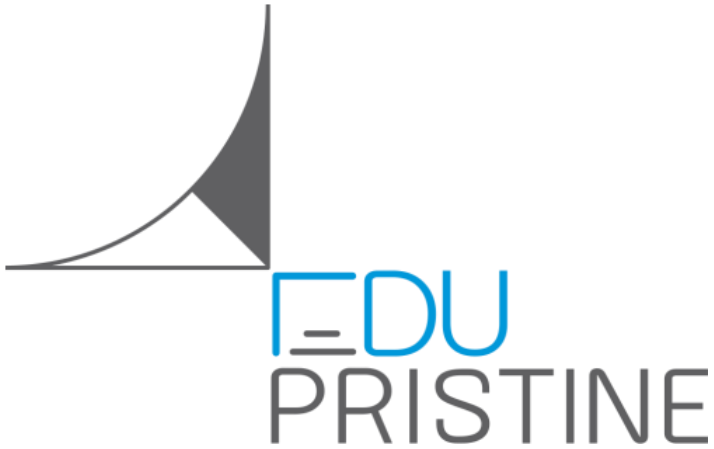


HADOOP

HDFS and Pseudo Cluster

Prerequisites

- JAVA
  - OOPs concepts
  - Serialization
  - Data Structures (Hash Map, Lists)
  - FILE I/O
- UNIX Commands (mv, cp, ls etc, mkdir, ps, vi)

Development Environment

- Install jdk 1.6, jre 6, eclipse

Introduction

What is Big Data

- Typically we work on excel sheet, ppt, word docs, code files. They are of the order 1-2Mb. Even a movie is just 1–2 Gb size.
- The BIG DATA we want to deal with, is of the order of Petabytes. 10^12 times size of ordinary files.

Every min. of Internet

What Happens in An Internet Minute?

Where is this Data

- This data is generated from multiple sources. The data that goes in the logs of google, facebook, linkedin, yahoo servers is of billion users of all around the world.
- What are users accessing, how long the user remains in site. All the meta data sites visited, friend's list, status. Torrent downloads In Every 5 minutes granularity google gets Petabytes information in it's server logs. Same goes for facebook, yahoo, AT&T, Airtel.

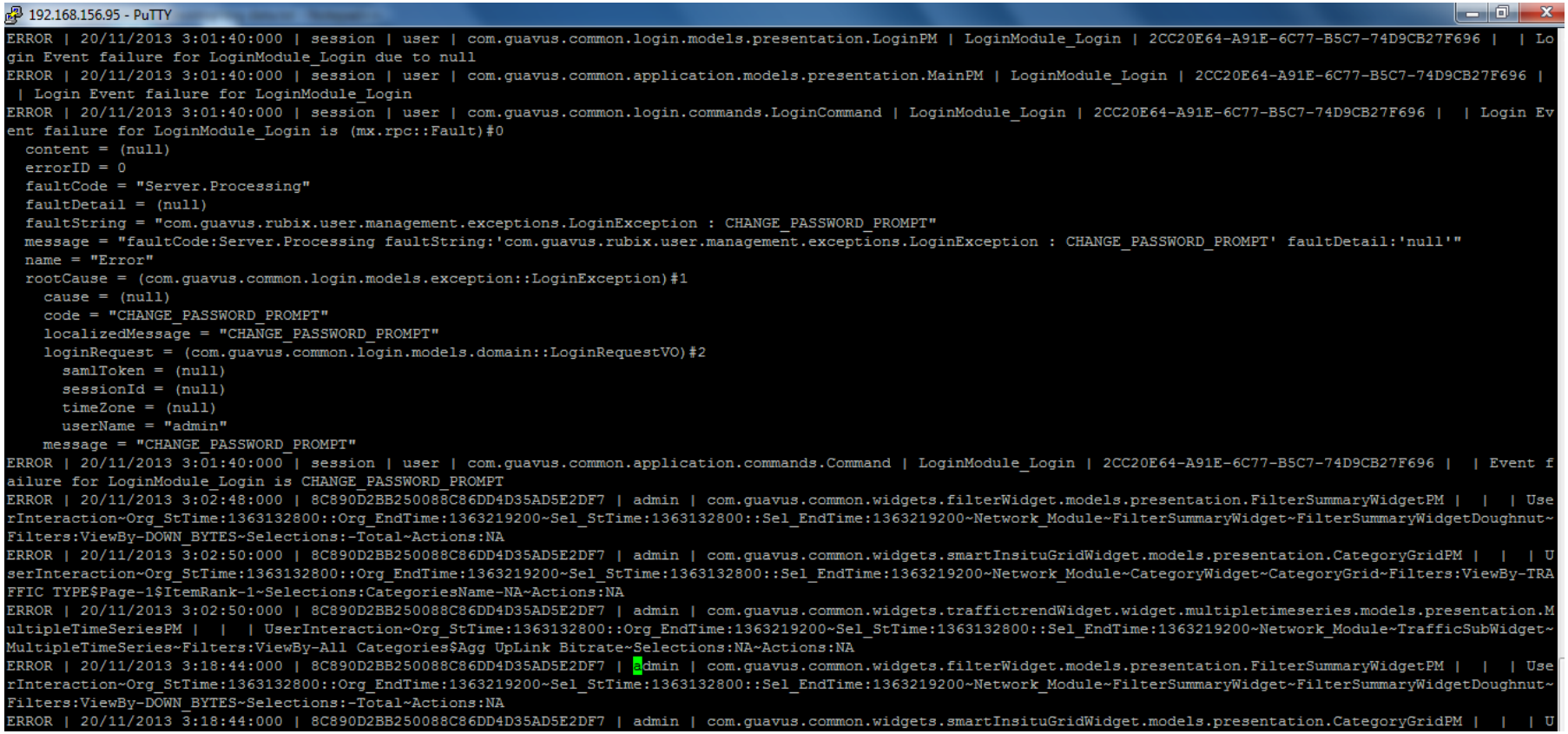
Why is this Data Needed

- In networks companies like airtel in India, AT&T in US monitor routers logs where information of subscriber phone id, call id is recorded to find top subscribers and install routers where traffic is more etc.
- Google, Amazon, Ebay they get logs so that ads and products can be recommended to customers.

Who can Use this Data

- Any organization / entity that has webserver or an information logging utility can utilize it's logs to find out information relevant for it's business, and take insights into data to increase revenue and decrease it’s costs.

How do the Logs Look



Challenges

- Problem is not getting this big data. Problem is **how to store, process and analyze this data.**

Case Study

Telecom Company

- Airmobile (50 million subscribers) wants to sell it's expensive $500 monthly plan to it's customers, for this it wants to find out its top subscribers and the total bytes they have downloaded(Internet data) using it's services in last one month.
- Also it wants to advertise it's roaming plan of $100 so that subscribers don't switch to other networks, when going to other cities. For this it wants to find out the minutes of usage (i.e., call duration) of top 10 thousand subscribers who have roamed in last 1 month .

Issues

- Different subscribers in different cities have different data plans. Almost all the subscribers are active each day of month.
- Data collection is huge every minute almost 1 million people visit 5–6 sites.
- Every day tera bytes of information is collected in airmobile servers of each city, which get discarded because of unavailability of storage.

Solution

- Introduced by Google was GFS (Google file system) and Map Reduce.
- Then Hadoop became open source and now is owned by apache.
- Hadoop is used by Facebook, Yahoo, Google, Twitter, linkedin, Rackspace.
- A new concept Hadoop Yarn which is a framework for job scheduling and cluster resource management has also come into existence.

How is HADOOP the Solution

- Storage -> HDFS A distributed file system where commodity hardware can be used to form clusters and store the huge data in distributed fashion. There is no need for high end hardwares.
- Process -> MAP Reduce Paradigm
- Analyze -> Hive, Pig MapReduce.
- It can easily scale to multiple nodes(1,500–2,000 nodes in a cluster), with just configuration change.

Applications of HADOOP

- Telecommunications -> To find out top subscribers for advertisement, find peak traffic rate to install routers at right places, for cost cutting.
- Recommendation systems -> Google Ads customized for all users.
- Data warehousing -> to store data and analyze it e.g., categorize data into http web or mobile, so that services by ISP can be customized accordingly.
- Market Research and Forecasting -> Forecast subscribers, traffic based on past data trend.
- Finance, social networking -> To predict trends and gain profit.

What it is Not

- Should be noted it's not OLAP (online analytical Processing) but batch / offline oriented
- It is not a database

Storage HDFS

Challenges

- Can this data be stored in 1 machine? Hard drives are approximately 500Gb in size. Even if you add external hard drives, you can't store the data in Peta bytes. Let's say you add external hard drives and store this data, you wouldn't be able to open or process that file because of insufficient RAM. And processing, it would take months to analyze this data.

HDFS Features

- Data is distributed over several machines, and replicated to ensure their durability to failure and high availability to parallel applications
- Designed for very large files (in GBs, TBs)
- Block oriented
- Unix like commands interface
- Write once and read many times
- Commodity hardware
- Fault Tolerant when nodes fail
- Scalable by adding new nodes

It is Not Designed for

- Small files.
- Multiple writes, arbitrary file modification -> Writes are always supported at the end of the file, modifications can't be made at random offsets of files.
- Low latency data access -> Since we are accessing huge amount of data it comes at the expense of time taken to access the data.

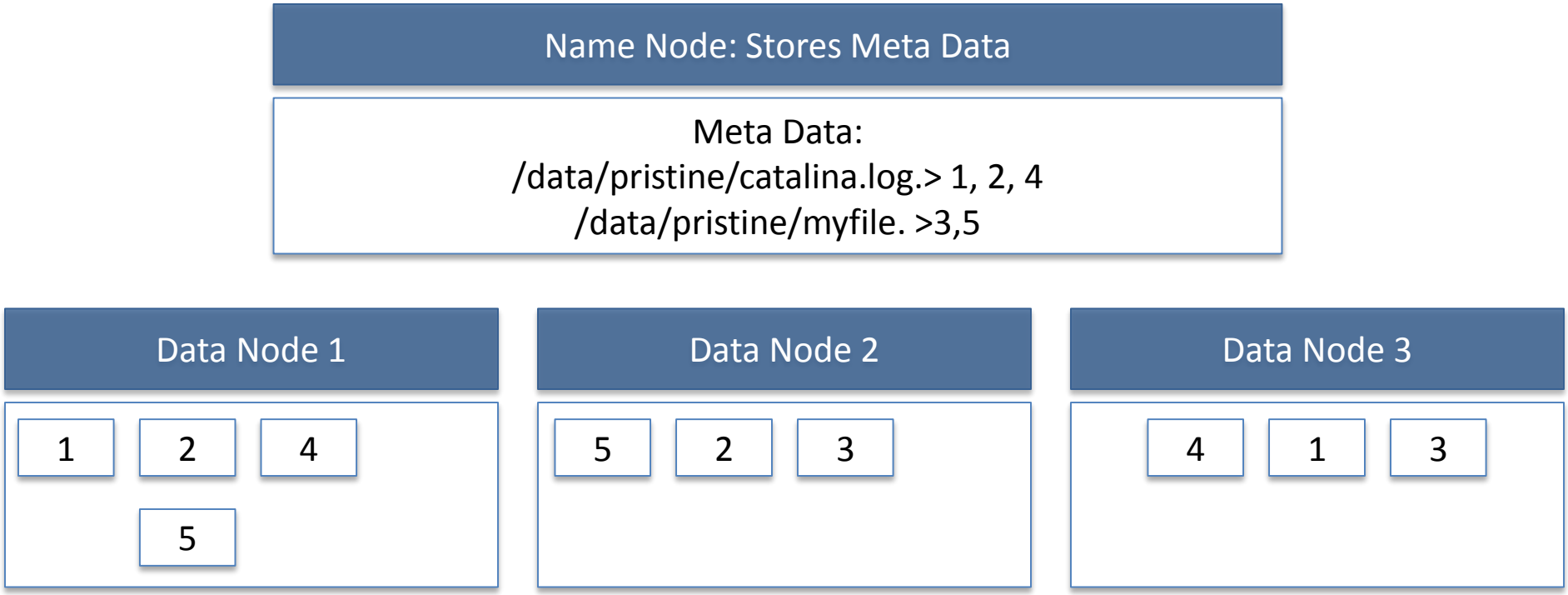
HDFS Design

- Individual files are broken into blocks of fixed size (typically 64 Mb >> 4Kb block structured FS), and stored across cluster of nodes. (Not necessarily on same machine). These files can be more than the size of individual machine's hard drive.
- Individual machines are called Data Nodes.
- So access to a file requires cooperation of several nodes.

Design Details

- Challenge -> Since several machines are involved in serving a file, loss of machine can result in failure of servicing the file request.
- HDFS Solution : Replicate each block across a number of machines (3 in the screenshot). Each block 64 Mb in size, this is to keep smaller metadata at name node for each file (list will be small).
- HDFS expects large files sequentially laid on disk for processing by MAP REDUCE. Small number of very large files. So for smaller data HDFS is not an optimal solution.

Architecture



Function of Name Node

- Name Node is controller and manager of HDFS. It knows the status and the metadata of all the files in HDFS.
- Metadata is -> file names, permissions, and locations of each block of file
- HDFS cluster can be accessed concurrently by multiple clients, even then this metadata information is never desynchronized. Hence, all this information is handled by a single machine.
- Since metadata is typically small, all this info. is stored in main memory of Name Node, allowing fast access to metadata.

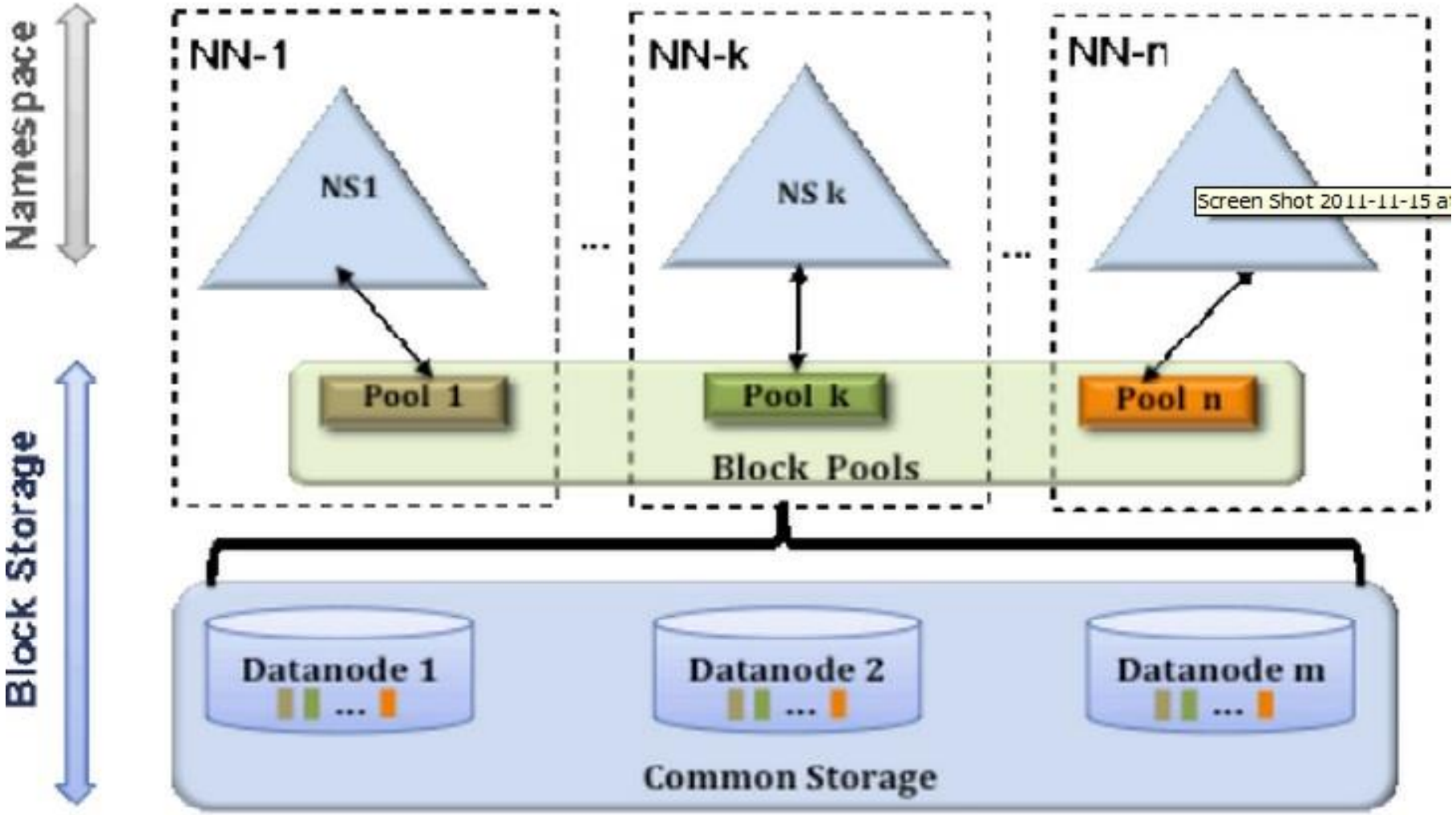
Name Node HA & Node Manager

- Since function of name node is very critical for overall health of HDFS, even if individual data nodes fail, HDFS can recover and function with a little less capacity however crash of name node can lose all the information and the complete file system irrecoverably. That's why metadata and involvement of name node in data transfer is kept minimal.

HighAvailibility(HA)

- In Hadoop 1.0 there is single NameNode which led to fact that if the name node fails then the complete process gets affected until the NameNode is restarted again which is taken care of by the introduction of HA in hadoop 2.0 onwards.
- The HDFS High Availability feature addresses the above problems by providing the option of running two redundant Name Nodes. Therefore, a typical HA cluster is created in two separate machines which are configured as NameNode. At any point in time, one of the NameNode is in an *Active* state, and the other is in a *Standby* state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if required.

HDFS High Availability



Source :- Hortonworks

Purpose of Secondary Name Node

- It is not backup of name node nor data nodes connect to this. It is just a helper of name node.
- It only performs periodic checkpoints.
- It communicates with name node and to take snapshots of HDFS metadata.
- These snapshots help minimize downtime and loss of data.

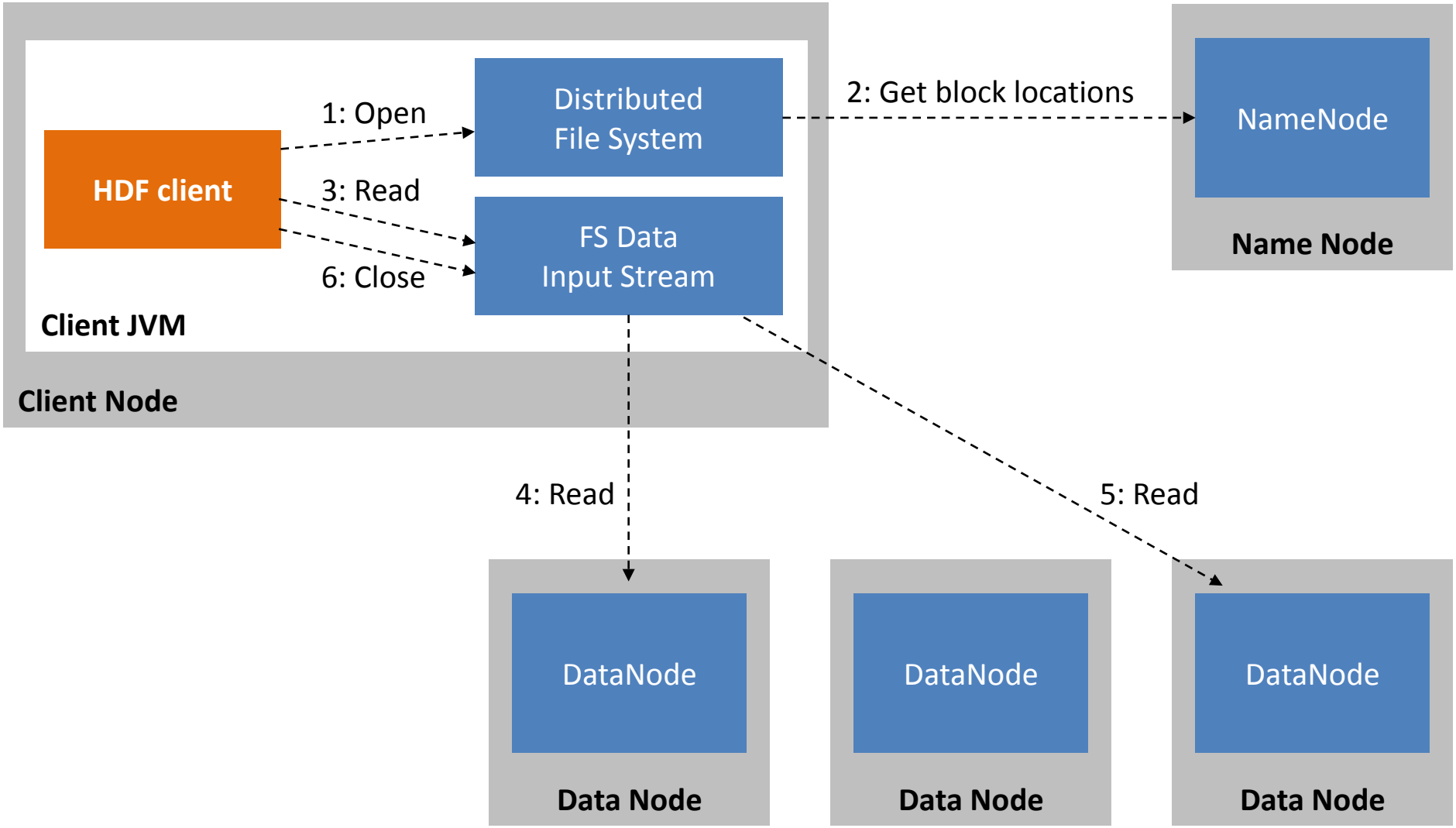
Purpose of Resource Manager

- New architecture of Hadoop divides the two major functions of the JobTracker : resource management and job life-cycle management into separate components.
- ResourceManager manages the global assignment of compute resources to applications.
- It also looks after the per-application ApplicationMaster which manages the application, scheduling and coordination.
- It has two main components: Scheduler and ApplicationsManager.
- Scheduler is responsible for allocating resources to the various running applications
- Applications Manager is responsible for accepting job-submissions, negotiating the first container for executing the application specific ApplicationMaster.

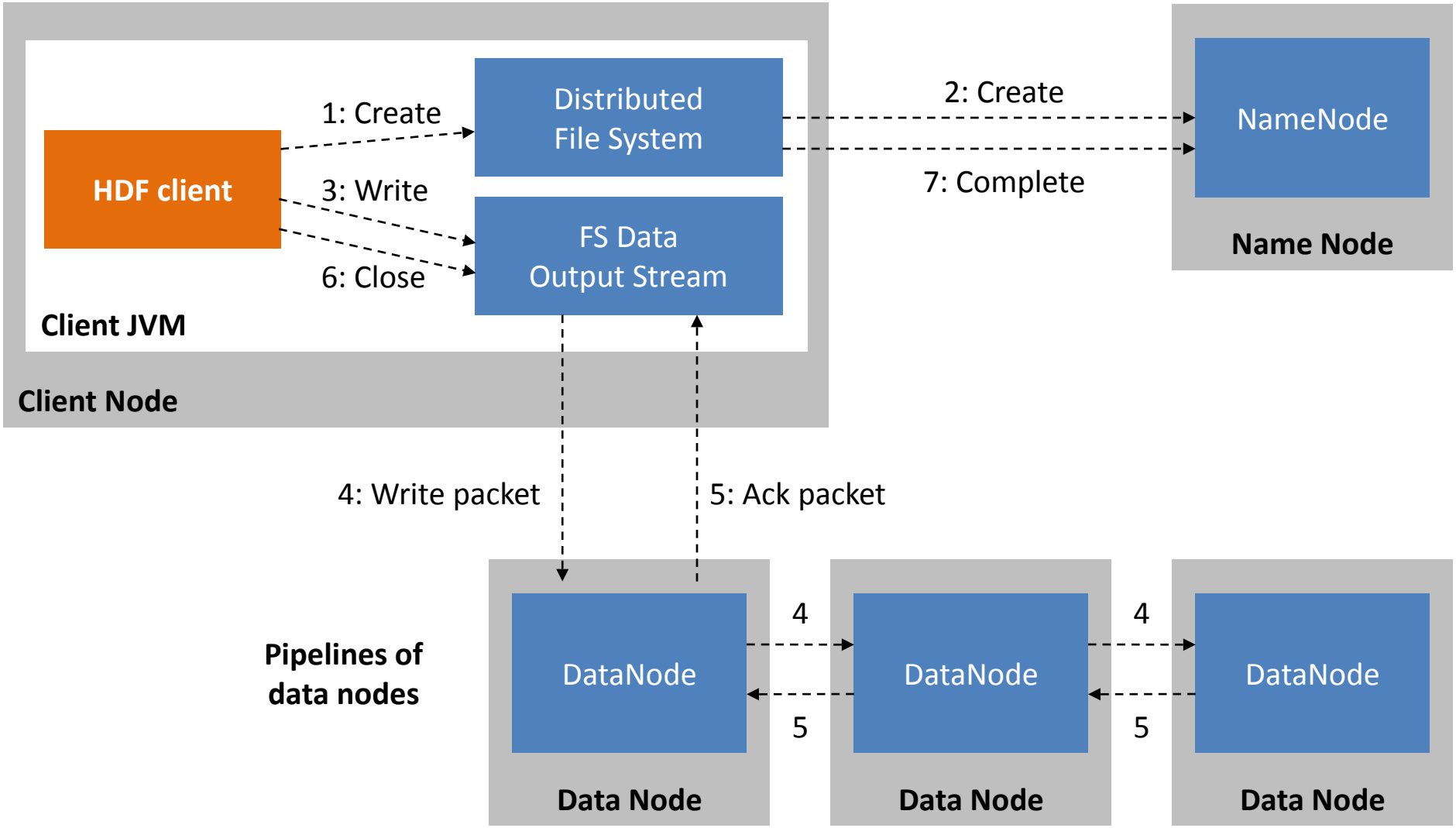
Overall Function

- To open a file, client contacts the name node retrieves list of locations of blocks of that file. These locations identify data nodes that hold the block. Clients then read the data directly from data nodes in parallel. Name Node is not involved in this stage.

