

1. Spark集群搭建

- 1.1. Spark版本选择
- 1.2. Spark编译
- 1.3. Spark依赖环境
 - 1.3.1. 安装DK
 - 1.3.2. 安装Scala
- 1.4. 安装Spark
 - 1.4.1. Spark单机模式
 - 1.4.2. Spark分布式集群
 - 1.4.3. Spark分布式高可用集群
 - 1.4.4. 配置Spark HistoryServer

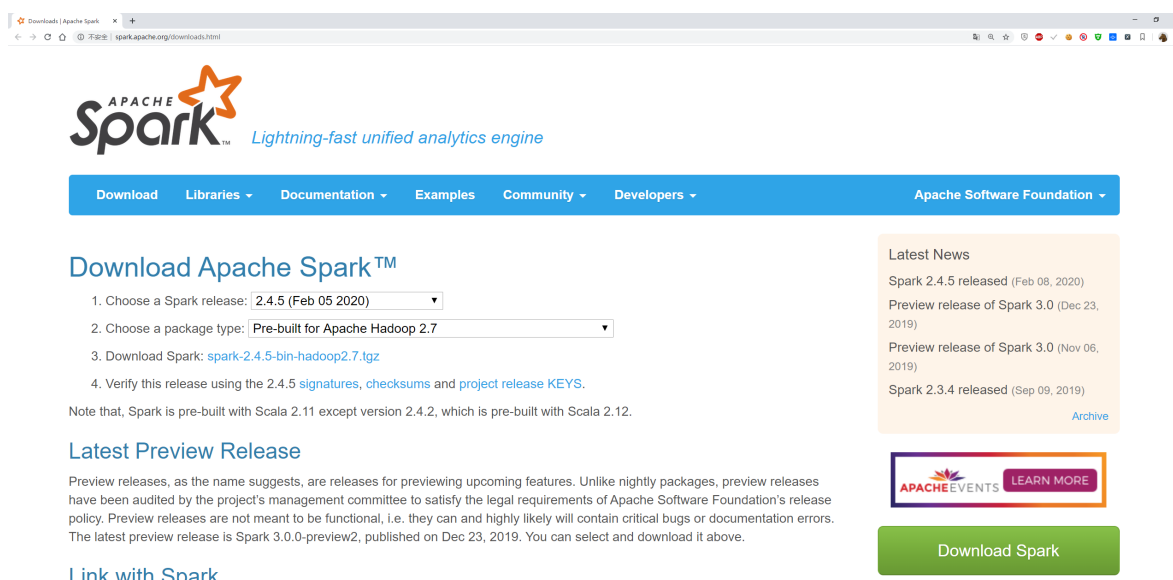
1. Spark集群搭建

1.1. Spark版本选择

三大主要版本：

spark-0.x
spark-1.x（主要spark-1.3和spark-1.6）
spark-2.x（最新spark-2.4.6）

官网首页：<http://spark.apache.org/downloads.html>



或者其他镜像站：

<https://mirrors.tuna.tsinghua.edu.cn/apache/spark/>
<https://downloads.apache.org/spark/spark-2.4.6/>

我们选择的版本：spark-2.4.6-bin-hadoop2.7.tgz

1.2. Spark编译

自行利用搜索引擎解决，可做可不做

官网：<http://spark.apache.org/docs/latest/building-spark.html>

1.3. Spark依赖环境

在官网文档中有一句话：<http://spark.apache.org/docs/latest/index.html#downloading>

Downloading

Get Spark from the [downloads page](#) of the project website. This documentation is for Spark version 2.4.5. Spark uses Hadoop's client libraries for HDFS and YARN. Downloads are pre-packaged for a handful of popular Hadoop versions. Users can also download a "Hadoop free" binary and run Spark with any Hadoop version [by augmenting Spark's classpath](#). Scala and Java users can include Spark in their projects using its Maven coordinates and in the future Python users can also install Spark from PyPI.

If you'd like to build Spark from source, visit [Building Spark](#).

Spark runs on both Windows and UNIX-like systems (e.g. Linux, Mac OS). It's easy to run locally on one machine — all you need is to have `java` installed on your system `PATH`, or the `JAVA_HOME` environment variable pointing to a Java installation.

Spark runs on Java 8, Python 2.7+/3.4+ and R 3.1+. For the Scala API, Spark 2.4.5 uses Scala 2.12. You will need to use a compatible Scala version (2.12.x).

