## 未标题-2

**JIU JIANG UNIVERSITY**

**《Hadoop实训课》**

院 系 信息科学与技术学院

专 业 计算机科学与技术

班 级 信息A1613

姓 名 施 诚

教 师 张 海

# Hadoop基础

IP：192.168.100.x MAC：255.255.255.0 默认网关：2

Linux命令

cat 查看文件内容 cat /etc 其中斜杠表示根目录，可以输入一部分，按tab键补全

onboot=no -> yes  
boootproto=dhcp -> static  
IPADDR=192.168.100.30  
GATEWAY=192.168.100.2  
DNS1=8.8.8.8

vi 编辑文件 vi /etc/sysconfig/network 配置主机名 a或i 进入编辑模式 esc 退出编辑模式 :wq 保存并退出 :quit 直接退出

service network restart 重启网卡配置 ping ...

SecureCRT.exe 修改主机名 cat /etc/sysconfig/network vi /etc/sysconfig/network

修改主机名与ip地址的映射

cat /etc/hosts

关闭防火墙 service iptables stop //临时关闭防火墙 chk config iptables off && setenforce 0 永久关闭

安装lrzsz（上传下载服务插件），虚拟机可以连上外网 yum install -y lrzsz 不交互

安装时间同步器 yum -y install ntpdate ntpdate -u ntp.api.bz date "+%Y-%m-%d %H:%M:%S"

添加一个自定义用户 add user 姓名的拼音 passwd 姓名的拼音 123456

注意：不要选中安装后自动启动虚拟机

192.168.46.134 hadoop01  
192.168.46.135 hadoop02  
192.168.46.136 hadoop03

vi /etc/profile

export JAVA\_HOME=/home/shicheng/export/servers/jdk  
export HADOOP\_HOME=/home/shicheng/export/servers/hadoop  
export PATH=$PATH:$HOME/bin:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin  
export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

hdfs 分布式存储

MapReduce （继承Mapper和Reduce） 数据计算（数据处理）

太麻烦，效率不是很高，现在已经不是主流

yarn 集群资源统一管理

配置hadoop cd ~server/hadoop/etc/hadoop

* mapred-env.sh

export JAVA\_HOME=/home/shicheng/servers/jdk

* hadoop-env.sh

export JAVA\_HOME=/home/shicheng/servers/jdk

* yarn-env.sh

export JAVA\_HOME=/home/shicheng/servers/jdk

* core-site.xml

<configuration>  
<!-- HDFS的默认路径，端口9000 -->  
<property>  
<name>fs.defaultFS</name>  
<value>hdfs://master:9000</value>  
</property>  
<!-- hadoop运行时，存放数据的临时文件的路径 -->  
<property>  
<name>hadoop.tmp.dir</name>  
<value>/home/shicheng/export/servers/hadoop/tmp</value>  
</property>  
</configuration>

* hdfs-site.xml：

<configuration>  
<property>  
<name>dfs.replication</name>  
<value>1</value>  
</property>  
<property>  
<name>dfs.permissions</name>  
<value>false</value>  
</property>  
</configuration>

* yarn-site.xml

<?xml version="1.0"?>

<configuration>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:18040</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:18030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:18025</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>master:18141</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:18088</value>

</property>

</configuration>

* mapred-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

* 修改slaves

hadoop01

hadoop02

hadoop03

* 将当前master节点上的jdk、hadoop以及 .bash\_profile分发到slave1、slave2上（在hadoop01的~目录下）

scp -r export/ hadoop02:~ scp -r export/ hadoop03:~ scp ~/.bash\_profile hadoop02:~/ scp ~/.bash\_profile hadoop03:~/

