

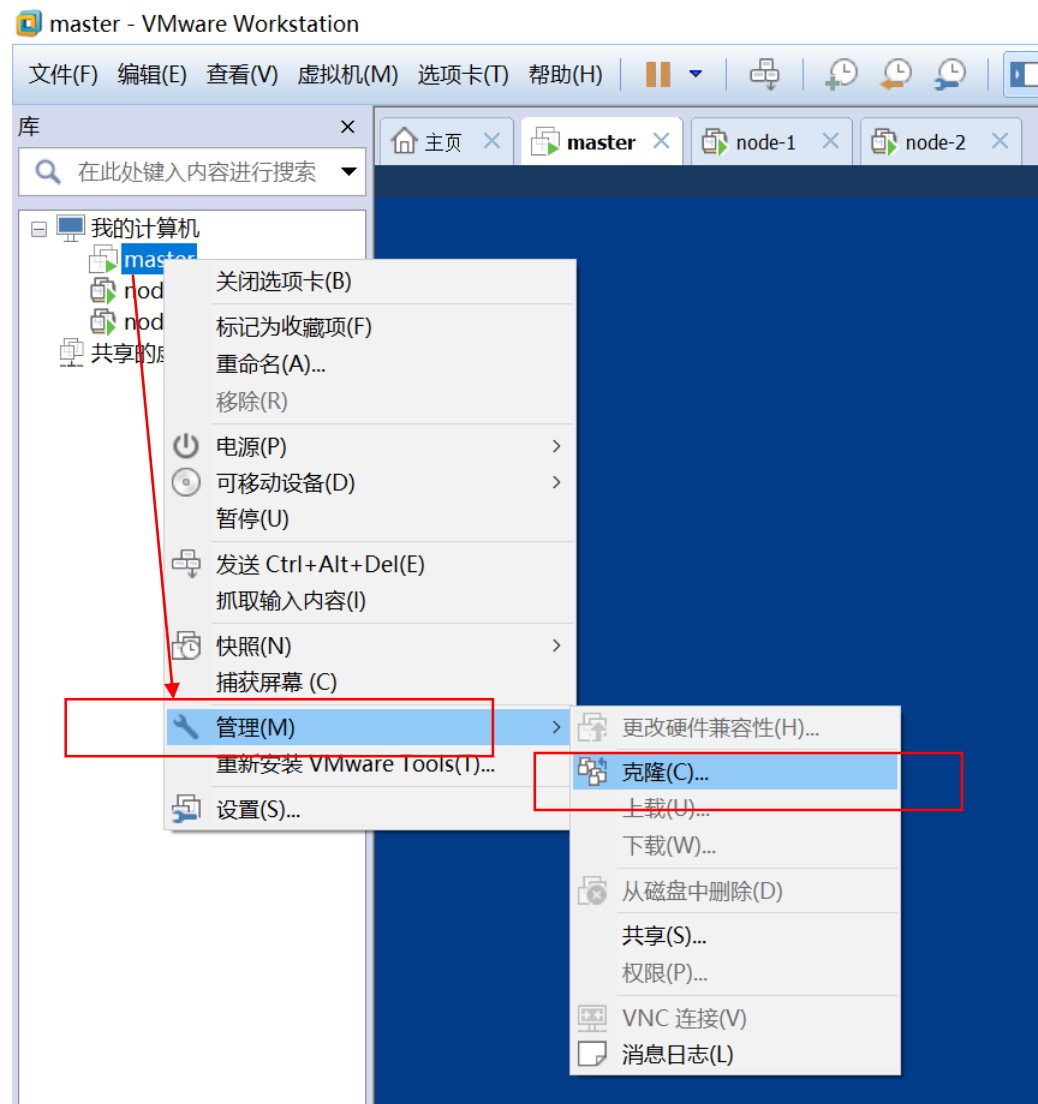
# 搭建开发环境

讲师: watermelon

## 搭建虚拟机开发环境及节点间互信

# 安装虚拟机

安装完master虚拟机后，从master虚拟机clone出另两个slave虚拟机



# 环境配置

机器名称	IP地址
master	192.168.5.134
node-1	192.168.5.135
node-2	192.168.5.136

修改/etc/hosts 文件，该文件是用来配置主机用到的DNS服务器信息，是用来记录局域网内各主机对应的hostname – ip用的。用户在进行网络连接时，首先查找这个文件，寻找主机名对应的ip

```
[root@master ~]# more /etc/hosts
#127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
#::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
#127.0.0.1 localhost.localdomain localhost
192.168.5.134 master
192.168.5.135 node-1
192.168.5.136 node-2
```

配置完成hosts文件后，局域网内的节点可以相互识别，可以从一个节点ping通另一个节点

```
[root@master ~]# ping 192.168.5.135
PING 192.168.5.135 (192.168.5.135) 56(84) bytes of data.
64 bytes from 192.168.5.135: icmp_seq=1 ttl=64 time=0.407 ms
64 bytes from 192.168.5.135: icmp_seq=2 ttl=64 time=0.299 ms
64 bytes from 192.168.5.135: icmp_seq=3 ttl=64 time=0.333 ms
64 bytes from 192.168.5.135: icmp_seq=4 ttl=64 time=0.323 ms
64 bytes from 192.168.5.135: icmp_seq=5 ttl=64 time=0.300 ms
64 bytes from 192.168.5.135: icmp_seq=6 ttl=64 time=0.269 ms
```