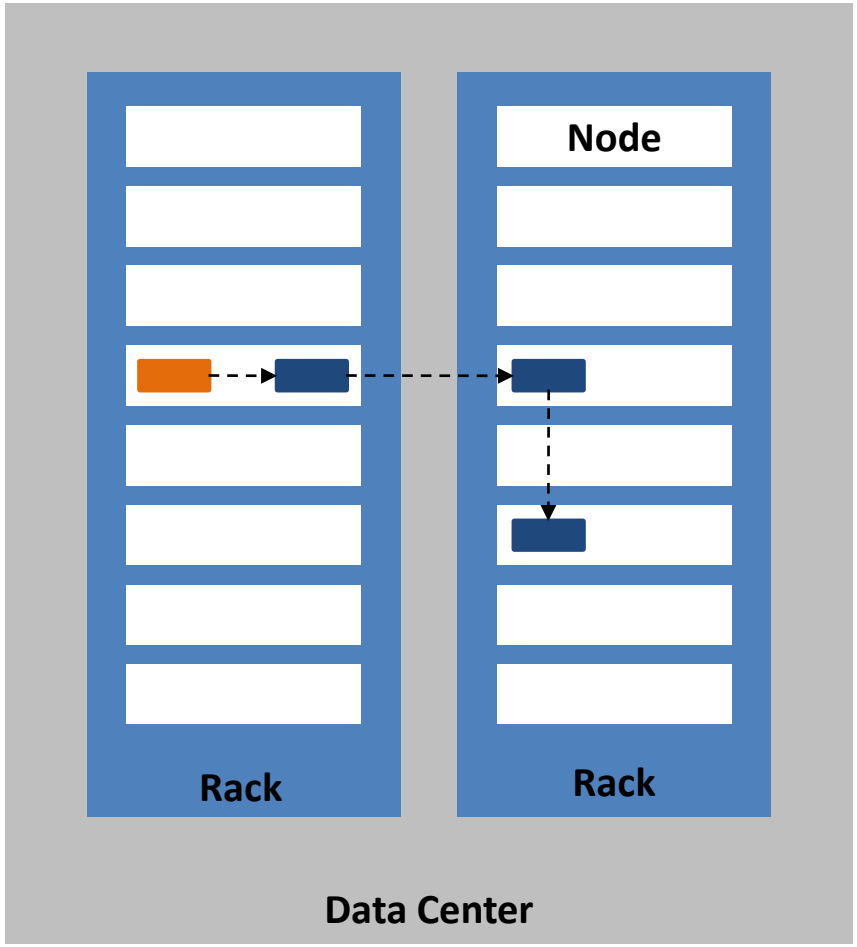
HDFS Read


HDF client
1: Open
3: Read
6: Close
Distributed File System
FS Data Input Stream
Client JVM
Client Node
2: Get block locations
NameNode
Name Node
4: Read
5: Read
DataNode
Data Node
DataNode
Data Node
DataNode
Data Node

HDFS Write


HDF client
1: Create
3: Write
6: Close
Distributed File System
FS Data Output Stream
Client JVM
Client Node
2: Create
7: Complete
NameNode
Name Node
4: Write packet
5: Ack packet
Pipelines of data nodes
DataNode
4
5
DataNode
4
5
DataNode
Data Node
Data Node
Data Node

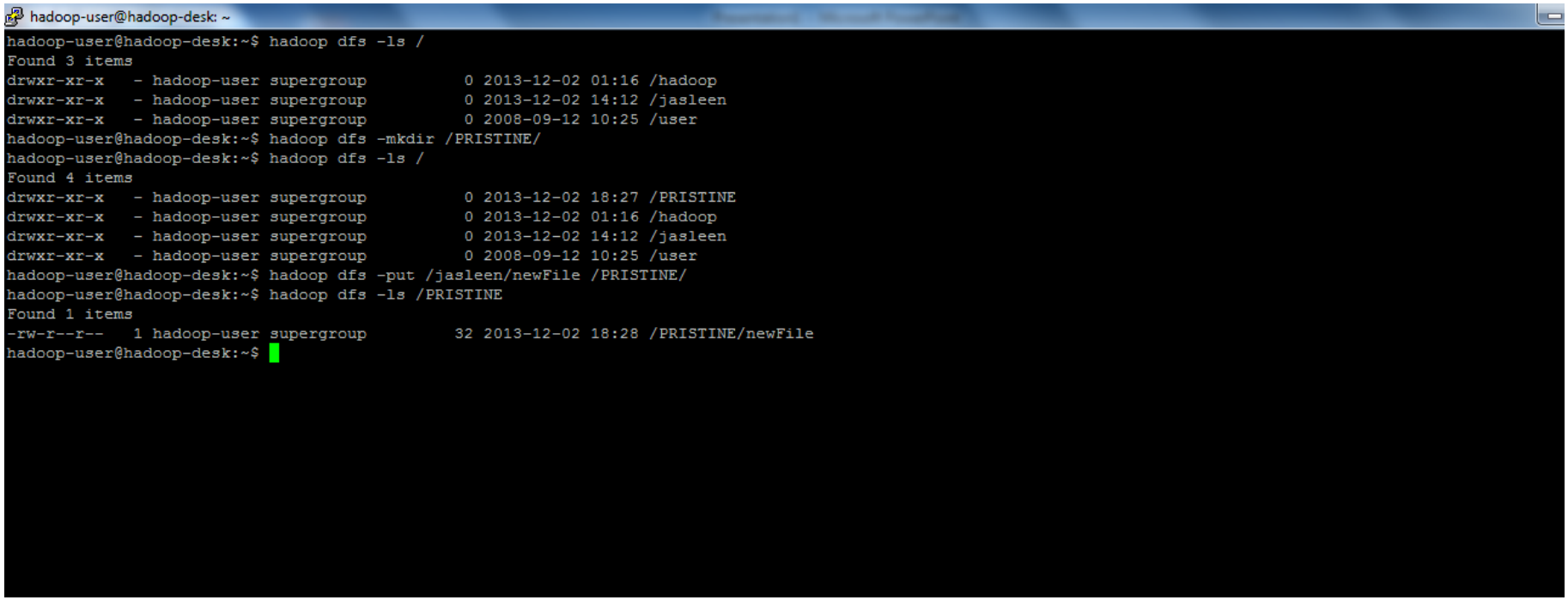
Replica Placement



How does it Run

- HDFS files are not part of ordinary file system. "ls" command will not list the files in HDFS.
- HADOOP runs as a separate process and in a different namespace isolated from normal FS on the OS.
- HDFS files/blocks are stored in a particular directory controlled by Datanode. Files are stored with block Ids. You can't interact with them using normal Unix commands.
- There are separate utilities similar to Unix that can be used to access the HDFS

Listing files in hdfs



Environment Needed for HADOOP

- The production and stable environment for HADOOP is Unix.
- You can run HADOOP on windows as well, but again you would need Unix like system (e.g., cygwin) to run HADOOP processes. For this cygwin should be run as a service in windows OS and can be a problem if user doesn't have admin rights
- Since most people have Windows OS, we will run HADOOP on preconfigured VM (given by Yahoo/CDH)

Configuring and Setting up HDFS

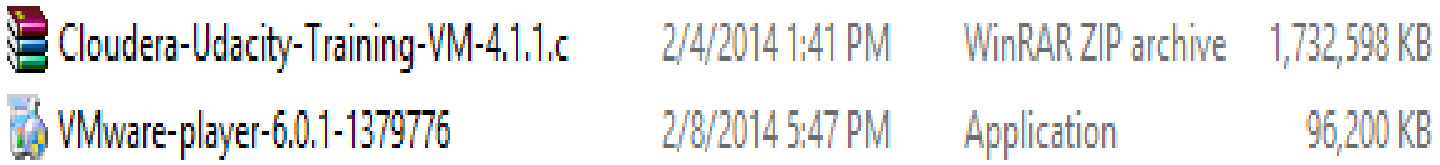
- Download vm player from http://www.vmware.com/support/download-player

HADOOP vm from cloudera can be downloaded from http://content.udacity-data.com/courses/ud617/Cloudera-Udacity-Training-VM-4.1.1.c.zip

- HADOOP has introduced several versions the vm image present in the following link is 0.18 version of HADOOP http://developer.yahoo.com/hadoop/tutorial/module3.html
- Download putty from http://www.putty.org/ and winscp http://winscp.net/eng/download.php

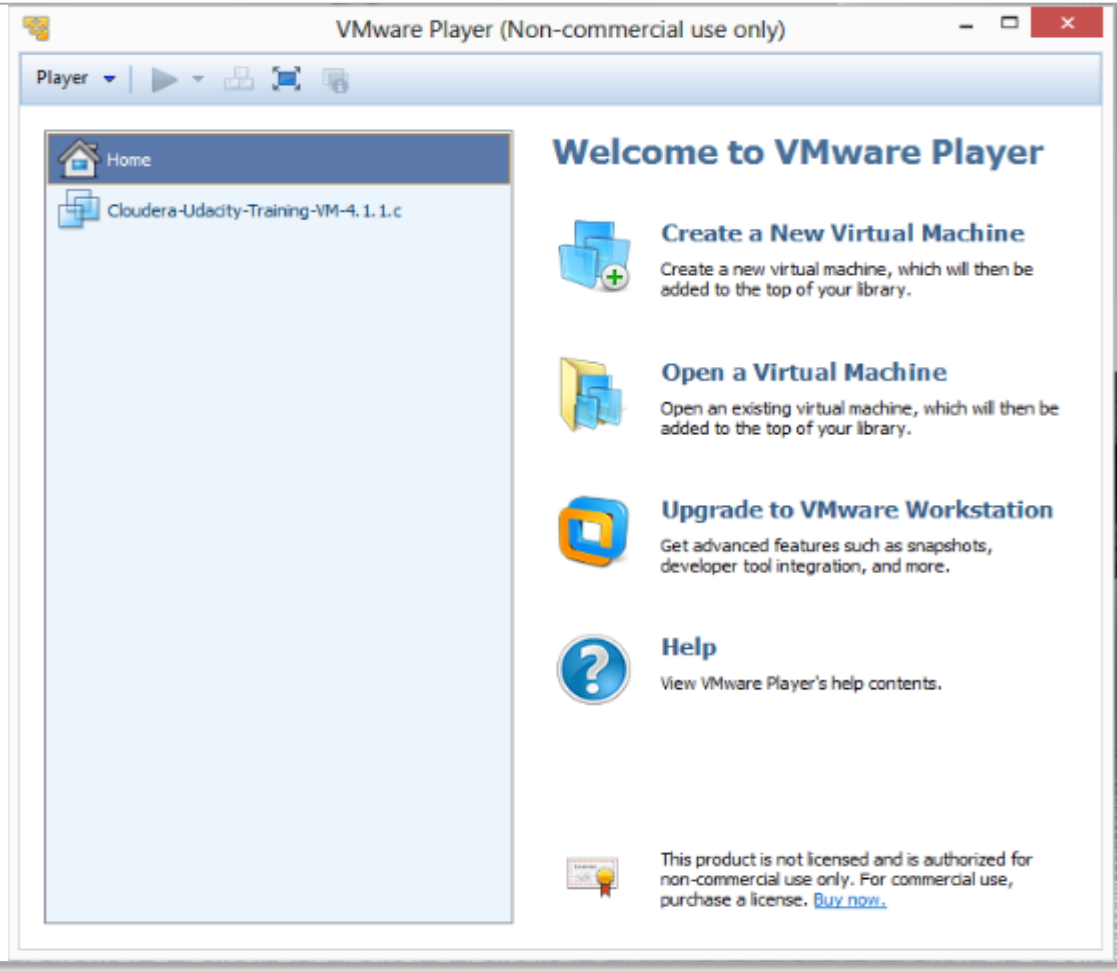
Steps

- After VMplayer is successfully installed, extract Cloudera-Udacity-4.1 zip to a folder



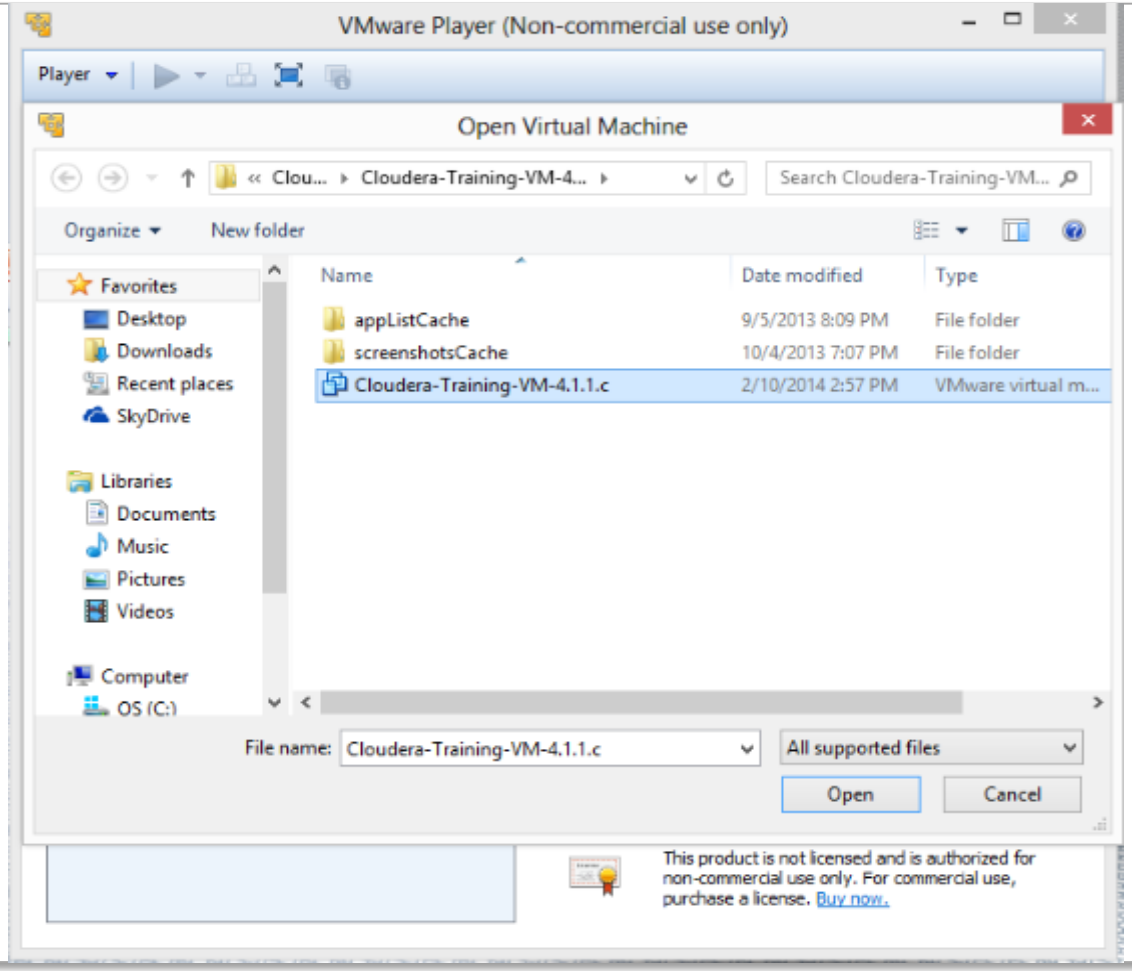
VMware Setup

- Double click on VM player quick launcher, following screen will be opened



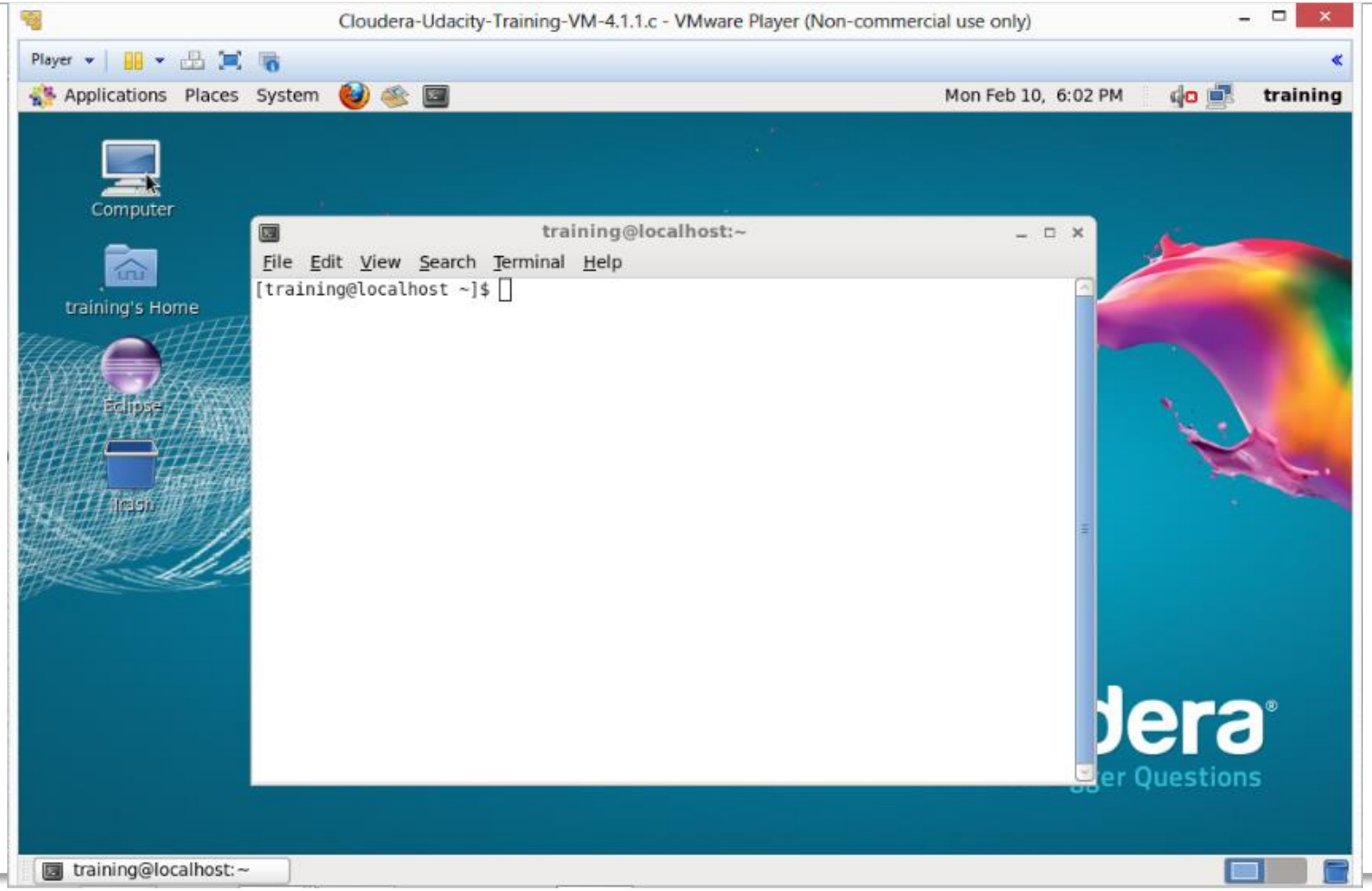
Cont.

- Click on ‘Open a Virtual Machine’ and direct to extracted image folder.



Cont.

- After opening the image following information will be displayed in vm player

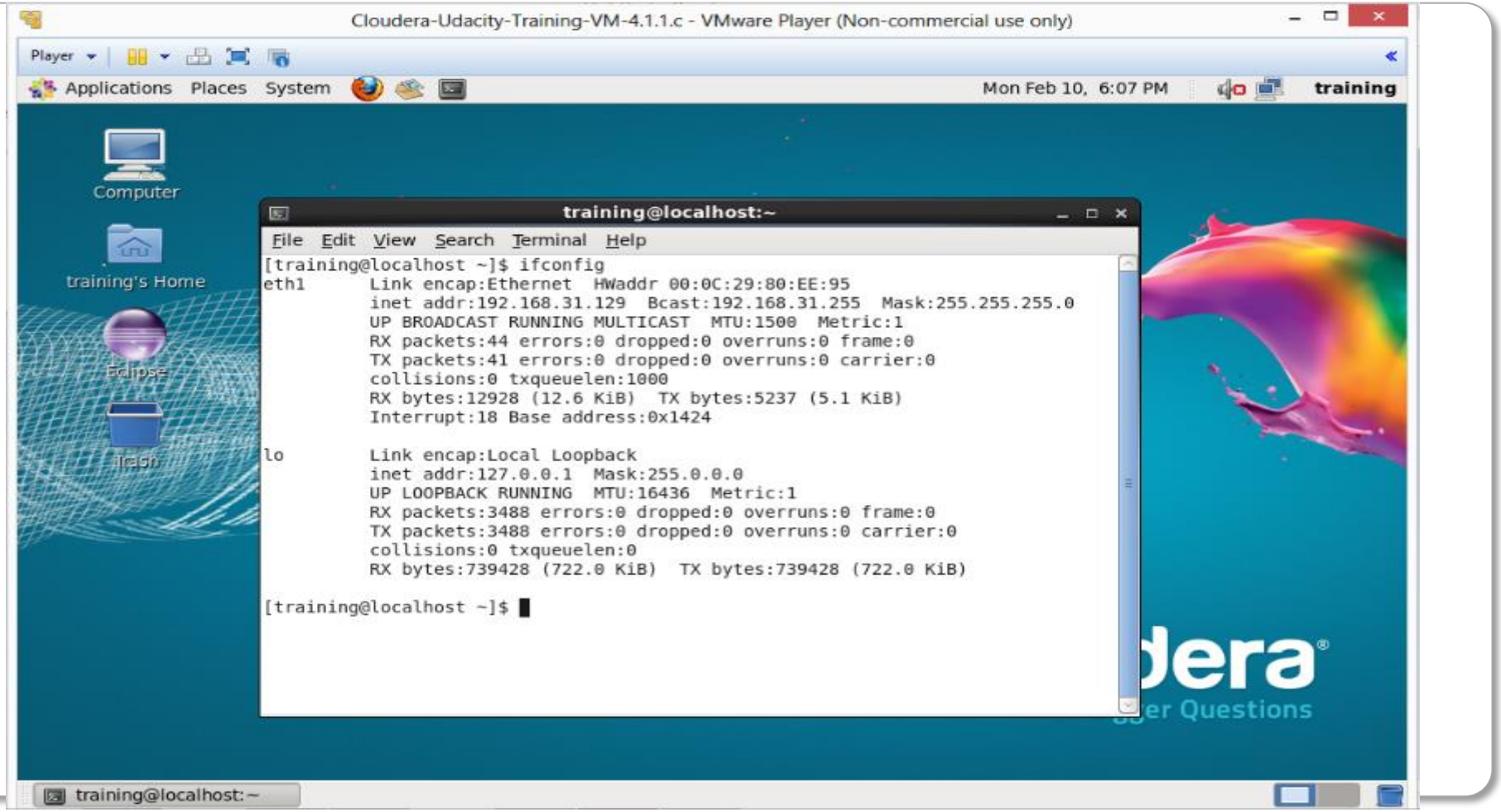


Cont.

