**大数据开发环境配置**

# 平台及软件版本：

1. 系统平台：centos7，内核版本：3.10.0-693.el7.x86\_64(docker 要求最低3.10)
2. Docker：用yum安装，自动选择最新版本
3. PyCharm：2018.3.4，Python环境使用Anaconda3
4. Intellij Idea:2018.3.4, JDK1.8
5. Scrapy：1.5.0。用pip安装，自动选择版本
6. MySql：5.7
7. Xshell\_wm
8. WinSCP
9. Hadoop：hadoop-2.7.7.tar.gz
10. Spark：spark-2.4.0-bin-hadoop2.7.tgz

# 搭建环境：

## 1 Centos 7：

安装VMWare WorkStation，使用镜像centos 7 安装centos系统

## 2 安装docker:

yum install docker

## 3 安装PyCharm和Intellij Idea

PyCharm使用Anaconda3的Python3环境

Intellij idea使用JDK 1.8

## 4 Scrapy

pip install scrapy

## 5 Mysql

使用阿里云端的Mysql

## 6 安装hadoop

关闭防火墙

systemctl stop firewalld

安装ssh：

yum install openssh\*

安装ssh的目的是：避免登录本机需要密码。只有单机部署hadoop，但也要把它理解成集群中只有一个机器

启动ssh

systemctl enable sshd

连接本机

ssh localhost

直接连接提示：22端口不可连接

安装（firewall-cmd 是 firewalld的字符界面管理工具）：

yum install firewalld firewall-config

systemctl start firewalld

永久开启22端口

firewall-cmd --zone=public --add-port=22/tcp --permanent

重启

service firewalld restart

重启ssh

service sshd restart

执行

ssh localhost

passwd root

提示输入密码，ssh 正常

生成key

cd ~/

ssh-keygen -t rsa -P '' -f ~/.ssh/id\_dsa

cd .ssh

cat id\_dsa.pub >> authorized\_keys

查看java版本

yum list installed |grep java

什么都没有，表示没有java

安装1.8（因为有些组件需要1.8）

yum -y install java-1.8.0-openjdk\*

找到java路径，设置环境变量 ll 是 LL

whereis javac

ll /usr/bin/javac

ll /etc/alternatives/javac

ll /usr/lib/jvm/java-1.7.0-openjdk.x86\_64/bin/javac：到这一步已经没有软连接了。因此，java\_home应该下面这个目录：

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.171-7.b10.el7.x86\_64

修改环境变量：

Bashrc：各个用户自己有一份

Profile：所有的用户共享

vi ~/.bashrc

添加：

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.171-7.b10.el7.x86\_64

source ~/.bashrc（使修改生效）

检验：

echo $JAVA\_HOME

java -version

$JAVA\_HOME/bin/java -version

现在开始安装hadoop

解压:

tar -zvxf hadoop-2.7.1.tar.gz

重命名：

mv ./hadoop-2.7.1/ ./hadoop

移动到local下：

mv ./hadoop /usr/local/

注意：以下内容是搭建hadoop为伪分布式模式

Hadoop 可以在单节点上以伪分布式的方式运行，Hadoop 进程以分离的 Java 进程来运行，节点既作为 NameNode 也作为 DataNode，同时，读取的是 HDFS 中的文件

vi **core-site.xml**

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>file:/usr/local/hadoop/tmp</value>

<description>Abase for other temporary directories.</description>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

**vi**hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop/tmp/dfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop/tmp/dfs/data</value>

</property>

</configuration>

**Hadoop配置文件说明**

Hadoop 的运行方式是由配置文件决定的（运行 Hadoop 时会读取配置文件），因此如果需要从伪分布式模式切换回非分布式模式，需要删除 core-site.xml 中的配置项。

