create a job

   Create a new job in E-MapReduce with the following parameters:

   ```
   jar ossref://yourBucket/yourPath/wordcountv2-1.0-SNAPSHOT.jar com
   .aliyun.emr.hadoop.examples.WordCount2 -Dwordcount.case.sensitive
   =true oss://yourBucket/yourPath/The_Sorrows_of_Young_Werther.txt
   oss://yourBucket/yourPath/output -skip oss://yourBucket/yourPath/
   patterns.txt
   ```

   Here, yourBucket represents your OSS bucket, and yourPath represents a path in the bucket. Configure the settings as needed. You need to download the files for processing related resources, which are *oss://yourBucket/yourPath/The_Sorrows_of_Young_Werther.txt* and *oss://yourBucket/yourPath/patterns.txt*, and then store them in OSS. You can download the resources needed for the jobs and store these resources in the corresponding directories in OSS.

   Download: *The_Sorrows_of_Young_Werther.txtpatterns.txt*

7. Create and run the execution plan

   Create the execution plan in E-MapReduce, associate it with the job, and then run the execution plan.

3.3 Create and run a Hive job.

This topic describes how to create and run a Hive job.

Use OSS in Hive

To read and write OSS in Hive, use the following statement to create an external table:

```
CREATE EXTERNAL TABLE eusers (
  userid INT)
 LOCATION 'oss://emr/users';
```

If your cluster version is obsolete and the preceding method is not supported, or if you want to access an OSS instance that is located in another region with the AccessKey of another account, follow these steps:

```
CREATE EXTERNAL TABLE eusers (
  userid INT)
 LOCATION 'oss://${AccessKeyId}:${AccessKeySecret}@${bucket}.${
endpoint}/users';
```

Parameter descriptions:

- ${accessKeyId}: The AccessKeyId of your account.
- ${accessKeySecret}: The secret key corresponding to the AccessKeyId.
- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists. The corresponding OSS instance should be in the region in which your cluster exists.

  For more information about specific values, see *OSS endpoints*.

Use Tez as the computing engine

Tez is implemented after E-MapReduce version 2.1.0+. Tez is a computing framework for optimizing complex DAG scheduling jobs. It dramatically improves the speed of Hive jobs in many scenarios.

You can optimize your jobs by setting Tez as the execution engine. Follow these steps:

```
set hive.execution.engine=tez
```

- Example 1

  Follow these steps:

  1. 1. Write the following script, save it as hiveSample1.sql, and then upload it to OSS.

     ```
     USE DEFAULT;
     ```

```
 set hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputF
ormat;
 set hive.stats.autogather=false;
 DROP TABLE emrusers;
 CREATE EXTERNAL TABLE emrusers (
   userid INT,
   movieid INT,
   rating INT,
   unixtime STRING )
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t'
  STORED AS TEXTFILE
  LOCATION 'oss://${bucket}/yourpath';
 SELECT COUNT(*) FROM emrusers;
 SELECT * from emrusers limit 100;
 SELECT movieid,count(userid) as usercount from emrusers group by
movieid order by usercount desc limit 50;
```

2. Data resources for testing.

   You can download required resources for Hive jobs from the following link and store the resources to the corresponding directory in your OSS instance.

   Download resources: *Public testing data*.

3. Create a job

   Create a new job in E-MapReduce using the following configuration:

   ```
   -f ossref://${bucket}/yourpath/hiveSample1.sql
   ```

   In this example, ${bucket} represents your OSS buckets, and yourPath represents a path in the bucket. You need to replace yourPath with where the Hive script is saved.

4. Create and run an execution plan.

   Create an execution plan. You can associate it with an existing cluster, or create a cluster as needed and associate the plan with the cluster. Save the job as **Run Manually**. Return to the previous page and click **Run Now** to run the job.

- Example 2

  Take scan in *HiBench* as an example.

  Follow these steps:

  1. Write the following script:

     ```
     USE DEFAULT;
      set hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputF
     ormat;
      set mapreduce.job.maps=12;
      set mapreduce.job.reduces=6;
      set hive.stats.autogather=false;
      DROP TABLE uservisits;
     ```

```
 CREATE EXTERNAL TABLE uservisits (sourceIP STRING,destURL STRING
,visitDate STRING,adRevenue DOUBLE,userAgent STRING,countryCode
STRING,languageCode STRING,searchWord STRING,duration INT ) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS SEQUENCEFILE
LOCATION 'oss://${bucket}/sample-data/hive/Scan/Input/uservisits';
```

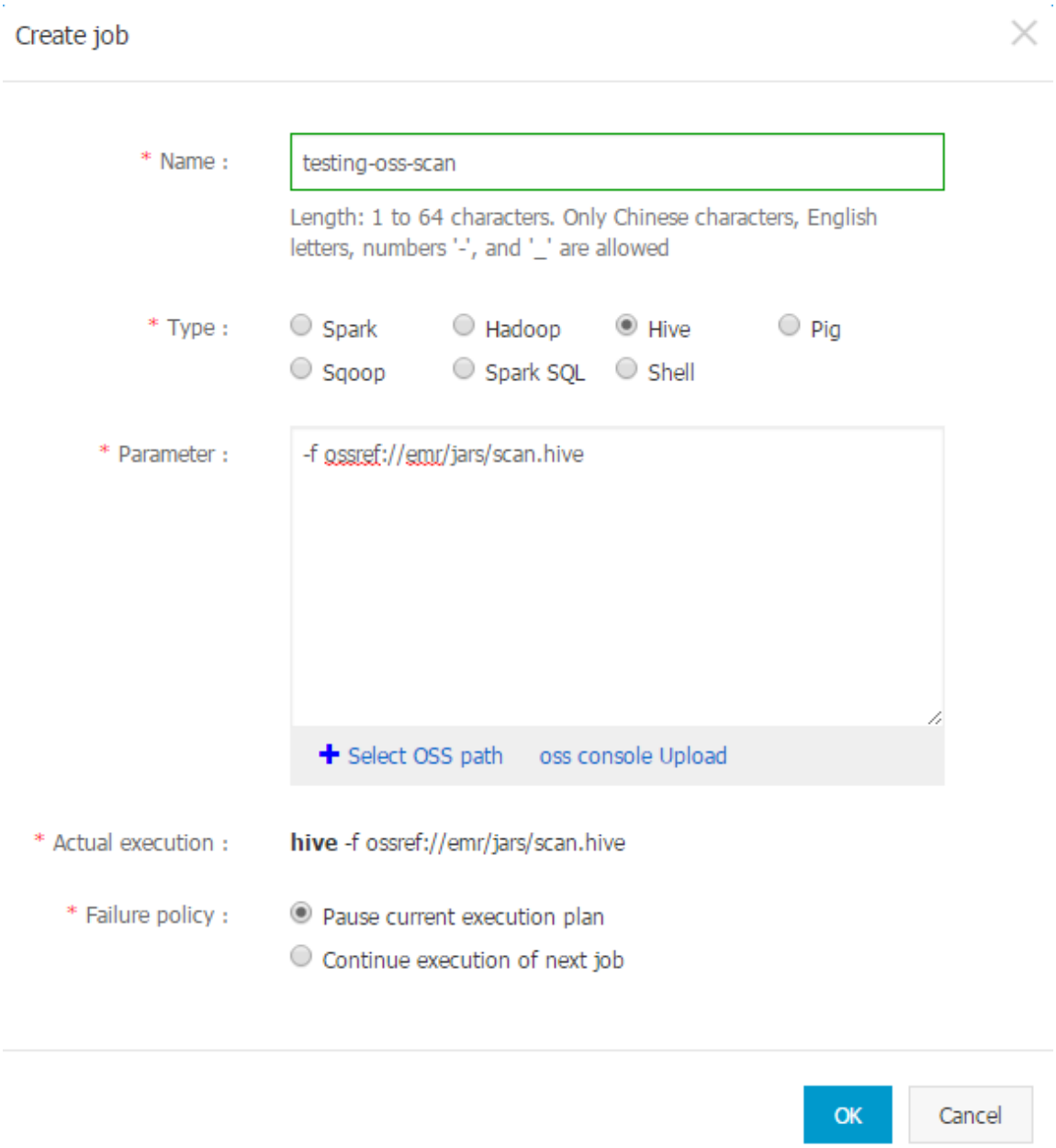
2. Prepare test data

   You can download required resources for the job from the following link and store the resources to the corresponding directory in your OSS instance.

   Download resources: *uservisits*

3. Create a job

   Store the script created in step 1 to OSS. For example, if the storage path is *oss://emr/jars/scan.hive*, follow these steps to create a job in E-MapReduce:



4. Create and run an execution plan.

   Create an execution plan. You can associate it with an existing cluster, or create a cluster as needed and associate the plan with the cluster. Save the job as **Run Manually**. Return to the previous page and click **Run Now** to run the job.

3.4 Create and run a Pig job

This topic describes how to create and run a Pig job.

Use OSS in Pig

Use the following format for OSS paths:

```
oss://${AccessKeyId}:${AccessKeySecret}@${bucket}.${endpoint}/${path}
```

Parameter descriptions:

- ${accessKeyId}: The AccessKey ID of your account.
- ${accessKeySecret}: The AccessKey Secret corresponding to the AccessKey ID.
- ${bucket}: The bucket corresponding to the AccessKey ID.
- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists, and the corresponding OSS should also be in the region in which your cluster exists.

  For more information about specific values, see *OSS endpoint*.
- ${path}: Path in the bucket.

Take script1-hadoop.pig in Pig as an example. Upload *tutorial.jar* and *excite.log.bz2* in Pig to OSS. Suppose that the URLs for uploading files are *oss://emr/jars/tutorial.jar* and *oss://emr/data/excite.log.bz2* respectively.

Follow these steps:

1. Create scripts

   Edit the path of the JAR file and the input and output paths in the script, as shown below: Note that the format of the OSS path is *oss://${accesskeyId}:${accessKeySecret}@${bucket}.${endpoint}/object/path*.

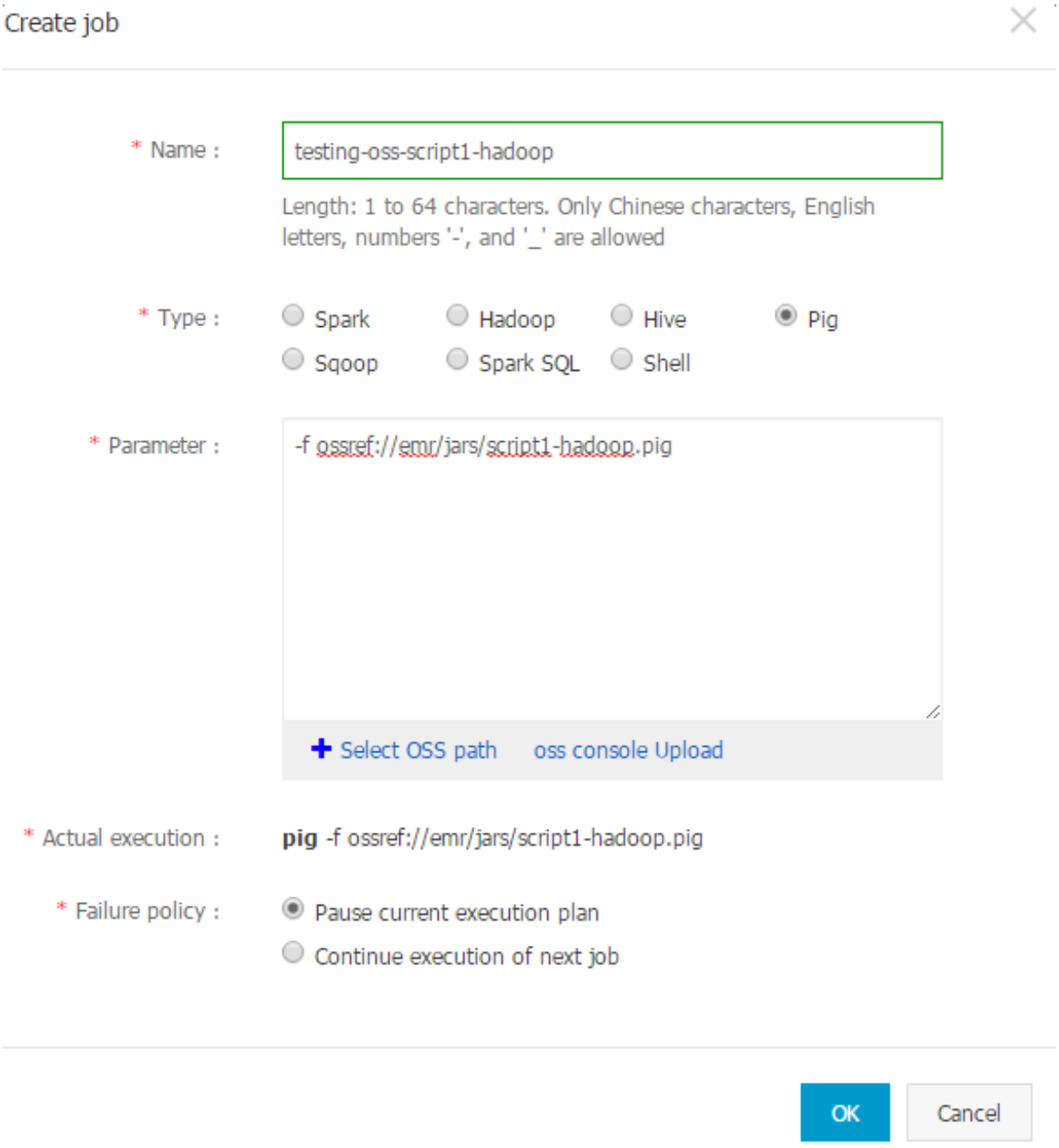
```
/*
 * Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements.  See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership.  The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing,
software
* distributed under the License is distributed on an "AS IS" BASIS,
```

```
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
* See the License for the specific language governing permissions
and
* limitations under the License.
*/
-- Query Phrase Popularity (Hadoop cluster)
-- This script processes a search query log file from the Excite
search engine and finds search phrases that occur with particular
high frequency during certain times of the day.
-- Register the tutorial JAR file so that the included UDFs can be
called in the script.
REGISTER oss://${AccessKeyId}:${AccessKeySecret}@${bucket}.${
endpoint}/data/tutorial.jar;
-- Use the  PigStorage function to load the excite log file into the
 ▒raw▒ bag as an array of records.
-- Input: (user,time,query)
raw = LOAD 'oss://${AccessKeyId}:${AccessKeySecret}@${bucket}.${
endpoint}/data/excite.log.bz2' USING PigStorage('\t') AS (user, time
, query);
-- Call the NonURLDetector UDF to remove records if the query field
is empty or a URL.
clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query
);
-- Call the ToLower UDF to change the query field to lowercase.
clean2 = FOREACH clean1 GENERATE user, time,     org.apache.pig.
tutorial.ToLower(query) as query;
-- Because the log file only contains queries for a single day, we
are only interested in the hour.
-- The excite query log timestamp format is YYMMDDHHMMSS.
-- Call the ExtractHour UDF to extract the hour (HH) from the time
field.
houred = FOREACH clean2 GENERATE user, org.apache.pig.tutorial.
ExtractHour(time) as hour, query;
-- Call the NGramGenerator UDF to compose the n-grams of the query.
ngramed1 = FOREACH houred GENERATE user, hour, flatten(org.apache.
pig.tutorial.NGramGenerator(query)) as ngram;
-- Use the  DISTINCT command to get the unique n-grams for all
records.
ngramed2 = DISTINCT ngramed1;
-- Use the  GROUP command to group records by n-gram and hour.
hour_frequency1 = GROUP ngramed2 BY (ngram, hour);
-- Use the  COUNT function to get the count (occurrences) of each n-
gram.
hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0),
COUNT($1) as count;
-- Use the  GROUP command to group records by n-gram only.
-- Each group now corresponds to a distinct n-gram and has the count
 for each hour.
uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;
-- For each group, identify the hour in which this n-gram is used
with a particularly high frequency.
-- Call the ScoreGenerator UDF to calculate a "popularity" score for
 the n-gram.
uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0),
flatten(org.apache.pig.tutorial.ScoreGenerator($1));
-- Use the  FOREACH-GENERATE command to assign names to the fields.
uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0 as
 ngram, $2 as score, $3 as count, $4 as mean;
-- Use the  FILTER command to move all records with a score less
than or equal to 2.0.
filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;
```

```
-- Use the  ORDER command to sort the remaining records by hour and
score.
ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour,
score;
-- Use the  PigStorage function to store the results.
-- Output: (hour, n-gram, score, count, average_counts_among
_all_hours)
STORE ordered_uniq_frequency INTO 'oss://${AccessKeyId}:${AccessKeyS
ecret}@${bucket}.${endpoint}/data/script1-hadoop-results' USING
PigStorage();
```

2. Create a job

   Store the script created in step 1 to OSS. For example, if the storage path is *oss://emr/jars/script1-hadoop.pig*, follow these steps to create a job in E-MapReduce:



3. Create and run an execution plan.

Create the execution plan in E-MapReduce and add the created Pig job to the execution plan.

Select **Run Now** as the policy so that the script1-hadoop job can run in the selected cluster.

3.5 Create a Hadoop streaming job

This topic describes how to use PythonH to create a Hadoop streaming job.