- This vm image is configured with 2 users root and training. The password of training user is training.
- As soon as the vm is loaded correctly, an IP is assigned to it . In the screenshot attached the IP is 192.168.31.129.
- The user training is sudo user of the system.
- You can also connect to this vm using ssh.

Cont.

- The ip of vm can be found by typing ifconfig in terminal

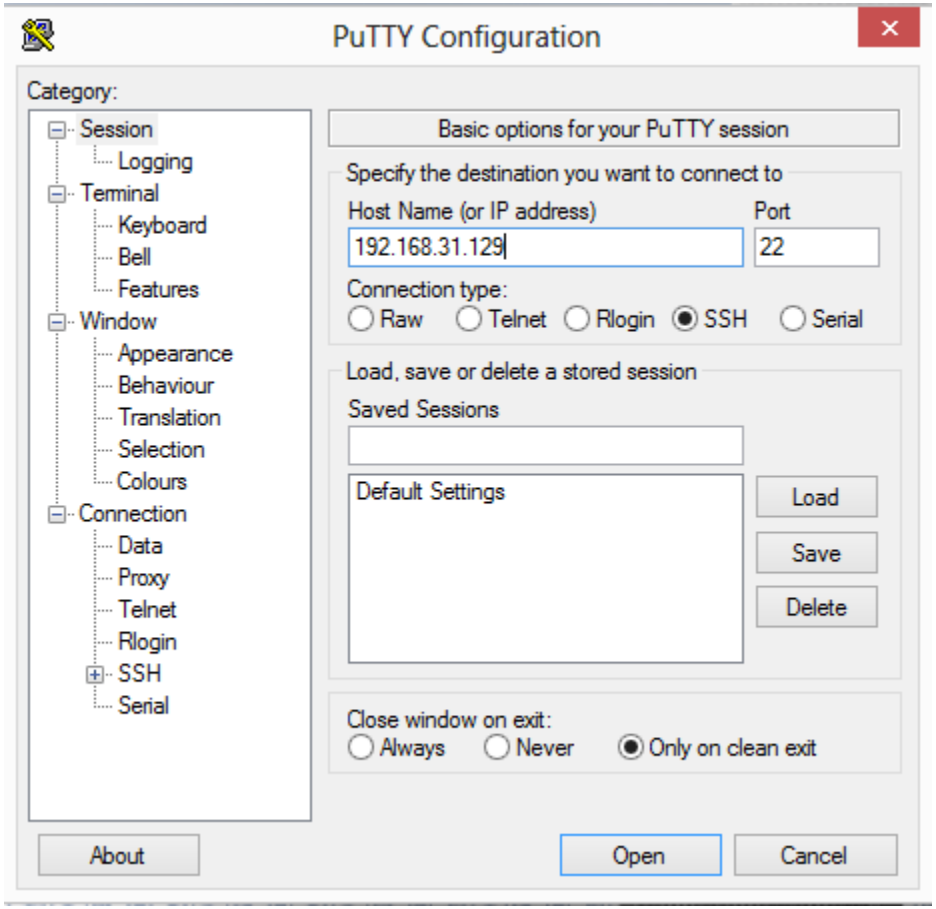


Cont.

- To use the vm either directly use vm or putty.
- Whichever UI, student finds suitable can be used. Both are pointing to the same server.

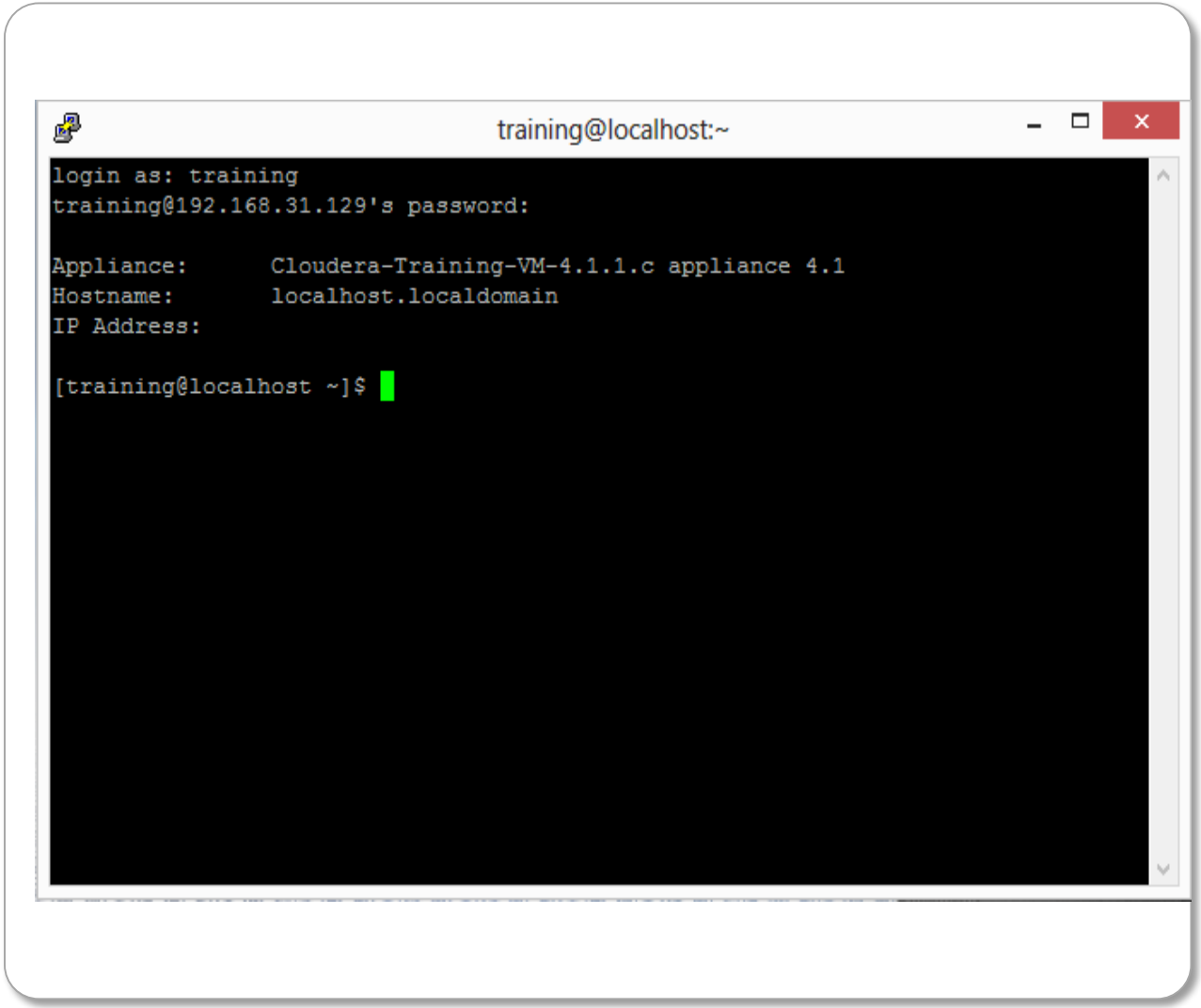
Cont.

- Double click on putty.exe and enter the IP of vm



Cont.

- Login training
- Password training
- After login following screen should appear



Cont.

- To check if HADOOP processes are running in the system, type
- ps –ef | grep java, name node, secondary name node, data node will be shown

```
cloudera@cloudera-vm:~$ ps -ef | grep java
hbase     1630     1  0 Dec07 ?        00:01:59 /usr/lib/jvm/java-6-sun/bin/java -Xmx1000m -ea -XX:+UseConcMarkSweepGC -XX:+CMSIncrementalMode -ea -XX:+UseConcMarkSweep
GC -XX:+CMSIncrementalMode -ea -XX:+UseConcMarkSweepGC -XX:+CMSIncrementalMode -Dhbase.log.dir=/usr/lib/hbase/logs -Dhbase.log.file=hbase-hbase-master-cloudera-vm.log -
Dhbase.home.dir=/usr/lib/hbase -Dhbase.id.str=hbase -Dhbase.root.logger=INFO,DRFA -classpath /usr/lib/hbase/conf:/usr/lib/jvm/java-6-sun/lib/tools.jar:/usr/lib/hbase:/u
sr/lib/hbase/hbase-0.90.1-cdh3u0.jar:/usr/lib/hbase/hbase-0.90.1-cdh3u0-tests.jar:/usr/lib/hbase/lib/activation-1.1.jar:/usr/lib/hbase/lib/asm-3.1.jar:/usr/lib/hbase/li
b/avro-1.3.3.jar:/usr/lib/hbase/lib/commons-cli-1.2.jar:/usr/lib/hbase/lib/commons-codec-1.4.jar:/usr/lib/hbase/lib/commons-el-1.0.jar:/usr/lib/hbase/lib/commons-httpcl
ient-3.1.jar:/usr/lib/hbase/lib/commons-lang-2.5.jar:/usr/lib/hbase/lib/commons-logging-1.1.1.jar:/usr/lib/hbase/lib/commons-net-1.4.1.jar:/usr/lib/hbase/lib/core-3.1.1
.jar:/usr/lib/hbase/lib/guava-r06.jar:/usr/lib/hbase/lib/hadoop-core.jar:/usr/lib/hbase/lib/hbase-0.90.1-cdh3u0.jar:/usr/lib/hbase/lib/jackson-core-asl-1.5.2.jar:/usr/l
ib/hbase/lib/jackson-jaxrs-1.5.5.jar:/usr/lib/hbase/lib/jackson-mapper-asl-1.5.2.jar:/usr/lib/hbase/lib/jackson-xc-1.5.5.jar:/usr/lib/hbase/lib/jasper-compiler-5.5.23.j
ar:/usr/lib/hbase/lib/jasper-runtime-5.5.23.jar:/usr/lib/hbase/lib/jaxb-api-2.1.jar:/usr/lib/hbase/lib/jaxb-impl-2.1.12.jar:/usr/lib/hbase/lib/jersey-core-1.4.jar:/usr/
lib/hbase/lib/jersey-json-1.4.jar:/usr/lib/hbase/lib/jersey-server-1.4.jar:/usr/lib/hbase/lib/jettison-1.1.jar:/usr/lib/hbase/lib/jetty-6.1.26.jar:/usr/lib/hbase/lib/je
tty-util-6.1.26.jar:/usr/lib/hbase/lib/jruby-complete-1.0.3.jar:/usr/lib/hbase/lib/jsp-2.1-6.1.14.jar:/usr/lib/hbase/lib/jsp-api-2.1-6.1.14.jar:/usr/lib/hbase/lib/jsp-a
pi-2.1.jar:/usr/lib/hbase/lib/jsr311-api-1.1.1.jar:/usr/lib/hbase/lib/log4j-1.2.16.jar:/usr/lib/hbase/lib/protobuf-java-2.3.0.jar:/usr/lib/hbase/lib/servlet-api-2.5-6.1
.14.jar:/usr/lib/hbase/lib/servlet-api-2.5.jar:/usr/lib/hbase/lib/slf4j-api-1.5.8.jar:/usr/lib/hbase/lib/slf4j-log4j12-1.5.8.jar:/usr/lib/hbase/lib/stax-api-1.0.1.jar:/
usr/lib/hbase/lib/thrift-0.2.0.jar:/usr/lib/hbase/lib/xmlenc-0.52.jar:/usr/lib/hbase/lib/zookeeper.jar org.apache.hadoop.hbase.master.HMaster start
hue       1772  1725  0 Dec07 ?        00:00:44 /usr/lib/jvm/java-6-sun/bin/java -Dproc_jar -Xmx1000m -Dlog4j.configuration=log4j.properties -Dhadoop.log.dir=/usr/lib/h
adoop-0.20/logs -Dhadoop.log.file=hadoop.log -Dhadoop.home.dir=/usr/lib/hadoop-0.20 -Dhadoop.id.str= -Dhadoop.root.logger=INFO,console -Djava.library.path=/usr/lib/hadoop-0.20/lib/native/Linux-i386-32 -Dhadoop.policy.file=hadoop-policy.xml -classpath /etc/hive/conf:/usr/share/hue/apps/beeswax/src/beeswax/../../../../desktop/conf:/usr/lib/hadoop-0.20/conf:/usr/lib/jvm/java-6-sun/lib/tools.jar:/usr/lib/hadoop-0.20:/usr/lib/hadoop-0.20/hadoop-core-0.20.2-cdh3u0.jar:/usr/lib/hadoop-0.20/lib/ant-contrib-1.0b3.jar:/usr/lib/hadoop-0.20/lib/aspectjrt-1.6.5.jar:/usr/lib/hadoop-0.20/lib/aspectjtools-1.6.5.jar:/usr/lib/hadoop-0.20/lib/commons-cli-1.2.jar:/usr/lib/hadoop-0.20/lib/commons-codec-1.4.jar:/usr/lib/hadoop-0.20/lib/commons-daemon-1.0.1.jar:/usr/lib/hadoop-0.20/lib/commons-el-1.0.jar:/usr/lib/hadoop-0.20/lib/commons-httpclient-3.0.1.jar:/usr/lib/hadoop-0.20/lib/commons-logging-1.0.4.jar:/usr/lib/hadoop-0.20/lib/commons-logging-api-1.0.4.jar:/usr/lib/hadoop-0.20/lib/commons-net-1.4.1.jar:/usr/lib/hadoop-0.20/lib/core-3.1.1.jar:/usr/lib/hadoop-0.20/lib/hadoop-fairscheduler-0.20.2-cdh3u0.jar:/usr/lib/hadoop-0.20/lib/hsqldb-1.8.0.10.jar:/usr/lib/hadoop-0.20/lib/hue-plugins-1.2.0.jar:/usr/lib/hadoop-0.20/lib/jackson-core-asl-1.5.2.jar:/usr/lib/hadoop-0.20/lib/jackson-mapper-asl-1.5.2.jar:/usr/lib/hadoop-0.20/lib/jasper-compiler-5.5.12.jar:/usr/lib/hadoop-0.20/lib/jasper-runtime-5.5.12.jar:/usr/lib/hadoop-0.20/lib/jets3t-0.6.1.jar:/usr/lib/hadoop-0.20/lib/jetty-6.1.26.jar:/usr/lib/hadoop-0.20/lib/jetty-servlet-tester-6.1.26.jar:/usr/lib/hadoop-0.20/lib/jetty-util-6.1.26.jar:/usr/lib/hadoop-0.20/lib/jsch-0.1.42.jar:/usr/lib/hadoop-0.20/lib/junit-4.5.jar:/usr/lib/hadoop-0.20/lib/kfs-0.2.2.jar:/usr/lib/hadoop-0.20/lib/log4j-1.2.15.jar:/usr/lib/hadoop-0.20/lib/mockito-all-1.8.2.jar:/usr/lib/hadoop-0.20/lib/oro-2.0.8.jar:/usr/lib/hadoop-0.20/lib/servlet-api-2.5-20081211.jar:/usr/lib/hadoop-0.20/lib/servlet-api-2.5-6.1.14.jar:/usr/lib/hadoop-0.20/lib/slf4j-api-1.4.3.jar:/usr/lib/hadoop-0.20/lib/slf4j-log4j12-1.4.3.jar:/usr/lib/hadoop-0.20/lib/xmlenc-0.52.jar:/usr/lib/hadoop-0.20/lib/jsp-2.1/jsp-2.1.jar:/usr/lib/hadoop-0.20/lib/jsp-2.1/jsp-api-2.1.jar:/usr/lib/hive/lib/hive-shims-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/hbase-0.90.1-cdh3u0.jar:/usr/lib/hive/lib/hive-service-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/datanucleus-connectionpool-2.0.3.jar:/usr/lib/hive/lib/datanucleus-rdbms-2.0.3.jar:/usr/lib/hive/lib/hive-exec-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/asm-3.1.jar:/usr/lib/hive/lib/commons-collections-3.2.1.jar:/usr/lib/hive/lib/antlr-runtime-3.0.1.jar:/usr/lib/hive/lib/commons-lang-2.4.jar:/usr/lib/hive/lib/commons-logging-1.0.4.jar:/usr/lib/hive/lib/hive-serde-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/hive-hwi-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/thrift-0.5.0.jar:/usr/lib/hive/lib/commons-logging-api-1.0.4.jar:/usr/lib/hive/lib/junit-3.8.1.jar:/usr/lib/hive/lib/hive-contrib-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/hive-cli-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/jline-0.9.94.jar:/usr/lib/hive/lib/log4j-1.2.15.jar:/usr/lib/hive/lib/guava-r06.jar:/usr/lib/hive/lib/hive-jdbc-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/hive-common-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/stringtemplate-3.1b1.jar:/usr/lib/hive/lib/json.jar:/usr/lib/hive/lib/hive-anttasks-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/jdo2-api-2.3-ec.jar:/usr/lib/hive/lib/slf4j-log4j12-1.6.1.jar:/usr/lib/hive/lib/velocity-1.5.jar:/usr/lib/hive/lib/commons-cli-1.2.jar:/usr/lib/hive/lib/commons-dbcp-1.4.jar:/usr/lib/hive/lib/hbase-0.90.1-cdh3u0-tests.jar:/usr/lib/hive/lib/commons-codec-1.3.jar:/usr/lib/hive/lib/datanucleus-core-2.0.3.jar:/usr/lib/hive/lib/commons-pool-1.5.4.jar:/usr/lib/hive/lib/zookeeper-3.3.1.jar:/usr/lib/hive/lib/datanucleus-enhancer-2.0.3.jar:/usr/lib/hive/lib/libthrift.jar:/usr/lib/hive/lib/derby.jar:/usr/lib/hive/lib/hive-metastore-0.7.0-cdh3u0.jar:/usr/lib/hive/lib/log4j-1.2.16.jar:/usr/lib/hive/lib/slf4j-api-1.6.1.jar:/usr/lib/hive/lib/ant-contrib-1.0b3.jar:/usr/lib/hive/lib/hive-hbase
oozie     1915     1  1 Dec07 ?        00:14:28 /usr/bin/java -Djava.util.logging.config.file=/var/lib/oozie/oozie-server/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Dderby.stream.error.file=/var/log/oozie/derby.log -Doozie.home.dir=/usr/lib/oozie -Doozie.config.dir=/etc/oozie -Doozie.log.dir=/var/log/oozie -Doozie.data.dir=/var/lib/oozie -Doozie.config.file=oozie-site.xml -Doozie.log4j.file=oozie-log4j.properties -Doozie.log4j.reload=10 -Doozie.http.hostname=cloudera-vm -Doozie.http.port=11000 -Doozie.base.url=http://cloudera-vm:11000/oozie -Djava.endorsed.dirs=/usr/lib/oozie/oozie-server/endorsed -classpath /usr/lib
```

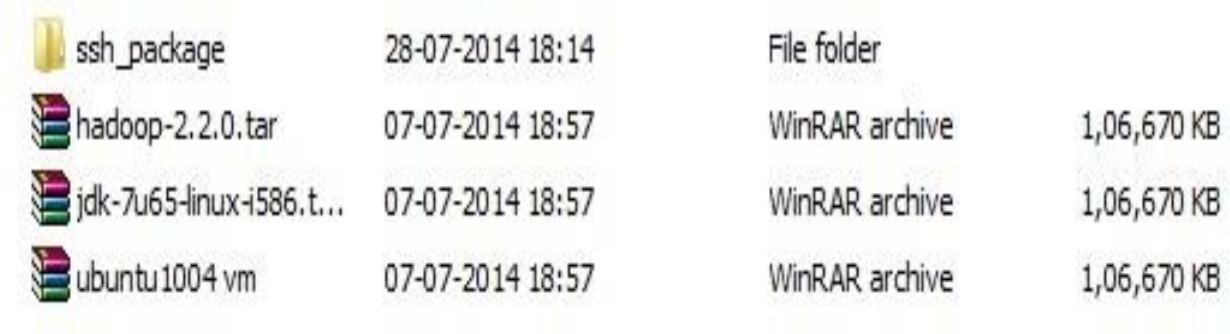
Hadoop Installation from tar ball. (Slides 47 – 66)

- Linux/Ubuntu environment
- SSH installation
- Java Installation
- Hadoop installation and file configuration

SSH Installation

SSH Installation

- Install Ubuntu 10.04 using vm player after extracting it from zip file.



- Before logging in, choose keyboard as USA.

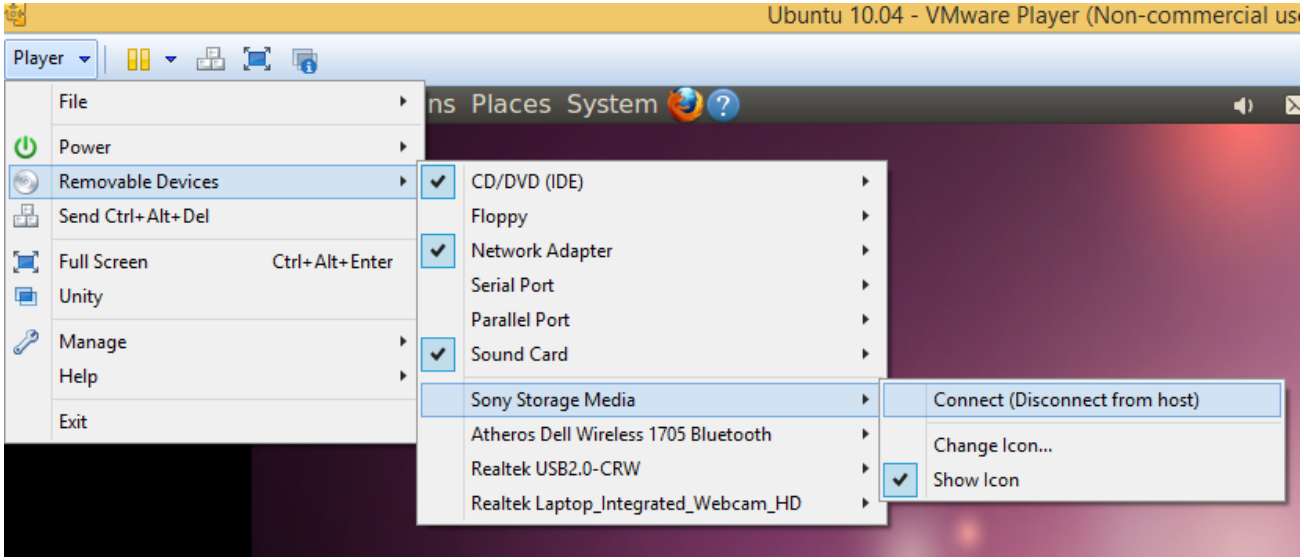
Username : user

Password : password

- Copy “ssh_package” to your ubuntu desktop using pendrive.

(Note: you won’t be able to use WINscp before installing ssh)

SSH Installation (Cont.)



- Open a new terminal (Cntrl + Alt + t) and run following commands.

cd /home/user/Desktop/ssh_package

sudo dpkg -i openssh-client_1%3a5.3p1-3ubuntu7_i386.deb

sudo dpkg -i openssh-server_1%3a5.3p1-3ubuntu7_i386.deb

(Note : first install client , then install server)

SSH Installation (Cont.)

```
user@ubuntu:~/Desktop/ssh_package$ sudo dpkg -i openssh-client_1%3a5.3p1-3ubuntu
7_i386.deb
[sudo] password for user:
(Reading database ... 122725 files and directories currently installed.)
Preparing to replace openssh-client 1:5.3p1-3ubuntu3 (using openssh-client_1%3a5
.3p1-3ubuntu7_i386.deb) ...
Unpacking replacement openssh-client ...
Setting up openssh-client (1:5.3p1-3ubuntu7) ...

Processing triggers for man-db ...
user@ubuntu:~/Desktop/ssh_package$ sudo dpkg -i openssh-server_1%3a5.3p1-3ubuntu
7_i386.deb
Selecting previously deselected package openssh-server.
(Reading database ... 122725 files and directories currently installed.)
Unpacking openssh-server (from openssh-server_1%3a5.3p1-3ubuntu7_i386.deb) ...
Setting up openssh-server (1:5.3p1-3ubuntu7) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
ssh start/running, process 1750

Processing triggers for ureadahead ...
ureadahead will be reprofiled on next reboot
Processing triggers for ufw ...
Processing triggers for man-db ...
```

- Now connect with Winscp and transfer hadoop and java setup to your ubuntu environment.