# 节点间互信(1)

Hadoop启动后，namenode通过SSH来启动和停止datanode上的各种进程。这里必须在节点间执行指令的时候是免密码登录的形式。而且方便跨节点执行指令的时候，也需要免密钥登录的形式

## 免密钥执行过程

- 1、分别在master和slave上分别输入指令：`ssh-keygen -t rsa`;
- 2、进入路径 `/root/.ssh` 下可以看到两个文件 `id-rsa`和`id_rsa.pub`，其中`id_rsa.pub`是公钥;
- 3、为了方便识别不同主机上的密钥名称，这里对master上的`id_rsa.pub`命名为：`authorized_keys_master.pub`，node-1上的命名为：`authorized_keys_node1.pub`，node-2上的命名为：`authorized_keys_node1.pub`;
- 4、把node-1和node-2上的公钥用`scp`命令传到master上;
- 5、把master和node-1、node-2的公钥信息保存到`authorized_keys`文件中，然后再把`authorized_keys`发送到node-1和node-2节点

# 节点间互信(2)

在master和node节点上输入指令

```
[root@master ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

公钥重命名: `mv id_rsa.pub authorized_keys_node1.pub`

```
[root@node-1 ~]# cd /root/.ssh
[root@node-1 .ssh]# ll
总用量 20
-rw-r--r--. 1 root root 1179 8月 20 00:24 authorized_keys
-rw-r--r--. 1 root root 408 7月 27 20:49 authorized_keys_node1.pub
-rw-----. 1 root root 1675 8月 20 00:17 id_rsa
-rw-r--r--. 1 root root 393 8月 20 00:17 id_rsa.pub
-rw-r--r--. 1 root root 353 7月 27 20:22 known_hosts
```

发送公钥到master节点: `scp -r /root/.ssh/ authorized_keys_node1.pub root@master:/root/.ssh/`

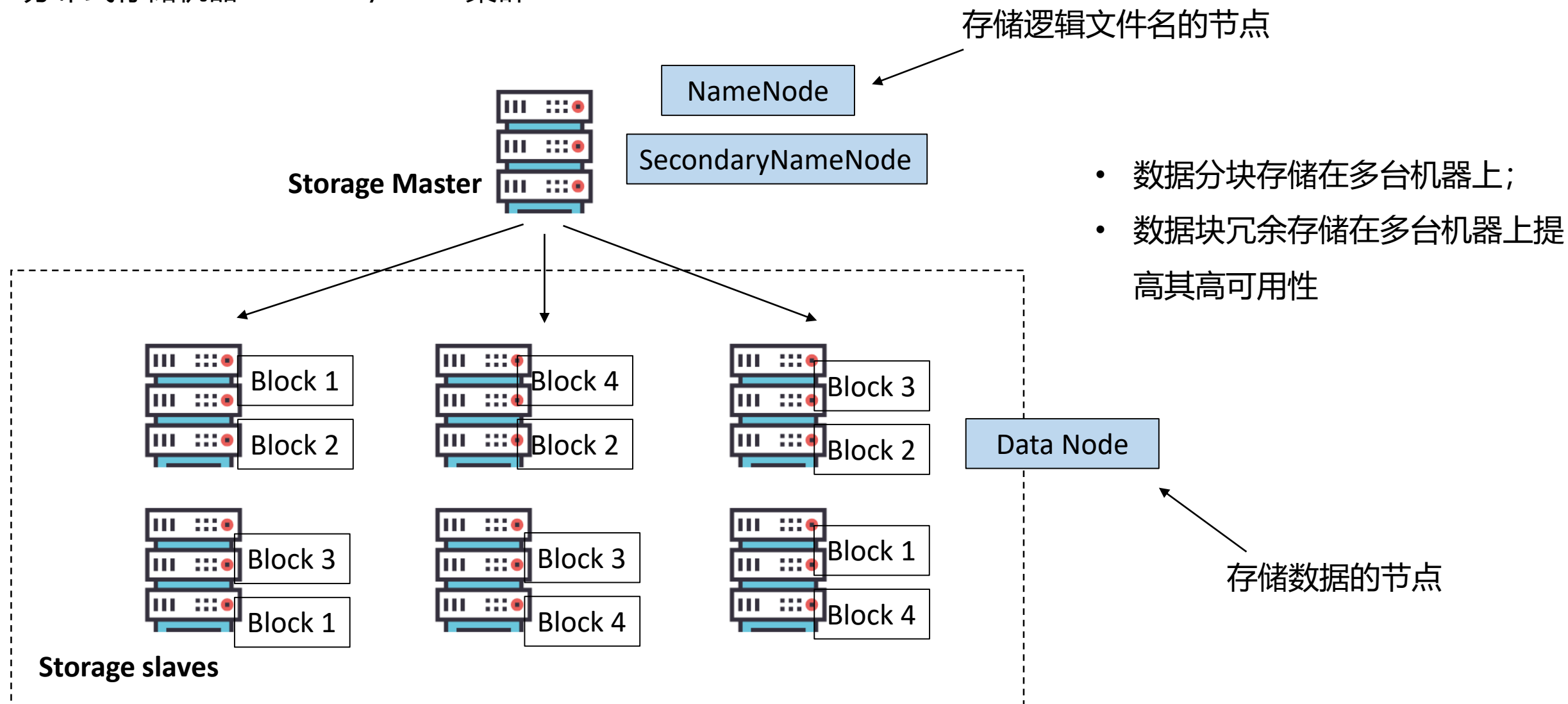
合并到authorized\_keys文件中: `cat authorized_keys_node1.pub >> authorized_keys`

