BIG DATA ADMINISTRATION EXERCISES

ABSTRACT

This document outlines the exercises to be undertaken in the Administration section of the Big Data course

QA Consulting Academy Team

Big Data Academy

Contents

Introduction 3
Starting the Virtual Machines 3
Step 1) Configure your theme 3
Step 2) Configure your IPs 3
Step 3) Edit your IP configuration files 4
Step 4) Add a new user 4
Networking 4
Exporting 5
Exercise 1: Installing Hadoop in Pseudo-Distributed Mode 6
Installing Hadoop in Pseudo-Distributed Mode from CDH 6
Testing the Hadoop Installation 8
Exercise 2: Creating a Hadoop Cluster 11
Installing Hadoop Software on the Cluster 11
Modifying the Hadoop Configuration Files 13
Setting up the File System 16
Formatting HDFS and Starting the HDFS Daemons 17
Creating Directories for YARN and MapReduce in HDFS 19
Starting the YARN and MapReduce daemons 19
Testing the Cluster 20
Exercise 3: Querying HDFS with Hive and Impala 22
Installing, Configuring and Starting a ZooKeeper Ensemble 22
Installing and Configuring Hive 23
Initialising the Hive Metastore in MySQL 25
Installing and Starting the Hive Metastore Service 25
Installing, Configuring and Starting HiveServer2 26
Modifying the HDFS Configuration for Impala 27
Installing, Configuring and Starting the Impala Daemons 27
Running Hive Queries 29
Running Impala Queries 32
Exercise 4: Configuring HDFS High Availability 34
Bringing Down Unneeded Servers 34
Bringing Down the Existing HDFS Configuration 35
Modifying the Hadoop Configuration 35
Installing JournalNode Software and Starting the JournalNodes 36
Initilialising the Shared Edits Directory and Starting the First NameNode 37

Installing, Bootstrapping and Starting the Standby NameNode 38
Installing, Formatting, and Starting the ZooKeeper Failover Controllers 39
Starting the DataNodes and Restarting the MapReduce Daemons 40
Testing the HDFS High Availability Deployment 40
Exercise 5: Breaking & Healing The Cluster 42
Challenges 43

Introduction

Take care if you are copy and pasting information from this exercise book as the formatting may cause issues, you may wish to enter it manually instead, though then you must be careful of typos.

For these exercises you will be working on a cluster of four virtual machines. A cluster of virtual machines is not recommended for production environments as it removes some of Hadoop's fault tolerance since all copies of a data block may be stored on one physical machine. This means if the physical host fails, all virtual machines on that host will fail. However, for demonstration and proof-of-concept purposes it suits us just fine.

Starting the Virtual Machines

Firstly you should install the programs we'll need to set up the virtual machines: Vagrant, Git Bash, and VirtualBox. You can find these online or install them from "*C:\LocalInstall\Downloads & Installers*".

Your virtual machines have been somewhat pre-prepared, all you have to do is set them off to initiate then make some installations and configurations for your purposes.

You will be distributed a folder called Vagrant boxes. In this folder you will be provided a number of vagrant box files which have been pre-downloaded to save time, these are required to run your virtual machines within vagrant. Take a **copy** of these files and place them with *"C:/Users/Administrator/.vagrant.d/boxes".*

Next you need to obtain the vagrant directory, which has been pre-created for you. This directory contains a variety of shell scripts, a shared directory and within some more configuration files. You've also got your Vagrantfile within this directory, which is the file that will complete most of the work. There are some vagrant command instructions at "*https://github.com/ajmulhollan1/vagrantcluster.git*" along with similar instructions to the below. However you do not need to clone this directory and you should follow the instructions given by your trainer.

Navigate to the directory where your Vagrantfile is installed and follow the below instructions:

Step 1) Configure your theme

You will be each assigned a group with which you will share a cluster system and be working together to set it up, so it may be wise to come up with a theme you can name your boxes. The example used in the Academy is "*Orange*, *Blue*, *Green, and Pink*". Each team member's virtual machines must be uniquely named. So you may have a range of "Orange1, Orange2... etc" or a similar running theme.

Once you've thought up your theme, all teams must change their individual vagrant file and update each instance of the words "*Orange*, *Green*, *Blue*, and *Pink*" with their own virtual machine names.

Step 2) Configure your IPs

Your trainer will assign each group a subnet range which will begin 192.168.XXX.YY. The XXX portion of your IP is for the entire team and is the subnet you will be communicating on. The YY portion of your IP will be different for every team member's virtual machine. You must communicate effectively with your team to ensure every machine has a unique IP.