Java Installation

Java Installation

- Download jdk for linux

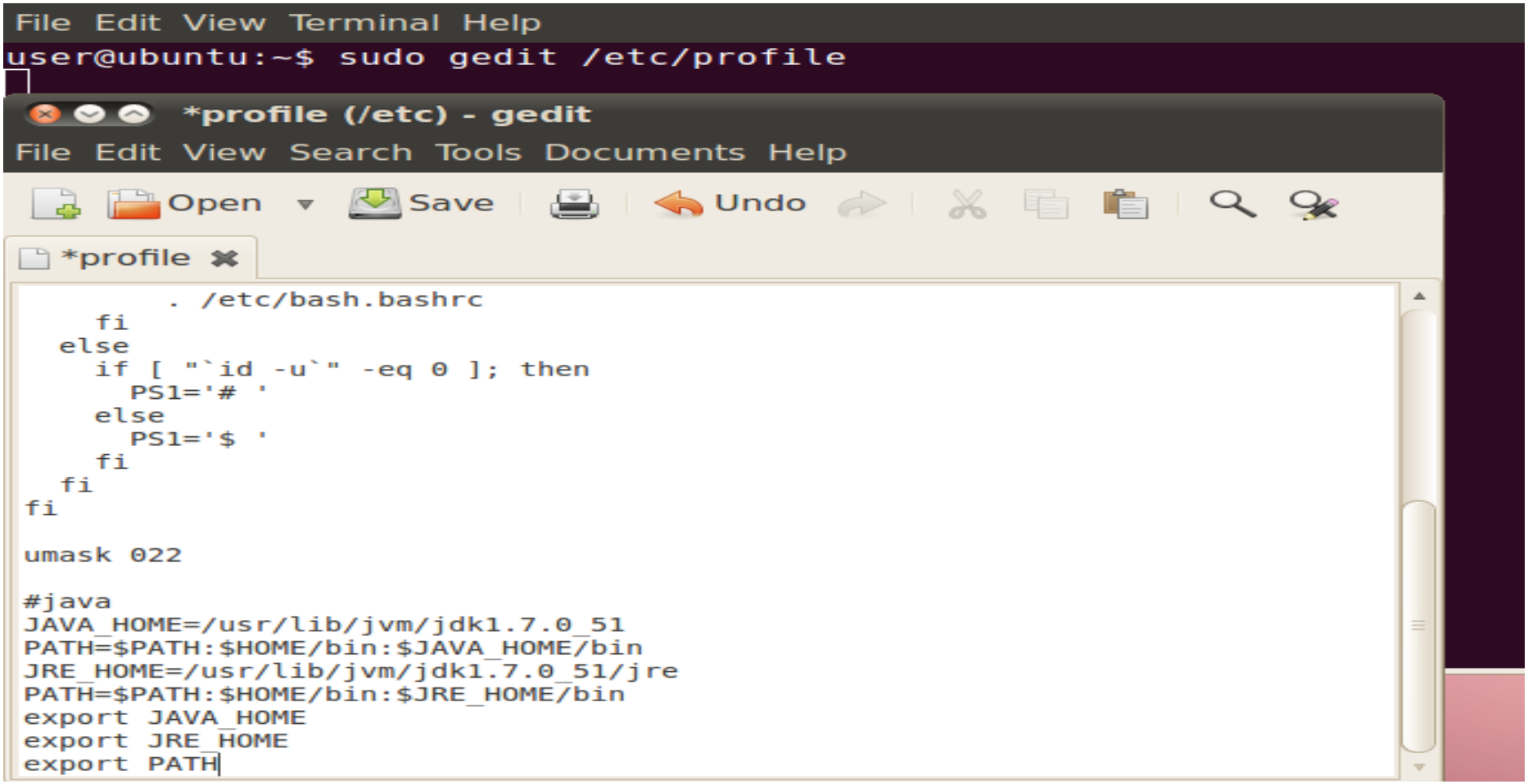
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

- Create a directory in root mode and install jdk from tar file.

```
user@ubuntu:~$ sudo su
root@ubuntu:/home/user# mkdir /usr/lib/jvm
root@ubuntu:/home/user# cp ./Desktop/jdk-7u51-linux-i586.tar.gz /usr/lib/jvm/
root@ubuntu:/home/user# cd /usr/lib/jvm/
root@ubuntu:/usr/lib/jvm# sudo tar xzvf jdk-7u51-linux-i586.tar.gz
```

Java Installation (Cont.)

- Restart your terminal and append append your /etc/profile



Java Installation (Cont.)

- Run following commands

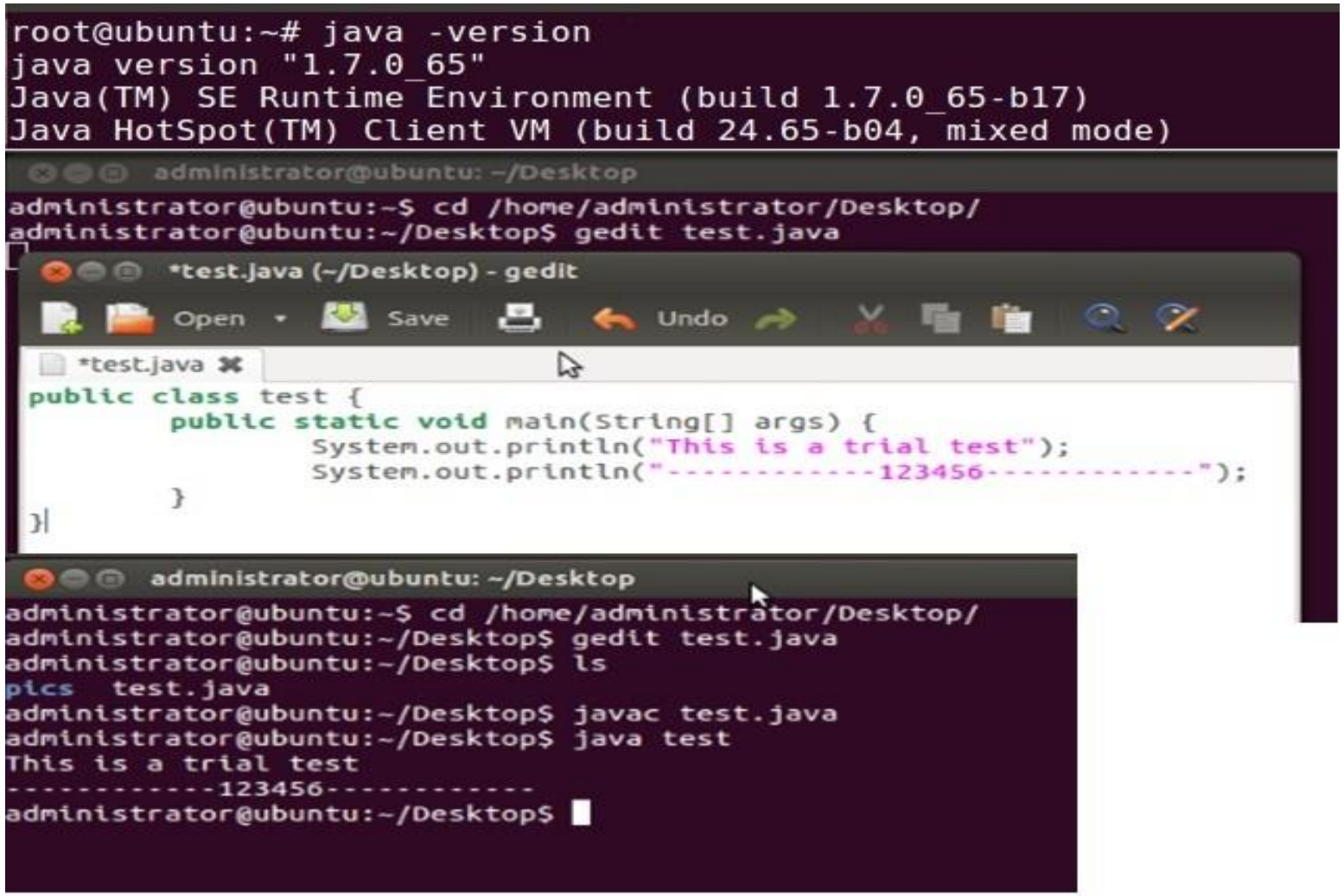
```
user@ubuntu:~$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.7.0_51/jre/bin/java" 1
user@ubuntu:~$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/lib/jvm/jdk1.7.0_51/bin/javac" 1
user@ubuntu:~$ sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/lib/jvm/jdk1.7.0_51/bin/javaws" 1
user@ubuntu:~$ sudo update-alternatives --install "/usr/bin/jps" "jps" "/usr/lib/jvm/jdk1.7.0_51/bin/jps" 1
user@ubuntu:~$
user@ubuntu:~$
user@ubuntu:~$ sudo update-alternatives --set java /usr/lib/jvm/jdk1.7.0_51/jre/bin/java
user@ubuntu:~$ sudo update-alternatives --set javac /usr/lib/jvm/jdk1.7.0_51/bin/javac
user@ubuntu:~$ sudo update-alternatives --set javaws /usr/lib/jvm/jdk1.7.0_51/bin/javaws
user@ubuntu:~$
```

- sudo update-alternatives --install "/usr/bin/java" "java“ "/usr/lib/jvm/jdk1.7.0_<java version>/jre/bin/java" 1
- sudo update-alternatives --install "/usr/bin/javac" "javac“ "/usr/lib/jvm/jdk1.7.0_<java version>/bin/javac" 1
- sudo update-alternatives --install "/usr/bin/javaws" "javaws“ "/usr/lib/jvm/jdk1.7.0_<java version>/bin/javaws" 1
- sudo update-alternatives --install "/usr/bin/jps" "jps“ "/usr/lib/jvm/jdk1.7.0_<java version>/bin/jps" 1

- sudo update-alternatives --set java /usr/lib/jvm/jdk1.7.0_<java version>/jre/bin/java
- sudo update-alternatives --set javac /usr/lib/jvm/jdk1.7.0_<java version>/bin/javac
- sudo update-alternatives --set javaws /usr/lib/jvm/jdk1.7.0_<java version>/bin/javaws

Java Installation (Cont.)

- Start and new terminal and check java installation by running a program.



Hadoop Installation

Hadoop Installation

- After proper installation of Vmware and Ubuntu and setting up java …go a/c to the screenshots as given below.

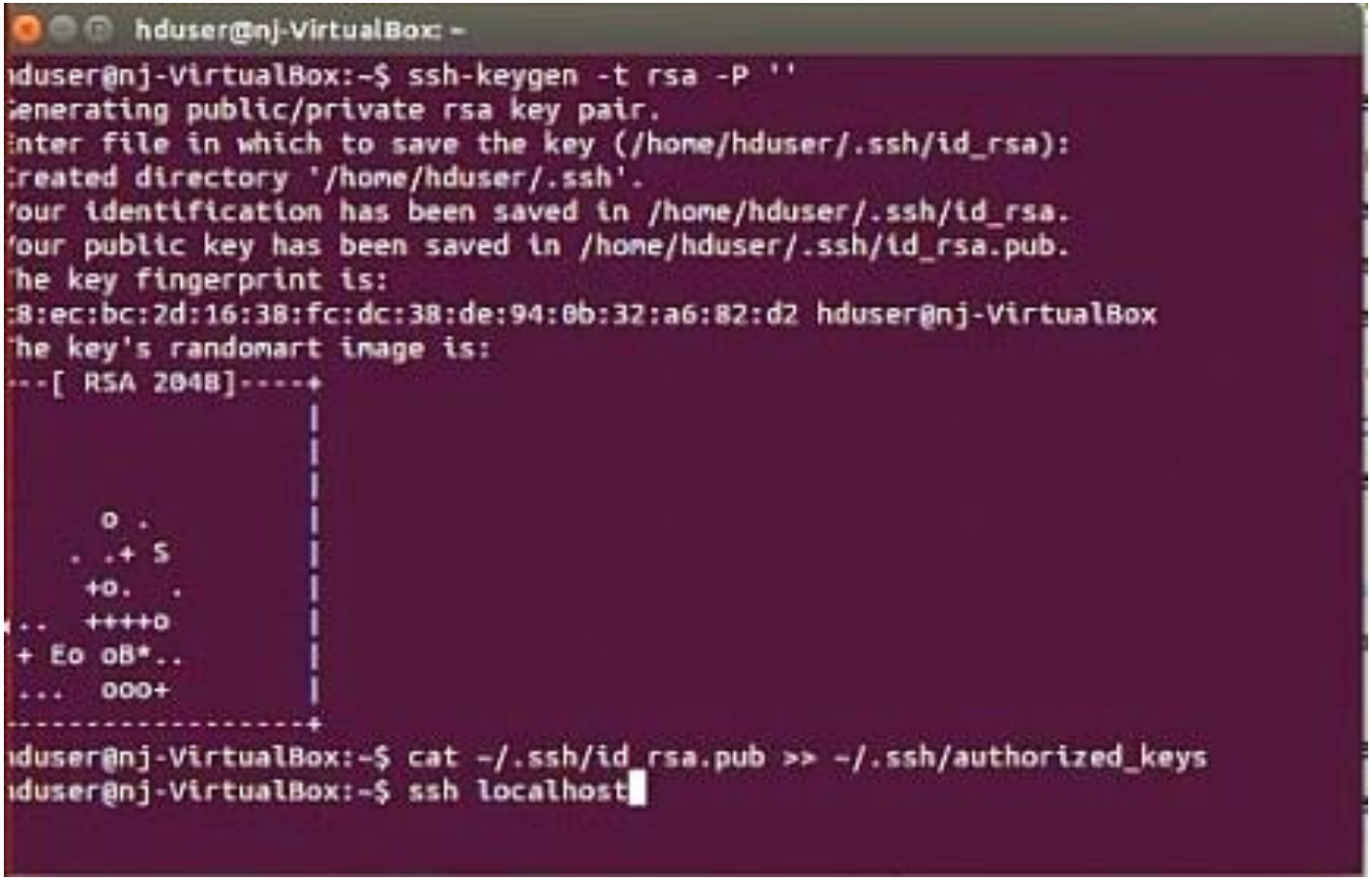
```
Oracle JRE 7 browser plugin installed

Setting up gsfonts-x11 (0.21) ...

user@ubuntu:~$ java -version
java version "1.7.0_60"
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
Java HotSpot(TM) Client VM (build 24.60-b09, mixed mode)
user@ubuntu:~$
```

- Adding a new user for hadoop and relogin into it.

```
user@ubuntu: ~
File Edit View Terminal Help

user@ubuntu:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
user@ubuntu:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
user@ubuntu:~$ sudo adduser hduser sudo
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Hadoop Installation (Cont.)

- After adding new user in group hadoop type the following commands for configuring ssh & taking ownership.



Hadoop Installation (Cont.)

Extract the hadoop 2.2.0 and copy it to a location.

Hadoop Installation (Cont.)

- Changing ownership of hadoop folder

```
hadoop-2.2.0/include/SerialUtils.hh
hadoop-2.2.0/include/TemplateFactory.hh
hadoop-2.2.0/include/hdfs.h
hadoop-2.2.0/include/StringUtils.hh
hduser@nj-VirtualBox:~/Downloads$ cd /usr/local
hduser@nj-VirtualBox:/usr/local$ sudo mv hadoop-2.2.0 hadoop
hduser@nj-VirtualBox:/usr/local$ ls
bin  etc  games  hadoop  include  lib  man  sbin  share  src
hduser@nj-VirtualBox:/usr/local$ ls -l
total 36
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 bin
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 etc
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 games
drwxr-xr-x 9 67974 users 4096 Oct  7  2013 hadoop
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 include
drwxr-xr-x 4 root  root  4096 Apr 17 11:25 lib
lrwxrwxrwx 1 root  root     9 Jun 13 17:37 man -> share/man
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 sbin
drwxr-xr-x 7 root  root  4096 Apr 17 11:26 share
drwxr-xr-x 2 root  root  4096 Apr 17 11:21 src
hduser@nj-VirtualBox:/usr/local$ sudo chown -R hduser:hadoop hadoop
```

- Ownership can be checked by typing “ ls -lrt “

Hadoop Installation (Cont.)

- All the commands are provided in a text file for ease.After copying the hadoop tar file extract it according to the commands given.

- Ownership can be checked by typing " ls -lrt "

Changing hadoop configuration files

All the files are in /usr/local/hadoop/etc/hadoop/

In **hadoop-env.sh** add,

export JAVA_HOME=/usr/lib/jvm/jdk/<java version should be mentioned here>

In **core-site.xml** add following between configuration tabs,

<property>

<name>fs.default.name</name>

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

</property>

In **yarn-site.xml** add following between configuration tabs,

<property>

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

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

Changing hadoop configuration files (Cont.)

In **mapred-site.xml copy the mapred-site.xml.template and rename it to mapred-site.xml &** add following between configuration tabs,

```
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
```

In **hdfs-site.xml** add following between configuration tabs,

```
<property>
        <name>dfs.replication</name>
        <value>1</value>
</property>
<property>
        <name>dfs.namenode.name.dir</name>
         <value>file:/home/hduser/mydata/hdfs/namenode</value>
 </property>
<property>
        <name>dfs.datanode.data.dir</name>
         <value>file:/home/hduser/mydata/hdfs/datanode</value>
 </property>
```

Changing hadoop configuration files (Cont.)

Finally update your $HOME/.bahsrc

cd $HOME

vi .bashrc

Append following lines in the end and save and exit

#Hadoop variables

export JAVA_HOME=/usr/lib/jvm/jdk/<your java version>

export HADOOP_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin

export HADOOP_MAPRED_HOME=$HADOOP_INSTALL

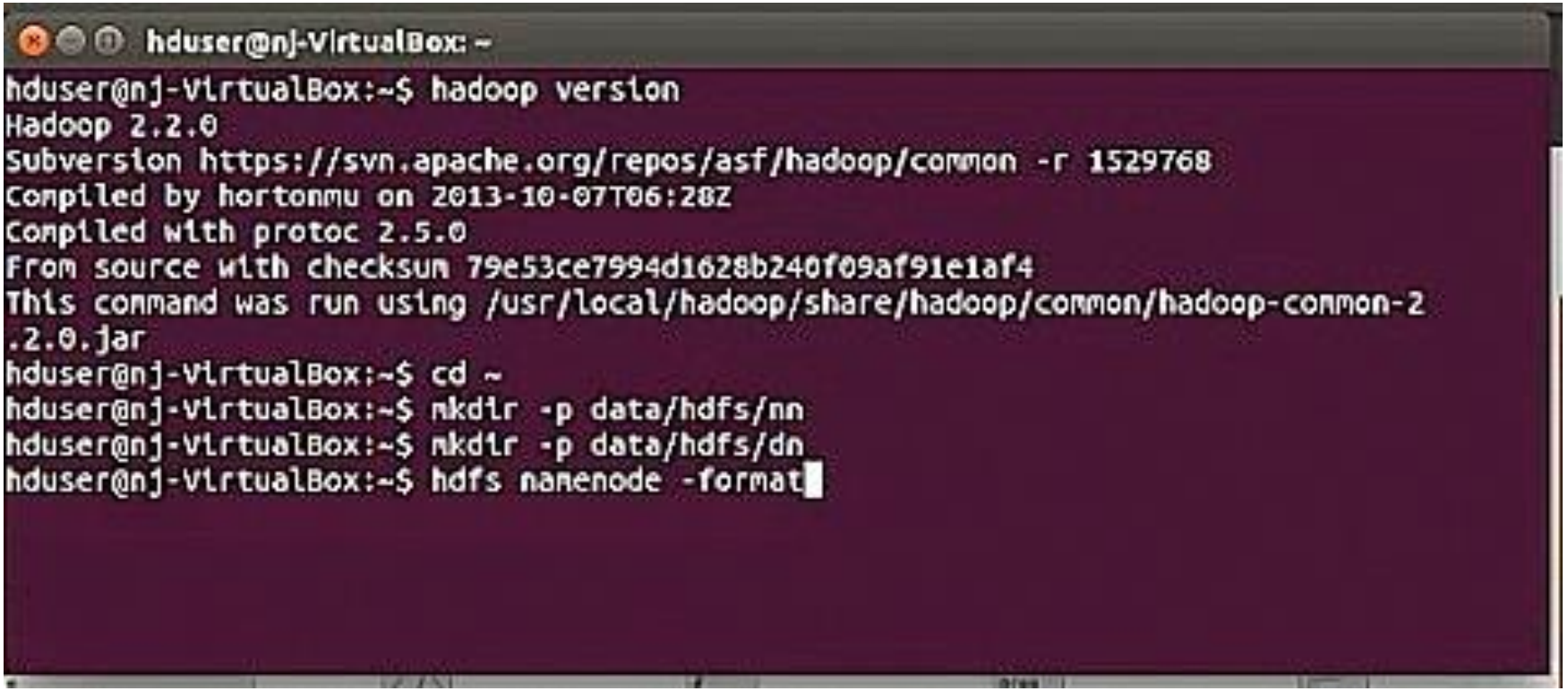
export HADOOP_COMMON_HOME=$HADOOP_INSTALL

export HADOOP_HDFS_HOME=$HADOOP_INSTALL

export YARN_HOME=$HADOOP_INSTALL

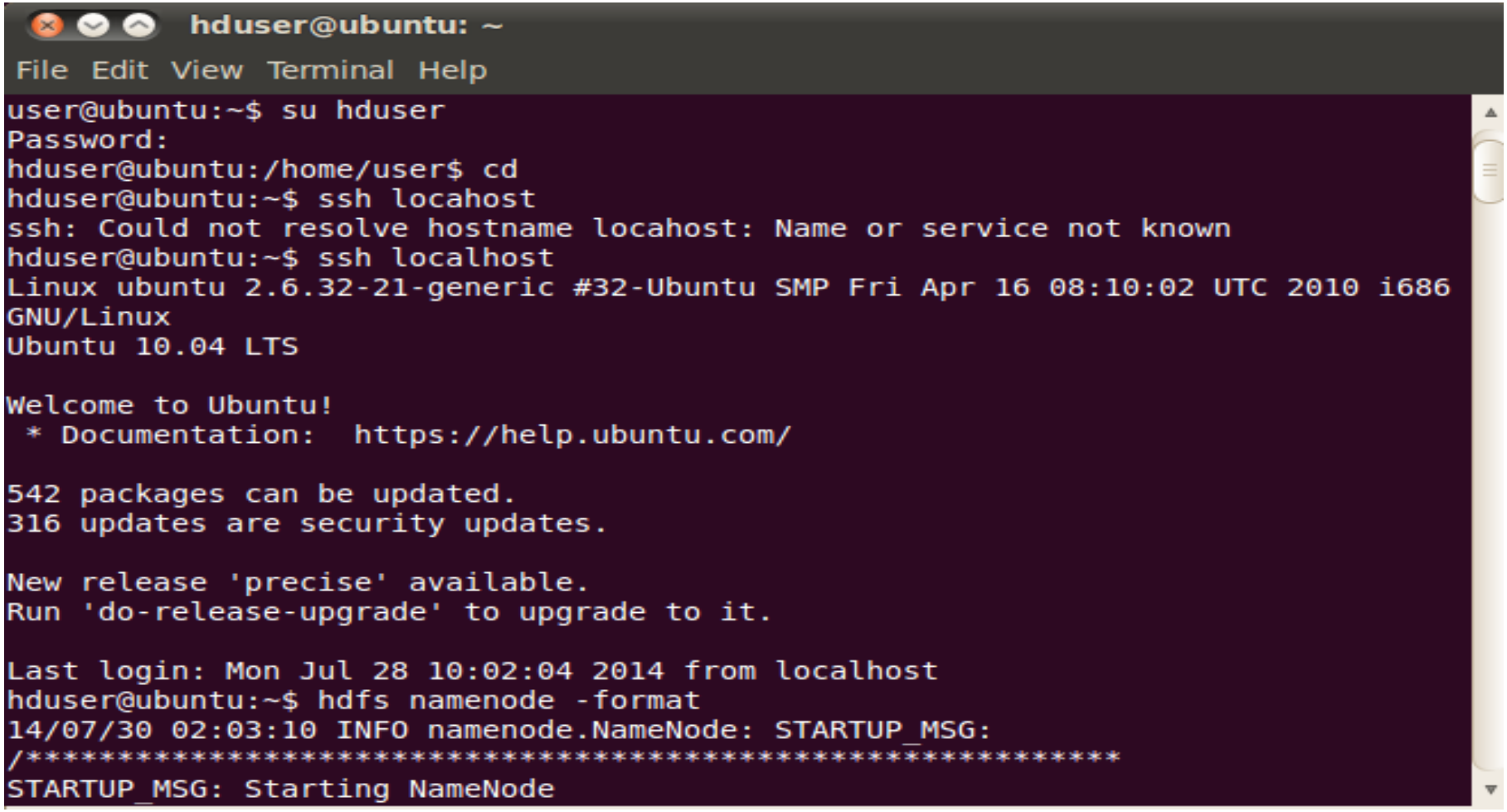
Changing hadoop configuration files (Cont.)

Checking hadoop version and making directories for name node and datanode.



Starting Hadoop

- Format namenode and start all daemons,



Starting Hadoop (Cont.)

- Restart all your hadoop-daemons and check daemons through “jps”.
- Starting hadoop and yarn services .
- **start-dfs.sh**

```
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-na
menode-ubuntu.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-da
tanode-ubuntu.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hd
user-secondarynamenode-ubuntu.out
hduser@ubuntu:~$ jps
4179 SecondaryNameNode
3966 DataNode
4309 Jps
3817 NameNode
hduser@ubuntu:~$
```

Starting yarn daemons

- **start-yarn.sh :-** for starting yarn services.