Use Python to create a Hadoop streaming job

The mapper code is as follows:

```
#! /usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

The reducer code is as follows:

```
#! /usr/bin/env python
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Suppose that the mapper code is saved as */home/hadoop/mapper.py* and the reducer code is saved as */home/hadoop/reducer.py*, and that the paths for input and output are */tmp/input* and */tmp/output* in the HDFS file system respectively. Submit the following Hadoop command in the E-MapReduce cluster.

```
hadoop jar /usr/lib/hadoop-current/share/hadoop/tools/lib/hadoop-
streaming-*.jar -file /home/hadoop/mapper.py -mapper mapper.py -file /
```

```
home/hadoop/reducer.py -reducer reducer.py -input /tmp/hosts -output /
tmp/output
```

3.6 Process Table Store data in Hive

This topic describes how to process Table Store data in Hive.

Connect Hive to Table Store

- Prepare a data table

  Create a table named pet and set the name field as the primary key.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

- The following example describes how to process Table Store data in Hive.

  1. Command line

     ```
     $ HADOOP_HOME=YourHadoopDir HADOOP_CLASSPATH=emr-tablestore-<
     version>.jar:tablestore-4.1.0-jar-with-dependencies.jar:joda-time-
     2.9.4.jar bin/hive
     ```

  2. Create an external table

     ```
     CREATE EXTERNAL TABLE pet
       (name STRING, owner STRING, species STRING, sex STRING, birth
     STRING, death STRING)
       STORED BY 'com.aliyun.openservices.tablestore.hive.TableStore
     StorageHandler'
       WITH SERDEPROPERTIES(
         "tablestore.columns.mapping"="name,owner,species,sex,birth,
     death")
       TBLPROPERTIES (
         "tablestore.endpoint"="YourEndpoint",
         "tablestore.access_key_id"="YourAccessKeyId",
         "tablestore.access_key_secret"="YourAccessKeySecret",
     ```

```
    "tablestore.table.name"="pet");
```

3. Insert data to the external table

```
INSERT INTO pet VALUES("Fluffy", "Harold", "cat", "f", "1993-02-04
", null);
INSERT INTO pet VALUES("Claws", "Gwen", "cat", "m", "1994-03-17",
null);
INSERT INTO pet VALUES("Buffy", "Harold", "dog", "f", "1989-05-13
", null);
INSERT INTO pet VALUES("Fang", "Benny", "dog", "m", "1990-08-27",
null);
INSERT INTO pet VALUES("Bowser", "Diane", "dog", "m", "1979-08-31
", "1995-07-29");
INSERT INTO pet VALUES("Chirpy", "Gwen", "bird", "f", "1998-09-11
", null);
INSERT INTO pet VALUES("Whistler", "Gwen", "bird", null, "1997-12-
09", null);
INSERT INTO pet VALUES("Slim", "Benny", "snake", "m", "1996-04-29
", null);
INSERT INTO pet VALUES("Puffball", "Diane", "hamster", "f", "1999-
03-30", null);
```

4. Query data

```
> SELECT * FROM pet;
Bowser  Diane   dog     m       1979-08-31      1995-07-29
Buffy   Harold  dog     f       1989-05-13      NULL
Chirpy  Gwen    bird    f       1998-09-11      NULL
Claws   Gwen    cat     m       1994-03-17      NULL
Fang    Benny   dog     m       1990-08-27      NULL
Fluffy  Harold  cat     f       1993-02-04      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
> SELECT * FROM pet WHERE birth > "1995-01-01";
Chirpy  Gwen    bird    f       1998-09-11      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
```

Data type conversion

	TINYINT	SMALLIN	INT	BIGINT	FLOAT	DOUBLE	BOOLEA	STRING	BINARY
INTEGER	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported	Supported (with loss of precision)	Supported (with loss of precision)			
DOUBLE	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported			

	TINYINT	SMALLIN	INT	BIGINT	FLOAT	DOUBLE	BOOLEA	STRING	BINARY
BOOLEAN							Supported		
STRING								Supported	
BINARY									Supported

Appendix

For the complete sample code, see

- *Process Table Store data in Hive*

3.7 Process Table Store data in MR

This topic describes how to process Table Store data in MR.

Connect MR to Table Store

- Prepare a data table

  Create a table named pet and set the name field as the primary key.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

- The following example describes how to process Table Store data in MR.

```
public class RowCounter {
    public static class RowCounterMapper
      extends Mapper<PrimaryKeyWritable, RowWritable, Text,
LongWritable> {
        private final static Text agg = new Text("TOTAL");
        private final static LongWritable one = new LongWritable(1);
        @Override public void map(PrimaryKeyWritable key, RowWritabl
e value,
            Context context) throws IOException, Interrupte
dException {
```

```
            context.write(agg, one);
        }
    }
    public static class IntSumReducer
      extends Reducer<Text,LongWritable,Text,LongWritable> {
        @Override public void reduce(Text key, Iterable<LongWritable
> values,
            Context context) throws IOException, Interrupte
dException {
            long sum = 0;
            for (LongWritable val : values) {
                sum += val.get();
            }
            context.write(key, new LongWritable(sum));
        }
    }
    private static RangeRowQueryCriteria fetchCriteria() {
        RangeRowQueryCriteria res = new RangeRowQueryCriteria("pet
");
        res.setMaxVersions(1);
        List<PrimaryKeyColumn> lower = new ArrayList<PrimaryKey
Column>();
        List<PrimaryKeyColumn> upper = new ArrayList<PrimaryKey
Column>();
        lower.add(new PrimaryKeyColumn("name", PrimaryKeyValue.
INF_MIN));
        upper.add(new PrimaryKeyColumn("name", PrimaryKeyValue.
INF_MAX));
        res.setInclusiveStartPrimaryKey(new PrimaryKey(lower));
        res.setExclusiveEndPrimaryKey(new PrimaryKey(upper));
        return res;
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        job.setJarByClass(RowCounter.class);
        job.setMapperClass(RowCounterMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);
        job.setInputFormatClass(TableStoreInputFormat.class);
        TableStore.setCredential(job, accessKeyId, accessKeySecret,
securityToken);
        TableStore.setEndpoint(job, endpoint, instance);
        TableStoreInputFormat.addCriteria(job, fetchCriteria());
        FileOutputFormat.setOutputPath(job, new Path(outputPath));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

- The following example describes how to use MR to write data to Table Store.

```
public static class OwnerMapper
  extends Mapper<PrimaryKeyWritable, RowWritable, Text, MapWritable>
 {
    @Override public void map(PrimaryKeyWritable key, RowWritable
row,
        Context context) throws IOException, InterruptedException {
        PrimaryKeyColumn pet = key.getPrimaryKey().getPrimary
KeyColumn("name");
        Column owner = row.getRow().getLatestColumn("owner");
        Column species = row.getRow().getLatestColumn("species");
```

```
        MapWritable m = new MapWritable();
        m.put(new Text(pet.getValue().asString()),
            new Text(species.getValue().asString()));
        context.write(new Text(owner.getValue().asString()), m);
    }
}
public static class IntoTableReducer
  extends Reducer<Text,MapWritable,Text,BatchWriteWritable> {
    @Override public void reduce(Text owner, Iterable<MapWritable>
pets,
        Context context) throws IOException, InterruptedException {
        List<PrimaryKeyColumn> pkeyCols = new ArrayList<PrimaryKey
Column>();
        pkeyCols.add(new PrimaryKeyColumn("owner",
                PrimaryKeyValue.fromString(owner.toString())));
        PrimaryKey pkey = new PrimaryKey(pkeyCols);
        List<Column> attrs = new ArrayList<Column>();
        for(MapWritable petMap: pets) {
            for(Map.Entry<Writable, Writable> pet: petMap.entrySet
()) {
                Text name = (Text) pet.getKey();
                Text species = (Text) pet.getValue();
                attrs.add(new Column(name.toString(),
                        ColumnValue.fromString(species.toString
())));
            }
        }
        RowPutChange putRow = new RowPutChange(outputTable, pkey)
            .addColumns(attrs);
        BatchWriteWritable batch = new BatchWriteWritable();
        batch.addRowChange(putRow);
        context.write(owner, batch);
    }
}
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, TableStoreOutputFormatExample.
class.getName());
    job.setMapperClass(OwnerMapper.class);
    job.setReducerClass(IntoTableReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(MapWritable.class);
    job.setInputFormatClass(TableStoreInputFormat.class);
    job.setOutputFormatClass(TableStoreOutputFormat.class);
    TableStore.setCredential(job, accessKeyId, accessKeySecret,
securityToken);
    TableStore.setEndpoint(job, endpoint, instance);
    TableStoreInputFormat.addCriteria(job, ...) ;
    TableStoreOutputFormat.setOutputTable(job, outputTable);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Appendix

For the complete sample code, see:

- *MR reads data from Table Store*
- *MR writes data to Table Store*

4 Create an HBase cluster and use the HBase storage service

This topic describes how to create and configure an HBase cluster and use the HBase storage service.

To make better use of HBase, we recommend that you use the following configurations when creating a cluster.

- Enable access from the public network.
- Select the zone where the server of the application that accesses HBase is located. Do not randomly assign a zone.
- Select four or more hardware nodes, including the master and slave nodes. E-MapReduce creates NameNode, DataNode, JournalNode, HMaster, RegionServer, and ZooKeeper roles on these nodes.
- Select 4-core 16 GB and 8-core 32 GB for servers. Lower configurations may impair the stability of HBase clusters.
- Select the SSD disk type for the data disk for cost-effectiveness. You can select Basic Disk as the disk type for services that are not frequently used but require a large amount of storage space.
- Configure the data capacity as needed.
- HBase clusters support memory expansion.

Configure HBase

You can use the *Software Configuration* feature on the Create Cluster page when creating an HBase cluster. You can optimize and modify the default parameters of HBase based in specific scenarios. See the following example:

```
{
  "configurations": [
    {
      "classification": "hbase-site",
      "properties": {
        "hbase.hregion.memstore.flush.size": "268435456",
        "hbase.regionserver.global.memstore.size": "0.5",
        "hbase.regionserver.global.memstore.lowerLimit": "0.6"
      }
    }
  ]
```

```
}
```

Some default configurations for HBase clusters are as follows:

key	value
zookeeper.session.timeout	180000
hbase.regionserver.global.memstore.size	0.35
hbase.regionserver.global.memstore.lowerLimit	0.3
hbase.hregion.memstore.flush.size	128MB

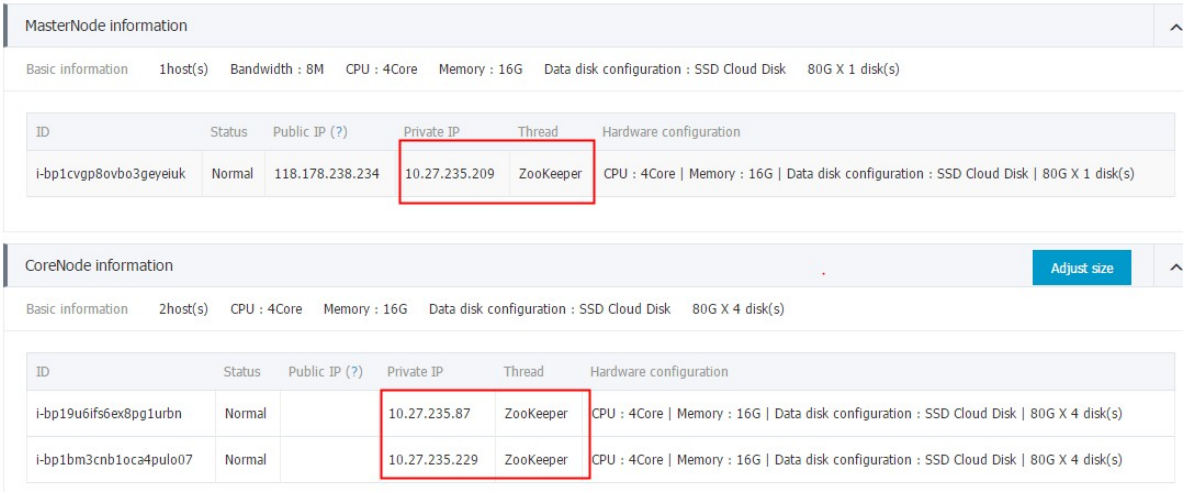
Access HBase

Note:

- Considering network performance, we recommend that you initiate access requests to HBase clusters created by E-MapReduce from ECS instances in the same zone.
- The ECS instance accessing HBase clusters must be in the same security group as the HBase clusters. Otherwise, the access fails. Therefore, if you create a Hadoop, Spark, or Hive cluster in E-MapReduce, and the cluster you create needs to access HBase, then make sure that you select the same security group as the HBase cluster.

After you have created an HBase cluster in the E-MapReduce console, you can use the HBase storage service. The procedure is as follows:

1. Get the master IP address and the address of the ZooKeeper cluster. On the Cluster Overview page in the E-MapReduce console, you can view the IP addresses of the master nodes and ZooKeeper access addresses (intranet IP addresses), as shown in the following figure.



For more information about master nodes with public IP addresses enabled, see *How to log on to a master node* and browse the HMaster WEB UI (*localhost:16010*).

2. You can connect to the master node of a cluster over SSH using HBase Shell. You can use SSH to access the master node of a cluster. Switch to an HDFS account and use HBase Shell to access the clusters. (For more information about HBase Shell, see the *Apache HBase website*.)

```
[root@emr-header-1 ~]# su hdfs
[hadoop@emr-header-1 root]$ hbase shell
HBase Shell; enter 'help<RETURN>' for a list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.1, r374488, Fri Aug 21 09:18:22 CST 2015
hbase(main):001:0>
```

3. Access the cluster using HBase Shell from another ECS node (within the same security group). Download the HBase-1.x resource from the Apache HBase website (*download link*). Extract the resources, modify *conf/hbase-site.xml*, and then add the ZooKeeper address of the cluster as follows:

```
<configuration>
     <property>
         <name>hbase.zookeeper.quorum</name>
         <value>$ZK_IP1,$ZK_IP2,$ZK_IP3</value>
     </property>
</configuration>
```

Then you can access the cluster using the **bin/hbase shell** command.

If the ECS instances are created using E-MapReduce, you only need to modify the *hbase-site.xml* file without downloading the HBase-1.x resource.

The configuration files of versions later than 3.2.0 are located in */etc/ecm/hbase-conf/hbase-site.xml*.

The configuration files of versions earlier than 3.2.0 are located in */etc/emr/hbase-conf/hbase-site.xml*.

4. To access the HBase cluster using APIs, add the following Maven dependency:

```
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-client</artifactId>
<version>1.1.1</version>
```

Configure the correct ZooKeeper address to connect to the cluster.

```
Configuration config = HBaseConfiguration.create();
config.set(HConstants.ZOOKEEPER_QUORUM,"$ZK_IP1,$ZK_IP2,$ZK_IP3");
Connection connection = ConnectionFactory.createConnection(config);
```

```
try {
     Table table = connection.getTable(TableName.valueOf("myLittleHB
aseTable"));
     try {
         //Do table operation
     }finally {
         if (table ! = null) table.close();
     }
} finally {
     connection.close();
}
```

For more information about the Apache HBase development, see the *Apache HBase* website.

Examples

- Prerequisites

  The ECS instance accessing the Hbase cluster must be in the same security group as the HBase cluster.

- Allow Spark to access HBase

  See *spark-hbase-connector*.

- Allow Hadoop to access HBase

  See *HBase MapReduce Examples*.

- Allow Hive to access HBase

  Only Hive running in clusters with a version of E-MapReduce 1.2.0 or later can access the HBase cluster. Follow these steps:

  1. Log on to the Hive cluster, and modify hosts by adding the following line:

     ```
     $zk_ip emr-cluster //$zk_ip is the IP address of the ZooKeeper
     node in the HBase cluster.
     ```

  2. For more information about how to operate Hive, see *Hive HBase Integration*.

5 Back up HBase

The HBase cluster created in E-MapReduce allows you to use the snapshot feature integrated into HBase to back up HBase tables, and export the backup to *OSS*.

Example:

1. Create an HBase cluster

   For more information, see *Create clusters*.