所有节点执行source ~/.bash\_profile

* 后续

hadoop namenode -format start-all.sh

jpa

主节点：1796 NameNode 2374 Jps 1978 SecondaryNameNode 2123 ResourceManager

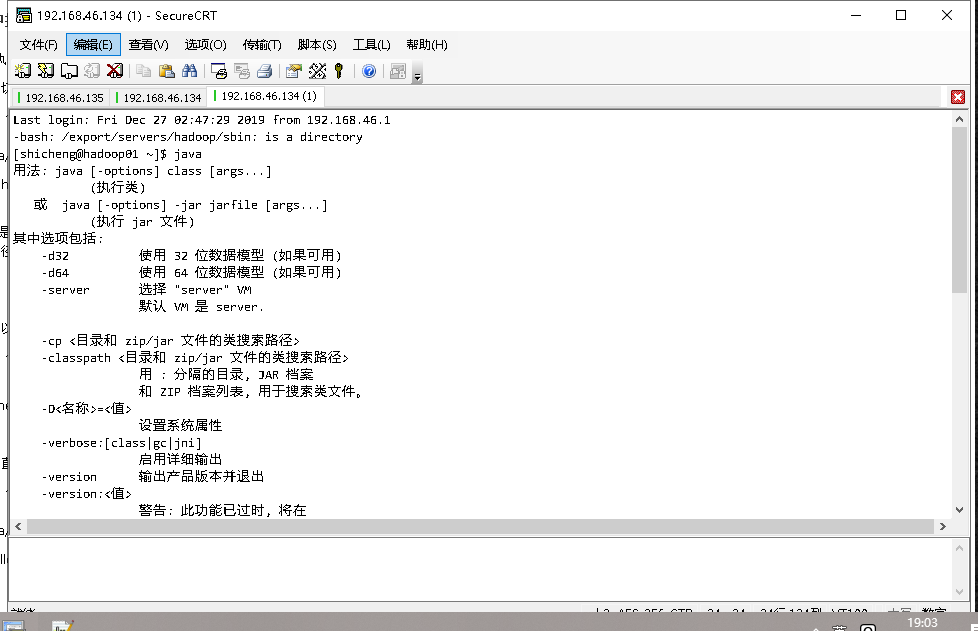
从节点： 1593 DataNode 1787 Jps 1692 NodeManager

hadoop fs 查看分布式集群中有哪些数据

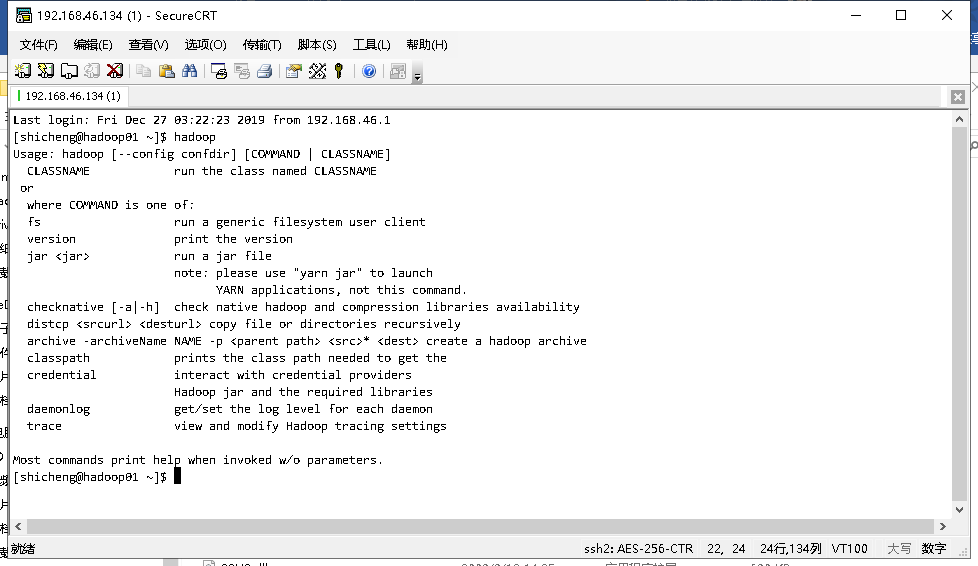
hdfs

hadoop fs -ls / hadoop fs -mrdir /words hadoop fs -rm /words hadoop fs -cat /words/xx hadoop fs -put ./aa /words 将本地文件存储到分布式环境中 hadoop fs -get /words/aa 将分布式环境中的文件下载到本地