```
hduser@ubuntu:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resource
manager-ubuntu.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-n
odemanager-ubuntu.out
hduser@ubuntu:~$ jps
1670 NameNode
1819 DataNode
2401 Jps
2008 SecondaryNameNode
2367 NodeManager
2212 ResourceManager
hduser@ubuntu:~$
```

NOTE:- to stop the daemons just omit start and write stop

Running MR program

- Running put command

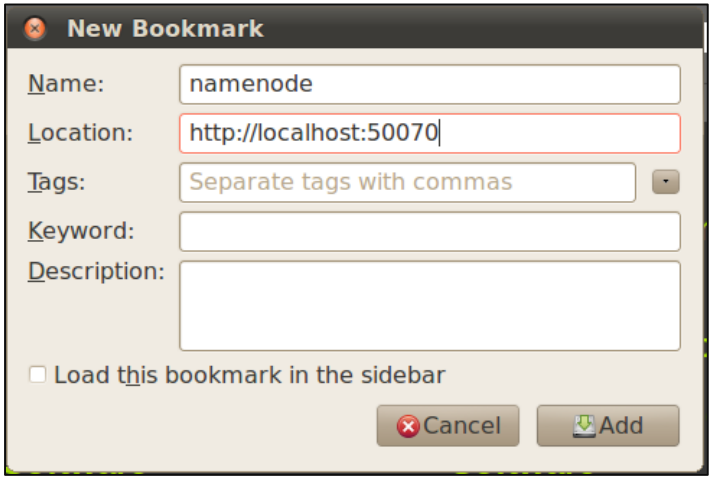
```
hadoop_user@ubuntu:~$ hadoop fs -mkdir /user/trial
Warning: $HADOOP_HOME is deprecated.

hadoop_user@ubuntu:~$ hadoop fs -put /home/user/Desktop/Data_File.txt /user/trial/
Warning: $HADOOP_HOME is deprecated.

hadoop_user@ubuntu:~$ hadoop fs -ls /user/trial
Warning: $HADOOP_HOME is deprecated.

Found 1 items
-rw-r--r--   1 hadoop_user supergroup    1198702 2014-03-26 04:08 /user/trial/Data_File.txt
hadoop_user@ubuntu:~$
```

- Running MR program

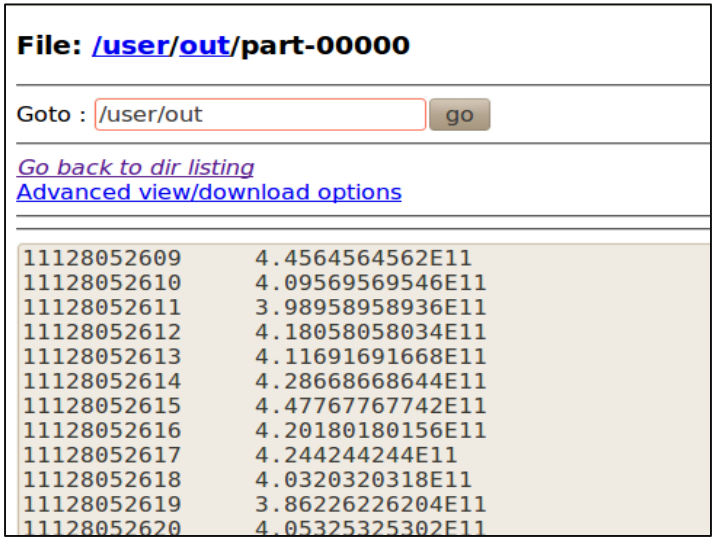
```
hadoop_user@ubuntu:~$ cd /home/user/Desktop/
hadoop_user@ubuntu:/home/user/Desktop$ hadoop jar Subscriber.jar aircel_runner /user/trial/Data_File.txt /user/out
```

Accesing HTTP interface

Create a new bookmark

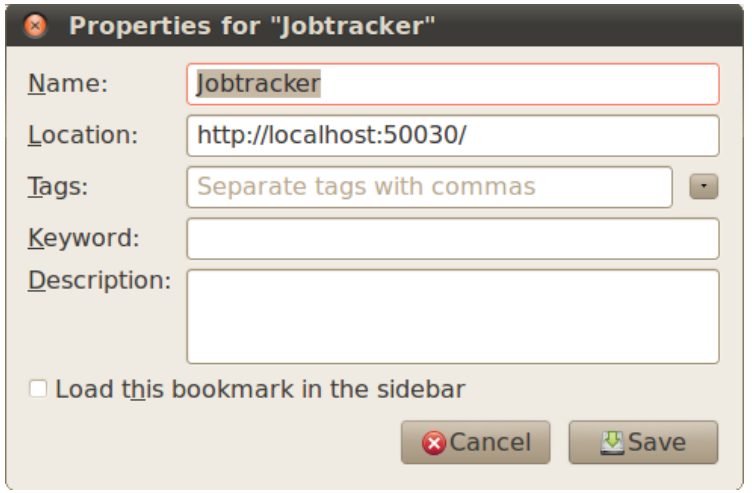


Viewing results of MR job



Accesing HTTP interface

Create a new bookmark



Pig installation

Installation

Pre Requisites:

- Working Hadoop (HDFS and Mapreduce) environment
- Pig tarball

Installation

- Create a directory and untar the pig tarball
- Give the ownership of that folder to “hadoop_user” user

```
hadoop_user@ubuntu:~$ sudo su
root@ubuntu:/home/hadoop_user# mkdir /usr/lib/pig/
root@ubuntu:/home/hadoop_user# cp /home/user/Desktop/pig-0.11.1.tar.gz /usr/lib/pig/
root@ubuntu:/home/hadoop_user# cd /usr/lib/pig/
root@ubuntu:/usr/lib/pig# tar xzvf pig-0.11.1.tar.gz
```

```
hadoop_user@ubuntu:~$ ls -lrt /usr/lib/pig
total 57068
drwxr-xr-x 15 root root     4096 2013-03-22 02:20 pig-0.11.1
-rw-r--r--  1 root root 58433159 2014-03-31 04:34 pig-0.11.1.tar.gz
hadoop_user@ubuntu:~$ sudo chown -R hadoop_user:hadoop /usr/lib/pig/
hadoop_user@ubuntu:~$ ls -lrt /usr/lib/pig
total 57068
drwxr-xr-x 15 hadoop_user hadoop     4096 2013-03-22 02:20 pig-0.11.1
-rw-r--r--  1 hadoop_user hadoop 58433159 2014-03-31 04:34 pig-0.11.1.tar.gz
hadoop_user@ubuntu:~$
hadoop_user@ubuntu:~$
hadoop_user@ubuntu:~$
```

Installation

- Update the .bashrc file.

vi ~/.bashrc

export PIG_HOME=/usr/lib/pig/pig-0.11.1/

export PATH=$PATH:$PIG_HOME/bin

```
export HADOOP_HOME=/usr/lib/hadoop/hadoop-1.2.1/
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_51
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:HADOOP
export PIG_HOME=/usr/lib/pig/pig-0.11.1/
export PATH=$PATH:$PIG_HOME/bin
```

Checking

Checking

- Upload a file from your local filesystem to your HDFS.
- Load that file using PigStorage() and dump its contents on console

```
grunt> cd /user/pig
grunt> pwd
hdfs://localhost:54310/user/pig
grunt> ls
hdfs://localhost:54310/user/pig/abc<r 1>        74
grunt> A = LOAD '/user/pig/abc' using PigStorage(' ') as (num:int,name:chararray,age:int,sal:double);
grunt> dump A;
```

```
Job DAG:
job_201403310949_0001


2014-03-31 09:56:27,168 [main] INFO  org.apache.pig.backend.hadoop.exe
 Success!
2014-03-31 09:56:27,176 [main] INFO  org.apache.pig.data.SchemaTupleBa
ll not generate code.
2014-03-31 09:56:27,180 [main] INFO  org.apache.hadoop.mapreduce.lib.i
cess : 1
2014-03-31 09:56:27,180 [main] INFO  org.apache.pig.backend.hadoop.exe
s to process : 1
(1,qwerty,29,12312.2321312)
(2,asdfgh,23,12345.23213)
(3,zxcvbn,22,54321.12321)
grunt>
```

Closing Single node-cluster

Closing

- First stop all the java processes using *"stop-all.sh".*
- Then logout from hadoop dedicated user by *"exit".*

Note: Improper closing or exiting may corrupt the HDFS system. On restarting, it may open in safe mode and then unable to load all the data on data nodes. Thus you may need to reformat namenode causing all the data to get lost

```
hduser@ubuntu:~$ stop-all.sh
This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
no proxyserver to stop
hduser@ubuntu:~$ exit
logout
Connection to localhost closed.
hduser@ubuntu:~$
```

Safemode

You may have to manually exit safe mode using following command

Hadoop dfsadmin –safemode leave

```
hadoop_user@ubuntu:~$ hadoop dfsadmin -safemode leave
Warning: $HADOOP_HOME is deprecated.