```
[root@master ~]# cd /root/.ssh
[root@master .ssh]# ll
总用量 28
-rw-r--r--. 1 root root 1179 8月 20 00:21 authorized_keys
-rw-r--r--. 1 root root 408 8月 3 14:17 authorized_keys_master.pub
-rw-r--r--. 1 root root 393 8月 20 00:19 authorized_keys_node1.pub
-rw-r--r--. 1 root root 393 8月 20 00:20 authorized_keys_node2.pub
-rw-----. 1 root root 1675 8月 20 00:16 id_rsa
-rw-r--r--. 1 root root 393 8月 20 00:16 id_rsa.pub
-rw-r--r--. 1 root root 353 7月 27 20:22 known_hosts
```

## HDFS的安装及应用

# 分布式存储(HDFS)-结构

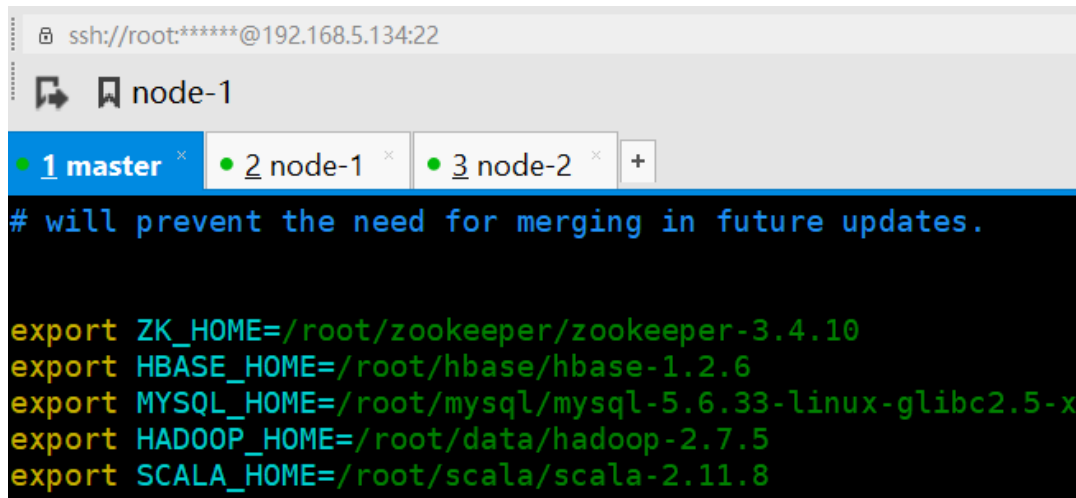
分布式存储机器：master/slave 集群



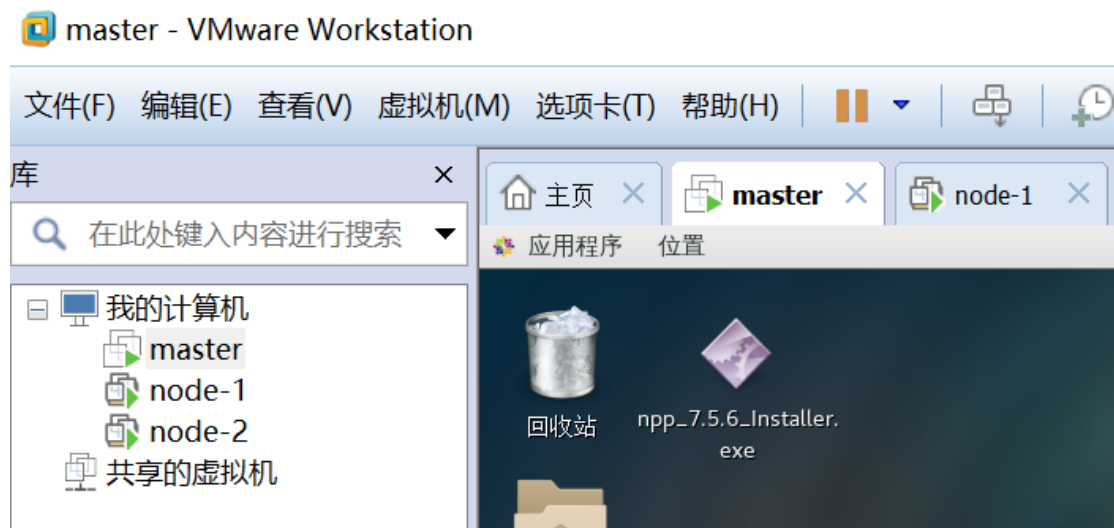


# 分布式存储(HDFS)-进入虚拟机

通过xshell登录搭建的虚拟机环境