2. Create a table

```
>create 'test','cf'
```

3. Add data

```
> put 'test','a','cf:c1',1
> put 'test','a','cf:c2',2
> put 'test','b','cf:c1',3
> put 'test','b','cf:c2',4
> put 'test','c','cf:c1',5
> put 'test','c','cf:c2',6
```

4. Create a snapshot

```
hbase snapshot create -n test_snapshot -t test
```

   List snapshots

```
>list_snapshots
SNAPSHOT                TABLE + CREATION TIME
 test_snapshot          test (Sun Sep 04 20:31:00 +0800 2016)
1 row(s) in 0.2080 seconds
```

5. Export the snapshot to OSS.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot
test_snapshot -copy-to oss://$accessKeyId:$accessKeySecret@$bucket.
oss-cn-hangzhou-internal.aliyuncs.com/hbase/snapshot/test
```

   Note:

   Access OSS using *internal endpoints*.

6. Create another HBase cluster

7. Export snapshots from OSS

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot
test_snapshot -copy-from oss://$accessKeyId:$accessKeySecret@$bucket
```

```
.oss-cn-hangzhou-internal.aliyuncs.com/hbase/snapshot/test -copy-to
 /hbase/
```

8. Restore data from snapshots

```
>restore _snapshot 'test_snapshot'
```

```
>scan 'test'
ROW                     COLUMN+CELL
 a                      column=cf:c1, timestamp=1472992081375, value
=1
 a                      column=cf:c2, timestamp=1472992090434, value
=2
 b                      column=cf:c1, timestamp=1472992104339, value
=3
 b                      column=cf:c2, timestamp=1472992099611, value
=4
 c                      column=cf:c1, timestamp=1472992112657, value
=5
 c                      column=cf:c2, timestamp=1472992118964, value
=6
3 row(s) in 0.0540 seconds
```

9. Create new tables from snapshots

```
>clone_snapshot 'test_snapshot','test_2'
```

```
>scan 'test_2'
ROW                     COLUMN+CELL
 a                      column=cf:c1, timestamp=1472992081375, value
=1
 a                      column=cf:c2, timestamp=1472992090434, value
=2
 b                      column=cf:c1, timestamp=1472992104339, value
=3
 b                      column=cf:c2, timestamp=1472992099611, value
=4
 c                      column=cf:c1, timestamp=1472992112657, value
=5
 c                      column=cf:c2, timestamp=1472992118964, value
=6
3 row(s) in 0.0540 seconds
```

6 Guide for E-MapReduce cluster operations

This topic describes multiple operation methods for EMR clusters, which make it easier for you to independently perform operations on components.

Note:

For later cluster versions (3.2 and later), all operations on components can be performed through the Cluster Management page in the console. We recommend that you manage components using web pages.

Some general environment variables

Enter the env command to see environment variable configurations that are similar to those in the following example. The actual configurations can be different. This example is for reference only.

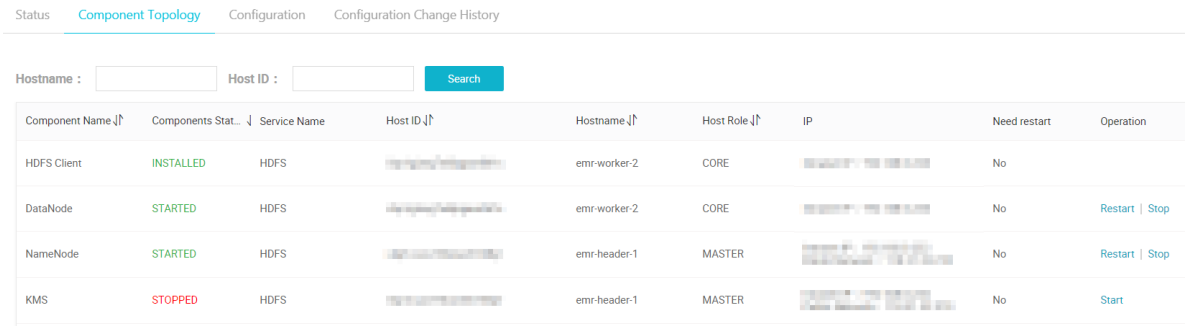
```
JAVA_HOME=/usr/lib/jvm/java
HADOOP_HOME=/usr/lib/hadoop-current
HADOOP_CLASSPATH=/usr/lib/hbase-current/lib/*:/usr/lib/tez-current/*:/
usr/lib/tez-current/lib/*:/etc/emr/tez-conf:/usr/lib/hbase-current/lib
/*:/usr/lib/tez-current/*:/usr/lib/tez-current/lib/*:/etc/emr/tez-conf
:/opt/apps/extra-jars/*:/opt/apps/extra-jars/*
HADOOP_CONF_DIR=/etc/emr/hadoop-conf
SPARK_HOME=/usr/lib/spark-current
SPARK_CONF_DIR=/etc/emr/spark-conf
HBASE_HOME=/usr/lib/hbase-current
HBASE_CONF_DIR=/etc/emr/hbase-conf
HIVE_HOME=/usr/lib/hive-current
HIVE_CONF_DIR=/etc/emr/hive-conf
PIG_HOME=/usr/lib/pig-current
PIG_CONF_DIR=/etc/emr/pig-conf
TEZ_HOME=/usr/lib/tez-current
TEZ_CONF_DIR=/etc/emr/tez-conf
ZEPPELIN_HOME=/usr/lib/zeppelin-current
ZEPPELIN_CONF_DIR=/etc/emr/zeppelin-conf
HUE_HOME=/usr/lib/hue-current
HUE_CONF_DIR=/etc/emr/hue-conf
PRESTO_HOME=/usr/lib/presto-current
PRESTO_CONF_DIR=/etc/emr/presot-conf
```

Start and stop a component using the Web UI

You can use the Web UI of the E-MapReduce service to start, stop, and restart the component that runs on the specified ECS instance. Operations on different components are similar, for example the HDFS service. Follow these steps to start, stop, and restart the DataNode component for the emr-worker-1 instance.

1. On the **Cluster list** page, click **Manage** in the **Actions** column for the cluster to perform operations on.

2. On the **Clusters and Services** page, click the **HDFS** link to go to the HDFS service page.
3. Click the **Component Topology** tab to see the list of components that run on all instances in the cluster.
4. Click **Start** in the **Actions** column for the DataNode component that runs on the *emr-worker-1* instance. Enter the commit record in the dialog box and click **OK**. Refresh the page after 10 seconds. The value in the **Components Status** column for DataNode switches from **STOPPED** to **STARTED**.



5. After the component is started, click **Restart** in the **Actions** column. Enter the commit record in the dialog box and click **OK**.

   Refresh the page after 40 seconds. The value in the Components Status column for DataNode switches from **STOPPED** to **STARTED**.
6. Click **Stop** in the **Actions** column. Enter the commit record in the dialog box and click **OK**.

   Refresh the page after 10 seconds. The value of the **Components Status** column for DataNode switches from **STARTED** to **STOPPED**.

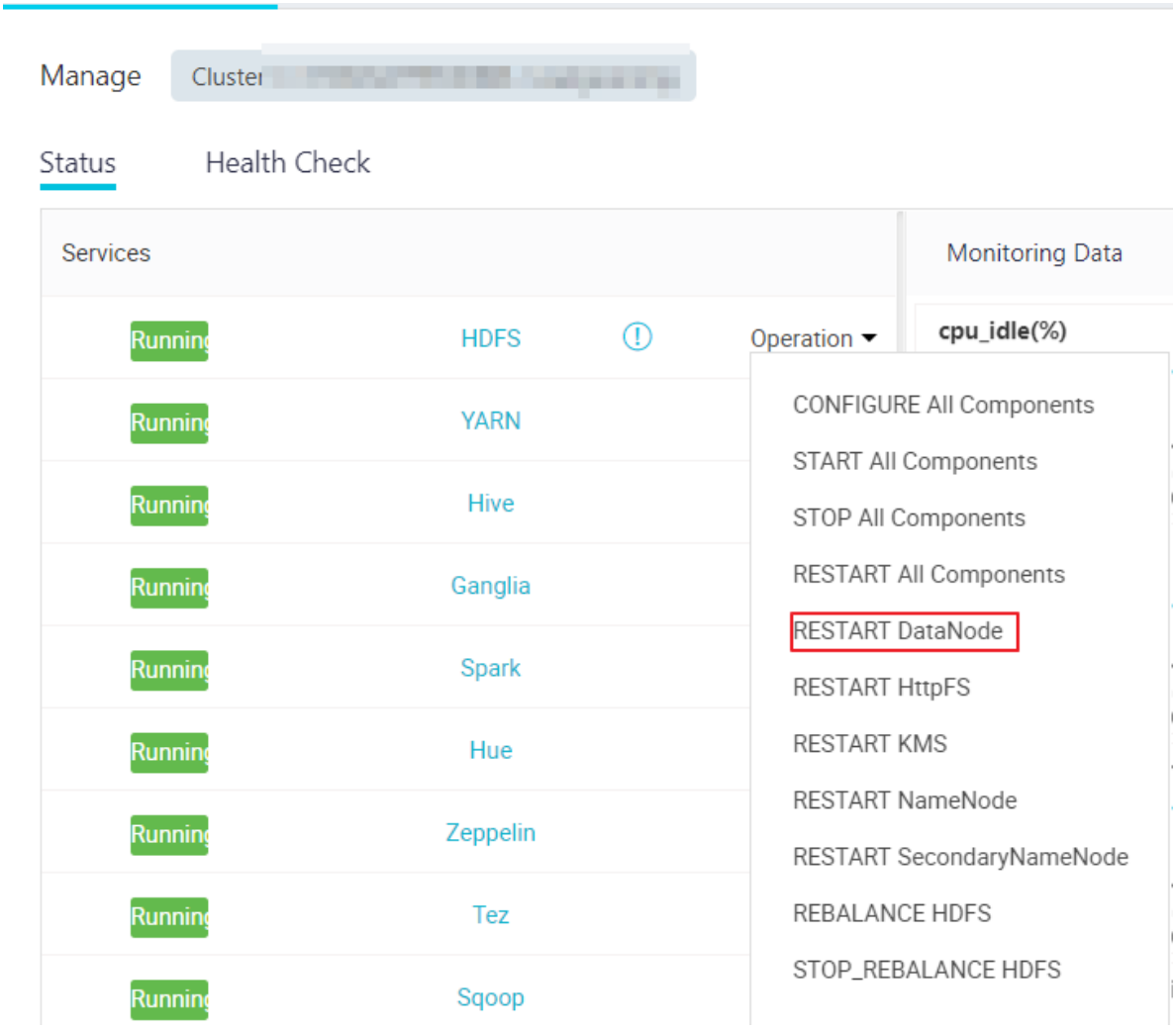
Perform bulk operations on components using the Web UI

You can perform bulk operations on components for multiple ECS instances instead of a single specified ECS instance. The HDFS service as an example. Follow these steps to restart the DataNode components for all instances.

1. On the **Cluster list** page, click **Manage** in the **Actions** column for the cluster to perform operations on.
2. On the **Clusters and Services** page, click **Actions** for **HDFS** in the Services list.
3. Select **RESTART DataNode** from the Actions drop-down list. Enter the commit record in the dialog box and click **OK**.

   Click **HDFS** and click the **Component Topology** tab. View the status of each process on the Component Topology tab page.

Note:
Console reports an error if you restart nodes manually after performing a rolling restart of the cluster.



Start and stop a component using CLI

- YARN

  Account: hadoop

  - ResourceManager (a master component)

    ```
    //Starts a component.
    /usr/lib/hadoop-current/sbin/yarn-daemon.sh start resourcemanager
    //Stops a component.
    /usr/lib/hadoop-current/sbin/yarn-daemon.sh stop resourcemanager
    ```

  - NodeManager (a core component)

    ```
    //Starts a component.
    ```

```
/usr/lib/hadoop-current/sbin/yarn-daemon.sh start nodemanager
//Stops a component.
/usr/lib/hadoop-current/sbin/yarn-daemon.sh stop nodemanager
```

- JobHistoryServer (a master component)

```
//Starts a component.
/usr/lib/hadoop-current/sbin/mr-jobhistory-daemon.sh start
historyserver
//Stops a component.
/usr/lib/hadoop-current/sbin/mr-jobhistory-daemon.sh stop
historyserver
```

- JobHistoryServer (a master component)

```
//Starts a component.
/usr/lib/hadoop-current/sbin/mr-jobhistory-daemon.sh start
historyserver
//Stops a component.
/usr/lib/hadoop-current/sbin/mr-jobhistory-daemon.sh stop
historyserver
```

- WebProxyServer (a master component)

```
//Starts a component.
/usr/lib/hadoop-current/sbin/yarn-daemon.sh start proxyserver
//Stops a component.
/usr/lib/hadoop-current/sbin/yarn-daemon.sh stop proxyserver
```

- HDFS

  Account: hdfs

  - NameNode (a master component)

```
//Starts a component.
/usr/lib/hadoop-current/sbin/hadoop-daemon.sh start namenode
//Stops a component.
/usr/lib/hadoop-current/sbin/hadoop-daemon.sh stop namenode
```

  - DataNode (a core component)

```
//Starts a component.
/usr/lib/hadoop-current/sbin/hadoop-daemon.sh start datanode
//Stops a component.
/usr/lib/hadoop-current/sbin/hadoop-daemon.sh stop datanode
```

- Hive

  Account: hadoop

  — MetaStore (a master component)

```
//Starts a component. You can set the HADOOP_HEAPSIZE environment
variable to a greater value for a larger maximum memory.
```

```
HADOOP_HEAPSIZE=512 /usr/lib/hive-current/bin/hive --service
metastore >/var/log/hive/metastore.log 2>&1 &
```

- HiveServer2 (a master component)

```
//Starts a component.
HADOOP_HEAPSIZE=512 /usr/lib/hive-current/bin/hive --service
hiveserver2 >/var/log/hive/hiveserver2.log 2>&1 &
```

- HBase

  Operational account: hdfs

  Note: The following example is based on the HBase service. An error occurs when you start a component without corresponding configurations.

  - HMaster (a master component)

```
//Starts a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh start master
//Restarts a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh restart master
//Stops a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh stop master
```

  - HRegionServer (a core component)

```
//Starts a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh start regionserver
//Restarts a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh restart regionserver
//Stops a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh stop regionserver
```

  - ThriftServer (a master component)

```
//Starts a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh start thrift -p 9099 >/
var/log/hive/thriftserver.log 2>&1 &
//Stops a component.
/usr/lib/hbase-current/bin/hbase-daemon.sh stop thrift
```

- Hue

  Account: hadoop

```
//Starts a component.
su -l root -c "${HUE_HOME}/build/env/bin/supervisor >/dev/null 2>&1
 &"
//Stops a component.
ps aux | grep hue     //Lists information about the currently
running Hue processes.
kill -9 huepid        //Kills the Hue process by its PID.
```

- Zeppelin

Account: hadoop

```
//Starts a component. You can set the HADOOP_HEAPSIZE environment
variable to a greater value for a larger maximum memory.
su -l root -c "ZEPPELIN_MEM=\"-Xmx512m -Xms512m\" ${ZEPPELIN_HOME}/
bin/zeppelin-daemon.sh start"
//Stops a component.
su -l root -c "${ZEPPELIN_HOME}/bin/zeppelin-daemon.sh stop"
```

- Presto

  Account: hadoop

  — PrestoServer (a master component)

  ```
  //Starts a component.
  /usr/lib/presto-current/bin/launcher --config=/usr/lib/presto-
  current/etc/coordinator-config.properties start
  //Stops a component.
  /usr/lib/presto-current/bin/launcher --config=/usr/lib/presto-
  current/etc/coordinator-config.properties stop
  ```

  — (a core component)

  ```
  //Starts a component.
  /usr/lib/presto-current/bin/launcher --config=/usr/lib/presto-
  current/etc/worker-config.properties start
  //Stops a component.
  /usr/lib/presto-current/bin/launcher --config=/usr/lib/presto-
  current/etc/worker-config.properties stop
  ```

Perform bulk operations on components using CLI

Write script commands to perform bulk operations for all worker (core) nodes. In an EMR cluster , all worker nodes that run in the hadoop account and the hdfs account are accessible from the master node using SSH.

For example, you can run the following commands to stop the NodeManager components for all worker nodes. Assume that the number of worker nodes is 10.

```
for i in `seq 1 10`;do ssh emr-worker-$i /usr/lib/hadoop-current/sbin/
yarn-daemon.sh stop nodemanager;done
```

7 E-MapReduce SDK release notes

E-MapReduce SDK release notes for each version.

Note:

- emr-core: Supports the interaction between data sources in OSS and Hadoop/Spark. It exists in the cluster by default. You do not need to package emr-code as part of a job.
- emr-tablestore: Supports the interaction between data sources in Table Store and Hadoop/Hive/Spark. Package emr-tablestore in the JAR file.
- emr-mns_2.10/emr-mns_2.11: Supports Spark to read data sources in MNS. Package emr-mns_2.10/emr-mns_2.11 in the JAR file.
- emr-ons_2.10/emr-ons_2.11: Supports Spark to read data sources in Message Queue (MQ). Package emr-ons_2.10/emr-ons_2.11 in the JAR file.
- emr-logservice_2.10/emr-logservice_2.11: Supports Spark to read data sources in Log Service. Package emr-logservice_2.10/emr-logservice_2.11 in the JAR file.
- emr-maxcompute_2.10/emr-maxcompute_2.11: Supports Spark to read data sources in MaxCompute. Package emr-maxcompute_2.10/emr-maxcompute_2.11 in the JAR file.

```
<! --Supports interaction with data sources in OSS-->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-core</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! --Supports interaction with data sources in Table Store
-->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-tablestore</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! --Supports interaction with data sources in MNS, MQ, Log
Service, and MaxCompute (in the Spark 1.x environment) -->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-mns_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-logservice_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-maxcompute_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
```

```
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-ons_2.10</artifactId>
        <version>1.4.1</version>
    </dependency>
    <! --Supports interaction with data sources in MNS, MQ, Log
Service, and MaxCompute (in the Spark 2.x environment)-->
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-mns_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-logservice_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-maxcompute_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-ons_2.11</artifactId>
        <version>1.4.1</version>
    </dependency>
```

- v1.4.1
  - MaxCompute: Fixes the problem of truncation for the DATETIME values.
  - MaxCompute: Fixes the thread-safety problem for the SimpleDateFormat class.
- v1.4.0
  - MaxCompute: Adds implementation based on the DataSource class. Only versions 2.x and above of Spark are supported.
  - Log Service: Adds implementation based on Direct API. Only versions 2.x and above of Spark are supported.
  - OTS: Optimizes read and write operations.
  - Fixes the bug that the access key of a user is replaced by the access key of a cluster app role when reading data sources in Log Service.
- v1.3.2
  - Fixes some bugs in Table Store.
- v1.3.1
  - Fixes the problem that NullPointerExceptions are thrown in some scenarios when Spark interacts with Log Service.
  - E-MapReduce SDK supports the Spark 2.x environments since this version.