此外，伪分布式虽然只需要配置 fs.defaultFS 和 dfs.replication 就可以运行（官方教程如此），不过若没有配置 hadoop.tmp.dir 参数，则默认使用的临时目录为 /tmp/hadoo-hadoop，而这个目录在重启时有可能被系统清理掉，导致必须重新执行 format 才行。所以需要进行设置，同时也指定 dfs.namenode.name.dir 和 dfs.datanode.data.dir，否则在接下来的步骤中可能会出错。

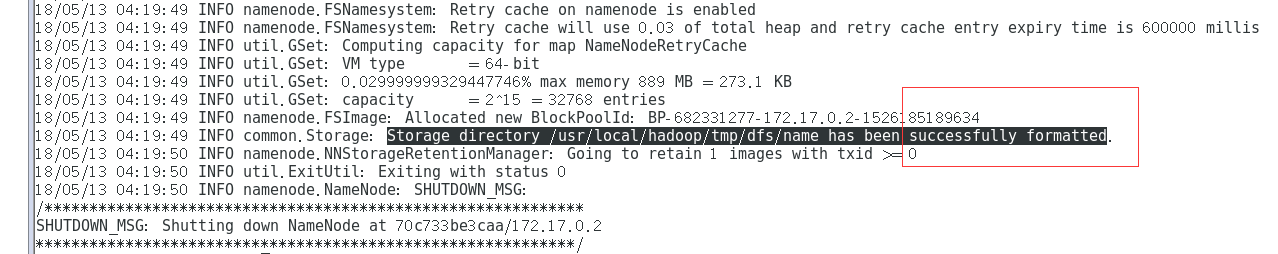
配置文件中的目录启动hdfs会自动创建

添加以下内容到~/.bashrc：

export HADOOP\_HOME=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_HOME/bin:$JAVA\_HOME/bin