```
ssh://root:*****@192.168.5.134:22
node-1
1 master x 2 node-1 x 3 node-2 x +
# will prevent the need for merging in future updates.
export ZK_HOME=/root/zookeeper/zookeeper-3.4.10
export HBASE_HOME=/root/hbase/hbase-1.2.6
export MYSQL_HOME=/root/mysql/mysql-5.6.33-linux-glibc2.5-x
export HADOOP_HOME=/root/data/hadoop-2.7.5
export SCALA_HOME=/root/scala/scala-2.11.8
```



配置的节点信息:

master: 192.168.5.134

node-1: 192.168.5.135

node-2: 192.168.5.136

# 分布式存储(HDFS)-安装和配置

下载Hadoop <http://hadoop.apache.org/releases.html>

需要修改的配置文件内容, 请查看“**开发环境搭建.sh**”文件

解压文件: `tar -zxvf hadoop-2.7.5.tar.gz`

修改配置文件: 都在etc目录下面,, 要修改core-site.xml、hdfs-site.xml、hadoop-env.sh、slaves

修改环境变量: `vim /etc/profile`

使修改后环境变量生效: `source /etc/profile`

同步master的文件到slave节点: `scp -r /root/hdfsV2.0 root@node-1:/root/`

格式化: `hdfs namenode -formats`

启动start-dfs.sh

启动stop-dfs.sh

# 分布式存储(HDFS)-查看是否启动进程

在master和slave节点分别查看启动的进程

```
[root@master ~]# jps
22385 NameNode
27553 Master
37922 QuorumPeerMain
38182 Kafka
27255 ResourceManager
38760 ConsoleConsumer
22569 SecondaryNameNode
44077 Jps
```

master节点的NameNode和  
SecondaryNameNode已经启动

```
[root@node-1 ~]# jps
19669 Jps
6312 DataNode
10472 Worker
10298 NodeManager
```

```
[root@node-2 ~]# jps
10402 Worker
19956 Jps
6281 DataNode
10236 NodeManager
```

slave节点的DataNode已经启动

# 分布式存储(HDFS)-查看是否配置正确

输入HDFS默认监控地址: <http://master:50070>

查看节点信息

Hadoop

Overview

Datanodes

Datanode Volume Failures

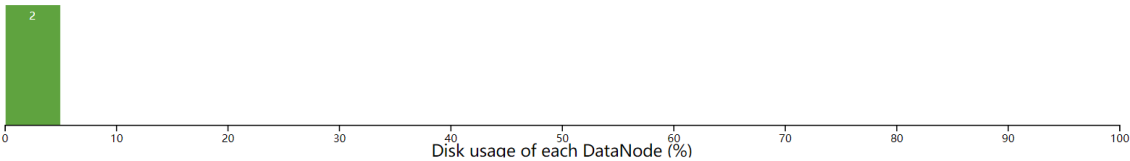
Snapshot

Startup Progress

Utilities

## Datanode Information

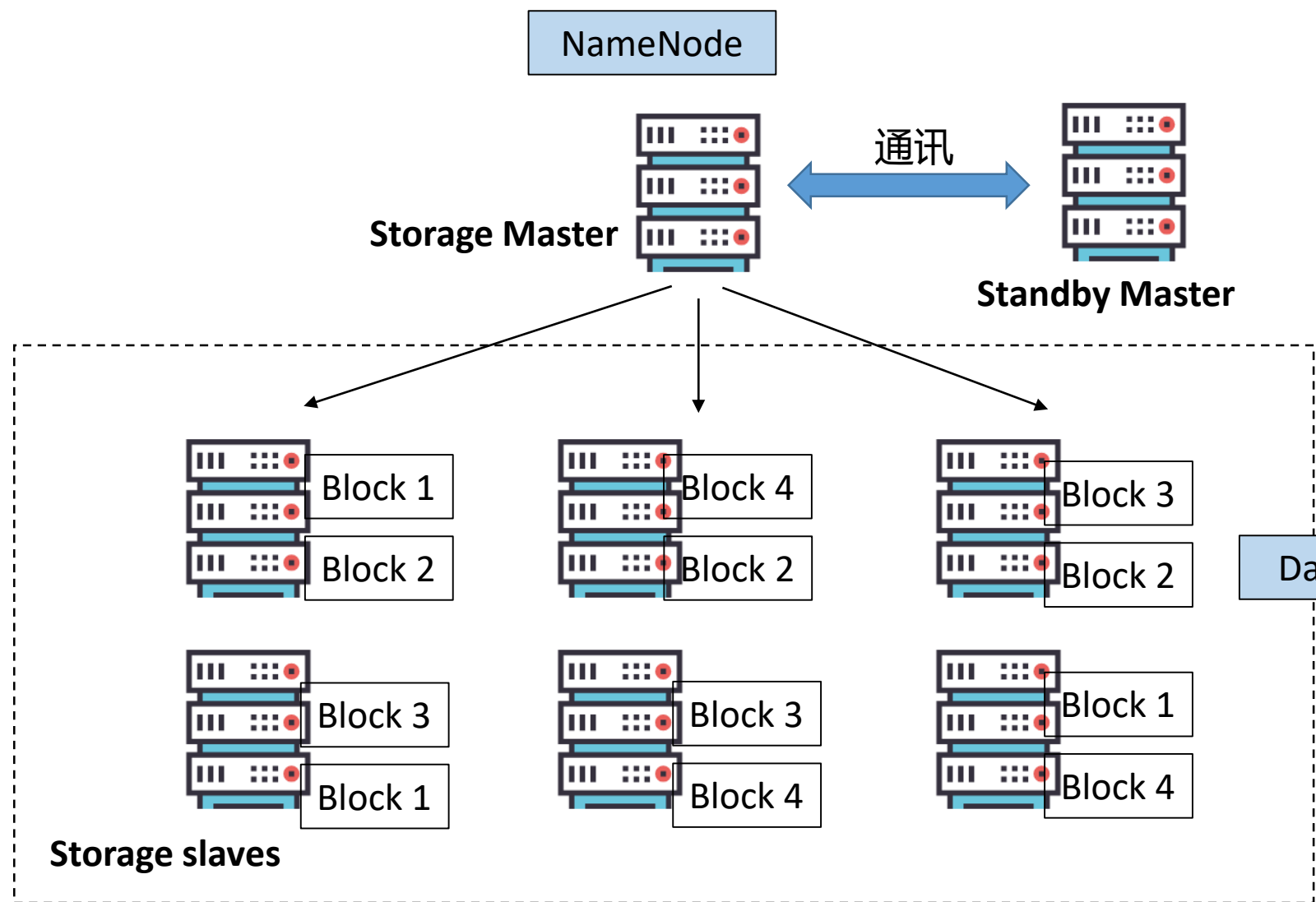
Datanode usage histogram



Configured Capacity:	33.97 GB
DFS Used:	16 KB (0%)
Non DFS Used:	15.99 GB
DFS Remaining:	17.98 GB (52.93%)
Block Pool Used:	16 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)

## zookeeper的安装及应用

# zookeeper作用



## zk实现了去中心化的集群模式

standby master 和storage master实时通讯, 把storage master数据同步过来, 如果storage master挂掉了, 那么standby master会启动, 利用这种机制, 保证集群的高可用性

Hbase中regionserver通过zookeeper来协调;

Kafka使用zookeeper作为其分布式协调框架

# 下载和安装

下载地址: <https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/stable/>

解压: `tar -zxvf zookeeper-3.4.10.tar.gz`

进入目录: `cd /root/zookeeper/zookeeper-3.4.10/conf`

`cp zoo_sample.cfg zoo.cfg`

修改zoo.cfg文件:

配置环境变量: `vim /etc/profile`

启动: `zkServer.sh start`

关闭: `zkServer.sh stop`

查看状态: `zkServer.sh status`

**详细配置过程见本章的“开发环境搭建.sh”**

# 查看zookeeper是否启动

jps查看当前进程