- v1.3.0
  - Supports Hadoop MapReduce, Spark, SparkSQL, and Hive to read data in Table Store.
  - MNS and Log Service support the MetaService service provided by E-MapReduce. Based on the MetaService service, you can access data in MNS and Log Service without AccessKeys.
  - Upgrades some dependencies.
- v1.1.3.1

  SDK:
  - Fixes the problem of dependency conflict between MNS and Spark/Hadoop packages.
  - Fixes the problem that NullPointerExceptions are thrown in some scenarios when Spark Streaming interacts with MNS.
  - Fixes some bugs for the Python SDK.
  - Supports custom time and locations in the scenarios when Spark Streaming integrates with Loghub.
- Core
  - Fixes the problem that Hadoop does not support the Snappy native files. Currently, E-MapReduce supports processing the Snappy files that Log Service have archived to OSS.
  - Fixes the problem that Spark does not support the Snappy files.
  - Fixes the problem that OSS does not support the two algorithms of the OutputCommitter class in Apache Hadoop 2.7.2.
  - Optimizes the performance of Hadoop/Spark reading and writing data in OSS.
  - Fixes the problem that a Log4j exception is thrown when Spark prints a job.
- v1.1.2
  - Fixes the problem that the *`ConnectionClosedException`* is thrown when a job is reading data in OSS.
  - Fixes the problem that some Hadoop commands are not available when accessing OSS data sources.
  - Fixes the *`java.text.ParseException`*: Unparseable date problem.
  - Optimizes the support of emr-core for local debugging.
  - Interprets the "_$folder$" files created in the earlier versions as directory paths instead of regular files.
  - Adds a retry mechanism for Hadoop/Spark failing to read data in OSS.

- v1.1.1
  - Fixes the imbalance of disk usage caused by writing temporary files in OSS locally.
  - Removes the $_folder$ tag files created during OSS directory creation in a job execution.
- v1.1.0
  - Upgrades the LogHub SDK to 0.6.2. Abandons the Client DB mode and uses the Server DB mode instead.
  - Upgrades the OSS SDK to 2.2.0. Fixes the run-time exceptions caused by the bugs of the OSS SDK.
  - Adds support for MNS.
  - Compatibility.
    - For the versions 1.0.x SDKs.
      - Interface:
        - Compatible
      - Namespace:
        - Incompatible: Adjusts the package structure. Modifies the package name from com.aliyun to com.aliyun.emr.
  - Modifies the groupId of the project from com.aliyun to com.aliyun.emr. The modified dependency in the POM file is as follows:

```
<dependency>
      <groupId>com.aliyun.emr</groupId>
      <artifactId>emr-sdk_2.10</artifactId>
      <version>1.1.3.1</version>
  </dependency>
```

- v1.0.5
  - Optimizes the LoghubUtils interface and parameter input.
  - Optimizes the output format of data in LogStore. Adds the *topic* and the *source* fields.
  - Adds the parameter configurations for the time interval of pulling data in LogStore. Parameter name: **spark.logservice.fetch.interval.millis**. Default value: 200. Unit: milliseconds.
  - Upgrades the ODPS SDK to 0.20.7-public.
- v1.0.4

  - Downgrades the dependency of Guava to 11.0.2 to avoid a conflict with the dependency of Guava in Hadoop.
  - The MapReduce task supports files greater than 5 GB.
- v1.0.3
  - Adds configuration parameters related to the OSS Client.
- v1.0.2
  - Fixes the bug that the OSS URIs are resolved incorrectly.
- v1.0.1
  - Optimize the settings for OSS URIs.
  - Adds support for MQ.
  - Adds support for Log Service.
  - Supports the Append Object feature of OSS.
  - Supports uploading data in OSS using the multipart upload API.
  - Supports copying data from OSS using the upload part copy mode.

8 API reference

8.1 Description of calls

8.1.1 Request structure

Request structure

Endpoints

emr.aliyuncs.com

Protocols

The system supports HTTP and HTTPS communications. We recommend that you send requests through HTTPS for enhanced security.

Methods

You can send requests using the HTTP GET method. In an HTTP GET request, parameters are included as part of the URL.

Parameters

Each request must specify the operation to be executed, namely, the action parameter (for example, CreateDatabase), and each operation must contain the public request parameters and the specific request parameters of the specified operation.

Encoding

Requests and responses are encoded using the UTF-8 character set.

8.1.2 Common parameters

Common parameters

Common request parameters

Common request parameters refer to the request parameters that each API requires.

Parameters

Name	Type	Required	Description
Format	String	No	The format of the returned value. Valid values: JSON and

Name	Type	Required	Description
			XML. Default value: XML.
Version	String	Yes	The version of the API. Format: YYYY-MM-DD. The current version is 2016-04-08.
AccessKeyId	String	Yes	The Key ID provided by Alibaba Cloud for you to access services.
Signature	String	Yes	The result string of the signature. For more information about how to calculate a signature, see Sign signatures.
SignatureMethod	String	Yse	The mode of the signature. HMAC-SHA1 is supported currently.
Timestamp	String	Yes	The timestamp to request. The format of the date follows the ISO8601 standard and uses UTC time. Format: YYYY-MM-DDThh:mm:ssZ. For example, 2013-08-15T12:00:00Z indicates 20:00:00, Aug 15, 2013, Beijing time.
SignatureVersion	String	Yes	The algorithm version of the signature. The current version is 1.0.
SignatureNonce	String	Yes	The unique random number that is used to prevent network replay attacks. You

Name	Type	Required	Description
			must use different random numbers for the requests.

Examples

```
https://emr.aliyuncs.com/
? Format=json
&Version=2016-04-08
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&OwnerId=12345678
&Timestamp=2014-10-10T12:00:00Z
```

Common response parameters

Each time you make a call to an API, the system returns a unique identification code (RequestId), regardless whether the request is successful.

- Examples
  - Successful response examples

    Returned XML results include information about whether the request is successful and specific scenario information. Examples

    ```
    {
      "RequestId": "4C467B38-3910-447D-87BC-AC049166F216",
      /* Returned results
    */
    }
    ```

  - Error response examples

    No result is returned after the API call fails. You can find the cause of error according to Error codes. When the HTTP request fails, a status code in the 4xx of 5xx format is returned. The response contains specific error codes and error information. The message body also contains a globally unique RequestId and the HostId of the site you request to access. If you cannot find the cause of the error, contact Alibaba Cloud After-Sales Support and provide the HostId and the RequestId for help.

    ```
    {
      "RequestId": "7463B73D-35CC-4D19-A010-6B8D65D242EF",
      "HostId": "emr.aliyuncs.com",
      "Code": "UnsupportedOperation",
      "Message": "The specified action is not supported."
    ```

```
}
```

- Common error codes

  For more information, see *ECS common error codes*

8.1.3 Sign signatures

ECS performs identity authentication on each access request. Therefore, whether submitted through HTTP or HTTPS, a request must contain signature information.

ECS uses symmetric encryption based on an Access Key ID and Access Key Secret to verify the identity of a request sender. The Access Key ID and Access Key Secret are issued to users by Alibaba Cloud (users can request and manage keys on the website of Alibaba Cloud). The Access Key ID indicates the identity of the user. The Access Key Secret is the secret key used to encrypt the signature string and to verify the signature string on the server. It must be kept strictly confidential and only be known to Alibaba Cloud and authenticated users.

Endpoint

Follow the instructions below to sign the signatures for your requests.

1. Use request parameters to construct a Canonicalized Query String.

   a. Sort parameters

   Sort all request parameters (including common request parameters and user-defined parameters for the given request interfaces described in this document, excluding the Signature parameter mentioned in "Common request parameters") alphabetically by the parameter name.

   Note:
   If you use the GET method to submit requests, these parameters are included in the request URI that is the part after the question mark (?) following the ampersand (&) in the URI .

   b. Encode parameters

   Encode the name and value of each request parameter. The names and values must undergo URL encoding using the UTF-8 character set. The URL encoding rules are as follows:

   - Uppercase letters (A–Z), lowercase letters (a–z), numbers (0–9), hyphens (-), underscore s (_), periods (.), and tildes (~) are not encoded.

- Other characters are encoded in "%XY" format, with XY representing the character ASCII code in hexadecimal notation. The double quotation mark (") is coded as %22.
- Extended UTF-8 characters are encoded in “%XY%ZA… “
- Note that an English space is encoded as %20 instead of a plus sign (+).

Note:

This encoding method is similar to but different from the application/x-www-form-urlencoded MIME encoding algorithm (such as the implementation of the java.net.URLEncoder class provided by the Java standard library). You can use this encoding method directly by replacing the plus sign (+) in the encoded string with "%20", the asterisk (*) with "%2A", and change "%7E" back to the tilde (~) to conform to the encoding rules described above. This algorithm can be implemented using the following percentEncode() method:

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws
UnsupportedEncodingException {
  return value ! = null ? URLEncoder.encode(value, ENCODING).
replace("+", "%20").replace("*", "%2A").replace("%7E", "~") :
null;
}
```

c. Connect the encoded parameter names and values with equal signs (=).

d. Then, sort the parameter name and value pairs connected by equal signs in alphabetical order, and connect them with an ampersand (&) to produce the canonicalized query string.

2. Follow the rules below to use the canonicalized query string to construct the string for signature calculation:

```
StringToSign=
  HTTPMethod + “&” +
  percentEncode(“/”) + ”&” +
   percentEncode(CanonicalizedQueryString)
```

In this formulation, HTTPMethod is the HTTP method (such as GET) used for request submission. percentEncode(“/“) is the encoded value (%2F) for the character “/“ according to the URL encoding rules described in 1.ii. percentEncode (CanonicalizedQueryString) is the encoded string for the canonicalized query string constructed in Step 1 according to the URL encoding rules described in 1.ii.

3. Use the string for signature calculation to calculate the HMAC value of the signature based on RFC2104.

Note:

The key used for signature calculation is your secret access key adding the ampersand “&” (ASCII:38) and it is based on the SHA1 hashing algorithm.

4. Encode the HMAC value into a string based on Base64 encoding rules to obtain the signature value.
5. Add the obtained signature value to request parameters as the Signature parameter to complete the request signing process.

Note:

The obtained signature value requires URL encoding based on the RFC3986 rule like other parameters before it is submitted to the GPDB server as the final request parameter value.

Take "DescribeDBInstances" as an example, the request URL before a signature is :

```
http://ecs.aliyuncs.com/?TimeStamp=2016-02-23T12:46:24Z&Format=XML&
AccessKeyId=testid&Action=DescribeRegions&SignatureMethod=HMAC-SHA1&
SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2014-05-
26&SignatureVersion=1.0
```

The corresponding StringToSign is:

```
GET&%2F&AccessKeyId%3Dtestid&Action%3DDescribeRegions&Format%3DXML
&SignatureMethod%3DHMAC-SHA1&SignatureNonce%3D3ee8c1b8-83d3-44af-
a94f-4e0ad82fd6cf&SignatureVersion%3D1.0&TimeStamp%3D2016-02-23T12%
253A46%253A24Z&Version%3D2014-05-26
```

Assume that the value of the Access Key Secret parameter is testsecret. Then the key used for HMAC calculation is testsecret&, and the calculated signature value is:

```
CT9X0VtwR86fNWSnsc6v8YGOjuE=
```

The signed request URL is (with the Signature parameter added):

```
http://ecs.aliyuncs.com/?SignatureVersion=1.0&Action=DescribeRegions
&Format=XML&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&
Version=2014-05-26&AccessKeyId=testid&Signature=CT9X0VtwR86fNWSnsc6v
```

```
8YGOjuE%3D&SignatureMethod=HMAC-SHA1&TimeStamp=2016-02-23T12%3A46%
3A24Z
```

8.2 List of operations by function

List of operations by function

Clusters

- *CreateCluster* Creates a cluster.
- *DescribeCluster* Queries the details of a cluster.
- *ListClusters* Queries the list of clusters.
- *ModifyClusterName* Modifies the name of a cluster.
- *ReleaseCluster* Releases a cluster.
- *ResizeCluster* Resizes a cluster.

Jobs

- *CreateJob* Creates a job.
- *ModifyJob* Modifies a job.
- *DescribeJob* Queries the details of a job.
- *DeleteJob* Deletes a job.
- *ListJobs* Queries the list of jobs.

Execution plans

- *CreateExecutionPlan* Creates an execution plan.
- *DescribeExecutionPlan* Queries the details of an execution plan.
- *ListExecutionPlans* Queries the list of execution jobs.
- *ModifyExecutionPlan* Modifies an execution plan.
- *DeleteExecutionPlan* Deletes an execution plan.
- *RunExecutionPlan* Runs an execution plan.
- *ResumeExecutionPlanScheduler* Starts periodic scheduling for an execution plan.
- *SuspendExecutionPlanScheduler* Stops periodic scheduling for an execution plan.
- *ListExecutionPlanInstances* Queries execution plan instances.
- *ListJobExecutionInstances* Queries the list of job instances.
- *KillExecutionJobInstance* Stops running a job instance.
- *ListJobInstanceWorkers* Queries the running nodes of a job instance.

Resources

- *API Error Center*

8.3 Cluster

8.3.1 CreateCluster

You can this operation to create a cluster.

Request parameters

Field	Type	Required	Default value	Description
Name	String	Yes	None	The name of the cluster. Length constraints: Minimum length of 1 character. Maximum length of 64 characters. Only Chinese characters, letters, numbers, hyphens (-), and underscores (_) are allowed.
RegionId	String	Yes	None	Currently, the available regions are China East 1 (Hangzhou), China East 2 (Shanghai), China South 1 (Shenzhen), China North 2 (Beijing), China North 3 (Zhangjiakou), US West 1 (Silicon Valley), Asia Pacific SE 1 (Singapore), and Germany (Frankfurt).

Field	Type	Required	Default value	Description
ZoneId	String	No	None	The zone location ID of the supported available zone in the China East 1 (Hangzhou) region : cn-hangzhou -b. The zone location IDs of the supported available zones in the China East 2 (Shanghai) region : cn-beijing-a, cn-beijing-b, and cn-beijing-c.
LogEnable	Boolean	Yes	None	Specifies whether to enable or disable storing logs. Make sure you have activated *OSS* to use this function.
LogPath	String	No. It is required if LogEnable== true.	None	The location of the log that is stored in OSS. The format is: oss ://bucketname/dir
SecurityGroupId	String	No	None	The ID of a security group. You can create a security group in the ECS console and use it. Note: If you are using an existing security group, the default security group policy is applied to this security

Field	Type	Required	Default value	Description
				group. Only port 22 is open at the inbound and all ports are open at the outbound.
SecurityGroupName	String	No	None	If the ID of the security group is not specified, this name is used to create a new security group. After the cluster has been created, you can see the ID of the created security group in the detailed information of the cluster. The default security group policy is applied to this security group. Only port 22 is open at the inbound and all ports are open at the outbound.
IsOpenPublicIp	Boolean	No	true	Specifies whether to set the public IP address. The default bandwidth is 8 MB.
EmrVer	String	Yes	None	The product version of E-MapReduce. For example, emr-2.4.1, emr-3.0.1.
ClusterType	String	Yes	None	The type of the cluster. Currently

Field	Type	Required	Default value	Description
				, only the Hadoop type is supported .
HighAvailabilityEnable	Boolean	No	false	Specifies whether to enable or disable high availability. If high availability is enabled, a minimum of two master nodes is required.
MasterPwdEnable	Boolean	No	None	Indicates whether to set the root passwords for master nodes.
MasterPwd	String	No. It is required if MasterPwdEnable==true.	None	The password must meet the following requirements. Length constraints: Minimum length of 8 characters. Maximum length of 30 characters . It must contain three types (uppercase letters , lowercase letters, numbers , and special symbols) of characters.
EcsOrder	String	Yes	None	The machine information of ESC instances contained by clusters. It is in JSON format. For

Field	Type	Required	Default value	Description
				example ， ”[{‘NodeCount’:1, ‘NodeType’:’MASTER’,’Instanc {‘NodeCount’:2, ‘NodeType’:’CORE’,’InstanceTy *EcsOrder*
ChargeType	String	Yes	None	The billing method. PostPaid : Pay-As-You-Go. PrePaid: Subscription.
Period	Integer	Only required when ChargeType== PrePaid	None	The length of the subscription. Unit : months. Valid values: 1, 2, 3, 4 , 5, 6, 7, 8, 9, 12, 24, and 36.
BootstrapActions	List *BootstrapAction*	No	None	The lists of bootstrap actions . The maximum number is 16. If the maximum is exceeded, only the first 16 lists are retained.
Configurations	String	No	None	Provides the path of an OSS file . See the user guide for the file content.
VpcId	String	No	None	VPC ID
VSwitchId	String	No	None	The ID of VSwitch
NetType	String	No	classic	Valid values: classic and vpc . Default value: classic.
IoOptimized	Boolean	No	true	Specifies whether to enable or disable IO optimization.

Field	Type	Required	Default value	Description
				Default value: true.
instanceGeneration	String	No	None	The generation of the ECS instance. Set the value to ecs-1 or ecs-2.

Response parameters

Field	Type	Description
ClusterId	String	The ID of the cluster.
EmrOrderId	String	The ID of the EMR service order.
MasterOrderId	String	The ID of the master node service order.
CoreOrderId	String	The ID of the core node service order.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=CreateCluster
&Name=smokeTestCreateCluster1
&ClusterType=HADOOP
&HighAvailabilityEnable=false
&SecurityGroupId=sg-234r6xoqe
&LogEnable=false
&EmrVer=EMR+1.1.0
&ZoneId=cn-hangzhou-b
&IsOpenPublicIp=true
&RegionId=cn-hangzhou
&MasterPwdEnable=false
&VpcId=vpc-239kkz237
&VSwitch=vsw-234iqq7ae
&NetType=vpc
&EcsOrder=[{"nodeCount":1,"nodeType":"master","instanceType":"ecs.n1
.large","diskType":"CLOUD_EFFICIENCY","diskCapacity":80}]
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07",
    "InstanceId": "C-13A570B821D4BAB3"
```

```
}
```

8.3.2 ModifyClusterName

You can call this operation to modify the name of a cluster.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the cluster.
Name	String	Yes	None	The new name of the cluster. The same naming rules for creating the cluster are required. Length constraints: Minimum length of 1 character. Maximum length of 64 characters. Only Chinese characters, letters, numbers, hyphens (-), and underscores (_) are allowed.
RegionId	String	Yes	None	

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ModifyClusterName
&Id=C-13A570B821D4BAB3
&Name=Cluster_Name
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.3.3 DescribeCluster

You can call this operation to query the details of a cluster.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the cluster.
RegionId	String	Yes	None	The region in which the cluster is deployed.