Step 3) Edit your IP configuration files

Open the shared folder and open the two "eth-1" files in Notepad ++ and edit the IP ADDRESS ONLY to match your individual IPs in the vagrant file. Note you only need to do this for the CentOS boxes, as the Ubuntu ones do not require additional configuration.

If you have any issues throughout this configuration, call your trainer over for further assistance.

Ensure you have VirtualBox & Git Bash open. Change your PWD (Present Working Directory) to the Cloudera Vagrant folder in git bash. Note you'll need to use a command similar to cd "*/c/Users/Administrator/Desktop/Cloudera\ Vagrant"*.

Enter the command 'vagrant up' to begin the process. This will take some time but by the end you should have four virtual machines. To start with, we will mostly be running from 'Orange', but as time goes on we will be setting up a full virtual cluster.

Step 4) Add a new user

You have been given the script file "*adduser.sh"* which will ask you a series of questions to enable the creation of a new user. You will need to perform this on each machine.

Before you execute the script, change your current directory to the shared folder, and, if you are running this on a centos machine, change the user to 'root'. When asked for a password, enter 'vagrant'.

```
$ cd /tmp/shared
```

```
$ su
```

To execute the script type "sh adduser.sh" from the same directory as the file and enter your details as described. Simply press enter to use the default options for the next two options beyond the user group.

At the end of this script it will ask you to specify which OS you are running. Enter either Ubuntu or centos depending on which your machine is, and it will automatically add your user to the sudoers file.

To exit the root terminal, just type 'exit'. To check your user has been created, run the following command. Your username should appear at the end.

```
$ sudo cut –d: -f1 /etc/passwd
```

To change your password, use the following command.

```
$ sudo passwd username
```

Once you've created your new user, log out of the vagrant account and log in to your unique account. Use this account throughout the rest of the course.

Networking

To set up the network between your virtual machines, follow the steps below. You should already have your fours hosts set up using Vagrant. Note that you will have to do something similar when you create your cluster in your groups, so remember this well.

On each host you need to locate the /etc/hosts file and edit it. This file will allow the virtual machines to know what else is in the cluster. To do this you will need the IP address of each virtual

machine and the name you have given it, you can find this using the command below. The example below uses examples and should not be directly copied, but your structure should remain the same.

```
$ ifconfig                \\ finds the IP address of your machine
                          \\ you want the 'inet addr' for eth1
```

```
$ sudo gedit /etc/hosts
```

Each file needs to contain something like the following:

```
#127.0.0.1          <vm name> localhost.localdomain localhost          //comment out!
::1                 localhost6.localdomain6 localhost6
192.168.1.164 orange
192.168.1.27 blue
192.168.1.57 green
192.168.1.28 pink
```

At this point you need to check if the machine IP addresses match what was set in your VagrantFile. If your eth1 IP information does not appear when using the 'ifconfig' command you should restart the network using 'sudo service network restart'.

Before we continue, we will disable the firewalls. Later, when you connect with everyone else in your team, it would be useful to try setting up the firewalls to accept certain communications rather than turning them off completely, but for now, we will continue with them off for ease of use.

```
$ sudo service iptables stop          \\ centos
$ sudo ufw disable                    \\ Ubuntu
```

Exporting

At this point you should export your virtual boxes and reimport them into VirtualBox. Vagrant has been very useful for creating and setting up our machines, but it could cause problems with our cluster due to the fickle nature of shutting down machines and bringing them back up.

To do this, simply open VirtualBox, click on File -> Export Appliance. Export each machine individually as a .ova file. This may take some time.

Once you have exported everything, import it into VirtualBox, open up each virtual machine, and make sure the network is still working and everything is as you set it up.

Exercise 1: Installing Hadoop in Pseudo-Distributed Mode

In this exercise you will install and test Hadoop in a pseudo-distributed mode using the Cloudera Distribution of Hadoop (CDH). This is basically setting up Hadoop on just one machine to start off with. It means we can quickly get an environment up and running to play around with and is useful for demonstration, however you wouldn't find it in a live production environment apart from for testing. Try and understand what each command is doing as we go through, but don't worry too much if it doesn't click immediately, you will get plenty of experience with it as we go on. Bear in mind that most of what is below is Hadoop-centric, not Cloudera specific. That is, you could go and use a Hortonworks distribution in much the same way, and there is plenty of material out there to walk you through this.

Installing Hadoop in Pseudo-Distributed Mode from CDH

When we talk about installing CDH in pseudo-distributed mode this means you are installing a Hadoop cluster on a single machine that will run all six core Hadoop daemons (NameNode, Secondary NameNode, DataNode, ResourceManager, NodeManager and JobHistoryServer). As a note: with a single DataNode, the replication factor is set to one. This is useful for testing environments and is useful to know how to set up, at least as a starting point.