```
[root@node-1 data]# jps
28839 QuorumPeerMain
6312 DataNode
10472 Worker
10298 NodeManager
28860 Jps
```

Zookeeper进程启动了

zkServer.sh status命令可查看状况，会有一个leader，两个follower

本节内容讲了zookeeper的安装搭建，为后面讲Spark Streaming开发流式计算标签中搭建kafka环境做了准备。Kafka将元数据信息（topic，partition信息等）保存在Zookeeper中，以及存储消费的offset信息



## Spark的安装及应用

# 下载和安装

下载地址 <http://spark.apache.org/downloads.html>

解压: `tar -zxvf spark-2.3.0-bin-hadoop2.7.tg`

配置slaves

配置spark-env.sh

Master配置完成后分发到slave节点

配置环境变量

*详细配置过程见本章的“开发环境搭建.sh”*

# 查看Spark是否启动成功

```
[root@master sbin]# jps
54000 QuorumPeerMain
22385 NameNode
27553 Master
57793 Master
57859 Jps
27255 ResourceManager
22569 SecondaryNameNode
```


master节点的master进程启动

```
[root@node-1 ~]# jps
29072 QuorumPeerMain
32966 Jps
6312 DataNode
10472 Worker
10298 NodeManager
```

```
[root@node-2 ~]# jps
10402 Worker
31698 Jps
6281 DataNode
10236 NodeManager
```

slave节点的worker已经启动

web端打开spark监控窗口 <http://master:8080/>

 **Spark Master at spark://master:7077**

URL: spark://master:7077  
REST URL: spark://master:6066 (cluster mode)  
Alive Workers: 3  
Cores in use: 3 Total, 0 Used  
Memory in use: 3.0 GB Total, 0.0 B Used  
Applications: 0 Running, 4 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

**Workers (3)**

Worker Id	Address	State	Cores	Memory
<a href="#">worker-20180904185510-192.168.5.135-45207</a>	192.168.5.135:45207	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
<a href="#">worker-20180916154921-192.168.5.136-46136</a>	192.168.5.136:46136	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
<a href="#">worker-20180916154924-192.168.5.135-46345</a>	192.168.5.135:46345	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 (4)**

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

## MySQL的安装及应用

# MySQL的安装

下载地址 <https://dev.mysql.com/downloads/mysql/>

解压 `tar -xvzf mysql-5.6.33-linux-glibc2.5-x86_64.tar.gz`

配置 `my.cnf`

启动服务 `service mysqld start`

`service mysqld stop`

`service mysqld status`

启动MySQL: `cd /root/mysql/mysql-5.6.33-linux-glibc2.5-x86_64/bin`

`./mysql -u root`

# MySQL在画像中应用

- 搭建Hive环境时，作为hive的元数据库；
- 存储画像标签相关的元数据；
- 标签数据波动监控的预警（hive、hbase）

## Hive的安装及应用

Hive是利用SQL语法结构对存储在HDFS中的大数据集进行读、写管理

## Hive安装方式

内嵌模式：元数据保存在内嵌的derby中

本地模式：本地安装mysql代替derby存储元数据



远程模式：远程安装mysql代替derby存储元数据



# Hive的安装及应用

下载地址 <https://mirrors.tuna.tsinghua.edu.cn/apache/hive/stable-2/>

解压安装包 `tar -zxvf apache-hive-2.3.3-bin.tar.gz`

在conf目录下新建配置文件hive-site.xml

上传驱动包

MySQL作为元数据管理,初始化配置

## Hbase的安装及应用

# Hbase的安装

下载地址 <http://mirrors.shu.edu.cn/apache/hbase>

解压安装包 `tar -zxvf apache-hive-2.3.3-bin.tar.gz`

在conf目录下进行配置 `hbase-site.xml`、`hbase-env.sh`、`regionservers`

将master配置拷贝到slave

配置hbase环境变量

启动hbase：先需要启动hdfs和zookeeper： `start-dfs.sh` 、 `zkServer.sh start`  
`start-hbase.sh`

访问hbase的web监控 <http://master:16010>

# Hbase是否启动