运行格式化命令：

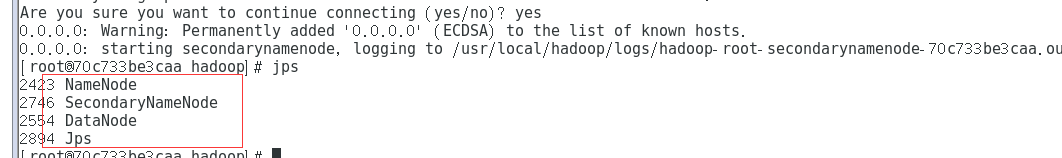
hadoop namenode -format

出现这个表示正常

启动hdfs

./sbin/start-dfs.sh

Jps



出现这几个进程，则表示hdfs正常

运行示例，验证hadoop是否正常

在hdfs文件系统根下创建input目录

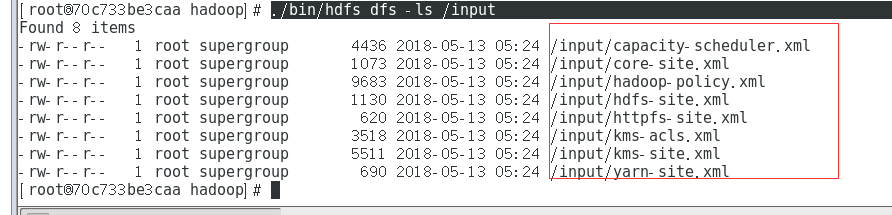
./bin/hdfs dfs -mkdir -p /input

上传文件到hdfs。用put命令

./bin/hdfs dfs -put ./etc/hadoop/\*.xml /input

验证是否上传了：

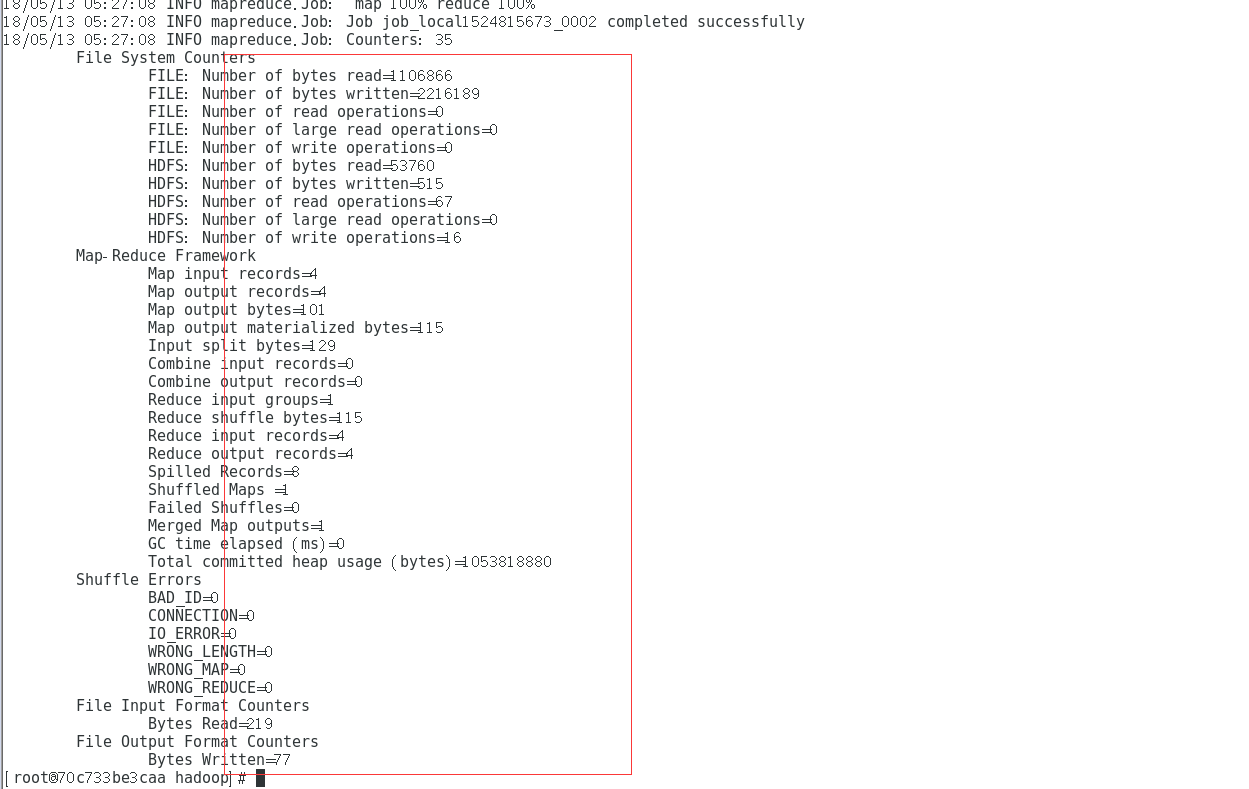
./bin/hdfs dfs -ls /input



表示正常

运行jar包，验证hadoop最终是否正常

./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar grep /input /output 'dfs[a-z.]+'