1. For our pseudo-distributed cluster we'll use the base host, in our case this is orange. Check to see which of your VMs has the most resources allocated to it – this is your base and what you should use to start with on the following exercises. It should be running Cent-OS. To start things off you will need to create a repository file by completing the following steps.

    a. Navigate to /etc/yum.repos.d/ in your file system.

    ```
    $ cd /etc/yum.repos.d/
    ```

    b. Create a new file called cloudera-cdh5.repo

    ```
    $ sudo gedit cloudera-cdh5.repo
    ```

    c. This file is going to set up 'directions' for our machine to find the installation files for CDH 5. When the file is open add the following details to it, save, and close the file. You will need to do these steps for your other centos machine later, do it now if you prefer or be sure to remember the steps. The installation of Hadoop will differ depending on what operating system you are using, and we will look at some other options later.

    ```
    [cloudera-cdh5]
    # Packages for Cloudera's Distribution for Hadoop, Version 5, on RedHat or CentOS 6 x86_64
    name=Cloudera's Distribution for Hadoop, Version 5
    baseurl=https://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/
    gpgkey =https://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
    gpgcheck=1
    ```

2. From the terminal window on orange enter the following:

```
$ sudo yum install hadoop-conf-pseudo
```

You may need to confirm you want to continue by entering 'y'.
This command installs the core Hadoop package, init scripts for the Hadoop daemons and the configuration files required for Hadoop to operate in pseudo-distributed mode, as well as any dependencies.

3. Format the Namenode on orange:

```
$ sudo –u hdfs hdfs namenode -format
```

4. The configuration files are in /etc/hadoop/conf which is a symbolic link from /etc/alternatives/hadoop-conf
Change the directory to /etc/hadoop/conf and inspect the various *-site.xml files. The example below uses gedit but feel free to use whatever method you choose. Replace the *-site.xml in the command prompt to the name of the xml file you are inspecting. If you wish to look at what is in your current directory to see the names, use the 'ls' command. We'll be editing some of these later.

```
$ cd /etc/hadoop/conf
$ ls
$ gedit *-site.xml
```

5. Start Hadoop's HDFS daemons.
On orange:

```
$ sudo service hadoop-hdfs-namenode start
$ sudo service hadoop-hdfs-secondarynamenode start
$ sudo service hadoop-hdfs-datanode start
```

6. Create the staging directory in HDFS. YARN and MapReduce will use this directory.
On orange:

```
$ sudo –u hdfs hadoop fs –mkdir /tmp
$ sudo –u hdfs hadoop fs –mkdir –p /tmp/hadoop-
yarn/staging/history/done_intermediate
$ sudo –u hdfs hadoop fs –chown –R mapred:mapred /tmp/hadoop-
yarn/staging
$ sudo –u hdfs hadoop fs –chmod –R 1777 /tmp
```

7. Create the YARN log directory.
On orange:

```
$ sudo –u hdfs hadoop fs –mkdir –p /var/log/hadoop-yarn
$ sudo –u hdfs hadoop fs –chown yarn:mapred /var/log/hadoop-yarn
```

8. Start the YARN and MapReduce daemons.
   On orange:

   ```
   $ sudo service hadoop-yarn-resourcemanager start
   $ sudo service hadoop-yarn-nodemanager start
   $ sudo service hadoop-mapreduce-historyserver start
   ```

9. Before the next step you should install a useful piece of kit. You will only need to install this on your Cent-OS machines as the Ubuntu machines already have it. This program can list us all the JVMs currently running, which means we can see which of our hadoop daemons are up at any time on that particular machine. Your machines should be running Java 1.7.0, but if not, just be sure to change this in the commands below (you can find your java version using the command **java –version**).

   ```
   $ sudo yum install -y java-1.7.0-openjdk*

   $ sudo apt-get install –y openjdk-7-jdk

   $ sudo apt-get install –y openjdk-7-jre
   ```

10. Check to ensure all six daemons are running.
    On orange:

    ```
    $ sudo jps
    ```

    You should see the six Hadoop daemons running.

11. Finally, create a user directory for the user.
    On orange:

    ```
    $ sudo –u hdfs hadoop fs –mkdir –p /user/<user name>
    $ sudo –u hdfs hadoop fs –chown <user name> /user/<user name>
    ```

Testing the Hadoop Installation

We will now test the Hadoop installation by uploading some data.