Response parameters

Field	Type	Description
ClusterInfo	*ClusterInfo*	The detailed information about the cluster.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=DescribeCluster
&Id=500003771
&PageSize=50
&PageNumber=1
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

8.3.4 ReleaseCluster

You can call this operation to release a cluster.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the cluster.
ForceRelease	Boolean	No	false	Specifies whether to force the cluster to release. For more information, see *Release a cluster*.
RegionId	String	Yes	None	The region in which the cluster is deployed.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ReleaseCluster
&ForceRelease=false
&Id=C-13A570B821D4BAB3
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
```

```
}
```

8.3.5 ResizeCluster

You can call this operation to resize a cluster.

Request parameters

Field	Type	Required	Default value	Description
ClusterId	String	Yes	None	The ID of the cluster.
RegionId	String	Yes	None	
NewMasterInstances	Integer	Yes	None	The number of master nodes after resizing.
NewCoreInstances	Integer	Yes	None	The number of core nodes after resizing.
NewTaskInstances	Integer	Yes	None	A reserved word. Set the value to 0.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ResizeCluster
&NewMasterInstances=1
&NewCoreInstances=3
&NewTaskInstances=0
&ClusterId=500003112
&amp;RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
```

```
}
```

8.3.6 ListClusters

You can call this operation to query a list of clusters.

Request parameters

Field	Type	Required	Default value	Description
StatusList	Array<String>	No	None	Filter the clusters by status. For more information about cluster status, see *Status of a cluster*. Use a string array in the following format: [“CREATING”, “RUNNING”, “IDLE”]
RegionId	String	Yes	None	
PageNumber	Integer	No	1	The page number to query.
PageSize	Integer	No	10	The number of rows to display per page.

Response parameters

Field	Type	Description
Cluster	Array<ClusterInfoSimple>	A ClusterInfoSimple array. For the structure of ClusterInfoSimple, see the following table.
TotalCount	Integer	The total number of records.
PageNumber	Integer	The current page number.
PageSize	Integer	The number of records that appear on the current page.

ClusterInfoSimple

Field	Type	Description
Id	String	The ID of the cluster.
Name	String	The name of the cluster.
ChargeType	String	The billing method for the cluster. PrePaid: Subscription. PostPaid: Pay-As-You-Go.
ExpiredTime	Long	The expiration time of the cluster. The data type is TIMESTAMP. For example, 1459236255438.
Type	String	The type of the cluster. Valid values: HADOOP and HBASE.
CreateTime	Long	The creation time of the cluster. The data type is TIMESTAMP. For example, 1459236255438.
RunningTime	Integer	The running time of the cluster. Unit: seconds.
Status	String	The status of the cluster. For more information about cluster status, see *Status of a cluster*.
FailReason	*FailReason*	The cause of cluster creation failure.
CreateType	String	ON-DEMAND: Clusters that are created dynamically using execution plans. MANUAL: Clusters that are created manually.

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListClusters
&PageSize=100
&StatusList=%5B%22CREATING%22%2C+%22RUNNING%22%2C+%22IDLE%22%2C+%
22RELEASING%22%2C+%22CREATE_FAILED%22%2C+%22RELEASE_FAILED%22%5D
&PageNumber=1
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

8.4 Job

8.4.1 CreateJob

You can call this operation to create a job.

Request parameters

Field	Type	Required	Default value	Description
Name	String	Yes	None	The name of the job.
Type	String	Yes	None	The type of the job. Valid values : HADOOP, SPARK, HIVE, PIG, SQOOP, SPARK_SQL, and SHELL.
RunParameter	String	Yes	None	The parameter of the job.
FailAct	String	Yes	None	The action to take after failure . STOP: Stops creating the job . CONTINUE : Continues creating the subsequent jobs.
RegionId	String	Yes	None	

Response parameters

Field	Type	Description
Id	String	The ID of the job.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=CreateJob
&Name=CreateJobApiTest-SPARK
&Type=SPARK
```

```
&EnvParam=--class%20org.apache.spark.examples.SparkPi%20--master%
20yarn-client%20%E2%80%93num-executors%202%20--executor-memory%202g
%20--executor-cores%202%20/opt/apps/spark-1.6.0-bin-hadoop2.6/lib/
spark-examples*.jar%2010
&FailAct=STOP
&amp;RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07",
    "InstanceId": "J-13A570B821D4BAB3"
}
```

8.4.2 ModifyJob

You can call this operation to modify a job.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the job.
Name	String	Yes	None	The name of the job.
Type	String	Yes	None	The type of the job. Valid values: HADOOP, HIVE, PIG, SPARK, SQOOP, SPARK_SQL, and SHELL.
RunParameter	String	Yes	None	The parameter of the job.
FailAct	String	Yes	None	The action to take after failure. STOP: Stops modifying the job. CONTINUE: Continues modifying the subsequent jobs.
RegionId	String	Yes	None	

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ModifyJob
&Id=J-13A570B821D4BAB3
&Name=test
&Type=HIVE
&EnvParam=-f%20ossref://emr/count.sql
&FailAct=STOP
&amp;RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.4.3 DescribeJob

You can call this operation to query the details of a job.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the job.
RegionId	String	Yes	None	The ID of the region.

Response parameters

Field	Type	Description
Id	String	The ID of the job.
Name	String	The name of the job.
Type	String	The type of the job. Valid values: HADOOP, SPARK , HIVE, PIG, SQOOP, SPARK_SQL, and SHELL.
RunParameter	String	The parameter of the job.

Field	Type	Description
FailAct	String	The action to take after failure . STOP: Stops querying the details of a job. CONTINUE: Continues querying the details of the subsequent jobs.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=DescribeJob
&Id=J-13A570B821D4BAB3
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07",
    "Id": "J-13A570B821D4BAB3",
    "Name": "test",
    "FailAct": "CONTINUE",
    "Type": "HIVE",
    "RunParameter": "-f%20ossref://emr/count.sql"
}
```

8.4.4 DeleteJob

You can call this operation to delete a job.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the job.
RegionId	String	Yes	None	The ID of the region.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=DeleteJob
```

```
&Id=J-13A570B821D4BAB3
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07",
}
```

8.4.5 ListJobs

You can call this operation to query a list of jobs.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
PageNumber	Integer	No	1	The page number to query. The minimum value is 1.
PageSize	Integer	No	10	The number of rows to display per page.

Response parameters

Field	Type	Description
Jobs	Array<Job>	The array of the jobs.
TotalCount	Integer	The total number of rows.
PageNumber	Integer	The current page number.
PageSize	Integer	The number of rows to display per page.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListJobs
&RegionId=cn-hangzhou
&PageNumber=1
&PageSize=20
```

```
&Common request parameters
```

- Sample responses

  JSON format

8.5 Execution plan

8.5.1 CreateExecutionPlan

You can call this operation to create an execution plan.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	The ID of the region.
Name	String	Yes	None	The name of the execution plan.
Strategy	String	Yes	None	The running policy of the execution plan. RUN_MANUALLY: Manual execution. Run the execution plan only when triggered by users. SCHEDULE: Periodic scheduling. Run the execution plan on the set interval.
StartTime	Long	No. It is required if Strategy== SCHEDULE.	None	The start time of the periodic scheduling. Unit: millisecond. The format is TIMESTAMP (UTC). For example, 1497516900000.

Field	Type	Required	Default value	Description
TimeUnit	String	No. It is required if Strategy== SCHEDULE.	None	The time interval for periodic scheduling: DAY : The unit is the day. HOUR: The unit is the hour.
TimeInterval	Integer	No. It is required if Strategy== SCHEDULE.	None	The time interval . If the unit is the day, set the value to 1. If the unit is the hour, set the value to a number in the range from 1 to 23.
JobIdList	Array	Yes	None	The array of the job IDs. For example, [“J-90D5FC26DA8C7CA7”,”J-A5B1F503752DE01F”].
CreateClusterOnDemand	Boolean	No	false	Specifies whether to create a cluster on demand dynamically.
ClusterId	String	No. It is required if CreateClusterOnDemand= false.	None	The ID of the existing cluster.
When you set CreateClusterOnDemand to true, the cluster is created on demand according to the settings of the following parameters.				

Field	Type	Required	Default value	Description
ClusterName	Boolean	Yes	None	The name of the newly created cluster.
ZoneId	String	No	None	The zone location ID. For example, cn-hangzhou-b.
LogEnable	Boolean	Yes	None	Specifies whether to enable storing logs. Before using this function, make sure you have activated *OSS*.
LogPath	String	Required only when LogEnable ==true	None	The location of the log that is stored in OSS. The format is: oss ://bucketname/dir
SecurityGroupId	String	Yes	None	The ID of a security group. You can create a security group in the ECS console and use it. Note: If you are using an existing security group, the default security group policy is applied to this security group: Only port 22 is open at the inbound and all ports are open at the outbound.
IsOpenPublicIp	Boolean	No	true	Specifies whether to enable the public network IP address. The

Field	Type	Required	Default value	Description
				default bandwidth is 8 MB.
EmrVer	String	Yes	None	The product version of E-MapReduce. For example, EMR 1.0.0 and EMR 1.1.0.
ClusterType	String	Yes		The type of the cluster. Hadoop is supported. Hbase is not supported.
HighAvailabilityEnable	Boolean	No	false	Specifies whether to enable high availability. If it is enabled, a minimum of two master nodes is required.
EcsOrder	*EcsOrder*	Yes	None	The machine information of ECS instances contained by the cluster. The format is JSON. For example, [{"nodeCount":3, "nodeType":"MASTER", "instanceType":"ecs.n1.large", "diskType":"CLOUD_EFFICIENCY", "diskCapacity":80, diskCount":1}]
BootstrapActions	List *BootstrapAction*	No	None	The lists of bootstrap actions. The maximum

Field	Type	Required	Default value	Description
				number is 16. If the maximum is exceeded, only the first 16 lists are retained.
Configurations	String	No	None	The OSS file path. For the contents of this file, see the user manual.
VpcId	String	No	None	The VPC ID.
VSwitchId	String	No	None	The ID of VSwitch.
NetType	String	No	classic	Valid values: classic and vpc. Default value: classic.
IoOptimized	Boolean	No	true	Specifies whether to enable IO optimization. The default value is true.
InstanceGeneration	String	No	None	The generation of the ECS instance. Set the value to ecs-1 or ecs-2.

Response parameters

Field	Type	Description
Id	String	The ID of the execution plan.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=CreateExecutionPlan
&ClusterName=Test
&ClusterType=HADOOP
&LogEnable=false
&SecurityGroupId=sg-234r6xoqe
&IsOpenPublicIp=true
&ZoneId=cn-hangzhou-b
```

```
&EmrVer=EMR+1.1.0
&EcsOrder=[{"nodeCount":3,"nodeType":"master","instanceType":"ecs.n1
.large","diskType":"CLOUD_EFFICIENCY","diskCapacity":80}]
&MasterIndex=1
&Name=MyJobFlow
&StartTime=1497516900000
&TimeInterval=1
&Strategy=RUN_MANUALLY
&CreateClusterOnDemand=true
&JobIdList=%5B%22J-90D5FC26DA8C7CA7%22%2C%22J-A5B1F503752DE01F%22%5D
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07",
    "Id":"WF-13A570B821D4BAB3"
}
```

8.5.2 DescribeExecutionPlan

You can call this operation to query the details of an execution plan.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the execution plan.
RegionId	String	Yes	None	The ID of the region.

Response parameters

Field	Type	Description
Id	String	The ID of the execution plan.
Name	String	The name of the execution plan.
Strategy	String	The running policy of the execution plan. RUN_MANUALLY: Manual execution. Run the execution plan only when triggered by users. SCHEDULE: Periodic scheduling. Run the execution plan on the set interval.

Field	Type	Description
StartTime	Long	The start time of the periodic scheduling.
TimeUnit	String	The time interval for periodic scheduling. DAY: The unit is the day. HOUR: The unit is the hour.
TimeInterval	Integer	The time interval. If the unit is the day, set the value to 1. If the unit is the hour, set the value to a number in the range from 1 to 23.
JobInfoList	*JobInfo*	The list of the associated jobs.
CreateClusterOnDemand	Boolean	Indicates whether to create a dynamic cluster on demand.
ClusterId	String	The ID of the cluster. If CreateClusterOnDemand== false, this value is returned. Otherwise, no value is returned because it is a cluster created on demand.
ClusterInfo	*ClusterInfo*	The detailed information about the cluster. If CreateClus terOnDemand==true, this value is returned.

Examples

- Sample requests

```
https://emr.aliyuncs.com//?Action=DescribeExecutionPlan
&Id=WF-13A570B821D4BAB3
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

JSON format

```
```

8.5.3 ListExecutionPlans

You can call this operation to query a list of execution plans.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
PageNumber	Integer	No	1	The page number to query.
PageSize	Integer	No	10	The number of rows to display per page.

Response parameters

Field	Type	Required
ExecutionPlans	Array<*ExecutionPlanInfo*>	The list of the execution plans.
TotalCount	Integer	The total number of rows.
PageNumber	Integer	The current page number.
PageSize	Integer	The number of rows to display per page.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListExecutionPlans
&PageSize=50
&PageNumber=1
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

8.5.4 ModifyExecutionPlan

You can call this operation to modify en execution plan.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	The ID of the region.
Id	String	Yes	None	The ID of the execution plan.
Name	String	Yes	None	The name of the execution plan.
Strategy	String	Yes	None	The running policy of the execution plan. RUN_MANUALLY: Manual execution. The execution plan runs only when triggered by users. SCHEDULE: Periodic scheduling. The execution plan runs based on the set interval.
StartTime	Long	No. It is required if Strategy== SCHEDULE.	None	The time when periodic scheduling takes effect.
TimeUnit	String	No. It is required if Strategy== SCHEDULE.	None	The time interval for periodic scheduling. DAY: The unit is the day. HOUR: The unit is the hour.
TimeInterval	Integer	No. It is required if Strategy== SCHEDULE.	None	The time interval. If the unit is the day, set the value to 1. If the unit is

Field	Type	Required	Default value	Description
				the hour, set the value to a number in the range from 1 to 23.
JobIdList	Array	Yes	None	The array of the job IDs. For example, [“J-24H770B821D3C56A”,”J-8A37Y0B821D3C90R”]
CreateClusterOnDemand	Boolean	No	false	Specifies whether to create a cluster on demand dynamically.
ClusterId	String	No. It is required if CreateClusterOnDemand== false.	None	The ID of the existing cluster.
When you set CreateClusterOnDemand to true, the cluster is created on demand according to the settings of the following parameters.				
ClusterName	Boolean	No	None	The name of the newly created cluster.
ZoneId	String	No	None	The zone location ID. For example, cn-hangzhou-b.
LogEnable	Boolean	Yes	None	Specifies whether to enable or disable storing

Field	Type	Required	Default value	Description
				logs. Make sure you have activated *OSS* before using this function.
LogPath	String	Only required when LogEnable ==true	None	The location of the log that is stored in OSS. The format is: oss ://bucketname/dir
SecurityGroupId	String	Yes	None	The ID of a security group. You can create a security group in the ECS console and use it. Note: If you are using an existing security group, the default security group policy is applied to this security group. Only port 22 is open at the inbound and all ports are open at the outbound.
IsOpenPublicIp	Boolean	No	true	Specifies whether to enable a public network IP address. The default bandwidth is 8 MB.
EmrVer	String	Yes	None	The product version of E-MapReduce. For example: EMR-2.4.1 or EMR-3.0.1

Field	Type	Required	Default value	Description
ClusterType	String	Yes	None	The type of the cluster. HADOOP is supported. HBASE is not supported.
HighAvailabilityEnable	Boolean	No	false	Specifies whether to enable or disable high availability. If high availability is enabled, a minimum of two master nodes is required.
EcsOrder	*EcsOrder*	Yes	None	The machine information of the ECS instances contained by the cluster is in JSON format. For example, [{“nodeCount”:3, “nodeType”:”MASTER”, “instanceType”:”ecs.n1.large”, “diskType”:”CLOUD_EFFICIENCY”, “diskCapacity”:80, diskCount”:1}]
BootstrapActions	List *BootstrapAction*	No	None	The lists of bootstrap actions. The maximum number is 16. If the maximum is exceeded, only the first 16 lists are retained.