能看到如下界面，恭喜你，成功了。

注意，现阶段建议不要用hadoop3.0及以上。原因，问题太多了，新手不建议用，并且，生态中的组件，版本还没跟上去。

查看运行结果的命令（查看的是位于 HDFS 中的输出结果）

./bin/hdfs dfs -cat /output/\*

将运行结果取回到本地

删除本地目录

rm -r ./output

从hdfs取结果

./bin/hdfs dfs -get /output ./output

查看本地结果

cat ./output/\*

配置yarn（其实可以不用配置，因为hdfs并不一定非要用yarn。考虑到后面我们要用spark，做真正集群需要进行资源管控，先在这里启用yarn）

**vi** mapred-site.xml

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

**vi** yarn-site.xml

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

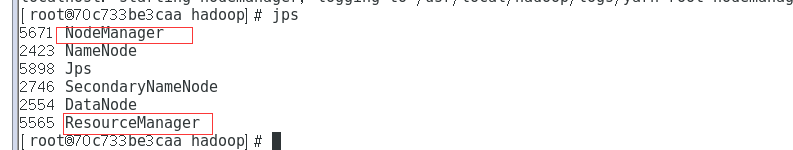
</configuration>

这里有个非常重要的玩意：shuffle 我们后面再解释

启动yarn

**./sbin/start-yarn.sh**

**jps**



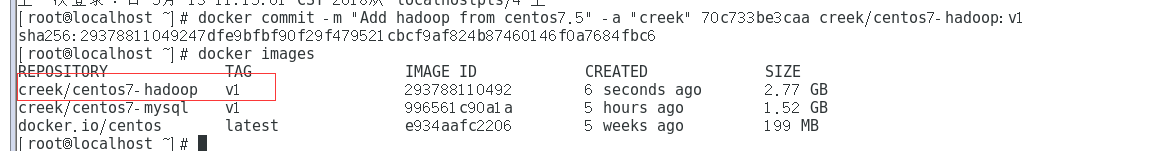
Yarn就是启动红框两个进程

开启历史服务器，才能在Web中查看任务运行情况（不用这个也行）

./sbin/mr-jobhistory-daemon.sh start historyserver

到此，我们再创建一个镜像。这个是在之前的容器上面加了一层。目的就是：备份。

docker commit -m "Add hadoop from centos7.5" -a "creek" 70c733be3caa creek/centos7-hadoop:v1



整个hadoop正常的话，原则上可以看到如下的界面。然而，这里你看不到。原因是docker里面没装浏览器，并且docker里面的端口也没映射到宿主。

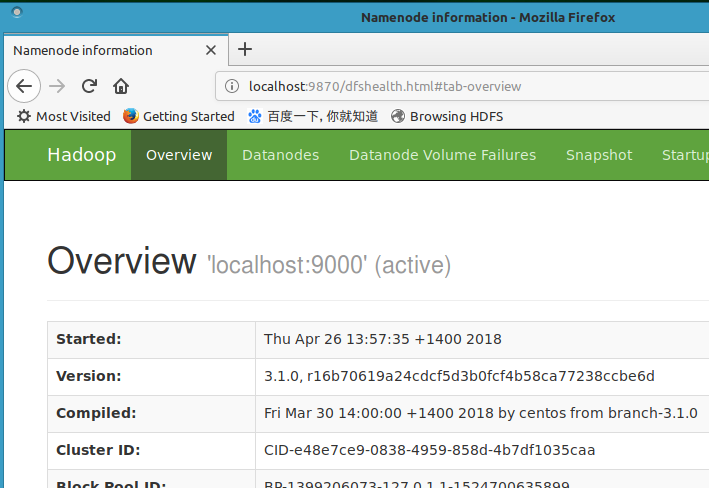
需要停止hdfs和yarn，使用：

./sbin/stop-dfs.sh

./sbin/stop-yarn.sh

若是突然不想使用yarn了，把mapred-site.xml重名了即可

这里就不用完全分布式了。原因很简单：没得个几百台机器，没几百PB数据没必要。数据量越大hadoop越有优势，数据少还到处分布，计算移来移去反倒浪费资源。基于伪分布式搞全分布式也有配个slaves+master而已，3.0以后slaves改名为works



## 7 安装spark

这里，spark并一定非要用hadoop。但是一般情况要用。因为spark没有存储，况且，目前业界spark最好的存储就是hadoop的hdfs，同时，spark一般情况也用yarn来调度，当然也有mesos。

tar -zxf spark-2.4.0-bin-hadoop2.7.tgz -C /usr/local/

cd /usr/local

mv ./ spark-2.4.0-bin-hadoop2.7 ./spark

设置hadoop路径

cd /usr/local/spark

cp ./conf/spark-env.sh.template ./conf/spark-env.sh

vi ./conf/spark-env.sh

在文件的顶部添加：

export SPARK\_DIST\_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)