```
[root@master conf]# jps
54000 QuorumPeerMain
22385 NameNode
66951 Jps
27255 ResourceManager
22569 SecondaryNameNode
66847 HMaster
```

先将zookeeper启动

先将hdfs启动

master节点的HMaster进程启动

```
[root@node-1 hbase]# jps
29072 QuorumPeerMain
6312 DataNode
39609 HRegionServer
10298 NodeManager
39661 Jps
```

```
[root@node-2 ~]# jps
37315 Jps
36087 QuorumPeerMain
6281 DataNode
10236 NodeManager
37246 HRegionServer
```

slave节点的HRegionServer已经启动

访问hbase的web监控 <http://master:16010>

## sqoop的安装及应用

# sqoop的安装及应用

下载地址： <http://mirrors.hust.edu.cn/apache/sqoop/1.4.7>

解压： `tar -zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz`

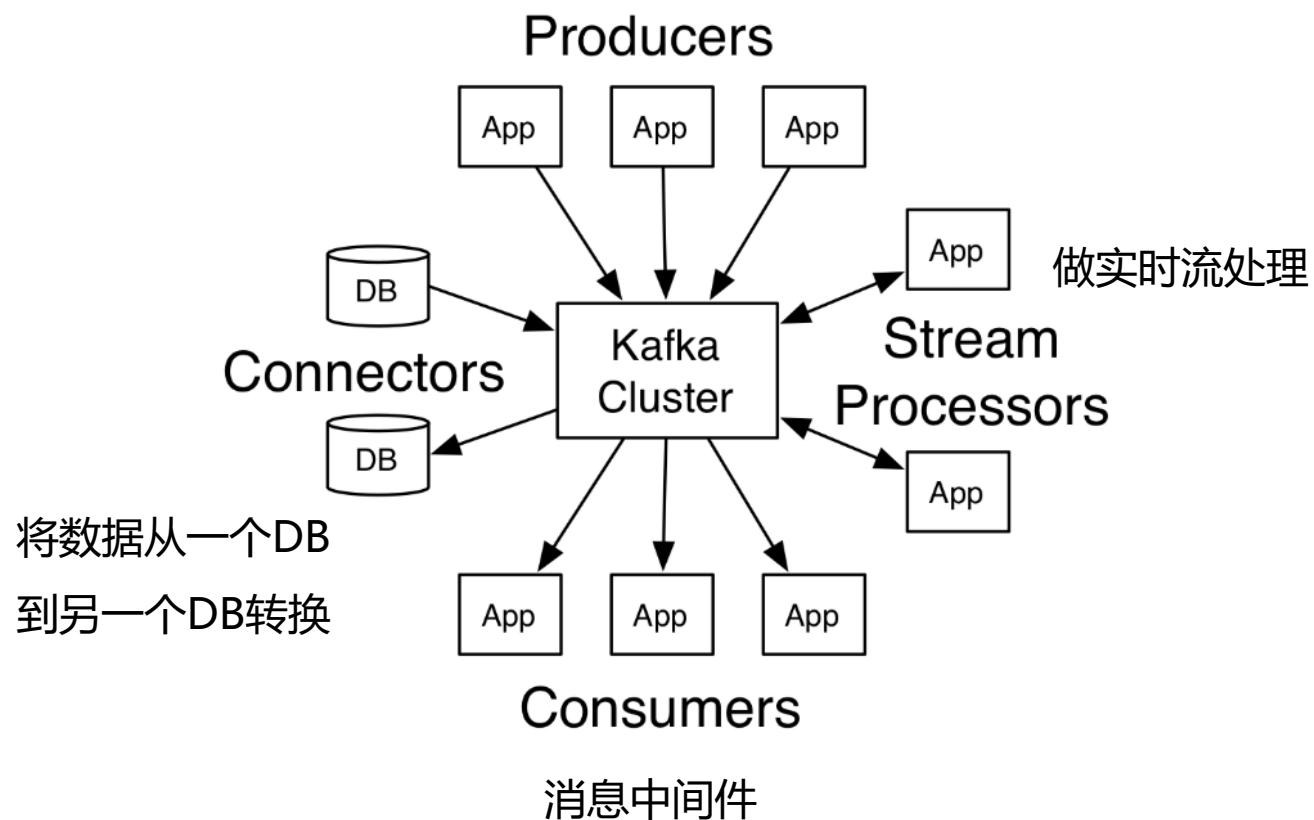