Note that support for Java 7, Python 2.6 and old Hadoop versions before 2.6.5 were removed as of Spark 2.2.0. Support for Scala 2.10 was removed as of 2.3.0. Support for Scala 2.11 is deprecated as of Spark 2.4.1 and will be removed in Spark 3.0.

所以总结：Spark-2.4 需要依赖：**Java 8+** 和 **Python 2.7+/3.4+** 和 **Scala 2.12** 和 R 3.1+

1.3.1. 安装JDK

详情见 JDK 安装文档。

1.3.2. 安装Scala

详情见 Scala 安装文档。

1.4. 安装Spark

1.4.1. Spark单机模式

1.4.2. Spark分布式集群

Spark也是一个主从架构的分布式计算引擎。主节点是Master，从节点是Worker。所以集群规划：

Server	Master	Worker
bigdata02	√	√
bigdata03		√
bigdata04		√
bigdata05		√

第一步：从官网下载对应版本的安装包

```
wget https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.6/spark-2.4.6-bin-hadoop2.7.tgz
```

第二步：上传安装包，然后解压缩安装到对应的目录

```
tar -zxvf ~/soft/spark-2.4.6-bin-hadoop2.7.tgz -C /home/bigdata/apps/
```

第三步：修改配置文件spark-env.sh

```
cd ~/apps/spark-2.4.6-bin-hadoop2.7/conf  
mv spark-env.sh.template spark-env.sh  
vim spark-env.sh
```

修改或添加如下内容：

```
export JAVA_HOME=/usr/local/java/jdk1.8.0_181  
export SPARK_MASTER_HOST=bigdata02  
export SPARK_MASTER_PORT=7077
```

第四步：修改配置文件slave

```
cd ~/apps/spark-2.4.6-bin-hadoop2.7/conf  
mv slaves.template slaves  
vi slaves
```

添加如下内容：

```
bigdata02  
bigdata03  
bigdata04  
bigdata05
```

第五步：分发安装到所有节点

```
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7/ bigdata03:~/apps/  
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7/ bigdata04:~/apps/  
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7/ bigdata05:~/apps/
```

第六步：配置环境变量

```
vi ~/.bashrc
```

```
export SPARK_HOME=/home/bigdata/apps/spark-2.4.6-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
scp -r ~/.bashrc bigdata03:~
scp -r ~/.bashrc bigdata04:~
scp -r ~/.bashrc bigdata05:~
```

```
source ~/.bashrc
```

切记：所有节点都要配置。

第七步：启动spark集群

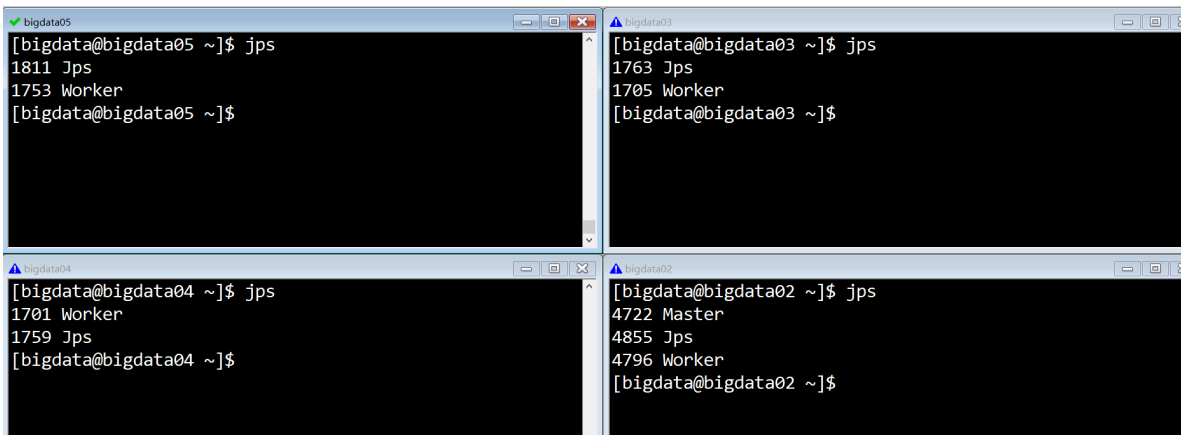
由于启动spark集群的命令是start-all.sh，而且刚好启动HDFS集群的命令也是start-all.sh，所以启动的时候注意区分。

```
$SPARK_HOME/sbin/start-all.sh
```