有这个spark就可以读写hdfs了

添加spark环境变量

vi ~/.bashrc

添加：

export SPARK\_HOME=/usr/local/spark

export PYTHONPATH=$SPARK\_HOME/python:$SPARK\_HOME/python/lib/py4j-0.10.4-src.zip:$PYTHONPATH

export PYSPARK\_PYTHON=python3

export PATH=$HADOOP\_HOME/bin:$SPARK\_HOME/bin:$PATH

验证安装

cd /usr/local/spark

bin/run-example SparkPi 2>&1 | grep "Pi is"



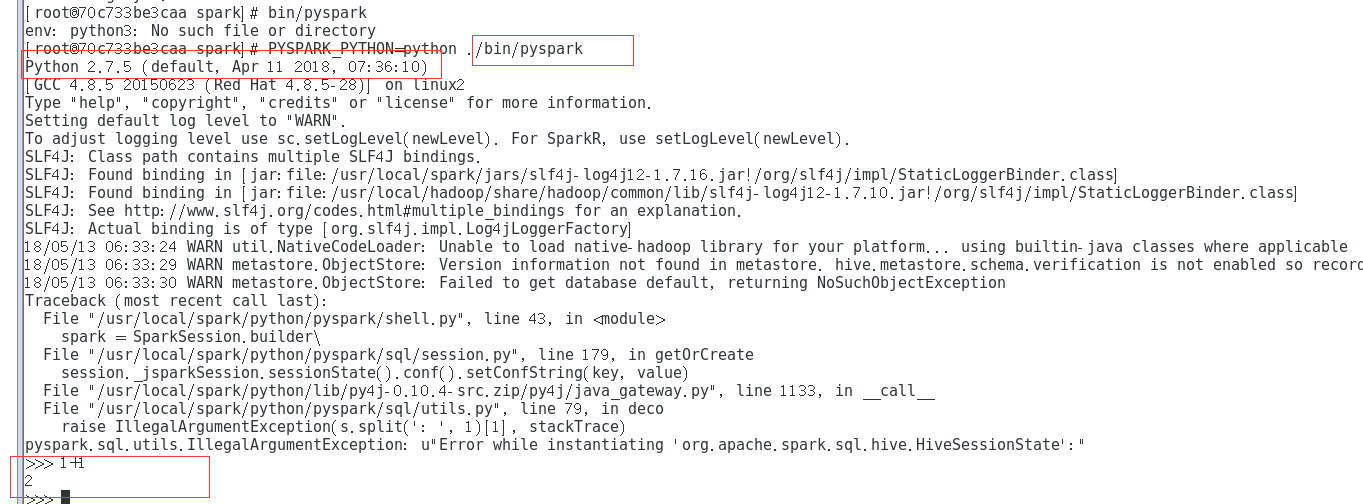
出现计算结果，表示正常

其实系统里面没有python3，只有python

所以这里在执行pyspark的时候，手动指定PYSPARK\_PYTHON。环境变量里面可以去掉。

暂时不去，后面有用

PYSPARK\_PYTHON=python ./bin/pyspark



./bin/pyspark --master <master-url>

这里可以指定spark的运行模式，解释如下：

Spark的运行模式取决于传递给SparkContext的Master URL的值。Master URL可以是以下任一种形式：

\* local 使用一个Worker线程本地化运行SPARK(完全不并行)

\* local[\*] 使用逻辑CPU个数数量的线程来本地化运行Spark

\* local[K] 使用K个Worker线程本地化运行Spark（理想情况下，K应该根据运行机器的CPU核数设定）

\* spark://HOST:PORT 连接到指定的Spark standalone master。默认端口是7077.

\* yarn-client 以客户端模式连接YARN集群。集群的位置可以在HADOOP\_CONF\_DIR 环境变量中找到。

\* yarn-cluster 以集群模式连接YARN集群。集群的位置可以在HADOOP\_CONF\_DIR 环境变量中找到。

\* mesos://HOST:PORT 连接到指定的Mesos集群。默认接口是5050。