Safe mode is OFF
hadoop_user@ubuntu:~$
```

This may cause some datablocks to corrupt and cause loss of data on HDFS.

Configurations

- Each component is configured using an XML file.
- NOTE: In HADOOP version 2.2 all the confs are done in a single file hadoop-site.xml
- In newer versions from 0.20 onwards this file is split into 3 files (Properties names are same only the files have been split)
- core-site.xml -> Core properties go here
- hdfs-site.xml -> HDFS Properties go here
- mapred-site.xml -> Map reduce properties are configured here
- Since we are using CDH4 it uses HADOOP version 2.0.0

Modes

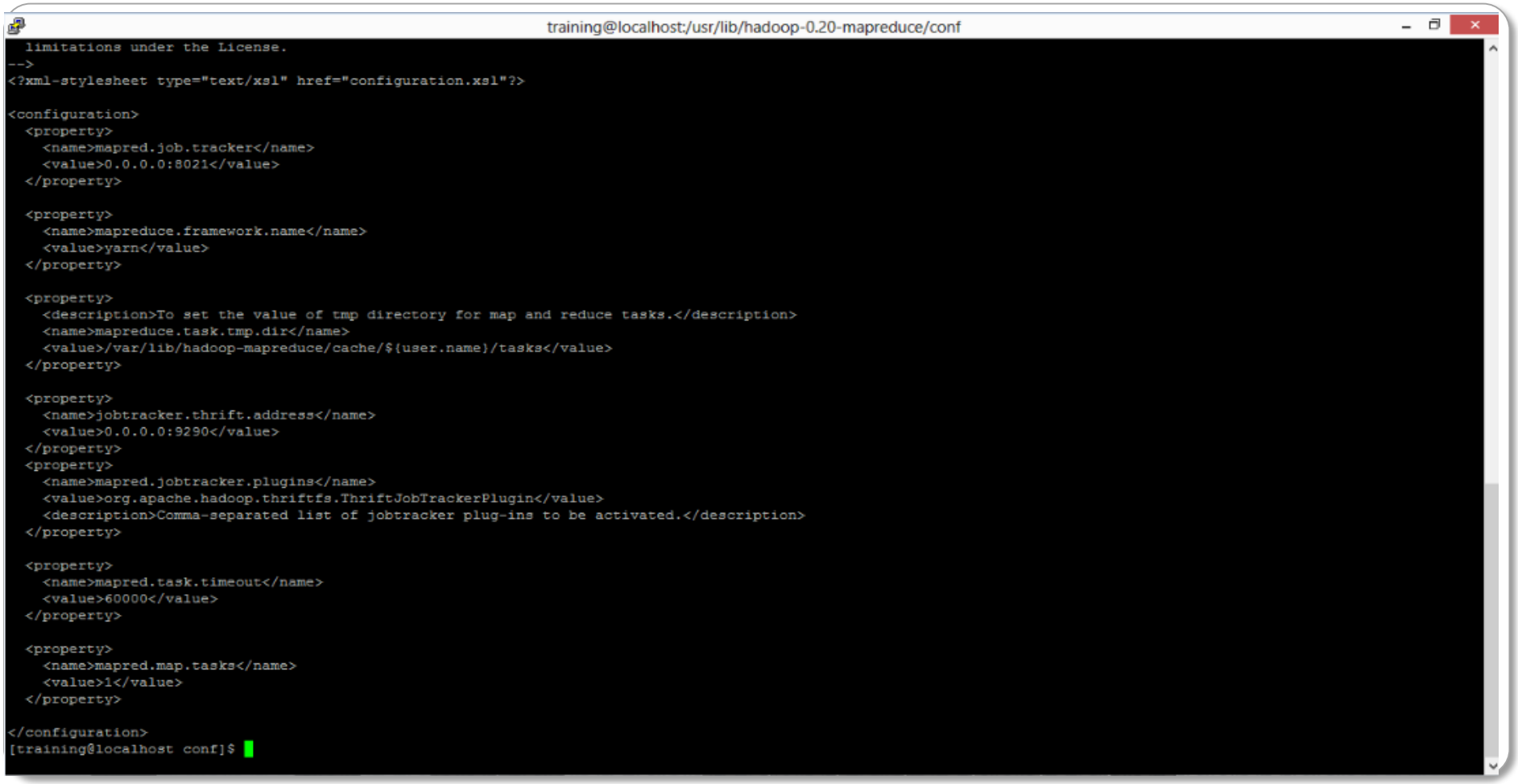
- Standalone mode-> No daemons run, everything runs in single process. Suitable for debugging MR programs
- Pseudo Distributed -> All HADOOP processes run on local machine, simulating small cluster of 1 node
- Fully Distributed -> HADOOP runs on a cluster of machines.

Properties

Component	Property	Standalone	Pseudo-distributed	Fully distributed
Core	fs.default.name	file:///(default)	hdfs://localhost/	hdfs://*namenode*/
HDFS	dfs.replication	N/A	1	3 (default)

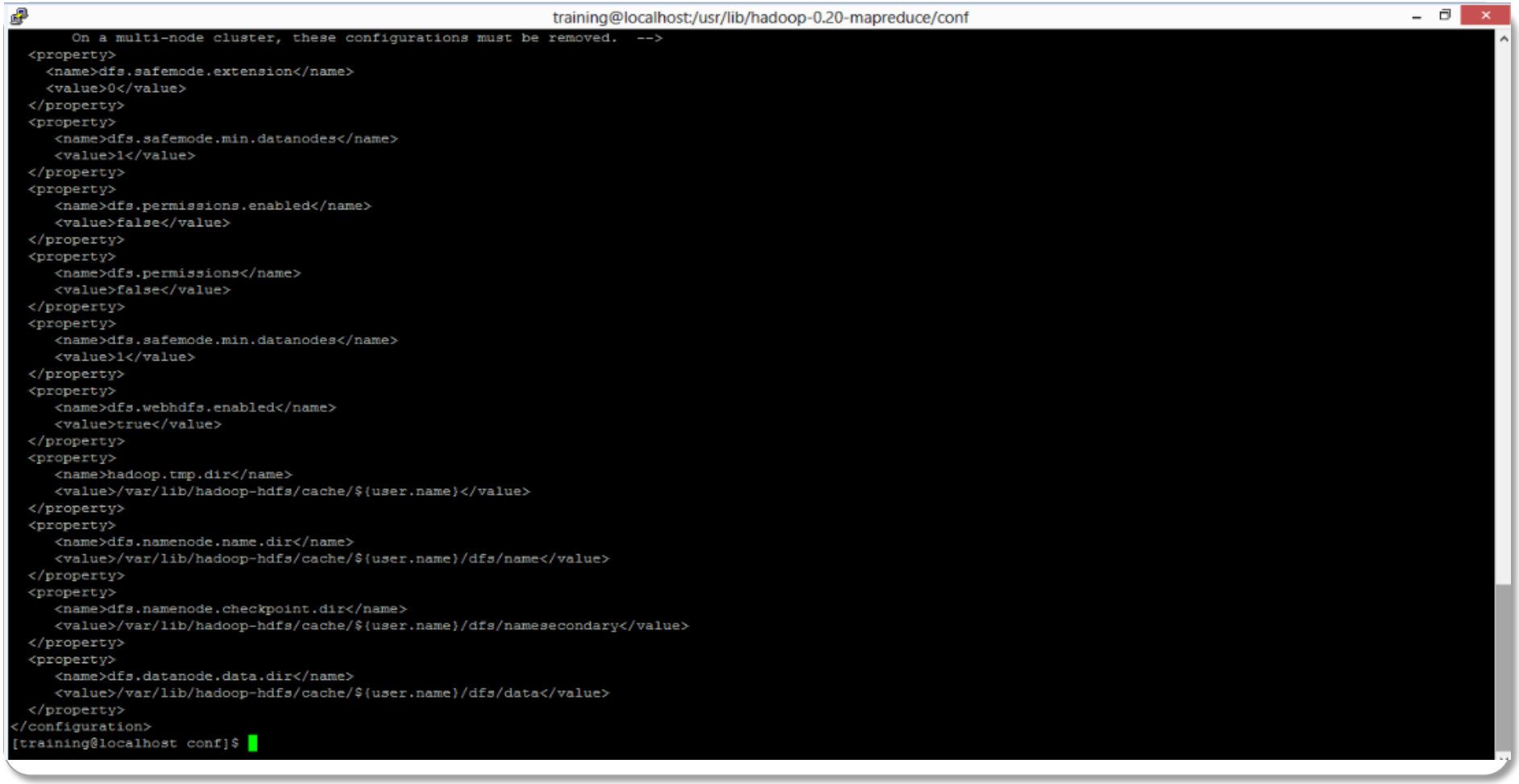
mapred-site.xml

- mapred-site.xml location /usr/local/hadoop/etc/hadoop/



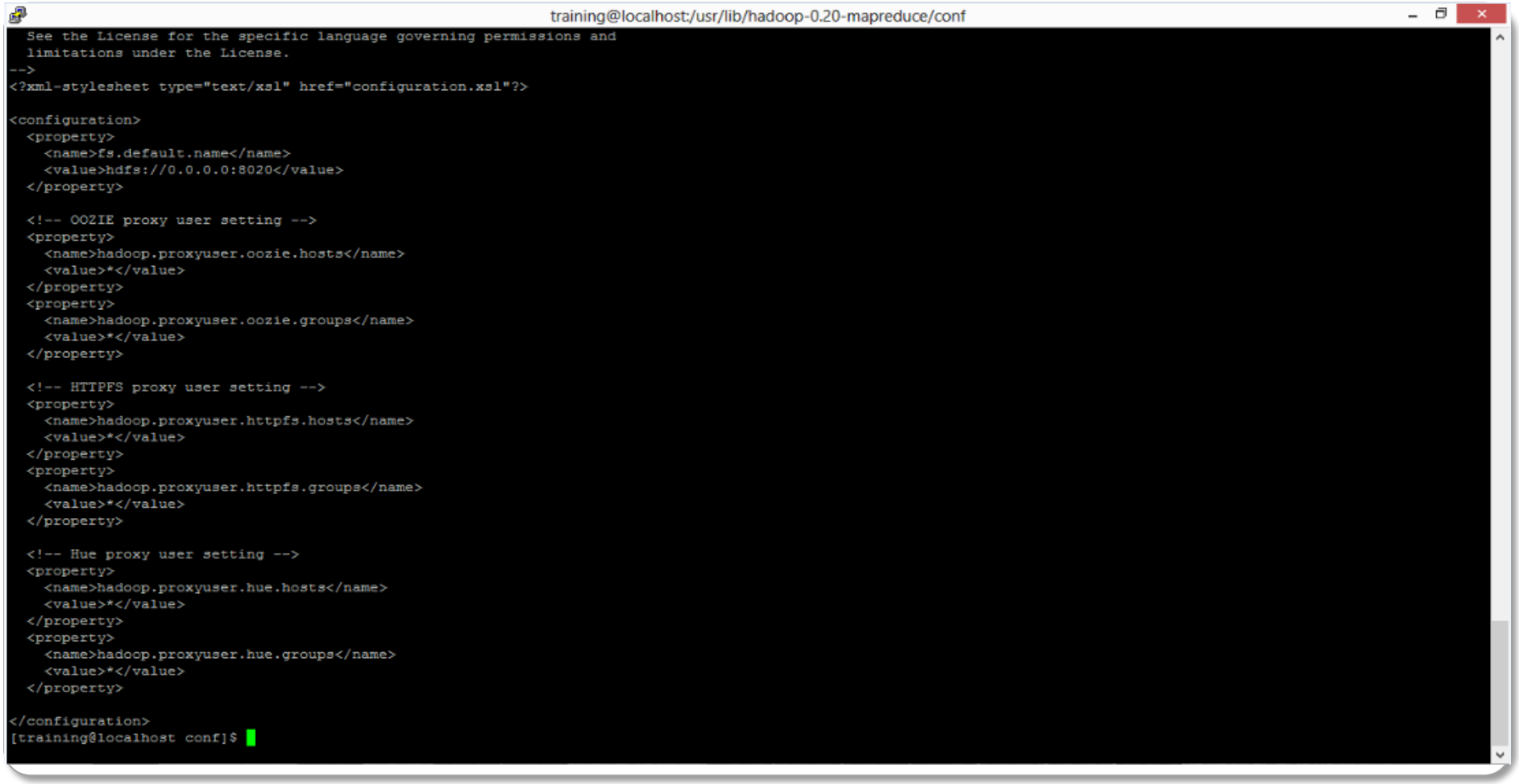
hdfs-site.xml

- hdfs-site.xml location /usr/local/hadoop/etc/hadoop/



core-site.xml

- core-site.xml location /usr/local/hadoop/etc/hadoop/



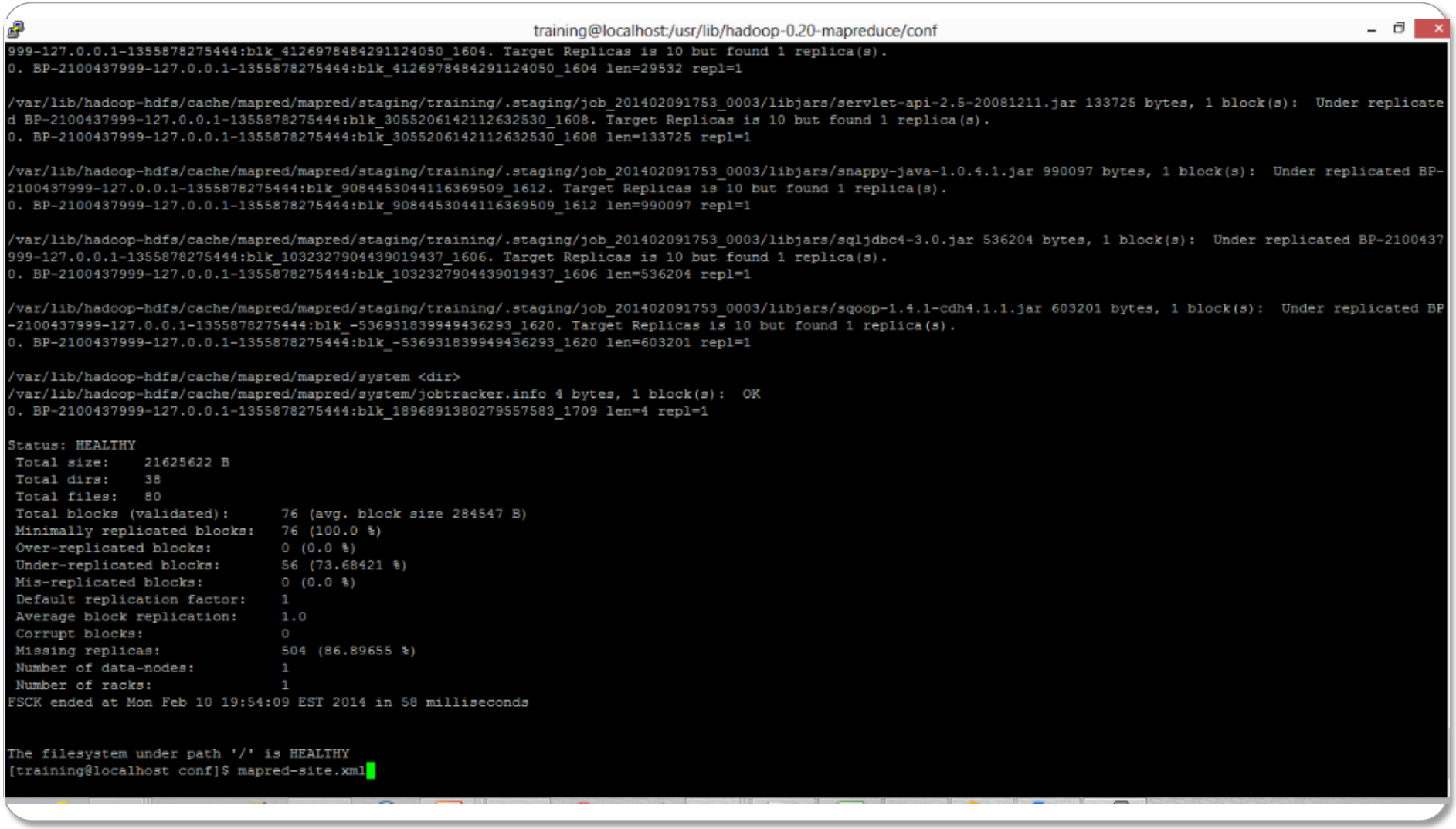
VM Configurations

- The VM provided is preconfigured with these properties, the properties can be verified by these files present in location /usr/lib/hadoop/conf

VM Configurations

- After checking the configurations, to check the status and health of HDFS do
- HADOOP fsck / -files –blocks
- This command gives the blocks and metadata information of HDFS i.e., directory namespace information, size, block Id, permissions etc
- If the setup is successful status will be shown HEALTHY

Status HDFS


training@localhost:/usr/lib/hadoop-0.20-mapreduce/conf
999-127.0.0.1-1355878275444:blk_4126978484291124050_1604. Target Replicas is 10 but found 1 replica(s).
0. BP-2100437999-127.0.0.1-1355878275444:blk_4126978484291124050_1604 len=29532 repl=1
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_201402091753_0003/libjars/servlet-api-2.5-20081211.jar 133725 bytes, 1 block(s): Under replicate
d BP-2100437999-127.0.0.1-1355878275444:blk_3055206142112632530_1608. Target Replicas is 10 but found 1 replica(s).
0. BP-2100437999-127.0.0.1-1355878275444:blk_3055206142112632530_1608 len=133725 repl=1
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_201402091753_0003/libjars/snappy-java-1.0.4.1.jar 990097 bytes, 1 block(s): Under replicated BP-
2100437999-127.0.0.1-1355878275444:blk_9084453044116369509_1612. Target Replicas is 10 but found 1 replica(s).
0. BP-2100437999-127.0.0.1-1355878275444:blk_9084453044116369509_1612 len=990097 repl=1
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_201402091753_0003/libjars/sqljdbc4-3.0.jar 536204 bytes, 1 block(s): Under replicated BP-2100437
999-127.0.0.1-1355878275444:blk_1032327904439019437_1606. Target Replicas is 10 but found 1 replica(s).
0. BP-2100437999-127.0.0.1-1355878275444:blk_1032327904439019437_1606 len=536204 repl=1
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_201402091753_0003/libjars/sqoop-1.4.1-cdh4.1.1.jar 603201 bytes, 1 block(s): Under replicated BP
-2100437999-127.0.0.1-1355878275444:blk_-536931839949436293_1620. Target Replicas is 10 but found 1 replica(s).
0. BP-2100437999-127.0.0.1-1355878275444:blk_-536931839949436293_1620 len=603201 repl=1
/var/lib/hadoop-hdfs/cache/mapred/mapred/system <dir>
/var/lib/hadoop-hdfs/cache/mapred/mapred/system/jobtracker.info 4 bytes, 1 block(s): OK
0. BP-2100437999-127.0.0.1-1355878275444:blk_1896891380279557583_1709 len=4 repl=1
Status: HEALTHY
Total size: 21625622 B
Total dirs: 38
Total files: 80
Total blocks (validated): 76 (avg. block size 284547 B)
Minimally replicated blocks: 76 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 56 (73.68421 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 504 (86.89655 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Mon Feb 10 19:54:09 EST 2014 in 58 milliseconds
The filesystem under path '/' is HEALTHY
[training@localhost conf]$ mapred-site.xml

HDFS Interfaces / Clients

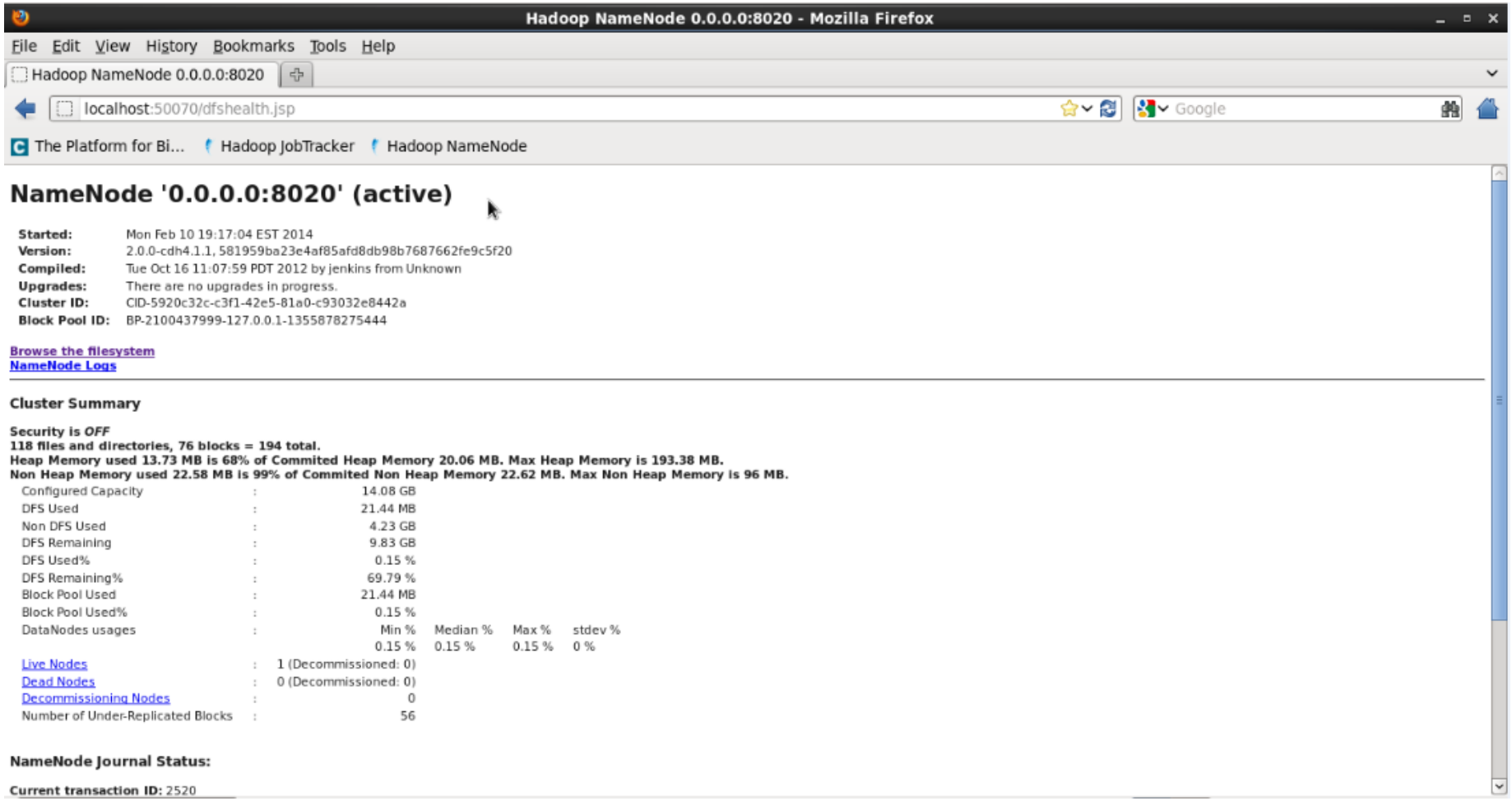
- HTTP Interfaces with browsers
- Using CLI (Command line) utility
- Programmatically using Java
- Using thrift APIs with other languages like C
- Most common use is with command line and Java APIs

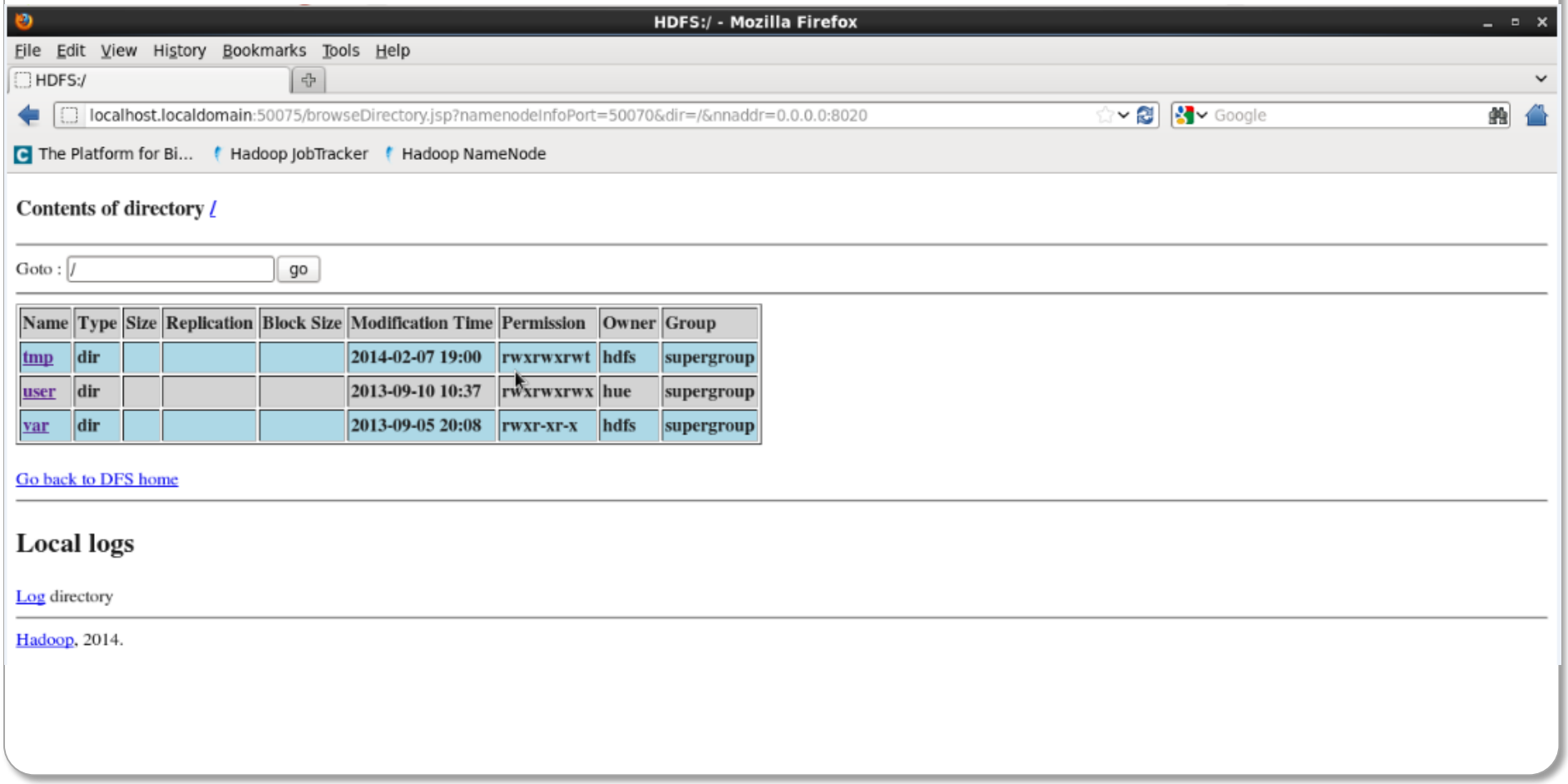
HTTP Interfaces

HTTP APIs

- HTTP: HDFS defines read only interface to browse directory listings and data over HTTP.
  HTTP web server runs on top of name node on port 50070, while file data is streamed from data nodes by their web servers running on 50075

NameNode GUI




HDFS:/ - Mozilla Firefox
File Edit View History Bookmarks Tools Help
HDFS:/
localhost.localdomain:50075/browseDirectory.jsp?namenodeInfoPort=50070&dir=/&nnaddr=0.0.0.0:8020
Google
The Platform for Bi...
Hadoop JobTracker
Hadoop NameNode
Contents of directory /
Goto : /
go
Name Type Size Replication Block Size Modification Time Permission Owner Group
tmp dir 2014-02-07 19:00 rwxrwxrwt hdfs supergroup
user dir 2013-09-10 10:37 rwxrwxrwx hue supergroup
var dir 2013-09-05 20:08 rwxr-xr-x hdfs supergroup
Go back to DFS home
Local logs
Log directory
Hadoop, 2014.

Command Line Utilities

HADOOP Shell commands

- http://hadoop.apache.org/docs/r0.18.3/hdfs_shell.html
- Some list of common commands are as :-

-**ls**- for listing files in hdfs.

Syntax :-- hadoop fs –ls

-**mkdir -** for creating a directory in hdfs.

Syntax :-- hadoop fs –mkdir

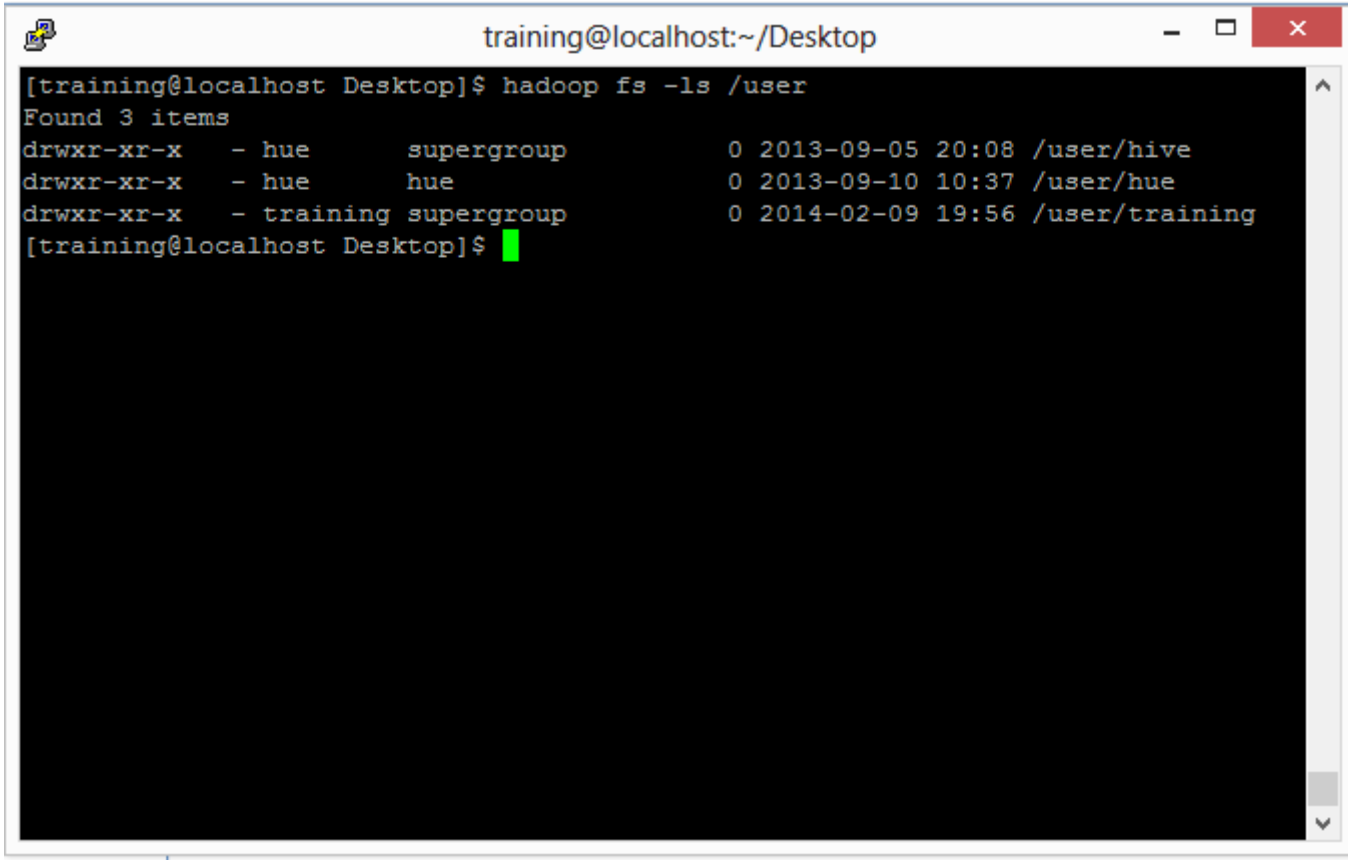
-**put –** for copying file to hdfs from local file system

Syntax :-- hadoop fs –put <source file dest.> <destn.>

Command to List Files and DIRS

- ls
- Usage: HADOOP fs -ls <args>
- For a file returns stat on the file with the following format:
  filename <number of replicas> filesize modification_date modification_time permissions userid groupid