第八步：验证集群启动是否成功

通过守护进程检查：

```
jps
```



通过web界面检查：<http://bigdata02:8080>

```
~/apps/spark-2.4.6-bin-hadoop2.7/bin/spark-shell \  
--master spark://bigdata02:7077 \  
--executor-memory 512m \  
--total-executor-cores 1
```

```
[bigdata@bigdata02 ~]$ ~/apps/spark-2.4.5-bin-hadoop2.7/bin/spark-shell \
> --master spark://bigdata02:7077 \
> --executor-memory 512m \
> --total-executor-cores 1
20/04/04 22:27:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context web UI available at http://bigdata02:4040
Spark context available as 'sc' (master = spark://bigdata02:7077, app id = app-20200404222729-0000).
Spark session available as 'spark'.
Welcome to

  ____      __
  /  _/____/  /  ___  _____/  ___/
 /  /_  /_  /  /  /  /_  /_  /_  /
/_  /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

version 2.4.5

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_73)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

提交wordcount代码：

```
sc.textFile("file:///home/bigdata/words.txt").flatMap(_._split("
")).map(_._1).reduceByKey(_+_).foreach(println)
```

```
sc.textFile("hdfs://hadoop277ha/sparkwc/input/words.txt").flatMap(_._split("
")).map(_._1).reduceByKey(_+_).collect()
```

```
sc.textFile("file:///home/bigdata/words.txt").flatMap(_._split("
")).map(_._1).reduceByKey(_+_).collect()
```

1.4.3. Spark分布式高可用集群

上面安装的普通分布式 spark 集群存在 SPOF 的问题，Hadoop 在 2.X 版本开始，已经利用 ZooKeeper 解决了单点故障问题。同样的策略，Spark 也利用 ZooKeeper 解决 Spark 集群的单点故障问题。

集群规划：

Server	Master	Worker
bigdata02	√	√
bigdata03		√
bigdata04	√	√
bigdata05		√

前提：由于Spark集群的高可用依赖于Zookeeper来实现，所以必须要准备一个可用的zookeeper集群！

我们可以安装一个全新的spark的高可用集群，但是也可以在原来的基础之上，安装高可用的集群！保证spark集群关机状态。

```
$SPARK_HOME/sbin/stop-all.sh
```

第一步：从官网下载对应版本的安装包

```
wget https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.6/spark-2.4.6-bin-hadoop2.7.tgz
```

第二步：上传安装包，然后解压缩安装到对应的目录

```
tar -zxvf spark-2.4.6-bin-hadoop2.7.tgz -C /home/bigdata/apps/
```

第三步：修改配置文件spark-env.sh

```
cd ~/apps/spark-2.4.6-bin-hadoop2.7/conf
mv spark-env.sh.template spark-env.sh
vim spark-env.sh
```

修改或添加如下内容：

```
export JAVA_HOME=/usr/local/java/jdk1.8.0_181
export SPARK_MASTER_PORT=7077
export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -
Dspark.deploy.zookeeper.url=bigdata02,bigdata03,bigdata04 -
Dspark.deploy.zookeeper.dir=/spark246"
```

参数解释：

-Dspark.deploy.recoveryMode=ZOOKEEPER

说明整个集群状态是通过zookeeper来维护的，整个集群状态的恢复也是通过zookeeper来维护的。就是说用zookeeper做了Spark的HA配置，Master(Active)挂掉的话，Master(standby)要想变成Master(Active)的话，Master(Standby)就要像zookeeper读取整个集群状态信息，然后进行恢复所有Worker和Driver的状态信息，和所有的Application状态信息

-Dspark.deploy.zookeeper.url=bigdata02,bigdata03,bigdata04

将所有配置了zookeeper，并且在这台机器上有可能做master(Active)的机器都配置进来（我用了3台，就配置了3台）

-Dspark.deploy.zookeeper.dir=/spark246

-Dspark.deploy.zookeeper.dir是保存spark的元数据，保存了spark的作业运行状态；zookeeper会保存spark集群的所有的状态信息，包括所有的workers信息，所有的AppIactions信息，所有的Driver信息

第四步：修改配置文件slave

```
cd ~/apps/spark-2.4.6-bin-hadoop2.7/conf
mv slaves.template slaves
vi slaves
```

添加如下内容：