将 `mysql-connector-java-5.1.44-bin.jar` 包放到 `/root/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0/lib`

查看sqoop是否安装成功： `sqoop help`

Sqoop是一个用来将Hadoop和关系型数据库中的数据相互转移的工具。后面章节讲如何将hive中的标签同步到业务系统中会用到

## kafka的安装及应用

# Kafka基本介绍

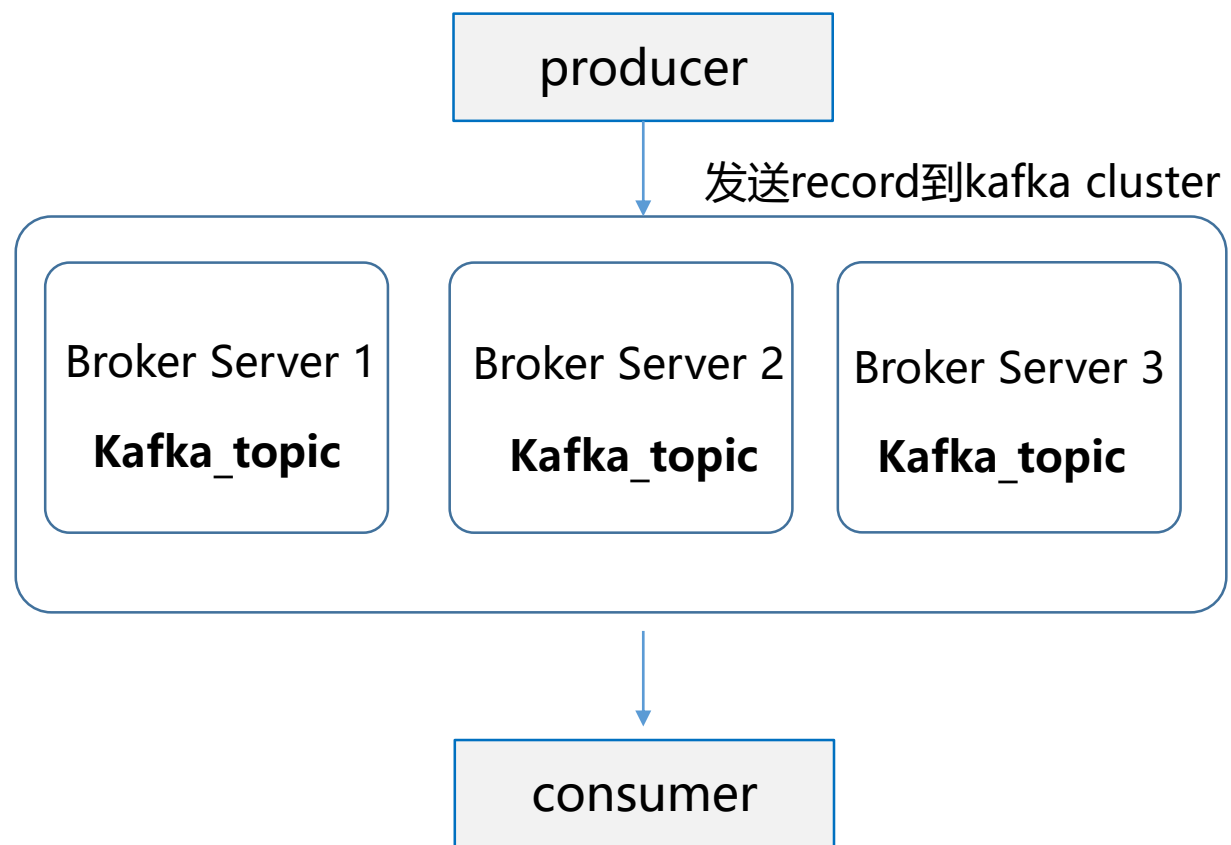


- producer: 产生信息的主体;
- consumer: 消费producer产生信息的主体;
- broker: 消息处理节点, 多个broker组成kafka集群;
- topic: 是数据主题, 是数据记录发布的地方,可以用来区分业务系统;
- partition: 是topic的分组, 每个partition都是一个有序队列
- offset: 用于定位消费者在每个partition中消费到的位置

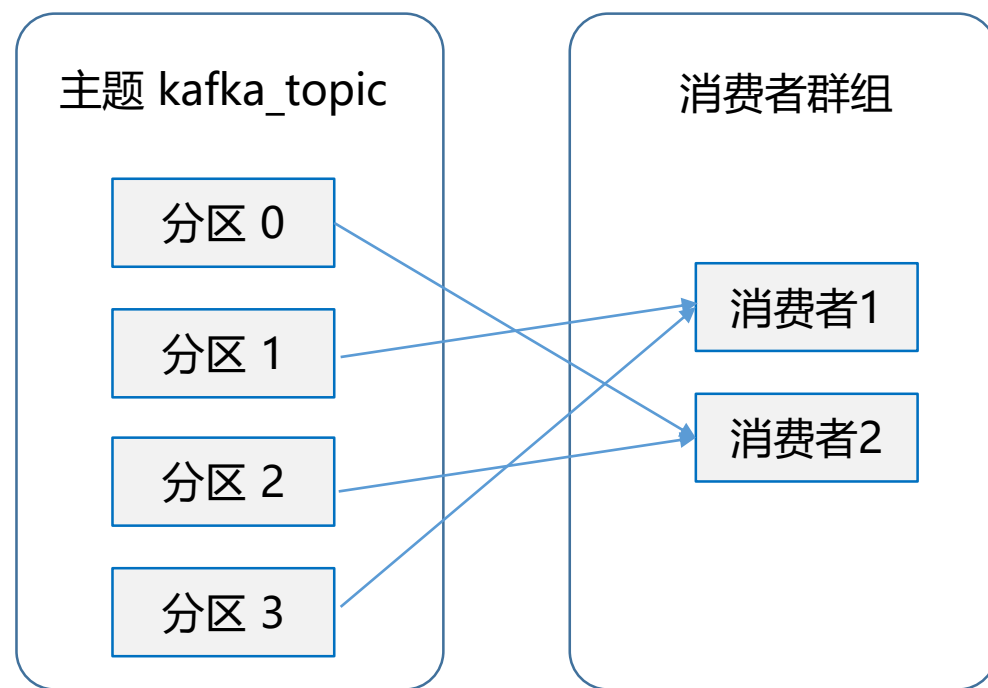
在本套课程中, spark streaming消费kafka数据, 创建实时类标签



# Kafka基本介绍



搭建kafka的时候需要在 server.properties中设置broker.id



Kafka中消费者从属于消费者群组，一个群组里的消费者消费同一个topic，每个消费者接受topic的一部分数据

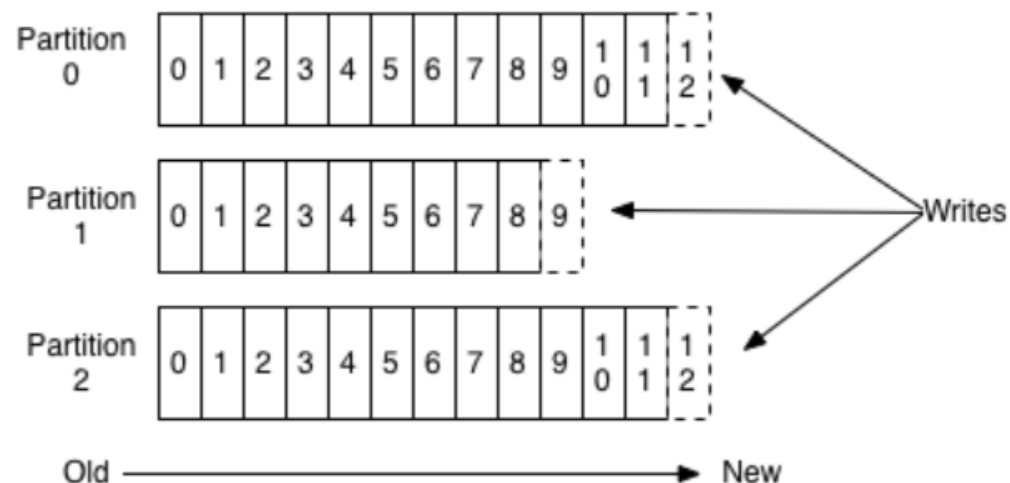
# 从kafka消费数据