- For a directory it returns list of its direct children as in unix. A directory is listed as:
  dirname <dir> modification_time modification_time permissions userid groupid

Example ls


training@localhost:~/Desktop
[training@localhost Desktop]$ hadoop fs -ls /user
Found 3 items
drwxr-xr-x   - hue   supergroup   0 2013-09-05 20:08 /user/hive
drwxr-xr-x   - hue   hue   0 2013-09-10 10:37 /user/hue
drwxr-xr-x   - training supergroup   0 2014-02-09 19:56 /user/training
[training@localhost Desktop]$

Practice Exercise

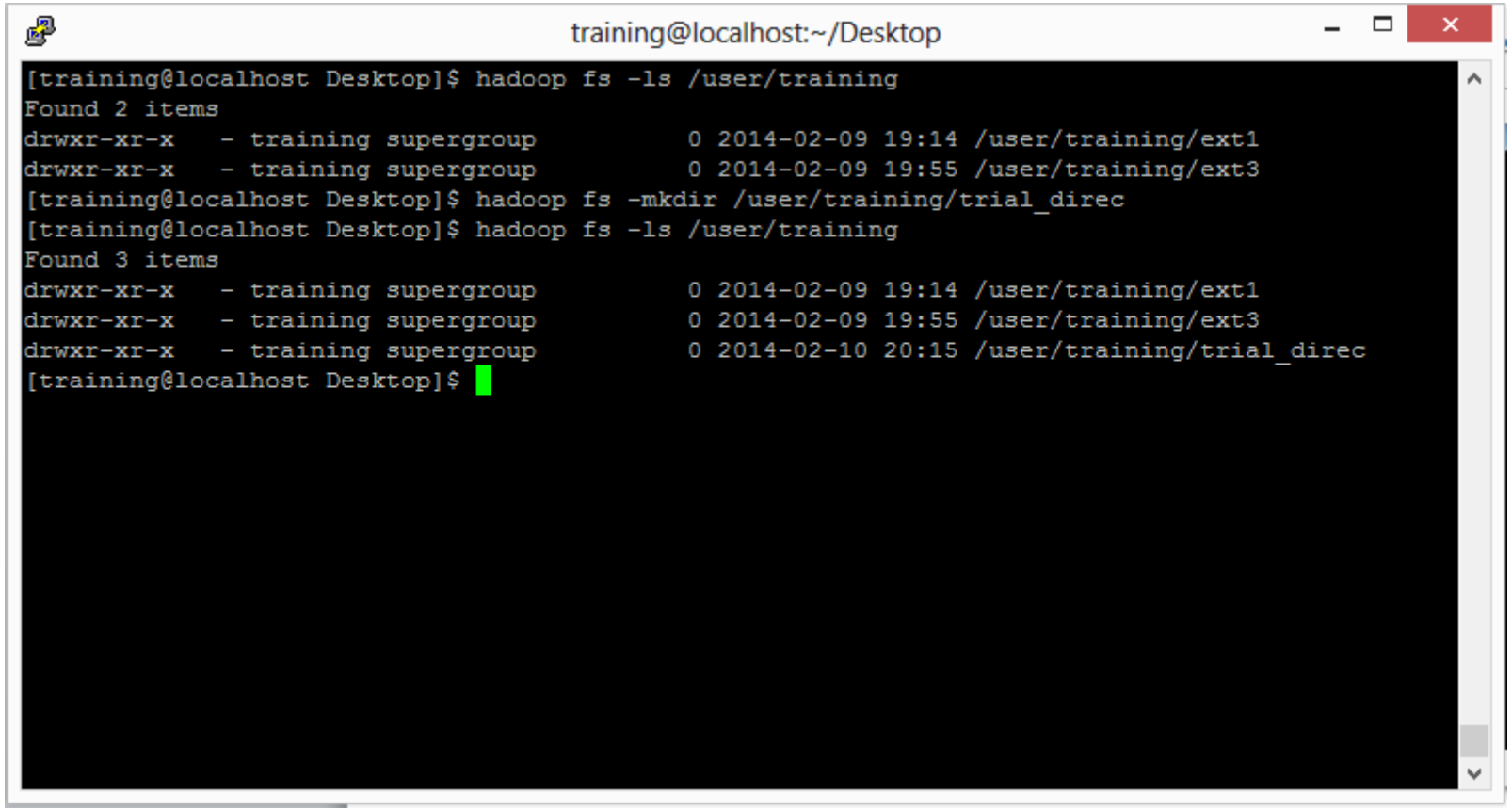
- Use command ls and see the output from command line and then do HADOOP ls

HADOOP mkdir

- mkdir – Takes path URI as argument and creates directory
- Usage: hadoop fs -mkdir <paths>

- Takes path uri's as argument and creates directories. The behavior is much like unix mkdir -p creating parent directories along the path
- Exit Code:
- Returns 0 on success and -1 on error

Example mkdir



Practice Exercise

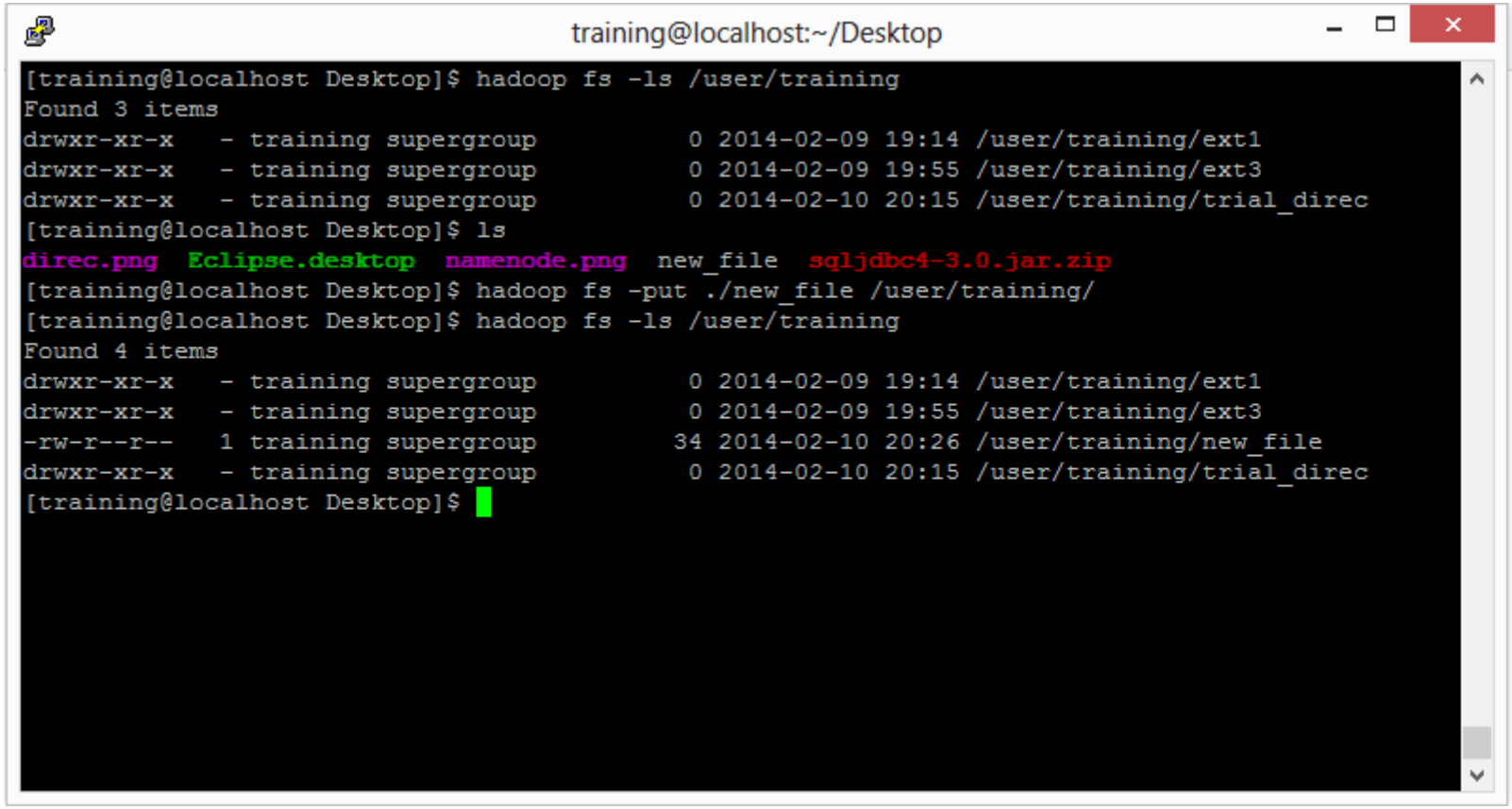
- Make a HADOOP directory using mkdir command with your name and then do ls for that directory

- mkdir <yourName>

HADOOP put

- Usage: hadoop fs -put <localsrc> ... <dst>
- Copy single src, or multiple srcs from local file system to the destination filesystem. Also reads input from stdin and writes to destination filesystem.

Example Put


training@localhost:~/Desktop
[training@localhost Desktop]$ hadoop fs -ls /user/training
Found 3 items
drwxr-xr-x   - training supergroup   0 2014-02-09 19:14 /user/training/ext1
drwxr-xr-x   - training supergroup   0 2014-02-09 19:55 /user/training/ext3
drwxr-xr-x   - training supergroup   0 2014-02-10 20:15 /user/training/trial_direc
[training@localhost Desktop]$ ls
direc.png Eclipse.desktop namenode.png new_file sqljdbc4-3.0.jar.zip
[training@localhost Desktop]$ hadoop fs -put ./new_file /user/training/
[training@localhost Desktop]$ hadoop fs -ls /user/training
Found 4 items
drwxr-xr-x   - training supergroup   0 2014-02-09 19:14 /user/training/ext1
drwxr-xr-x   - training supergroup   0 2014-02-09 19:55 /user/training/ext3
-rw-r--r--   1 training supergroup   34 2014-02-10 20:26 /user/training/new_file
drwxr-xr-x   - training supergroup   0 2014-02-10 20:15 /user/training/trial_direc
[training@localhost Desktop]$

Practice Exercise

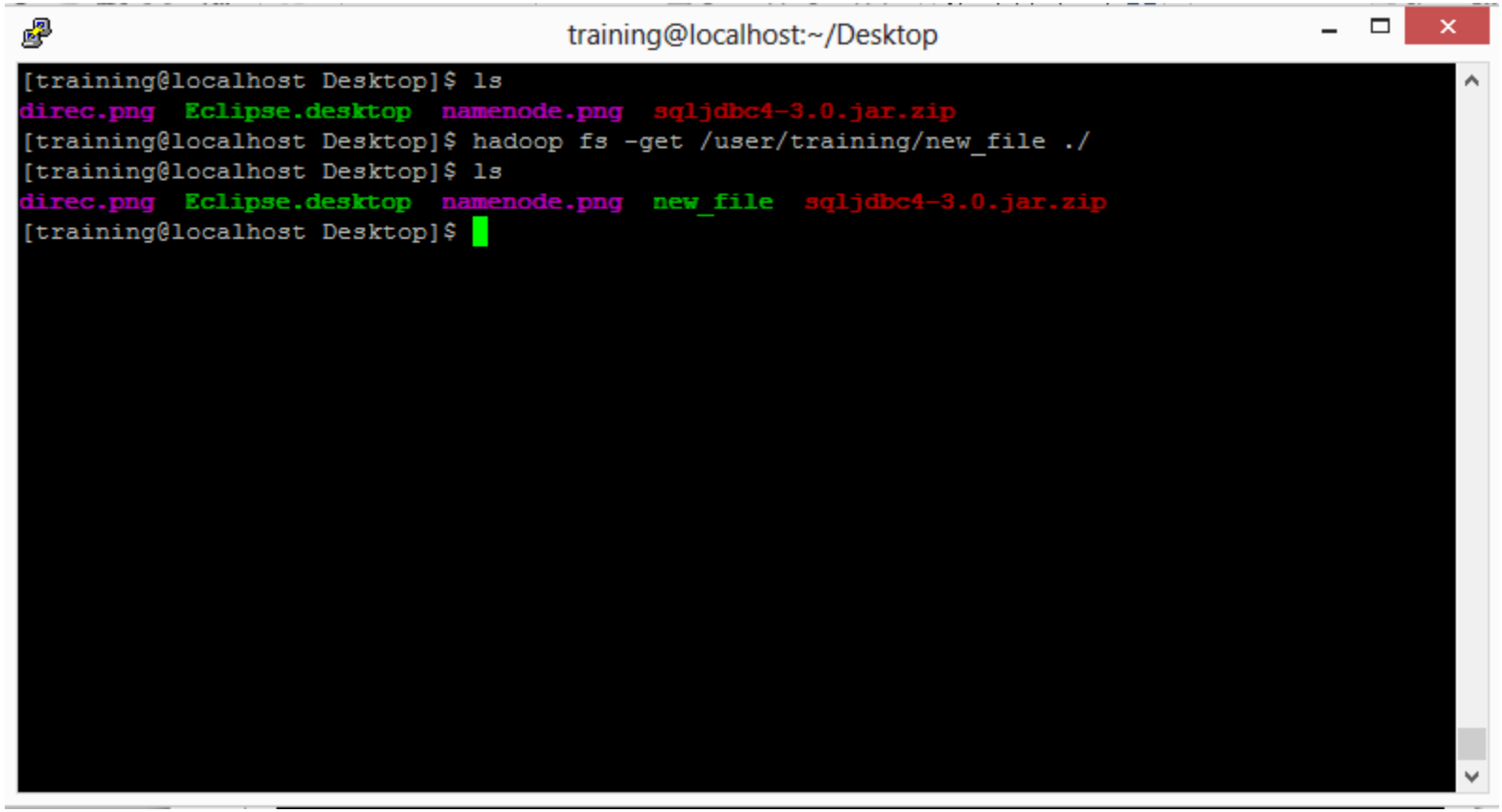
- Write a file in your local file system of vm using vi editor, save that file. Now put that file in your named directory in HDFS using put command.
- vi <myname>
- HADOOP dfs –put <myname> <hadoop destination>

HADOOP Get

- **Get**
- Usage: hadoop fs -get [-ignorecrc] [-crc] <src> <localdst>

- Copy files to the local file system. Files that fail the CRC check may be copied with the –ignorecrc option. Files and CRCs may be copied using the -crc option.

Example Get


training@localhost:~/Desktop
[training@localhost Desktop]$ ls
direc.png Eclipse.desktop namenode.png sqljdbc4-3.0.jar.zip
[training@localhost Desktop]$ hadoop fs -get /user/training/new_file ./
[training@localhost Desktop]$ ls
direc.png Eclipse.desktop namenode.png new_file sqljdbc4-3.0.jar.zip
[training@localhost Desktop]$

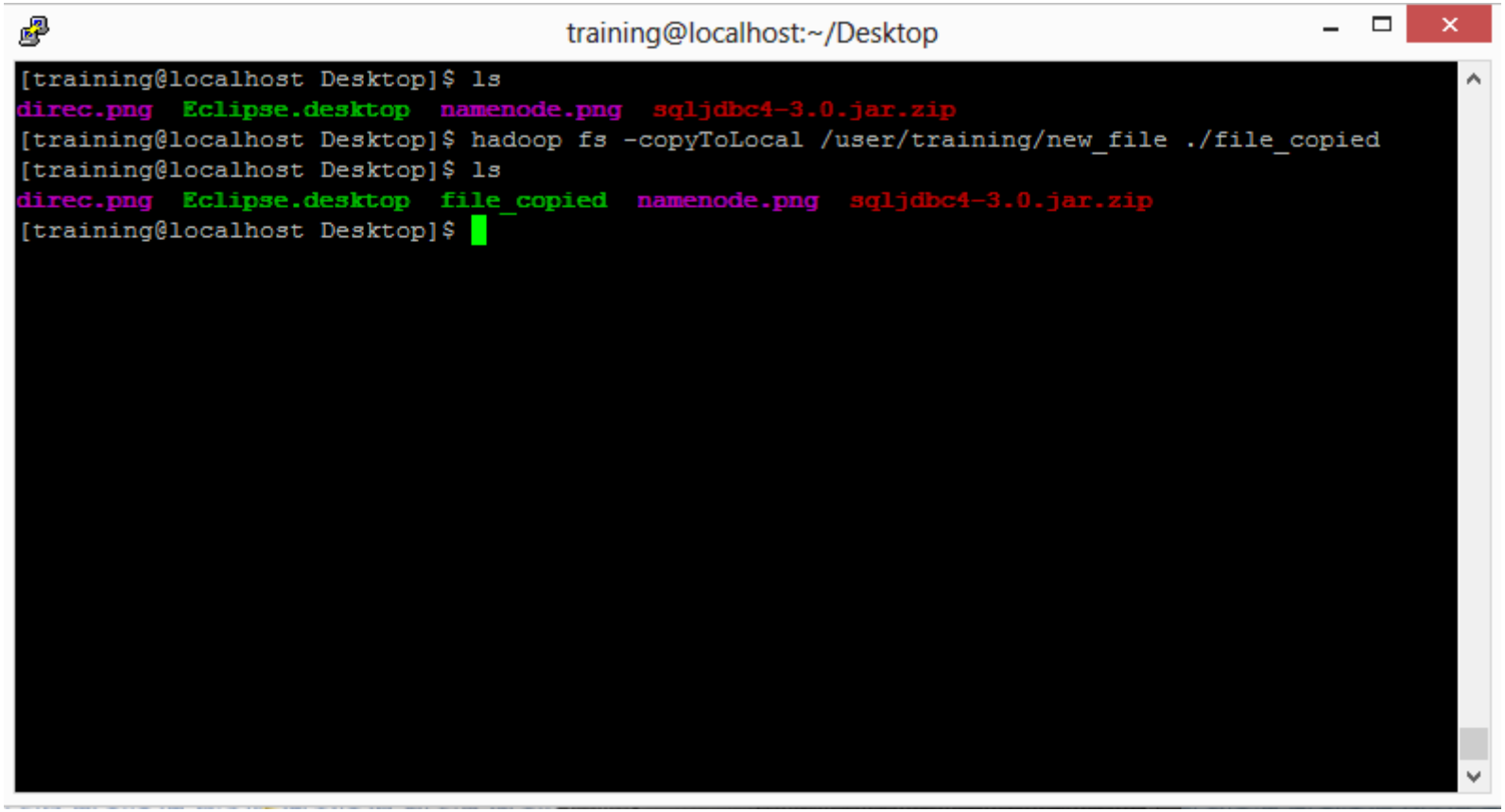
Practice Exercise

- Get the file you saved in HDFS to your local file system of VM, and read their contents and check if files are same.

HADOOP Copy to Local

- Usage: hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
- Similar to **get** command, except that the destination is restricted to a local file reference.

HADOOP Copy to Local



Java APIs

Java API's

- HDFS can be accessed with JAVA API's as well
- You can add/delete/view file system from them

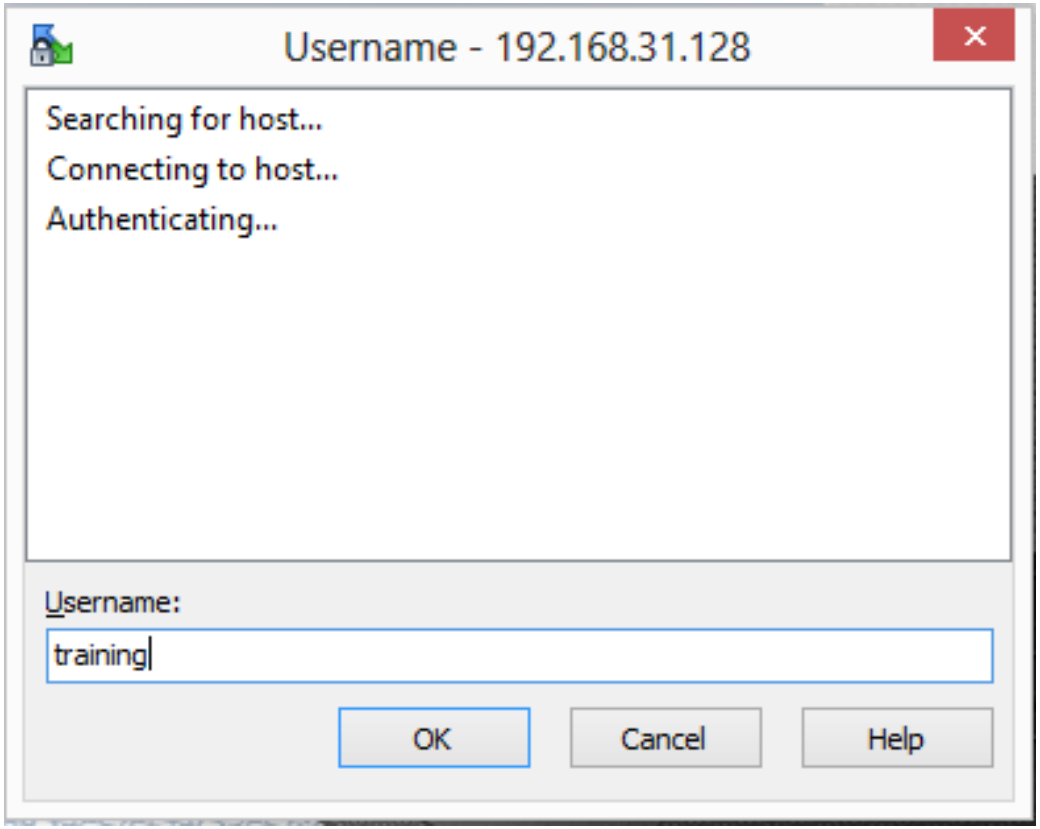
Environment

- To access hadoop API's hadoop jars should be there in build path of eclipse
- **From /usr/lib/hadoop/client-0.20**

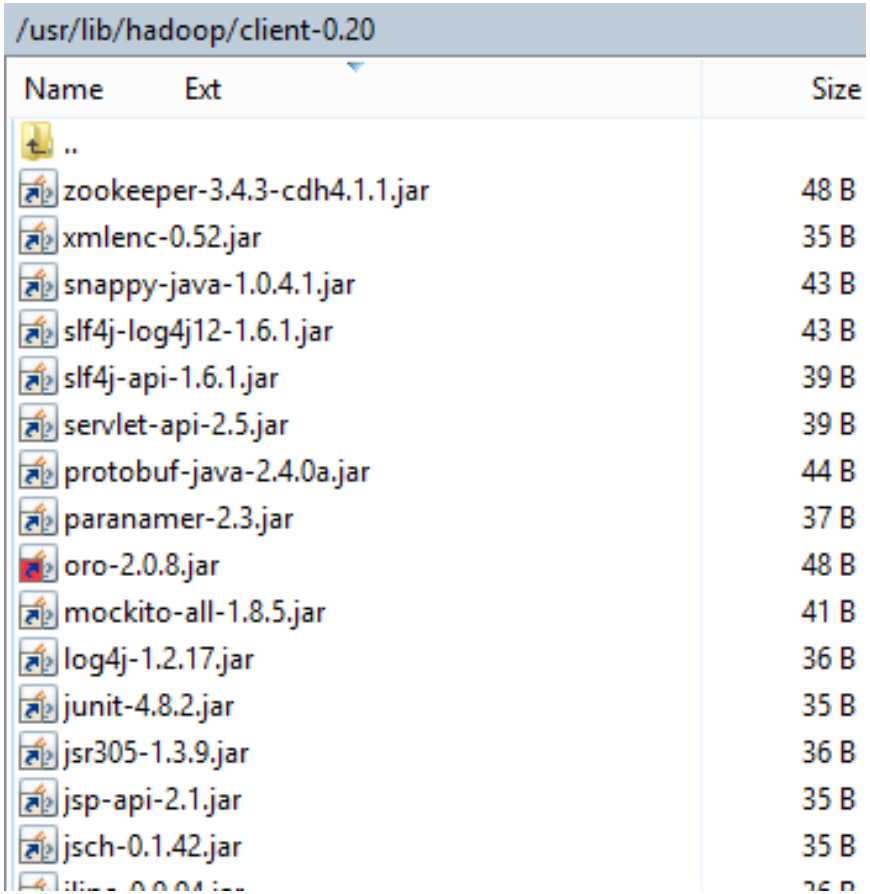
  copy this complete folder to your local system and add these jars into your build path

WinSCP

- Use WINSCP to copy the Jars to local system



- /usr/lib/hadoop/client-0.20

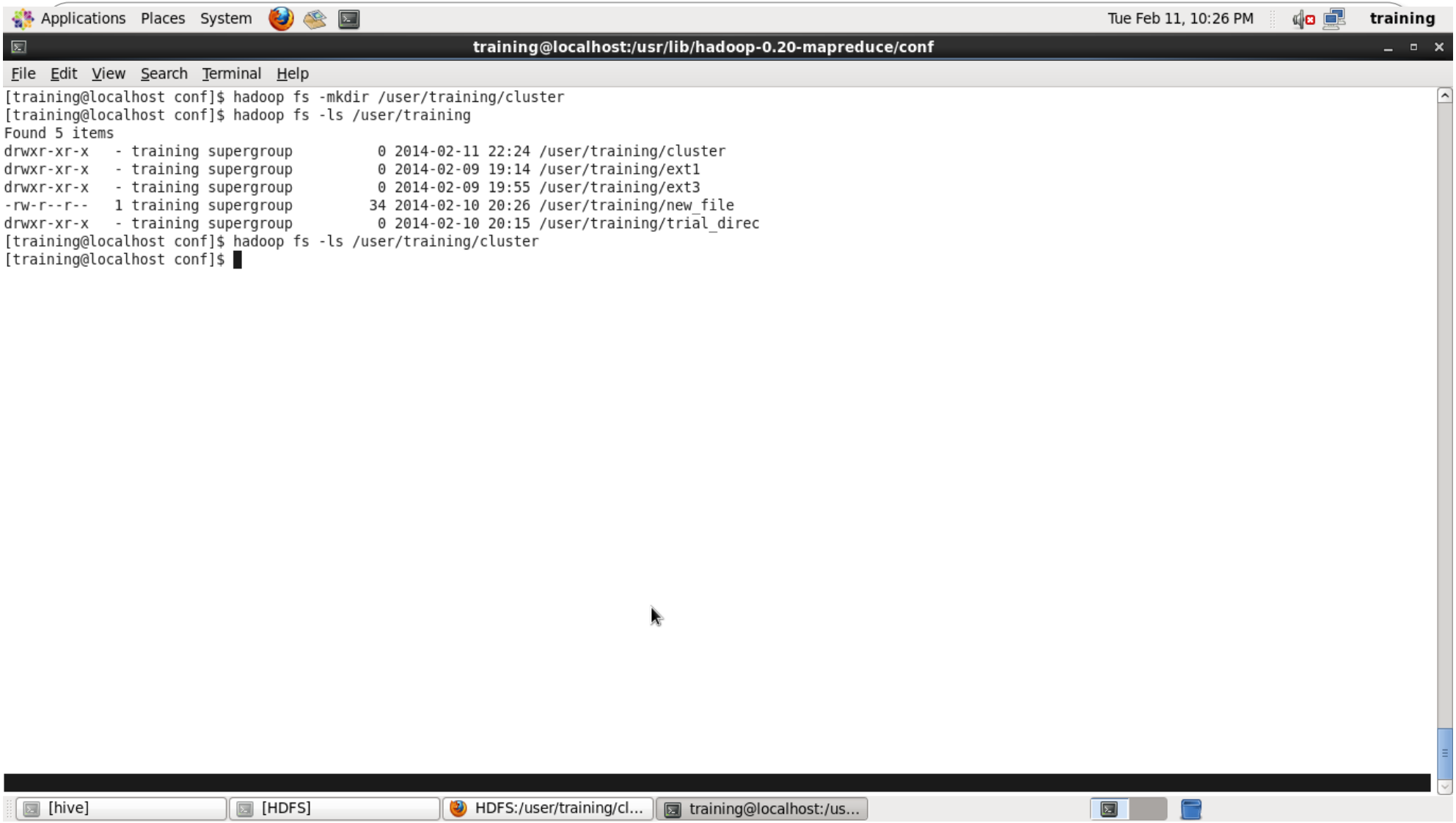


Project Settings

- Make a new project in eclipse
- Add these jars in it’s build path

Exercise

- We will put a file in HDFS and try to read it's contents with JAVA APIs


Applications Places System
Tue Feb 11, 10:26 PM
training
training@localhost:/usr/lib/hadoop-0.20-mapreduce/conf
File Edit View Search Terminal Help
[training@localhost conf]$ hadoop fs -mkdir /user/training/cluster
[training@localhost conf]$ hadoop fs -ls /user/training
Found 5 items
drwxr-xr-x - training supergroup 0 2014-02-11 22:24 /user/training/cluster
drwxr-xr-x - training supergroup 0 2014-02-09 19:14 /user/training/ext1
drwxr-xr-x - training supergroup 0 2014-02-09 19:55 /user/training/ext3
-rw-r--r-- 1 training supergroup 34 2014-02-10 20:26 /user/training/new_file
drwxr-xr-x - training supergroup 0 2014-02-10 20:15 /user/training/trial_direc
[training@localhost conf]$ hadoop fs -ls /user/training/cluster
[training@localhost conf]$
[hive]
[HDFS]
HDFS:/user/training/cl...
training@localhost:/us...

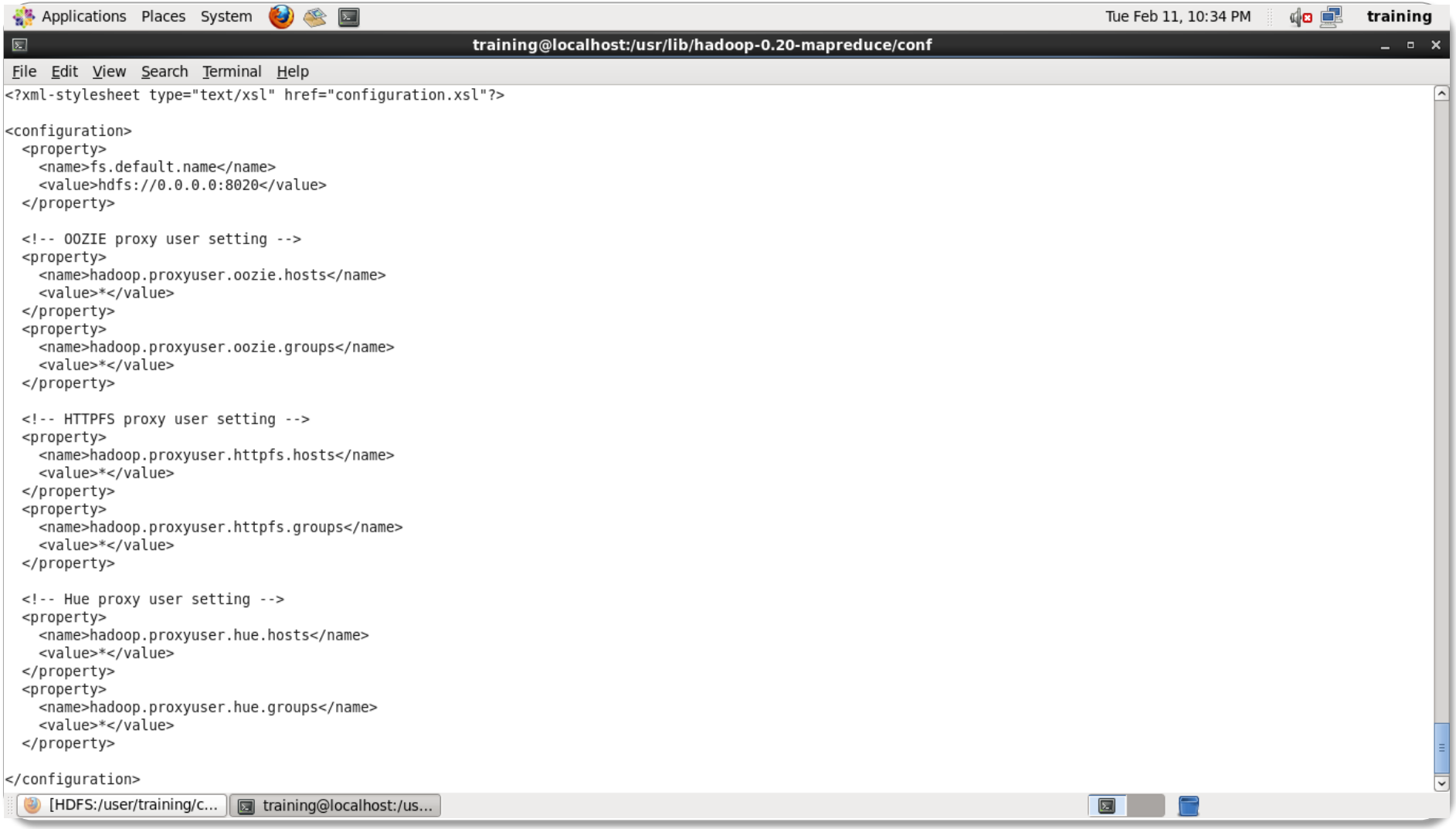
VM Environment

- The default configurations in vm are local host. With this configuration HADOOP doesn't allow external client connections.
- To ease the testing and our understanding we would like to connect the HADOOP from our local OS where eclipse is installed, so that we don't have to scp our jars or classes to vm.

Configurations Needed

- Change the NAMENODE/JOBTRACKER from local host to the IP of vm
- Files to be changed in /usr/lib/hadoop-0.20-mapreduce/conf are mapred-site.xml, hdfs-site.xml, core-site.xml

Editing Read Only File



- To edit the file you will have to do sudo su so that with root access you can change the file

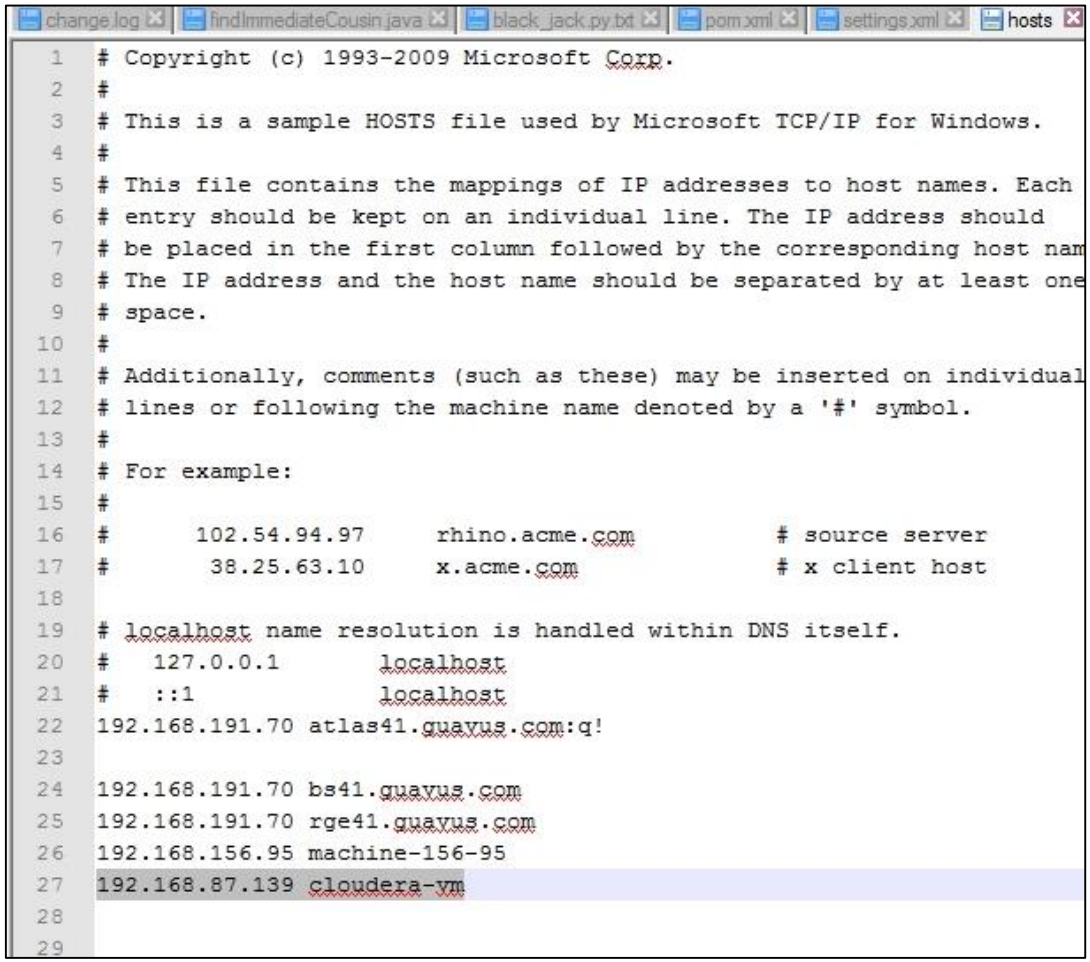
```
[training@localhost conf]$ sudo su
[root@localhost conf.pseudo.mr1]#
```

- You can now edit the file and change the localhost to IP of vm

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

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.31.128:8020</value>
  </property>
```