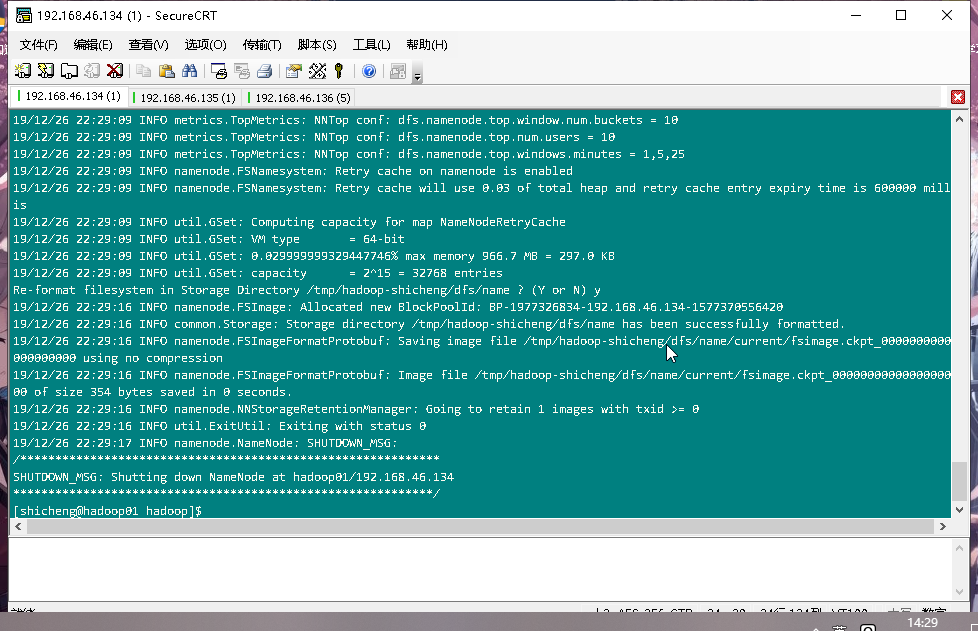
安装java



安装hadoop

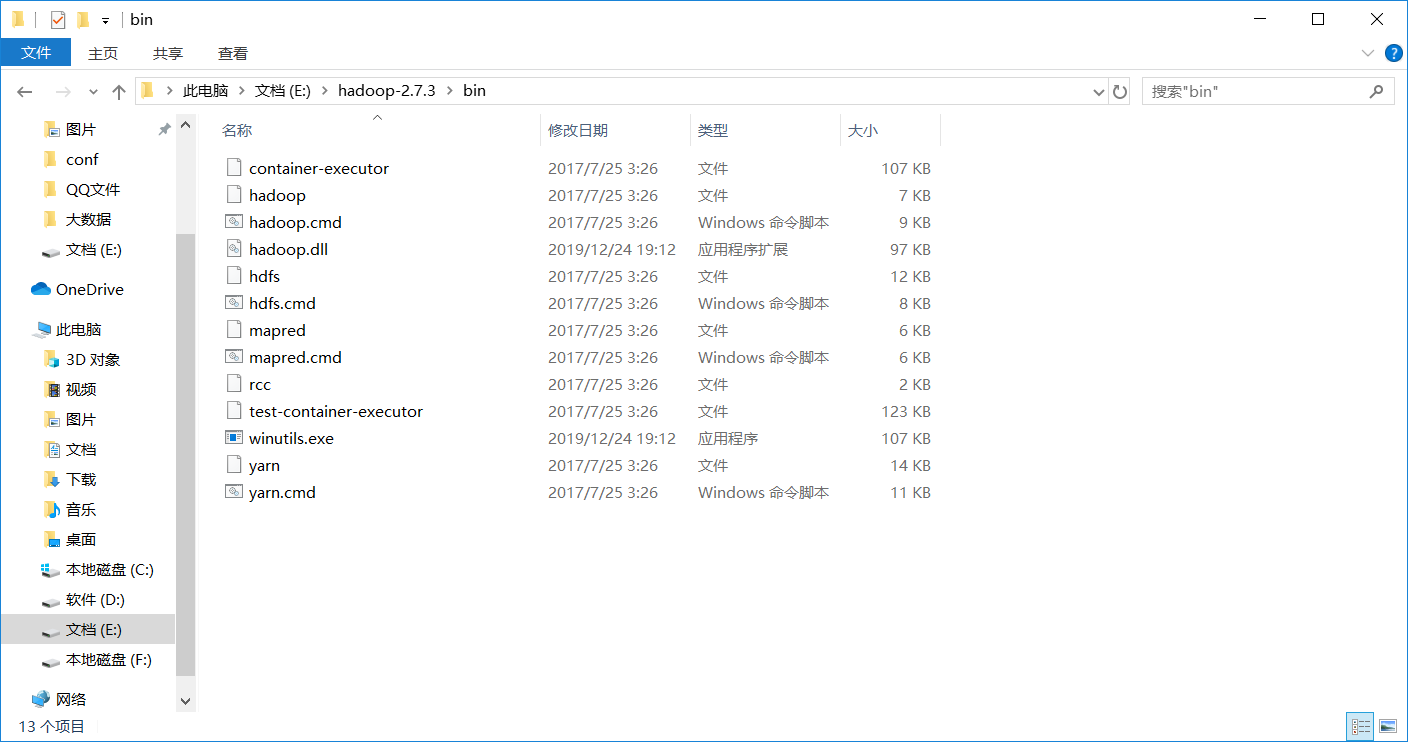


hadoop namenode -format



# 大数据分析工具

1. Windows装开发环境
2. 把 E:\hadoop-2.7.3\bin 下面的 hadoop.dll 和 winutils.xml 拷贝到 C:\Windows\System32

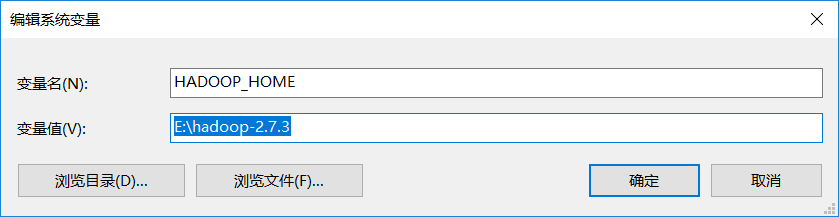


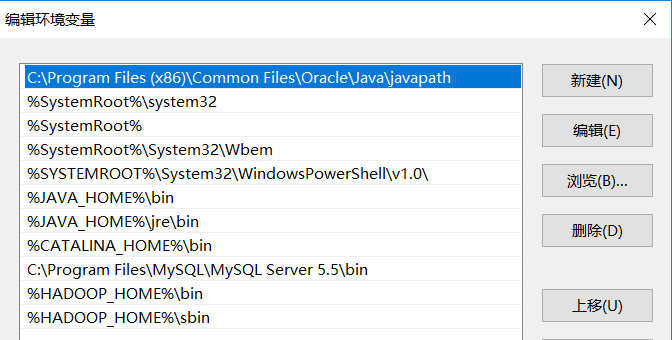
1. 配置环境变量

HADOOP\_HOME

%HADOOP\_HOME%\bin\

%HADOOP\_HOME%\sbin\



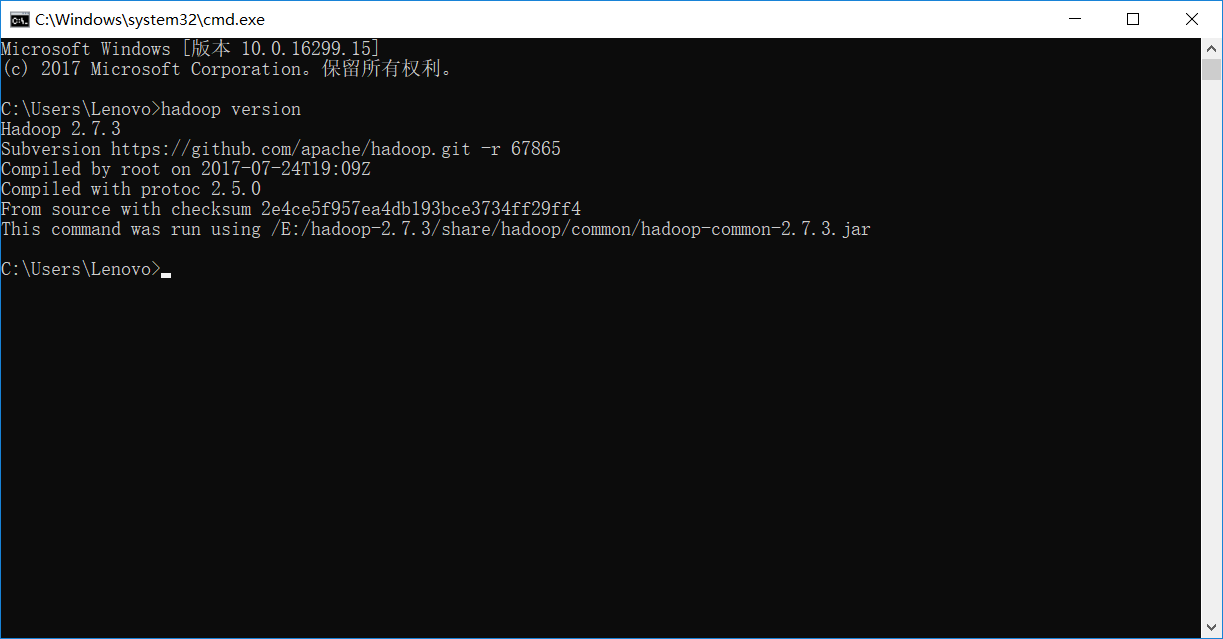


1. 修改hadoop-env.sh文件

将JAVA\_HOME=%JAVA\_HOME%改为JAVA\_HOME=C:\PROGRA~1\Java\jdk1.8.0\_191

PS: 因为Program Files中存在空格，若是直接使用会出现错误，可以进行如下处理： 只需要用PROGRA~1 代替Program Files

1. 用cmd输入hadoop version



1. 搭建项目

jar包引入

common： 的所有.jar以及 lib/中的所有.jar

hdfs： 的 lib/中的所有.jar

mapreduce： 的 lib/中的所有.jar  
yarn: 的 lib/中的所有.jar

1. 编写程序

代码如下：

package count;  
  
import java.io.IOException;  
import java.net.URI;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