Field	Type	Required	Default value	Description
Configurations	String	No	None	The OSS file path . For the contents of this file, see the user manual.
VpcId	String	No	None	VPC ID
VSwitchId	String	No	None	The ID of the VSwitch.
NetType	String	No	None	Valid values: classic and vpc . Default value: classic.
IoOptimized	Boolean	No	true	Indicates whether to enable IO optimization.
InstanceGeneration	String	No	No	The generation of ECS instances. Set the value to ecs-1 or ecs-2.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ModifyExecutionPlan
&Id=WF-13A570B821D4BAB3
&CreateClusterOnDemand=false
&ClusterId=C-358770B821D4B293
&JobIdList=%5B%22J-24H770B821D3C56A%22%2C%22J-8A37Y0B821D3C90R%22%5D
&Name=%E9%A2%84%E5%8F%91%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B_SSD%E7%9B%98HDFS%E5%86%99
&StartTime=1
&TimeInterval=1
&Strategy=RUN_MANUALLY
&RegionId=cn-hangzhou
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
```

```
}
```

8.5.5 DeleteExecutionPlan

You can call this operation to delete an execution plan.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the execution plan.
RegionId	String	Yes	None	The region in which the cluster is deployed.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=DeleteExecutionPlan
&Id=WF-13A570B821D4BAB3
&RegionId=cn-hangzhou
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.5.6 RunExecutionPlan

You can call this operation to run an execution plan.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
Id	String	Yes	None	The ID of the execution plan.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=RunExecutionPlan
&Id=WF-13A570B821D4BAB3
&RegionId=cn-hangzhou
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.5.7 ResumeExecutionPlanScheduler

You can call this operation to start the periodic scheduling of an execution plan.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
Id	String	Yes	None	The ID of the execution plan.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ResumeExecutionPlanScheduler
&Id=WF-13A570B821D4BAB3
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
```

```
}
```

8.5.8 SuspendExecutionPlanScheduler

You can call this operation to stop the periodic scheduling of an execution plan.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	The ID of the region.
Id	String	Yes	None	The ID of the execution plan, of which the periodic scheduling is stopped.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=SuspendExecutionPlanScheduler
&Id=WF-13A570B821D4BAB3
&RegionId=cn-hangzhou
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.5.9 ListExecutionPlanInstances

You can call this operation to query a list of execution plan instances.

Request parameters

Field	Type	Required	Default value	Description
Id	String	Yes	None	The ID of the execution plan.

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
ExecutionPlanIdList	Array<String>	Yes	None	A list of the IDs of the execution plans to query. It is a string array. For example, [“WF-5D93B00901730B5E”,”WF-5D93B00901 73ABCD”]
OnlyLastInstance	Boolean	No	false	Returns the last running logs (instances) of all the execution plans on the list if onlyLastInstance =true. If you set the value of the onlyLastInstance parameter to true, then you do not need to specify the PageNumber parameter and the PageSize parameter, and the settings do not take effect.
PageNumber	Integer	No	1	The page number to query.
PageSize	Integer	No	10	The number of rows to display per page.

Response parameters

Field	Type	Required
ExecutionPlanInstances	Array<*ExecutionPlanInstance*>	The details of the execution plan instances.

Field	Type	Required
TotalCount	Integer	The total number of records.
PageNumber	Integer	The current page number.
PageSize	Integer	The number of rows to display per page.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListExecutionPlanInstances
&ExecutionPlanIdList=%5B%22WF-5D93B00901730B5E%22%2C%22WF-5D93B00901
73ABCD%22%5D
&OnlyLastInstance=true
&PageNumber=1
&PageSize=50
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
```

8.5.10 ListJobExecutionInstances

You can call this operation to query a list of job instances.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
ExecutionPlanInstanceId	String	Yes	None	The ID of the execution plan instance.
pageNumber	Integer	No	1	The page number to query.
pageSize	Integer	No	10	The number of rows to display per page.

Response parameters

Field	Type	Required
JobInstances	Array<*JobInstance*>	The information about the job instances.
TotalCount	Integer	The total number of rows.
PageNumber	Integer	The current page number.
PageSize	Integer	The number of rows to display per page.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListJobExecutionInstances
&ExecutionPlanInstanceId=500006685
&PageSize=100
&PageNumber=1
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
```

8.5.11 KillExecutionJobInstance

You can call this operation to stop a job instance.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
JobInstanceId	String	Yes	None	The ID of the job instance.

Response parameters

Common response parameters

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=KillExecutionJobInstance
&JobInstanceId=WNE-F18B014FF410410F
```

```
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "RequestId": "34B08619-2636-49F9-AB4E-CD8D347B1E07"
}
```

8.5.12 ListJobInstanceWorkers

You can call this operation to query the running nodes of a job instance.

Request parameters

Field	Type	Required	Default value	Description
RegionId	String	Yes	None	
JobInstanceId	String	Yes	None	The ID of the job instance.

Response parameters

Field	Type	Description
JobInstanceWorkers	Array<*JobInstanceWorker*>	The information about the running nodes of the job instance.

Examples

- Sample requests

```
https://emr.aliyuncs.com/?Action=ListJobInstanceWorkers
&JobInstanceId=500006685
&RegionId=cn-hangzhou
&Common request parameters
```

- Sample responses

  JSON format

```
{
    "JobInstanceWorkers": {
        "JobInstanceWorkerInfo": [
            {
                "ApplicationId": "application_1458791367888_0002",
                "ContainerInfo": "container_1458791367888_0002_0
2_000001",
                "InstanceInfo": "10.24.28.118"
            },
```

```
            {
                "ApplicationId": "application_1458791367888_0002",
                "ContainerInfo": "container_1458791367888_0002_0
1_000002",
                "InstanceInfo": "10.24.28.11"
            },
            {
                "ApplicationId": "application_1458791367888_0002",
                "ContainerInfo": "container_1458791367888_0002_0
2_000002",
                "InstanceInfo": "10.24.28.11"
            },
            {
                "ApplicationId": "application_1458791367888_0002",
                "ContainerInfo": "container_1458791367888_0002_0
1_000001",
                "InstanceInfo": "10.24.28.118"
            }
        ]
    },
    "RequestId": "0C6F1167-4807-4503-B63F-8EDCB26E0BEE"
}
```

8.6 Data type

8.6.1 ClusterInfo

Name	Type	Description
Id	String	The ID of the cluster.
RegionId	String	The region in which the cluster has been deployed.
Name	String	The name of the cluster.
CreateType	String	ON-DEMAND: A cluster that is created on demand using the execution plan. MANUAL: A cluster that is created manually.
ChargeType	String	The billing method for the cluster. PREPAID: Subscription. POSTPAID: Pay-As-You-Go.
StartTime	Long	The creation time of the cluster in TIMESTAMP format. For example, 1459236255438.

Name	Type	Description
StopTime	Long	The stop time of the cluster in TIMESTAMP format. For example, 1459236255438.
LogEnable	Boolean	Specifies whether to enable storing the logs.
LogPath	String	The path of the stored log.
Status	String	*The status of the cluster.*
HighAvailabilityEnable	Boolean	Enables or disables high availability.
RunningTime	Integer	Running time. Unit: seconds.
MasterNodeTotal	Integer	The total number of master nodes.
CoreNodeTotal	Integer	The total number of core nodes.
FailReason	*FailReason*	The cause of the failure.
SoftwareInfo	Array<*SoftwareInfo*>	The information about the software.
ImageId	String	The ID of the used image.
SecurityGroupId	String	The ID of the security group.
SecurityGroupName	String	The name of the security group .
EcsInstances	Array<*ECSInstance*>	The information of the ECS instances.
BootstrapActions	List<*BootstrapAction*>	The list of bootstrap actions. The maximum number is 16. If the maximum number has been exceeded, only the first 16 lists are retained.
Configurations	String	The OSS file path. For the contents of this file, see the user manual.

8.6.2 EcsInstance

Name	Type	Description
NodeType	Integer	The type of the node. Valid values: master and core.
InstanceType	String	The type of the instance. The types include ecs.s3.large (4-core 8G), ecs.m1.medium (4-core 16G), ecs.m1.xlarge (8-core 32G), ecs.c1.large (8-core 16G), ecs.c2.large (16-core 32G), ecs.c2.xlarge (16-core 64G), ecs.n1.large (4-core 8G), ecs.n2.large (4-core 16G), ecs.n1.xlarge (8-core 16G), ecs.n2.xlarge (8-core 32G), ecs.n1.3xlarge (16-core 32G), and ecs.n2.3xlarge (16-core 64G).
CpuCore	Integer	The number of cores in the CPU.
MemoryCapacity	Integer	The memory capacity.
DiskType	String	The type of the disk. Valid values: CLOUD, CLOUD_EFFICIENCY, and CLOUD_SSD.
DiskCapacity	Integer	The capacity of a single disk. The range is from 40 GiB to 2,000 GiB.
BandWidth	String	The bandwidth of the network.
Nodes	Array<Node>	The details of the nodes.

- Node

Name	Type	Description
ZoneId	String	The ID of the available zone.
InstanceId	String	The ID of the instance.
Status	String	Status

Name	Type	Description
PubIp	String	The public network IP address.
InnerIp	String	The private network IP address.
DiskInfos	Array<DiskInfo>	The information of the disks.

- DiskInfo

Name	Type	Required
Device	String	The path of the device.
DiskName	String	The name of the disk.
DiskId	String	The ID of the disk.
Type	String	The type of the disk.
Size	String	The capacity of the disk.

8.6.3 EcsOrder

Name	Type	Description
InstanceType	String	ecs.s2.large (2-core, 4 GB), ecs.s3.large (4-core, 8 GB), ecs.m1.medium (4-core, 16 GB), ecs.m1.xlarge (8-core, 32 GB), ecs.c1.large (8-core, 32 GB), ecs.c2.large (12-core, 32 GB), ecs.c2.xlarge (16-core , 64 GB), ecs.n1.medium (2-core, 4 GB), ecs.n1.large (4-core, 8 GB), ecs.n1.xlarge (8 -core, 16 GB), ecs.n1.3xlarge (16-core, 32 GB), ecs.n1. 7xlarge (32-core, 64 GB), ecs. n2.large (4-core, 16 GB), ecs. n2.xlarge (8-core, 32 GB), ecs .n2.3xlarge (16-core, 64 GB), ecs.n2.7xlarge (32-core, 128 GB), ecs.sn1.large (4-core, 8 GB), ecs.sn1.xlarge (8-core, 16 GB), ecs.sn1.3xlarge (16-core, 32 GB), ecs.sn1.7xlarge

Name	Type	Description
		(32-core, 64 GB), ecs.sn2.large (4-core, 16 GB), ecs.sn2.xlarge (8-core, 32 GB), ecs.sn2.3xlarge (16-core, 64 GB), ecs.sn2.7xlarge (32-core, 128 GB)
NodeType	String	The type of the node. Valid values: MASTER and CORE.
DiskCapacity	Integer	The capacity of a single disk. The range is from 40 GB to 2,000 GB.
DiskCount	Integer	The number of disks. The range is from one to four (an integer).
NodeCount	Integer	Currently, the default number of the master node is one. The minimum number of core nodes is one.

8.6.4 BootstrapAction

Name	Type	Required	Description
Name	String	Yes	The name of the bootstrap action.
Path	String	Yes	The path of the script file in OSS.
Arg	String	No	The required argument string for the script file.

8.6.5 ExecutionPlanInfo

Name	Type	Description
Id	String	The ID of the execution plan.
Parameter	String	The name of the execution plan.

Name	Type	Description
CreateClusterOnDemand	Boolean	Indicates whether the execution plan is created on demand automatically.
Strategy	String	The method for implementing the execution plan. RUN_MANUALLY: indicates that you can implement an execution plan manually. SCHEDULE: indicates that an execution plan is automatically implemented at the scheduled time.
Status	String	The status of the schedule.
StartTime	String	The start time of the effective period of the schedule.
TimeUnit	String	The unit of the time interval of the periodic schedule. Valid values: DAY and HOUR.
TimeInterval	String	The time interval. If you specify DAY for TimeUnit, the value of this parameter must be 1. If you specify HOUR for TimeUnit, the value of this parameter ranges from 1 to 23.

8.6.6 ExecutionPlanInstance

Name	Type	Description
Id	String	The instance ID of the execution plan.
ExecutionPlanId	String	The ID of the execution plan.
ExecutionPlanName	String	The name of the execution plan.
StartTime	Long	The start time.
RunTime	Integer	The running time. Unit: seconds.
ClusterId	String	The ID of the cluster.

Name	Type	Description
ClusterName	String	The name of the cluster.
ClusterType	String	The cluster type.
Status	String	The status.

8.6.7 JobInfo

Name	Type	Description
Id	String	Job ID
Name	String	Job name
Type	String	Job type
RunParameter	String	Job running parameters
FailAct	String	Processing method when the job fails

8.6.8 JobInstance

Name	Type	Description
Id	String	The ID of the job instance.
JobName	String	The name of the job.
StartTime	Long	The start time of the job instance.
RunTime	Integer	The running time of the job instance. Unit: Seconds.
JobType	String	The type of the job.
JobId	String	The ID of the job.
ClusterId	String	The ID of the cluster.
Status	String	The status of the job instance.

8.6.9 JobInstanceWorker

Name	Type	Description
ApplicationId	String	The ID of the application.
InstanceInfo	String	The private network IP address of the node.

Name	Type	Description
ContainerInfo	String	The ID of the container.

8.6.10 SoftwareInfo

Name	Type	Description
EmrVer	String	The version number of E-MapReduce.
ClusterType	String	The type of the cluster.
Softwares	Array<Software>	The information about the software.

Software

Name	Type	Description
Name	String	The name of the software.
Version	String	The version of the software.

8.6.11 FailReason

The cause of failure.

Name	Type	Description
Error code	String	The error code.
ErrorMsg	String	The error message.
RequestId	String	The ID of the request.

8.7 Enumerations

Enumerations

Cluster

- createType (the type of the cluster creation):

Enumerator	Description
ON-DEMAND	Creates a cluster on demand.
MANUAL	Creates a cluster manually.

- clusterType (The type of the cluster):

Enumerator	Description
HADOOP	Apache Hadoop.
HBASE	Apache HBase.

- clusterStatus (the status of the cluster):

Enumerator	Description
CREATING	The cluster is being created.
CREATE_FAILED	The cluster fails to be created.
RUNNING	The cluster is running.
IDLE	The cluster is idle.
RELEASING	The cluster is being released.
RELEASE_FAILED	The cluster fails to be released.
RELEASED	The cluster has been released.
WAIT_FOR_PAY	The cluster has not been paid.
ABNORMAL	The status of the cluster is abnormal.

- DiskType (the type of the disk):

Enumerator	Description
CLOUD	Cloud disk.
CLOUD_EFFICIENCY	Ultra disk.
CLOUD_SSD	SSD disk.

- ecs instance status

Enumerator	Description
ABNORMAL	The status of the instance is abnormal.
NORMAL	The status of the instance is normal.
RESIZING	The instance is being resized.
INITIALIZING	The instance is being initialized.
RELEASED	The instance has been released.

Execution plan

- executionPlan Instance Status (the status of the execution plan instance):

Enumerator	Description
WF_INSTNACE_READY	The instance is not running.
WF_INSTNACE_RUNNING	The instance is running.
WF_INSTNACE_FAILED	The instance fails to run.
WF_INSTNACE_SUCCESS	The instance runs successfully.

- executionPlan status (the status of the execution plan):

Enumerator	Description
SCHEDULING	The execution plan is being scheduled.
NO_SCHEDULE	The execution plan has not been scheduled . Either the scheduling of the execution plan is paused or the execution plan is executed manually.

- Strategy

Enumerator	Description
RUN_MANUALLY	Manual execution.
SCHEDULE	Periodic scheduling.

- timeUnit (the unit of the time interval for periodic scheduling)

Enumerator	Description
DAY	The day.
HOUR	The hour.

Job

- jobType (the type of the job):

Enumerator	Description
SPARK	A Spark job.
HADOOP	A Hadoop MapReduce job.
HIVE	A Hive job.
PIG	A PIG job.

- jobFailAct (The action to take after failure):

Enumerator	Description
STOP	Stops running the job.
CONTINUE	Continues to run the subsequent jobs.

- job instance status

Enumerator	Description
JOB_INSTANCE_WAITING	The instance is waiting to run.
JOB_INSTANCE_SUBMISSION_FAILED	The instance fails to be submitted.
JOB_INSTANCE_RUNNING	The instance is running.
JOB_INSTANCE_SUCCESS	The instance runs successfully.
JOB_INSTANCE_FAILED	The instance fails to run.
JOB_INSTANCE_CANCELING	The instance is being canceled.
JOB_INSTANCE_CANCELED	The instance has been canceled.
JOB_INSTANCE_CANCEL_FAILED	The instance fails to be canceled.

9 Sqoop

Sqoop is an open-source application that is used to transfer data between different data stores. It supports various data stores.

Install Sqoop

Note:

Sqoop has been integrated with E-MapReduce since E-MapReduce 1.3. If you are using E-MapReduce 1.3 or later, you can skip this section.

If the version you are using is earlier than E-MapReduce 1.3, you can install Sqoop as follows:

1. Download Sqoop 1.4.6 from the official site (*Click to download*). If you cannot open the *sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz* file that you downloaded, try to download the file from the mirror site *http://mirror.bit.edu.cn/apache/sqoop/1.4.6/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz* by executing the following command.

   ```
   wget http://mirror.bit.edu.cn/apache/sqoop/1.4.6/sqoop-1.4.6.bin__ha
    doop-2.0.4-alpha.tar.gz
   ```

2. Execute the following command to extract the *sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz* file to the Master node.

   ```
   tar zxf sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
   ```

3. Install the MySQL driver to import data from MySQL. Download the latest version from the official site (*Click to download*). In addition, you can execute the following command to download the latest version (take version 5.1.38 as an example).

   ```
   wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector
   -java-5.1.38.tar.gz
   ```

4. Extract the jar file to the lib folder in the Sqoop folder.

Transfer data

Scenarios:

- MySQL -> HDFS
- HDFS -> MySQL
- MySQL -> Hive
- Hive -> MySQL

- Free-form query imports

> **Note:**
> You must switch your user account to hadoop before executing commands in later sections.