```
·//·messages·从kafka获取数据,将数据转为RDD  
messages.foreachRDD((rdd,·batchTime)·=>·{·....  
··import·org.apache.spark.streaming.kafka.HasOffsetRanges  
··val·offsetRanges·=·rdd.asInstanceOf[HasOffsetRanges].offsetRanges·//·获取偏移量信息  
··offsetRanges.foreach(offset·=>·println(offset.topic,·offset.partition,·offset.fromOffset,·  
·offset.untilOffset))  
··}  
·}
```

消费的topic、 partition、 fromoffset、 untiloffset

18/09/19 11:12:40 INFO JobSchedule

```
(countly_event,2,388373,388375)  
(countly_event,0,388405,388407)  
(countly_event,7,414706,414707)  
(countly_event,8,388404,388406)  
(countly_event,6,388368,388370)  
(countly_event,10,388371,388373)  
(countly_event,3,414706,414708)  
(countly_event,11,414710,414712)  
(countly_event,9,388408,388410)  
(countly_event,1,388409,388410)  
(countly_event,5,388408,388409)  
(countly_event,4,388376,388377)
```



# Kafka安装

下载地址 <https://www.apache.org/dyn/closer.cgi?path=/kafka/1.0.0/>

解压安装包 `tar -zxvf kafka_2.11-1.0.0.tgz`

在conf目录下进行配置 `server.properties`

创建对应的log日志文件夹

将master配置拷贝到slave

在slave节点配置broker.id

启动kafka：先需要启动zookeeper： `start-dfs.sh` 、 `zkServer.sh start`

*详细配置过程见本章的 “开发环境搭建.sh”*