```
bigdata02
bigdata03
bigdata04
bigdata05
```

第五步：把core-site.xml和hdfs-site.xml文件拷贝过来，放到\$SPARK_HOME/conf目录下

```
cd ~/apps/hadoop-2.7.7/etc/hadoop
cp core-site.xml hdfs-site.xml ~/apps/spark-2.4.6-bin-hadoop2.7/conf
```

第六步：分发安装到所有节点

```
cd ~/apps/
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7 bigdata02:~/apps/
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7 bigdata03:~/apps/
scp -r ~/apps/spark-2.4.6-bin-hadoop2.7 bigdata04:~/apps/
```

第六步：配置环境变量

```
vi ~/.bashrc
```

```
export SPARK_HOME=/home/bigdata/apps/spark-2.4.6-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
scp -r ~/.bashrc bigdata02:~
scp -r ~/.bashrc bigdata03:~
scp -r ~/.bashrc bigdata04:~
```

```
source ~/.bashrc
```

切记：所有节点都要配置。

第七步：启动spark集群

由于spark高可用集群在安装过程中，并没有配置文件指定谁是master，所以其实在哪个节点执行start-all.sh，当前节点就是其中之一的master

```
$SPARK_HOME/sbin/start-all.sh
```

那么另外一个master节点，记住，可以通过命令去到对应的master节点，手动启动

```
$SPARK_HOME/sbin/start-master.sh
```

第八步：验证集群启动是否成功

通过守护进程检查：

```
jps
```



```
bigdata04 [bigdata@bigdata04 ~]$ jps
1921 NodeManager
2531 Master
2468 Worker
2614 Jps
1850 DataNode
2010 ResourceManager
1788 QuorumPeerMain
2268 JournalNode
[bigdata@bigdata04 ~]$

bigdata03 [bigdata@bigdata03 ~]$ jps
1792 QuorumPeerMain
1941 DataNode
2262 JournalNode
1863 NameNode
2807 Jps
2362 DFSZKFailoverController
2717 Worker
2015 NodeManager
[bigdata@bigdata03 ~]$

bigdata05 [bigdata@bigdata05 ~]$ jps
2224 QuorumPeerMain
3360 Worker
2471 NodeManager
2360 DataNode
2297 ResourceManager
3437 Jps
[bigdata@bigdata05 ~]$

bigdata02 [bigdata@bigdata02 ~]$ jps
5603 JournalNode
6307 Jps
5254 NodeManager
5063 NameNode
4921 QuorumPeerMain
5177 DataNode
6233 Worker
5756 DFSZKFailoverController
6143 Master
[bigdata@bigdata02 ~]$
```

通过web界面检查: <http://bigdata02:8080> 或者 <http://bigdata04:8080>

Spark Master at spark://bigdata02:7077

URL: spark://bigdata02:7077

Alive Workers: 4

Cores in use: 4 Total, 0 Used

Memory in use: 4.0 GB Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (4)

Worker Id	Address	State	Cores	Memory
worker-20200404223254-192.168.123.152-35736	192.168.123.152:35736	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20200404223254-192.168.123.153-42243	192.168.123.153:42243	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20200404223254-192.168.123.154-36973	192.168.123.154:36973	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20200404223254-192.168.123.155-43641	192.168.123.155:43641	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Spark Master at spark://bigdata04:7077

URL: spark://bigdata04:7077

Alive Workers: 0

Cores in use: 0 Total, 0 Used

Memory in use: 0.0 B Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: STANDBY

Workers (0)

Worker Id	Address	State	Cores	Memory
-----------	---------	-------	-------	--------

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

通过例子程序来验证:

```
~/apps/spark-2.4.6-bin-hadoop2.7/bin/spark-shell \  
--master spark://bigdata02:7077,bigdata04:7077 \  
--executor-memory 512m \  
--total-executor-cores 1
```



```
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -
Dspark.history.retainedApplications=30 -
Dspark.history.fs.logDirectory=hdfs://hadoop277ha/spark246_history_log"
```

参数解释:

`spark.eventLog.dir=hdfs://hadoop277ha/sparklog`
Application在运行过程中所有的信息均记录在该属性指定的路径下

`spark.history.ui.port=18080`
调整WEBUI访问的端口号为18080

`spark.history.retainedApplications=30`
指定保存Application历史记录个数，如果超过这个值，旧的应用程序信息将被删除，这个是内存中的应用数，而不是页面上显示的应用数

`spark.history.fs.logDirectory=hdfs://hadoop277ha/sparklog`
配置了该属性后，在`start-history-server.sh` 时就无需再显式的指定路径，Spark History Server页面只展示该指定路径下的信息

刚才改了两个配置文件，全局同步：