```
su hadoop
```

- Import data from MySQL into HDFS

  Execute the following command on the Master node of the cluster:

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --check-column
 <col> --incremental <mode> --last-value <value> --target-dir <hdfs-
dir>
```

  Parameters:

  - dburi: the connection string of a database such as *jdbc:mysql://192.168.1.124:3306/*. When a connection string includes any parameter, you must enclose the connection string within single quotation marks (') such as *jdbc:mysql://192.168.1.124:3306/mydatabase? useUnicode=true*
  - dbname: the name of a database such as user.
  - username: the username that is used to log on to a database.
  - password: the password that username with the username.
  - tablename: the name of a MySQL table.
  - col: the name of a column to be queried.
  - mode: specifies how Sqoop determines which rows are new. Valid values: append and lastmodified.
  - value : specifies the maximum value of a column to be checked from the previous import.
  - hdfs-dir: the HDFS directory that you import data into such as*/user/hive/result*.

  For more information about parameters, see *Sqoop import*.

- Import data from HDFS into MySQL

You must create MySQL tables that comply with the data structure of HDFS in advance. Then you can execute the following command on the Master node of a cluster to specify a directory that you import data into.

```
sqoop export --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --export-dir <
hdfs-dir>
```

Parameters:

- dburi: the connection string of a database such as *jdbc:mysql://192.168.1.124:3306/*. When any parameter is included in a connection string, enclose the connection string within single quotation marks (') such as *jdbc:mysql://192.168.1.124:3306/mydatabase? useUnicode=true*
- dbname: the name of a database, such as user.
- username: the username that is used to log on to a database.
- password: the password that is associated with the username.
- tablename: the name of a MySQL table.
- hdfs-dir: the directory of HDFS from which you import data into MySQL such as */user/hive/result*.

For more information about parameters, see *Sqoop export*

- Import data from MySQL into Hive

When you execute the following command on the Master node of a cluster to import data from MySQL, a Hive table will be created as follows.

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --check-column <
col> --incremental <mode> --last-value <value> --fields-terminated-
by "\t" --lines-terminated-by "\n" --hive-import --target-dir <hdfs-
dir> --hive-table <hive-tablename>
```

Parameters:

- dburi: the connection string of a database such as *jdbc:mysql://192.168.1.124:3306/*. When a connection string includes any parameter, you must enclose the connection string within single quotation marks (') such as *jdbc:mysql://192.168.1.124:3306/mydatabase? useUnicode=true*
- dbname: the name of a database such as user.
- username: the username that is used to log on to a database.

- password: the password that is associated with the username.
- tablename: the name of a MySQL table.
- col: the name of a column to be queried.
- mode: specifies how Sqoop determine which rows are new. Valid values: append and lastmodified. When you import data into Hive by Sqoop, you cannot use the append mode.
- value: specifies the maximum value of a column to be queried from the previous import.
- hdfs-dir: the directory of HDFS from which you import data into MySQL such as *`/user/hive/result`*.
- hive-tablename: the table name of Hive such as xxx.yyy.

For more information about how to use parameters, see *Sqoop import*.

- Import data from Hive into MySQL

  You can refer to the previous command that is used to import data from HDFS into MySQL. In addition, you must specify the HDFS directory of Hive tables from which you import data into MySQL.

- Import data from MySQL into OSS

  The process is similar to importing data from MySQL into HDFS, except for the configuration of the target-dir parameter. You can execute the following command on the Master node of a cluster:

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --check-column <
col> --incremental <mode> --last-value <value> --target-dir <oss-dir
> --temporary-rootdir <oss-tmpdir>
```

Note:

- The endpoint of an OSS host can be: intranet endpoint, Internet endpoint, or VPC endpoint. For a classic network, you must specify the intranet endpoint. For example, the OSS intranet endpoint of the China (Hangzhou) region is *`oss-cn-hangzhou-internal.aliyuncs.com`*. For a VPC, you must specify the VPC endpoint. For example, the OSS VPC endpoint of the China (Hangzhou) region is *`vpc100-oss-cn-hangzhou.aliyuncs.com`*.
- Currently, when you import data into OSS, you cannot specify the **`delete-target-dir`** parameter. Otherwise, the error message Wrong FS occurs. When you want to overwrite

a directory, you can execute the **hadoop fs -rm -r osspath** command to remove this OSS directory before using Sqoop.

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --check-column <
col> --incremental <mode> --last-value <value> --target-dir <oss-dir
> --temporary-rootdir <oss-tmpdir>
```

Parameters:

- dburi: the connection string of a database such as *jdbc:mysql://192.168.1.124 :3306/* . When a connection string includes any parameter, enclose the connection string within single quotation marks (') such as *jdbc:mysql://192.168.1.124:3306/ mydatabase? useUnicode=true*
- dbname: the name of a database such as user.
- username: the username that is used to log on to a database.
- password: the password that is associated with the username.
- tablename: the name of a MySQL table.
- col: the name of a column to be queried.
- mode: used by Sqoop to determine which rows are new rows. Valid values: append and lastmodified.
- value: specifies the maximum value of a column to be queried from the previous import.
- oss-dir: the OSS directory that you import data into *oss://<accessid>:<accesskey>@<bucketname>.oss-cn-hangzhou-internal.aliyuncs.com/result*。
- oss-tmpdir: the temporary target folder. You must specify this parameter when you specify the append mode. If the destination directory already exists in HDFS, Sqoop will stop to import and overwrite that directory's contents. If you specify the append mode, Sqoop will import data to a temporary directory and then rename the files to the normal target directory in a manner that does not conflict with the existing filenames in that directory.

For more information about available parameters, see *Sqoop import*.

- Import data from OSS into MySQL

The process is similar to importing data from MySQL to HDFS, except for the configuration of the export-dir parameter. You must create MySQL tables that comply with the data structure of OSS in advance.

Then you can execute the following command on the Master node of a cluster to specify the directory from which you want to import data.

```
sqoop export --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --table <tablename> --export-dir <
oss-dir>
```

Parameters:

- dburi: the connection string of a database such as *jdbc:mysql://192.168.1.124 :3306/* . When a connection string includes any parameter, you must enclose this connection string within single quotation marks (') such as *jdbc:mysql://192.168.1. 124:3306/mydatabase? useUnicode=true*
- dbname: the name of a database such as user.
- username: the username that is used to log on to a database.
- password: the password that is associated with the username.
- tablename: the name of a MySQL table.
- oss-dir: the OSS directory that you import data into such as*oss://<accessid>:< accesskey>@<bucketname>.oss-cn-hangzhou-internal.aliyuncs.com/result* .
- oss-tmpdir: the temporary directory that you import data into. You must specify this parameter when you specify the append mode. If the destination directory already exists in HDFS, Sqoop will stop to import and overwrite that directory's contents. If you specify the append mode, Sqoop will import data to a temporary directory and then rename the files into the normal target directory in a manner that does not conflict with existing filenames in that directory.

Note:
The endpoint of an OSS host can be: intranet endpoint, Internet endpoint, or VPC endpoint. For a classic network, you must specify an intranet endpoint. For example, the OSS intranet endpoint of the China (Hangzhou) region is *oss-cn-hangzhou-internal.aliyuncs.com* . For a VPC, you must specify a VPC endpoint. For example, the OSS VPC endpoint of the China (Hangzhou) region is*vpc100-oss-cn-hangzhou.aliyuncs.com*.

For more information about available parameters, see *Sqoop export*.

- Free-form query imports

In addition to importing a set of MySQL tables, you can also import the result set of an arbitrary SQL query as follows:

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname> --username <
username> --password <password> --query <query-sql> --split-by <sp
-column> --hive-import --hive-table <hive-tablename> --target-dir <
hdfs-dir>
```

Parameters:

- dburi: the connection string of a database, such as *jdbc:mysql://192.168.1.124 :3306/* . When a connection string includes any parameter, you must enclose this connection string within single quotation marks (') such as *jdbc:mysql://192.168.1. 124:3306/mydatabase? useUnicode=true*
- dbname: the name of a database such as user.
- username: the username that is used to log on to a database.
- password: the password that is associated with the username.
- query-sql: the query statement such as `SELECT * FROM profile WHERE id>1 AND \ $CONDITIONS`. You must enclose the query statement that ends with `AND \$CONDITIONS` within double quotation marks (").
- sp-column: specifies the name of a column to be split. In general, the value of this parameter is the primary key of the MySQL table.
- hdfs-dir: the directory of HDFS from which you import data into MySQL such as */user/ hive/result*.
- hive-tablename: the name of a table that is used to import data to Hive such as xxx.yyy.

For more information about available parameters, see *Sqoop query import*.

10 SDK Reference

10.1 Create an AccessKey

To operate the E-MapReduce service by using APIs or the SDK, you must have an AccessKey.

Note:

You must keep your AccessKey.

1. Log on to the Alibaba Cloud official site: *https://www.aliyun.com/*.
2. Click **Console**.
3. Click **AccessKeys**.
4. Create and manage your AccessKey.

10.2 Java SDK

10.2.1 Download SDKs

Alibaba Cloud SDKs for E-MapReduce consists of two parts. The common part of Alibaba Cloud Java SDK is involved in aliyun-java-sdk-core. The E-MapReduce part is involved in aliyun-java-sdk-emr. We recommend that you can use Maven to manage your projects.

Maven dependencies

```
<dependency>
          <groupId>com.aliyun</groupId>
          <artifactId>aliyun-java-sdk-core</artifactId>
          <version>2.3.2</version>
     </dependency>
     <dependency>
          <groupId>com.aliyun</groupId>
          <artifactId>aliyun-java-sdk-ecs</artifactId>
          <version>2.2.3</version>
      </dependency>
```

Download links

aliyun-java-sdk-core-2.3.9.jar

aliyun-java-sdk-emr-2.2.2.jar

10.2.2 Get started

This section describes how to quickly create clusters, jobs, and execution plans by using Java SDKs.

Prerequisites

You can create a Maven project and then add Maven dependencies as follows:

```
<dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>2.3.9</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-emr</artifactId>
            <version>2.2.2</version>
         </dependency>
```

You can also download associated JAR files to your local disk. Take Eclipse as an example. You can download JAR files as follows:

1. Download the following JAR files:

   aliyun-java-sdk-core-2.3.9.jar

   aliyun-java-sdk-emr-2.2.2.jar
2. Copy the JAR files to your project folder.
3. In Eclipse, right-click your project name, and then select **Properties** > **Java Build Path** > **Add JARs**.
4. Select all JAR files that you copied in Step 2.

Now, you can use Alibaba Cloud E-MapReduce Open API Java SDK for your Eclipse project.

Initialize a client

```
IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", "<
Your-AccessKeyId>", "<Your-AccessKeySecret>");
    DefaultAcsClient client = new DefaultAcsClient(profile);
```

All operations on the E-MapReduce in SDK can be performed using this client.

Sample code

- Clusters

  — Create a cluster

  ```
  public static void main(String[] args) {
  ```

```
      IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
      DefaultAcsClient client = new DefaultAcsClient(profile);
      final CreateClusterRequest request = new CreateClus
terRequest();
      request.setName("Your-Cluster-Name");
      // if you did not specify security group id, it will create
a new security group with given name
      request.setSecurityGroupId("Your-Security-Group-Id"); // (1)
      request.setAutoRenew(false);
      request.setChargeType("PostPaid"); // PostPaid or PrePaid
      request.setClusterType("HADOOP"); // HADOOP or HBase (2)
      request.setEmrVer("EMR-1.3.0"); // emr image version
      request.setIsOpenPublicIp(true);
      request.setLogEnable(true);
      request.setLogPath("oss://Your-Bucket/Your-Folder");
      request.setMasterPwdEnable(true); // enable master password
      request.setMasterPwd("Aa123456789"); // set master node's
password
      request.setZoneId("cn-hangzhou-b"); // set zone
       // The I/O optimization parameters. The available hardware
 configurations, such as ECS instance types and cloud disk types
are determined by the specified ECS instance series.
      // For more information about available configurations, see
the Buy Now page of ECS.
      // https://ecs.console.aliyun.com/#/create/postpay/
      request.setIoOptimized(true); // You can specify I/O
optimization parameters.
      request.setInstanceGeneration("ecs-2"); // You can specify
ecs-2 as an ECS instance series. Valid values: ecs-1 and ecs-2.
      request.setNetType("classic"); // You can specify classic as
 a network type. Valid values: classic and vpc.
      List<CreateClusterRequest.EcsOrder> ecsOrders = new
ArrayList<CreateClusterRequest.EcsOrder>();
      CreateClusterRequest.EcsOrder masterOrder = new CreateClus
terRequest.EcsOrder();
      masterOrder.setIndex(1);
      masterOrder.setDiskCapacity(50);
      masterOrder.setDiskCount(2);
      masterOrder.setDiskType("CLOUD_EFFICIENCY"); // specify disk
 type (2)
      masterOrder.setInstanceType("ecs.n1.large"); // specify ecs
instance type
      masterOrder.setNodeCount(1);
      masterOrder.setNodeType("MASTER"); // MASTER or CORE (2)
      ecsOrders.add(masterOrder);
      CreateClusterRequest.EcsOrder coreOrder = new CreateClus
terRequest.EcsOrder();
      coreOrder.setIndex(2);
      coreOrder.setDiskCapacity(50);
      coreOrder.setDiskCount(4);
      coreOrder.setDiskType("CLOUD_EFFICIENCY");
      coreOrder.setInstanceType("ecs.n1.large");
      coreOrder.setNodeCount(3);
      coreOrder.setNodeType("CORE");
      ecsOrders.add(coreOrder);
      request.setEcsOrders(ecsOrders);
      try {
          CreateClusterResponse response = client.getAcsResponse(
request);
          String clusterId = response.getClusterId(); // cluster
id
```

```
            // TODO do something with this cluster
        } catch (Exception e) {
            // TODO do something
        }
    }
```

- When you create a cluster, you must specify a security group that hosts this cluster. If you did not specify the ID of a security group, then you must specify the name of this security group. You need to create a security group when you create a cluster.
- For more information, see *Enumeration*.
- The preceding code snippet creates a cluster in a classic network. If you want to create a cluster in a *VPC network*, you need to specify vpc in the request method and specify vpcid and vswitchid as follows:

```
request.setNetType("vpc"); // You can specify vpc as a network
type.
request.setVpcId("your-vpcId");
request.setVSwitchId("your-switchId");
```

- You can specify high availability parameters. For more information about high availability parameters, see the hardware configuration section of *Create a cluster*.

```
request.setHighAvailabilityEnable(true);
```

- You can specify available software components. For more information about available software components, see the software configuration section of *Create a cluster*.

```
List<String> soft = new ArrayList<String>();
soft.add("presto");
soft.add("oozie");
request.setOptionSoftWareLists(soft);
```

- You can specify a configuration item. For more information, click *here*.

```
Request. setconfigurations ("Oss: // your-bucket/your-conf.json
 ");
```

- You can specify a bootstrap operation. For more information, click *here*.

```
List<CreateClusterRequest.BootstrapAction> bootstrapActionLists
 = new ArrayList<CreateClusterRequest.BootstrapAction>();
CreateClusterRequest.BootstrapAction bootstrapActionList = new
CreateClusterRequest.BootstrapAction();
bootstrapActionList.setName("bootstrapName");
bootstrapActionList.setPath("oss://emr-agent-pack/bootstrap/run
-if.py");
bootstrapActionList.setArg("instance.isMaster=true mkdir -p /
tmp/abc");
bootstrapActionLists.add(bootstrapActionList);
```

```
request.setBootstrapActions(bootstrapActionLists);
```