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;  
  
public class myWord {  
public static void main(String[] args) throws Exception {  
String inputFold="hdfs://192.168.6141.10:9000/words";  
String outFold="hdfs://192.168.6141.10:9000/outwords";  
   Configuration conf=new Configuration();  
   FileSystem fs=FileSystem.get(new URI("hdfs://192.168.6.10:9000"),conf,"shicheng");  
   if(fs.exists(new Path(outFold))){  
   fs.delete(new Path(outFold), true);  
   }  
  
   Job job=Job.getInstance(conf,"myWordCountApp");  
   job.setJarByClass(myWord.class);//运行的主类  
     
   job.setMapperClass(myMapper.class);  
   job.setMapOutputKeyClass(Text.class);  
   job.setMapOutputValueClass(IntWritable.class);  
     
   job.setReducerClass(myReduce.class);  
   job.setOutputKeyClass(Text.class);  
   job.setOutputValueClass(LongWritable.class);  
     
   FileInputFormat.addInputPath(job, new Path(inputFold));  
   FileOutputFormat.setOutputPath(job, new Path(outFold));  
   System.exit(job.waitForCompletion(true)?0:1);  
}

public static class myMapper extends Mapper<LongWritable, Text, Text, IntWritable>{  
@Override  
protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context)  
throws IOException, InterruptedException {  
String[] words=value.toString().split(" ");  
for(String word:words){  
context.write(new Text(word),new IntWritable(1));  
}  
}  
}  
  
public static class myReduce extends Reducer<Text, IntWritable,Text, LongWritable>{  
@Override  
protected void reduce(Text key, Iterable<IntWritable> value,  
Reducer<Text, IntWritable, Text, LongWritable>.Context context) throws IOException, InterruptedException {  
   long total=0l;  
   for(IntWritable num:value){  
   total=total+1;  
   }  
   context.write(key, new LongWritable(total));  
}  
}  
}

1. 主机查看结果如下：