```
cd /home/bigdata/apps/spark-2.4.6-bin-hadoop2.7/conf
scp -r spark-defaults.conf spark-env.sh bigdata02:$PWD
scp -r spark-defaults.conf spark-env.sh bigdata03:$PWD
scp -r spark-defaults.conf spark-env.sh bigdata04:$PWD
```

第三步：在启动HistoryServer服务之前 `hdfs://hadoop277ha/sparklog` 目录要提前创建

```
hadoop fs -mkdir -p hdfs://hadoop277ha/spark246_history_log
```

第四步：启动spark historyserver

在 bigdata03 上启动 Spark HistoryServer

```
$SPARK_HOME/sbin/start-history-server.sh
```

第五：访问Spark History WebUI: <http://bigdata03:18080/>

第六步：执行一个任务程序

```
$SPARK_HOME/bin/spark-submit \  
--class org.apache.spark.examples.SparkPi \  
--master spark://bigdata02:7077,bigdata04:7077 \  
--executor-memory 512m \  
--total-executor-cores 2 \  
$SPARK_HOME/examples/jars/spark-examples_2.11-2.4.6.jar \  
100
```

执行结果：

```
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Finished task 95.0 in stage 0.0 (TID 95) in 24 ms on 192.168.123.102 (executor 0) (96/100)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Starting task 98.0 in stage 0.0 (TID 98, 192.168.123.105, executor 1, partition 98, PROCESS_LOCAL  
, 7740 bytes)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Finished task 96.0 in stage 0.0 (TID 96) in 30 ms on 192.168.123.105 (executor 1) (97/100)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Starting task 99.0 in stage 0.0 (TID 99, 192.168.123.102, executor 0, partition 99, PROCESS_LOCAL  
, 7740 bytes)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Finished task 97.0 in stage 0.0 (TID 97) in 45 ms on 192.168.123.102 (executor 0) (98/100)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Finished task 98.0 in stage 0.0 (TID 98) in 38 ms on 192.168.123.105 (executor 1) (99/100)  
20/09/04 16:42:41 INFO scheduler.TaskSetManager: Finished task 99.0 in stage 0.0 (TID 99) in 38 ms on 192.168.123.102 (executor 0) (100/100)  
20/09/04 16:42:41 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool  
20/09/04 16:42:41 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 5.080 s  
20/09/04 16:42:41 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 5.402354 s  
Pi is roughly 3.1422215142221512  
20/09/04 16:42:41 INFO server.AbstractConnector: Stopped Spark@22ee2d0{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}  
20/09/04 16:42:41 INFO ui.SparkUI: Stopped Spark web UI at http://bigdata03:4040  
20/09/04 16:42:41 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors  
20/09/04 16:42:41 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down  
20/09/04 16:42:41 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!  
20/09/04 16:42:42 INFO memory.MemoryStore: MemoryStore cleared  
20/09/04 16:42:42 INFO storage.BlockManager: BlockManager stopped  
20/09/04 16:42:42 INFO storage.BlockManagerMaster: BlockManagerMaster stopped  
20/09/04 16:42:42 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!  
20/09/04 16:42:42 INFO spark.SparkContext: Successfully stopped SparkContext  
20/09/04 16:42:42 INFO util.ShutdownHookManager: Shutdown hook called  
20/09/04 16:42:42 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-bc443839-1bf9-4b99-9220-22ad5ebf53d9  
20/09/04 16:42:42 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e19377d4-77b5-4b98-a497-889ff4a5452a  
[bigdata@bigdata03 jars]$
```

Spark Master at spark://bigda...

History Server

← → ↻ ⌂ 🔒 不安全 | bigdata03:18080

spark 2.4.6

History Server

Event log directory: hdfs://hadoop277ha/spark246_history_log

Last updated: 2020-09-04 16:43:59

Client local time zone: Asia/Shanghai

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
app-20200904164233-0001	Spark Pi	2020-09-04 16:42:31	2020-09-04 16:42:41	10 s	bigdata	2020-09-04 16:42:41	Download

Showing 1 to 1 of 1 entries

[Show incomplete applications](#)