- Query the details of a cluster

```
public static void main(String[] args) {
      IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
      DefaultAcsClient client = new DefaultAcsClient(profile);
      final DescribeClusterRequest request = new DescribeCl
usterRequest();
      request.setId("C-XXXXXXXXXXXXXXX"); // cluster id
      try {
          DescribeClusterResponse response = client.getAcsResponse
(request);
          DescribeClusterResponse.ClusterInfo clusterInfo =
response.getClusterInfo();
          // TODO do something with this cluster
      } catch (Exception e) {
          // TODO do something
      }
  }
```

- Query the list of clusters

```
public static void main(String[] args) {
          IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
          DefaultAcsClient client = new DefaultAcsClient(profile);
          final ListClustersRequest request = new ListCluste
rsRequest();
          request.setPageNumber(1);
          request.setIsDesc(true);
          request.setPageSize(20);
          try {
              ListClustersResponse response = client.getAcsResp
onse(request);
              List<ListClustersResponse.ClusterInfo> clusterInfos
 = response.getClusters();
              for (ListClustersResponse.ClusterInfo clusterInfo :
clusterInfos) {
                  // TODO do something with this cluster
              }
          } catch (Exception e) {
              // TODO do something
          }
      }
```

- Release a cluster

```
public static void main(String[] args) {
            IClientProfile profile = DefaultProfile.getProfile("cn
-hangzhou", "<Your AccessKeyId>", "<Your AccessKeySecret>");
            DefaultAcsClient client = new DefaultAcsClient(profile
);
            ReleaseClusterRequest request = new ReleaseClu
sterRequest();
            request.setId("C-XXXXXXXXXXXXXXX"); // specify the
cluster id you want to release.
            try {
                ReleaseClusterResponse response = client.
getAcsResponse(request);
```

```
            } catch (Exception e) {
                // TODO do something
            }
        }
```

- Jobs

  — Create a job

```
public static void main(String[] args) {
      IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
      DefaultAcsClient client = new DefaultAcsClient(profile);
      final CreateJobRequest request = new CreateJobRequest();
      request.setName("Your-Job-Name");
      request.setRunParameter("--master yarn-client --driver-
memory 4g --executor-memory 4g --executor-cores 2 --num-executors
 4 --class com.test.RemoteDebug ossref://Your-Bucket/Resource.jar
1000\"");
      request.setFailAct("CONTINUE"); // STOP or CONTINUE
      request.setType("SPARK"); // SPARK or HADOOP or HIVE or PIG
```

```
try {
            CreateJobResponse response = client.getAcsResponse(
request);
            String jobId = response.getId();
            // TODO do something with this job
        } catch (Exception e) {
            // TODO do something
        }
    }
```

  — Delete a job

> **Note:**
>
> When a job is being used by an execution plan, you cannot delete this job. You must delete this execution plan or modify this execution plan before you delete this job.

```
public static void main(String[] args) {
          IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
          DefaultAcsClient client = new DefaultAcsClient(profile);
          final DeleteJobRequest request = new DeleteJobRequest();
          request.setId("J-XXXXXXXXXXXXXXX"); // set  job id
          try {
              DeleteJobResponse response = client.getAcsResponse(
request);
          } catch (Exception e) {
              // TODO do something
          }
      }
```

- Execution plans

- Creating an execution plan

```
public static void main(String[] args) {
          IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
          DefaultAcsClient client = new DefaultAcsClient(profile);
          final CreateExecutionPlanRequest request = new
CreateExecutionPlanRequest();
          request.setName("Your-ExecutionPlan-Name");
          request.setCreateClusterOnDemand(false);
          request.setStrategy("RUN_MANUALLY"); // RUN_MANUALLY or
SCHEDULE
          request.setClusterId("C-XXXXXXXXXXXXXXX"); // specify
an existing running cluster
          List<String> jobIds = new ArrayList<String>();
          jobIds.add("J-XXXXXXXXXXXXXXX"); // specify a job
          request.setJobIdLists(jobIds);
          try {
              CreateExecutionPlanResponse response = client.
getAcsResponse(request);
              String executionPlanId = response.getId();
              // TODO do something with this execution plan
          } catch (Exception e) {
              // TODO do something
          }
      }
```

The preceding code snippet creates an execution plan with the RUN_MANUALLY type. In addition, an existing cluster is specified for this execution plan.

If you want to create an execution plan with the SCHEDULE type, you must modify the previous code snippet and add the following code snippet:

```
request.setStrategy("SCHEDULE"); // RUN_MANUALLY or SCHEDULE
          request.setStartTime(new Date().getTime()); // set start
 time
          request.setTimeUnit("DAY"); // DAY or HOUR
          request.setTimeInterval(1); // set time interval
```

If you want to create an execution plan for which a new cluster will be created to run jobs, you must modify the previous code snippet and add the following code snippet:

```
request.setCreateClusterOnDemand(true);
          request.setClusterType("HADOOP");
          request.setClusterName("Your-Cluster-Name");
          request.setEmrVer("EMR-1.3.0");
          request.setSecurityGroupId("Your-Security-Group-Id");
          request.setIsOpenPublicIp(true);
           // The I/O optimization parameters. The available
hardware configurations such as ECS instance types and cloud disk
types are determined by the specified ECS instance series.
          // For more information about how to select these
configurations, see the Buy Now page of Elastic Compute Service.
          // https://ecs.console.aliyun.com/#/create/postpay/
          request.setIoOptimized(true); // You can specify true to
 enable I/O optimization.
```

```
            request.setInstanceGeneration("ecs-2"); // You can
specify esc-2 as an ECS instance series. Valid values: ecs-1 and
ecs-2.
            request.setNetType("classic"); // You can specify
classic as a network type. Valid values: classic and vpc.
            request.setLogEnable(true);
            request.setLogPath("oss://xxx");
            request.setEcsOrders(); // TODO you can refer to the
 configurations when you create a cluster. Note that the type
of ecsOder is CreateExecutionPlanRequest.EcsOrder. The type of
ecsOder here is different from CreateClusterRequest.EcsOrder.
```

You can configure a cluster by specifying the previous parameters. For more informatio n about these parameters, see the section Create a cluster. For an execution plan that requires a new cluster to run jobs, a temporary cluster will be created whenever you execute this execution plan. The temporary cluster is created based on the specified cluster configurations and then released after the execution plan is completed. Unlike the general process of creating a cluster, you must specify a security group ID rather than a security group name when you create a cluster in this case.

The Create on demand option and the Periodic schedule option are not mutually exclusive. A new cluster is created when an execution plan starts at a scheduled time.

- Deleting an execution plan

```
public static void main(String[] args) {
            IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
            DefaultAcsClient client = new DefaultAcsClient(profile);
            final DeleteExecutionPlanRequest request = new
DeleteExecutionPlanRequest();
            request.setId("WF-XXXXXXXXXXXXXXX"); // set execution
plan id
            try {
                DeleteExecutionPlanResponse response = client.
getAcsResponse(request);
            } catch (Exception e) {
                // TODO do something
            }
        }
```

- Execute an execution plan

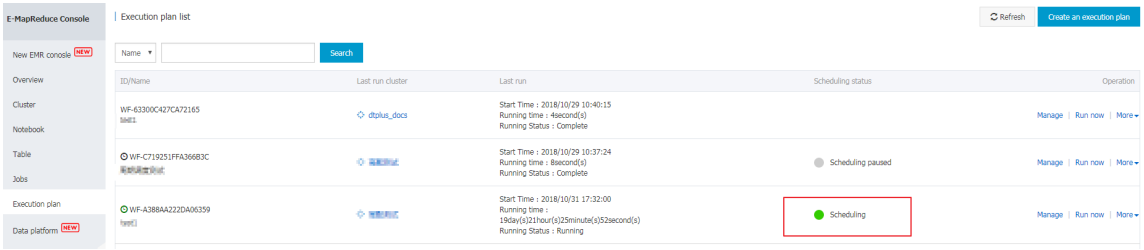
> **Note:**
> You cannot execute an execution plan that is running or being scheduled.

```
public static void main(String[] args) {
            IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your-AccessKeyId>", "<Your-AccessKeySecret>");
            DefaultAcsClient client = new DefaultAcsClient(profile);
            RunExecutionPlanRequest request = new RunExecuti
onPlanRequest();
```

```
            request.setId("WF-XXXXXXXXXXXXXXX"); // specify the
execution plan id which to run
            try {
                RunExecutionPlanResponse response = client.
getAcsResponse(request);
                String instanceId = response.getExecutionPlanInst
anceId();
                // TODO do something with this instance
            } catch (Exception e) {
                // TODO do something
            }
        }
```

- Suspend a scheduled execution plan

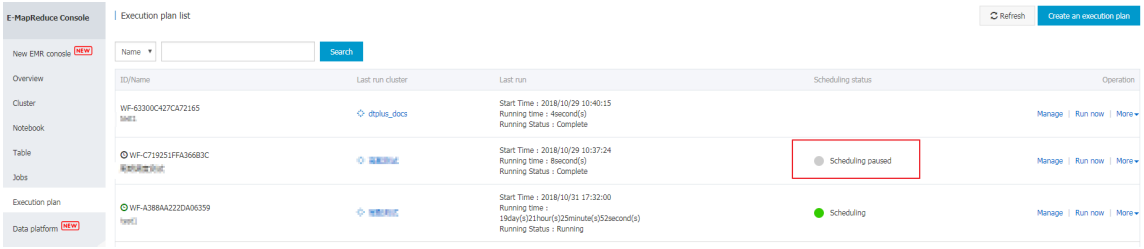
For an execution plan that will be executed periodically (as the following figure shows), you can suspend this execution plan by calling a method provided by the SDK as follows:




```
public static void main(String[] args) {
            IClientProfile clientProfile = DefaultProfile.GetProfile
("cn-shanghai", "<your-access-key-id>", "<your-access-key-secret
>");
            DefaultAcsClient client = new DefaultAcsClient(profile);
            SuspendExecutionPlanSchedulerRequest request = new
SuspendExecutionPlanSchedulerRequest();
            request.setId("WF-XXXXXXXXXXXXXXX"); // specify the
execution plan id you want to suspend
            try {
                SuspendExecutionPlanSchedulerResponse response =
client.getAcsResponse(request);
            } catch (Exception e) {
                // TODO do something
            }
        }
```

- Resume an execution plan

For an execution plan that will be executed periodically (as the following figure shows), you can resume this execution plan by calling a method provided by the SDK as follows:




```
public static void main(String[] args) {
          IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your AccessKeyId>", "<Your AccessKeySecret>");
          DefaultAcsClient client = new DefaultAcsClient(profile);
          ResumeExecutionPlanSchedulerRequest request = new
ResumeExecutionPlanSchedulerRequest();
          request.setId("WF-XXXXXXXXXXXXXXX"); // specify the
execution plan id you want to suspend
          try {
              ResumeExecutionPlanSchedulerResponse response =
client.getAcsResponse(request);
          } catch (Exception e) {
              // TODO do something
          }
      }
```

- Query the running logs of an execution plan

```
public static void main(String[] args) {
          IClientProfile profile = DefaultProfile.getProfile("cn-
hangzhou", "<Your AccessKeyId>", "<Your AccessKeySecret>");
          DefaultAcsClient client = new DefaultAcsClient(profile);
          ListExecutionPlanInstancesRequest request = new
ListExecutionPlanInstancesRequest();
          // specify execution plan ids
          List<String> executionPlanIds = new ArrayList<String>();
          executionPlanIds.add("WF-XXXXXXXXXXXXXX1");
          executionPlanIds.add("WF-XXXXXXXXXXXXXX2");
          executionPlanIds.add("WF-XXXXXXXXXXXXXX3");
          request.setExecutionPlanIdLists(executionPlanIds); // (1
)
          // specify order key (ordered by id)
          request.setIsDesc(true);
          // specify page number and page size, default page
number is 1 and default page size is 10.
          request.setPageSize(20);
          request.setPageNumber(1);
          // specify if you want to list latest instance for each
execution plan id.
          request.setOnlyLastInstance(true); // (2) default is
false
          try {
              ListExecutionPlanInstancesResponse response = client
.getAcsResponse(request);
              for (ListExecutionPlanInstancesResponse.ExecutionP
lanInstance instance : response.getExecutionPlanInstances()) {
                  // TODO do something with each instance
              }
          } catch (Exception e) {
              // TODO do something
          }
```

```
        }
```

- You can specify multiple execution plan IDs to query the running logs.
- If you query the last running log, only the last running log of a specified execution plan will be returned. You can use this feature to check whether the last execution plan has been executed successfully, or query the last running log of an execution plan.

10.2.3 Reference project

If you are still not clear about how to construct a project through the previous steps, or you want to have a project that can be directly imported to an IDE for running, you can download the following reference project code and import it to your IDE environment.

> **Note:**
> You need to change the information of the AccessKey and security group of this project to yours. For more information, see comments in the sample code.

Eclipse

Eclipse version used by the reference project: `Eclipse Java EE IDE for Web Developers. Version: Mars. 1 Release (4.5.1) Build id: 20150924-1200`

JDK version: `JDK 1.7 or later`

- Maven project

  emrmavensample.zip
- Normal project

  emrsample.zip

IntelliJ IDEA

IntelliJ IDEA version used by the reference project: `IntelliJ IDEA 15.0.1`

JDK version: `JDK 1.7 or later`

- Maven project

  emrmavensample.zip
- Normal project

  emrsample.zip

10.3 Python SDK

10.3.1 Install SDK

Python 2.7+

Install SDK

- You can install the SDK by using pip as follows.

  ```
  sudo pip install aliyun-python-sdk-emr
  ```

- You can update the E-MapReduce Python SDK as follows.

  ```
  sudo pip install aliyun-python-sdk-emr --upgrade
  ```

- You can uninstall the SDK as follows.

  ```
  sudo pip uninstall aliyun-python-sdk-emr
  ```

SDK source code

Download link

10.3.2 Sample code

Python sample code

Query the cluster list

```
#! /usr/bin/python
from aliyunsdkcore import client
from aliyunsdkemr.request.v20160408 import ListClustersRequest
clt = client.AcsClient('SFAW************','Nc2nZ6dQoiqck0
*************','cn-hangzhou') # set acessId and accessKey
request = ListClustersRequest.ListClustersRequest()
request.set_accept_format('xml') # xml or json
# You can set filter conditions to obtain clusters in the RUNNING or
 IDLE state. The state parameter is optional. You can choose not to
specify this parameter.
request.add_query_param('StatusList. 1', 'RUNNING')
request.add_query_param('StatusList. 2', 'IDLE')
result = clt.do_action(request)
print result
```

Create a cluster

```
#! /usr/bin/env python
from aliyunsdkcore import client
from aliyunsdkemr.request.v20160408 import CreateClusterRequest
clt = client.AcsClient('SFAW************','Nc2nZ6dQoiqck0
*************','cn-hangzhou') # set acessId and accessKey
request = CreateClusterRequest.CreateClusterRequest()
request.set_Name("pydemo")
request.set_ZoneId("cn-hangzhou-b")
request.set_LogEnable(False)
request.set_SecurityGroupId("sg-********")
request.set_IsOpenPublicIp(True)
request.set_ChargeType("PostPaid")
```

```
request.set_EmrVer("EMR-1.3.0")
request.set_ClusterType("HADOOP")
request.set_IoOptimized(True)
request.set_InstanceGeneration("ecs-2")
# set EcsOrder
request.add_query_param('EcsOrder. 1. NodeCount', '1')
request.add_query_param('EcsOrder. 1. NodeType', 'MASTER')
request.add_query_param('EcsOrder. 1. InstanceType', 'ecs.n1.large')
request.add_query_param('EcsOrder. 1. DiskType', 'CLOUD_EFFICIENCY')
request.add_query_param('EcsOrder. 1. DiskCapacity', '80')
request.add_query_param('EcsOrder. 1. DiskCount', '1')
request.add_query_param('EcsOrder. 1. Index', '1')
request.add_query_param('EcsOrder. 2. NodeCount', '3')
request.add_query_param('EcsOrder. 2. NodeType', 'CORE')
request.add_query_param('EcsOrder. 2. InstanceType', 'ecs.n1.large')
request.add_query_param('EcsOrder. 2. DiskType', 'CLOUD_EFFICIENCY')
request.add_query_param('EcsOrder. 2. DiskCapacity', '80')
request.add_query_param('EcsOrder. 2. DiskCount', '4')
request.add_query_param('EcsOrder. 2. Index', '2')
request.set_accept_format('json')
result = clt.do_action(request)
print result
```

Note:

All SDKs are generated automatically for various Alibaba Cloud services. You must modify the original code of some SDKs to apply to specific use cases. For example, in the current Python SDK, you must refer to the previous sample code when you use a List as a request parameter. If you specify a List with values of basic data types for request parameters, see configurations of the StatusList parameter in Query the list of clusters. If you specify a List with objects for request parameters, see configurations of the EcsOrder parameter in the Create a cluster section. For Python APIs that use a List as a request parameter such as BootstrapAction, you can refer to the same places described in the preceding sentence according to a List type. We recommend that you use the Java SDK in which a List can be used properly.

For more information, see *API overview*.