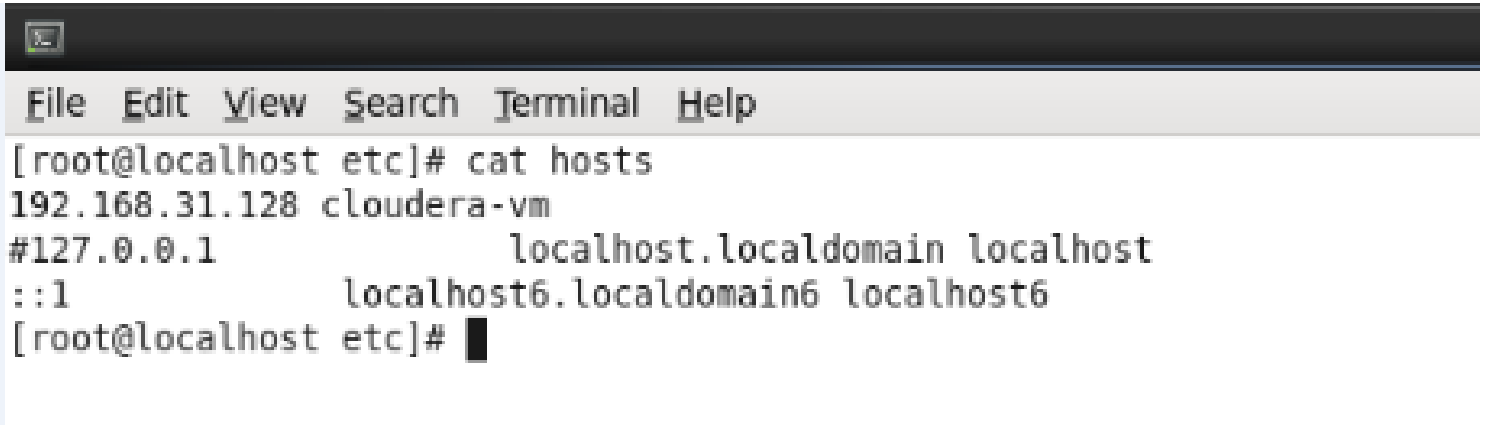
Changes to be Done in Local OS

- If you are running MR jobs/HDFS access from outside vm, host name entry needs to be done in the local OS as well, so that IP corresponding to the name can be pinged
- In running from windows C:\Windows\System32\drivers\etc\hosts, add the mapping 192.168.31.128 cloudera-vm


change.log
findImmediateCousin.java
black_jack.py.txt
pom.xml
settings.xml
hosts
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host nam
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 # 102.54.94.97 rhino.acme.com # source server
17 # 38.25.63.10 x.acme.com # x client host
18
19 # localhost name resolution is handled within DNS itself.
20 # 127.0.0.1 localhost
21 # ::1 localhost
22 192.168.191.70 atlas41.guavus.com:q!
23
24 192.168.191.70 bs41.guavus.com
25 192.168.191.70 rge41.guavus.com
26 192.168.156.95 machine-156-95
27 192.168.87.139 cloudera-vm
28
29

Changes to be Done

- Change the fs.default.name to hdfs://<IP OF VM>:8020 in /etc/hadoop-0.20/conf/core-site.xml
- In mapred-site.xml mapred.job.tracker to <IP OF VM>:8021
- In /etc/hosts make sure that there is no entry of 127.0.0.1 and there is a hostname for your VM IP


File Edit View Search Terminal Help
[root@localhost etc]# cat hosts
192.168.31.128 cloudera-vm
#127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
[root@localhost etc]#

Restart HADOOP Services

- To make your changes effective in hadoop you will have to restart namenode, data node and jobtracker

Use commands

- sudo service hadoop-hdfs-namenode restart
- sudo service hadoop-hdfs-datanode restart
- sudo service hadoop-0.20-mapreduce-jobtracker restart
- sudo service hadoop-0.20-mapreduce-tasktracker restart

Restart Name Node and Data Node

```
File  Edit  View  Search  Terminal  Help
[training@localhost etc]$ sudo service hadoop-hdfs-namenode restart
Stopping Hadoop namenode:                                  [  OK  ]
no namenode to stop
Starting Hadoop namenode:                                  [  OK  ]
starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-localhost.localdomain.out
[training@localhost etc]$
```

```
[training@localhost etc]$ sudo service hadoop-hdfs-datanode restart
Stopping Hadoop datanode:                                  [  OK  ]
stopping datanode
Starting Hadoop datanode:                                  [  OK  ]
starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-localhost.localdomain.out
[training@localhost etc]$
```

Health Check

- After doing all the changes check if HDFS is accessible
- HADOOP dfs –ls /

Troubleshooting Guide

- You can always check logs of namenode, datanode to see if any errors have occurred while starting namenode/datanode or hadoop services in /var/log/hadoop-0.20-mapreduce/logs and /usr/lib/hadoop-hdfs/logs
- In case you get the error

```
cloudera@cloudera-vm:/etc/hadoop-0.20/conf.pseudo$ hadoop dfs -ls /
13/12/08 05:44:57 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 0 time(s).
13/12/08 05:44:58 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 1 time(s).
13/12/08 05:44:59 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 2 time(s).
13/12/08 05:45:00 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 3 time(s).
13/12/08 05:45:01 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 4 time(s).
13/12/08 05:45:02 INFO ipc.Client: Retrying connect to server: /192.168.87.138:8020. Already tried 5 time(s).
```

- Issue must be with your host mapping, please check /etc/hosts to see if there is any entry for localhost IP 127.0.0.1. If it is there comment or delete that and restart your hdfs services

Distscp

- Tool for large inter/intra cluster copying.
- Can be used to copy large data from one hdfs cluster to another. Provided both are running on same HADOOP version
- HADOOP distscp hdfs://namenode1/foo hdfs://namenode2/foo2
- This will copy /foo directory in foo2
- This is implemented as MR job.(Will be covered in MR section)

Java APIs

APIs

- http://hadoop.apache.org/docs/current/api/org/apache/hadoop/fs/Path
- **Path** (URI aUri)
  - Construct a path from a URI. This URI is the path of HDFS
- The file system can be obtained by:
  - http://hadoop.apache.org/docs/current/api/org/apache/hadoop/fs/FileSystem.html

APIs

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;


public class ReadFile {

    public static void main(String[] args) throws Exception {

        try{

            Path path  = new Path("hdfs://cloudera-vm:8020/pristine/data");
            Configuration conf = new Configuration();
            FileSystem fs = FileSystem.get(path.toUri(), conf);
            BufferedReader br = new BufferedReader(new InputStreamReader(fs.open(path)));
            String line ;
                line = br.readLine();
            while(line!=null)
            {
                System.out.println(line);
                line = br.readLine();
            }


        }catch(Exception e)
        {
            System.out.println(e);
            e.printStackTrace();
        }
```

HADOOP CLASSES FOR HDFS

HDFS PATH, Expects host name of hadoop machine. Port 8020 of namenode.

FILE SYSTEM HADOOP

Output

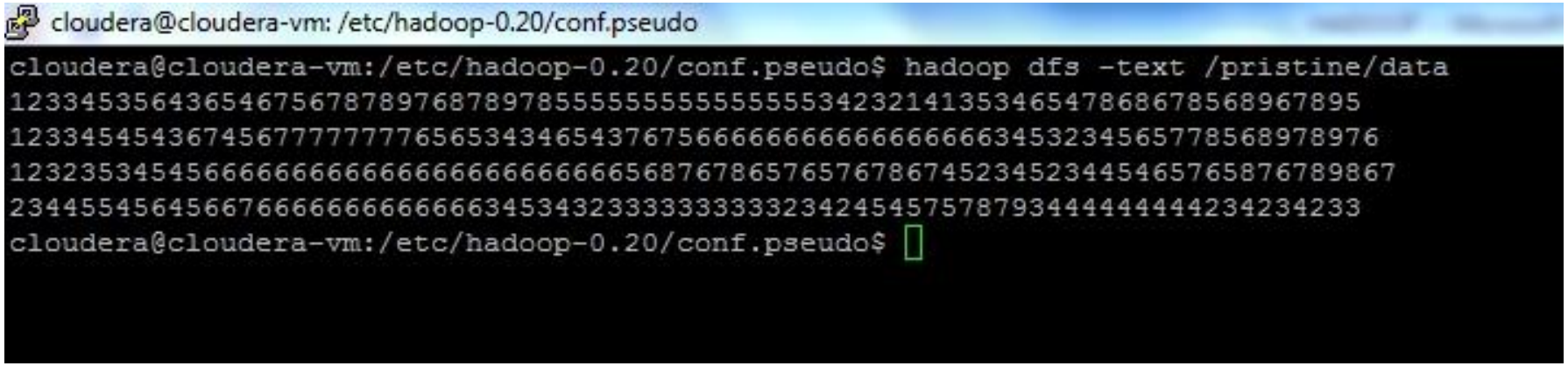
Mapper.java | RncAnnotation.java | IRncAnnotationRecord | IAnnotationRecord.ja | EdrAnnotationRecord. | ReadFile.java | FileCopyWithProgress

```
1  import java.io.BufferedReader;
2  import java.io.InputStreamReader;
3
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.FileSystem;
6  import org.apache.hadoop.fs.Path;
7
8
9  public class ReadFile {
10
11     public static void main(String[] args) throws Exception {
12
13         try{
14
15             Path path  = new Path("hdfs://cloudera-vm:8020/pristine/data");
16             Configuration conf = new Configuration();
17             FileSystem fs = FileSystem.get(path.toUri(), conf);
18             BufferedReader br = new BufferedReader(new InputStreamReader(fs.open(path)));
19             String line ;
20                 line = br.readLine();
21             while(line!=null)
22             {
23                 System.out.println(line);
```

Console

```
<terminated> ReadFile [Java Application] C:\Common\binary\com.sun.java.jdk.win32.x86_1.6.0.013\bin\javaw.exe (09-Dec-2013 4:29:31 PM)
12334535643654675678789768789785555555555555553423214135346547868678568967895
12334545436745677777777656534346543767566666666666666666634532345657785689789 76
12323534545666666666666666666666666656876786576576786745234523445465765876789867
23445545645667666666666666634534323333333333234245457578793444444444234234233
```

Verify with HDFS CONENTS



Seek

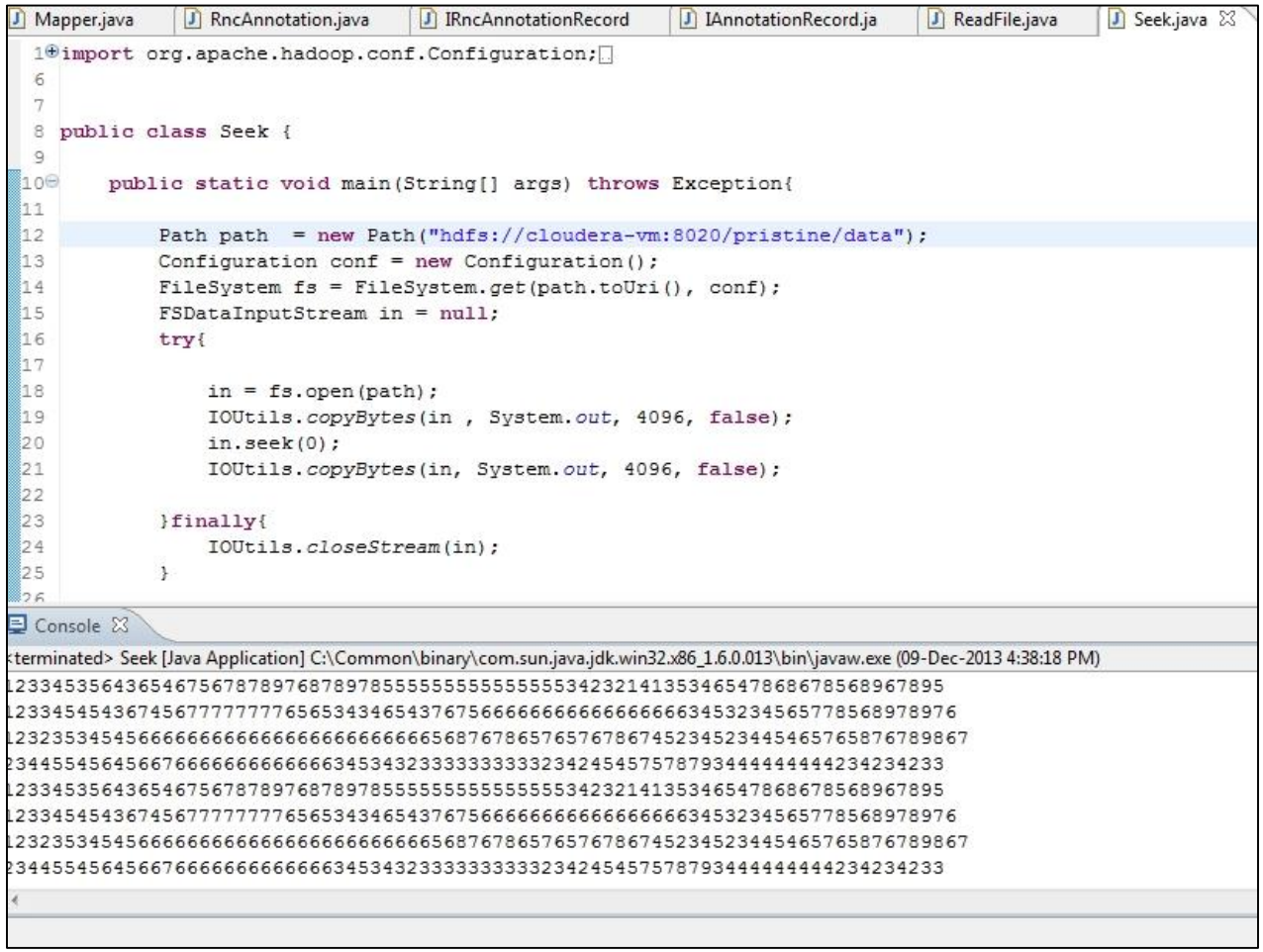
- Read from offset

Java API

- http://hadoop.apache.org/docs/current/api/org/apache/hadoop/fs/FSDataInputStream.html

```
import org.apache.hadoop.conf.Configuration;


public class Seek {

    public static void main(String[] args) throws Exception{

        Path path  = new Path("hdfs://cloudera-vm:8020/pristine/data");
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(path.toUri(), conf);
        FSDataInputStream in = null;
        try{

            in = fs.open(path);
            IOUtils.copyBytes(in , System.out, 4096, false);
            in.seek(0);
            IOUtils.copyBytes(in, System.out, 4096, false);

        }finally{
            IOUtils.closeStream(in);
        }

    }
}
```

Seek to desired offset

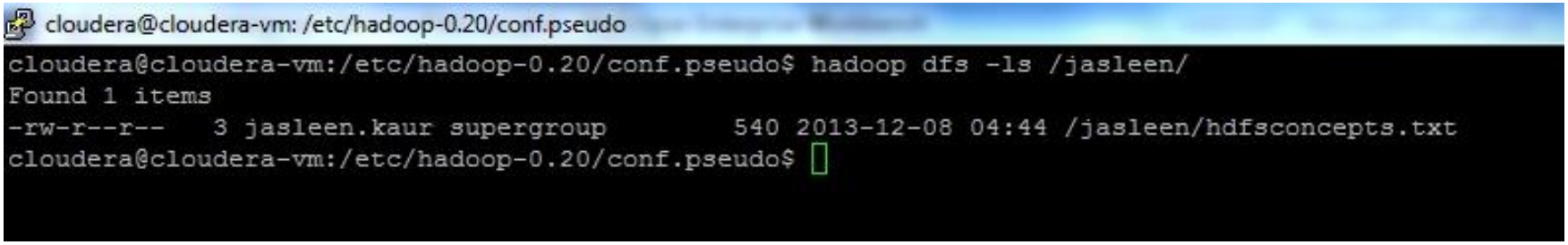

Mapper.java
RncAnnotation.java
IRncAnnotationRecord
IAnnotationRecord.ja
ReadFile.java
Seek.java
1 import org.apache.hadoop.conf.Configuration;
6
7
8 public class Seek {
9
10 public static void main(String[] args) throws Exception{
11
12 Path path = new Path("hdfs://cloudera-vm:8020/pristine/data");
13 Configuration conf = new Configuration();
14 FileSystem fs = FileSystem.get(path.toUri(), conf);
15 FSDataInputStream in = null;
16 try{
17
18 in = fs.open(path);
19 IOUtils.copyBytes(in , System.out, 4096, false);
20 in.seek(0);
21 IOUtils.copyBytes(in, System.out, 4096, false);
22
23 }finally{
24 IOUtils.closeStream(in);
25 }
Console
terminated> Seek [Java Application] C:\Common\binary\com.sun.java.jdk.win32.x86_1.6.0.013\bin\javaw.exe (09-Dec-2013 4:38:18 PM)
1233453564365467567878976878978555555555555555342321413534654786867856896789
1233454543674567777777765653434654376756666666666666666663453234565778568978976
1232353454566666666666666666666666656876786576576786745234523445465765876789867
2344554564566766666666666663453432333333333323424545757879344444444234234233
1233453564365467567878976878978555555555555555342321413534654786867856896789
1233454543674567777777765653434654376756666666666666666663453234565778568978976
1232353454566666666666666666666666656876786576576786745234523445465765876789867
2344554564566766666666666663453432333333333323424545757879344444444234234233

Copy File

```
import java.io.BufferedInputStream;


public class FileCopy {

    public static void main(String[] args) throws Exception {

        String localSrc = "C:\\Users\\jasleen.kaur\\Desktop\\hdfsconcepts.txt";
        Path path  = new Path("hdfs://cloudera-vm:8020/jasleen/hdfsconcepts.txt");


        Configuration conf = new Configuration();

        FileSystem fs = FileSystem.get(path.toUri(), conf);

        InputStream in  = new BufferedInputStream(new FileInputStream(localSrc));
        OutputStream out = fs.create(path, true);
        IOUtils.copyBytes(in, out, 4096, true);




    }

}
```

Path of File to be copied

Path of hdfs


cloudera@cloudera-vm: /etc/hadoop-0.20/conf.pseudo
cloudera@cloudera-vm:/etc/hadoop-0.20/conf.pseudo$ hadoop dfs -ls /jasleen/
Found 1 items
-rw-r--r--   3 jasleen.kaur supergroup   540 2013-12-08 04:44 /jasleen/hdfsconcepts.txt
cloudera@cloudera-vm:/etc/hadoop-0.20/conf.pseudo$

Case Study Exercise

- STORAGE: Put the provided sample logs of airmobile in HDFS directory with name airmobile with java APIs.
- READ: Try to read the file using FS commands.
- The fields in the given file represent field following msidn= “<subscriber Id>”, databytes =“Bytes Downloaded“, city=“<city”>, sitevisited=“<Site Visited”>. For analysis write a Java program to find out subscribers and their corresponding downloaded bytes.

Next Agenda

- Next we will try to find out how the same Java program implemented in last exercise can be done with Map Reduce.

Thank you!

Contact:

EduPristine

702, Raaj Chambers, Old Nagardas Road, Andheri (E), Mumbai-400 069. INDIA

www.edupristine.com

Ph. +91 22 3215 6191