1. Change directories to the /tmp/shared folder. You should see a file called shakespeare.txt.
   On orange:

   ```
   $ cd /tmp/shared
   $ ls
   $ tail shakespeare.txt
   ```

2. Create a directory in HDFS and upload the file to that directory. Check to make sure the file is in place.
   On orange, try the following statements and understand what they are returning:

```
$ hadoop fs –mkdir input
$ hadoop fs –put shakespeare.txt input
$ hadoop fs –ls /user
$ hadoop fs –ls /user/<user name>
$ hadoop fs –ls
$ hadoop fs –ls input
$ hadoop fs –tail input/shakespeare.txt
```

3. Run the hadoop fs –ls command to review the file permissions in HDFS.

```
$ hadoop fs –ls /user/<user_name>/input/shakespeare.txt
```

4. Locate information about the shakespeare file in the NameNode Web UI.

    a. Start Firefox

    b. Enter the following URL: http://orange:50070

    c. Click Utilities > “Browse the file system”, then navigate to the /user/<user_name>/data/shakespeare directory

    d. Verify that the permissions for the shakespeare file are identical to the permissions you observed when you ran the hadoop fs –ls command.

    e. Now select the shakespeare file in the NameNode Web UI file browser to bring up the File Information window.

    Observe that for each block in the file, only a single host – orange – is listed under Availability. This indicates that the file is replicated to a single host. CDH sets the replication number to 1 for a pseudo-distributed cluster.

    When you build a cluster with multiple nodes later in the course, the replication factor will be 3 and you will see 3 hosts listed under Availability.

    f. Choose Block 0 and note the value that appears in the Size field.
    You will need this value when you examine the file that contains this block.

    g. Select the value of the Block ID field and copy it.
    You will need this value for the next step of the exercise.

5. Locate the HDFS block in the file system.

    a. In your terminal session on orange, change to the /var/lib/hadoop-hdfs/cache/hdfs directory.

    ```
    $ cd /var/lib/hadoop-hdfs/cache/hdfs
    ```

b. Find the HDFS block within the /var/lib/hadoop-hdfs/cache/hdfs directory. For block_id, paste in the Block ID that you copied earlier.

```
$ sudo find . | grep <block_id>
```

c. Verify that two files appear in the find command output – one file with an extension .meta and another file without this extension.

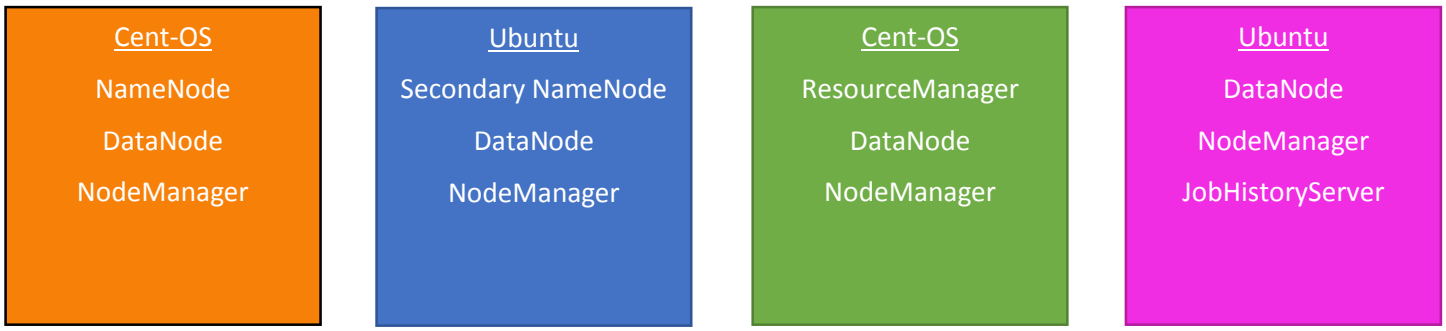
6. Use the sudo ls –l command to verify that the size of the file containing the HDFS block is exactly the size that was reported in the NameNode Web UI.

7. Remove the input directory, which contains the shakespeare file, from HDFS.

```
$ hadoop fs –rm –r input
```

Exercise 2: Creating a Hadoop Cluster

In this exercise you will create and test a fully distributed Hadoop cluster on your machine. You should have four hosts set up, but so far we have only been using one – orange. In this exercise you will stop the daemons running on orange, install Hadoop on the other hosts and then configure a cluster using all four hosts. See the image below to understand how your daemons will be deployed by the end of the exercise.



Having a limited number of hosts is not unusual for proof-of-concept projects in Hadoop. Bear in mind that some of your virtual machines run on different operating systems. The examples above don't mean you have to set up your cluster in this way, but it is the way these exercises will follow.

Installing Hadoop Software on the Cluster

In this task you will remove the Hadoop Secondarynamenode, ResourceManager and JobHistoryServer from orange and install Hadoop software on blue, green and pink.

When you installed Hadoop is pseudo-distributed mode on orange, you installed the NameNode, Secondary NameNode, DataNode, ResourceManager, NodeManager and JobHistoryServer on that host.

Now we want to install the following daemons on the other hosts.

- Secondary NameNode – blue
- DataNode – blue, green and pink
- NodeManager – blue, green and pink
- ResourceManager – green
- JobHistoryServer – pink

1. The first step is to stop the existing pseudo-distributed cluster running on orange. Enter the following commands on orange:

```
$ sudo service hadoop-hdfs-namenode stop
$ sudo service hadoop-hdfs-secondarynamenode stop
$ sudo service hadoop-hdfs-datanode stop
$ sudo service hadoop-yarn-resourcemanager stop
$ sudo service hadoop-yarn-nodemanager stop
```

```
$ sudo service hadoop-mapreduce-historyserver stop
```

2. Remove any log files that were created while the pseudo-distributed cluster was running. Enter the following command on orange:

```
$ sudo rm –rf /var/log/hadoop-*/*
```

Removing the log files that were created when you were running Hadoop on orange in pseudo-distributed mode will make it easier to locate the log files that are written after you start the distributed cluster.

3. Remove the Secondary NameNode, ResourceManager and JobHistoryServer from orange. On orange:

```
$ sudo yum remove -y hadoop-hdfs-secondarynamenode
$ sudo yum remove -y hadoop-yarn-resourcemanager
$ sudo yum remove -y hadoop-mapreduce-historyserver
```

4. Before you try to install anything on a node, remember that you need to set up a repository! We've seen how to do this for Cent-OS earlier, follow the steps below to set up one in Ubuntu.

    a. Create and open new file at /etc/apt/sources.list.d/cloudera.list

    ```
    sudo gedit /etc/apt/sources.list.d/cloudera.list
    ```

    b. Add the following to the document then save and close

    ```
    deb [arch=amd64]
    http://archive.cloudera.com/cdh5/ubuntu/trusty/amd64/cdh trusty-
    cdh5 contrib
    deb-src http://archive.cloudera.com/cdh5/ubuntu/trusty/amd64/cdh
    trusty-cdh5 contrib
    ```

5. Before using apt-get to install anything you should update it first.

```
$ sudo apt-get update
```

6. Install the Secondary NameNode on blue.
On blue:

```
$ sudo apt-get install hadoop-hdfs-secondarynamenode
```

7. Install the DataNode on blue, green and pink.
Run the following commands on blue, green and pink dependent upon the operating system

```
$ sudo yum install -y hadoop-hdfs-datanode
```

```
$ sudo apt-get install hadoop-hdfs-datanode
```

8. Install the ResourceManager on green.
   On green:

```
$ sudo yum install -y hadoop-yarn-resourcemanager
```

9. Install the NodeManager on blue, green and pink.
   On blue, green and pink:

```
$ sudo yum install -y hadoop-yarn-nodemanager
$ sudo apt-get install hadoop-yarn-nodemanager
```

10. Install the hadoop-mapreduce package on blue, green and pink. This package contains the mapreduce_shuffle auxiliary service which is needed for MapReduce jobs.
    On blue, green and pink:

```
$ sudo yum install -y hadoop-mapreduce
$ sudo apt-get install hadoop-mapreduce
```

11. Install the JobHistoryServer on pink.
    On pink:

```
$ sudo apt-get install hadoop-mapreduce-historyserver
```

Modifying the Hadoop Configuration Files

In this task you will edit four Hadoop configuration files on orange: core-site.xml, hdfs-site.xml, yarn-site.xml and mapred-site.xml. You will also edit the hadoop-env.sh script, setting environment variables that limit the heap size of Hadoop daemons.
When you have finished editing the files, you will copy them to blue, green and pink. When editing the configuration files you will need to use sudo e.g. sudo gedit core-site.xml. See below for an example of the format for the configuration file properties, you can have multiple properties.

```
<property>
        <name>property_name</name>
        <value>property_value</value>
</property>
```

1. Edit /etc/hadoop/conf/core-site.xml on orange, adding the following property and value between the <configuration> …. </configuration> tags. Add property names and values inside the <name> and <value> tags, and don't forget to use sudo when you edit the Hadoop configuration files. See below the table for what your file should include.

Name	Value
fs.defaultFS	hdfs://orange:8020

```
<configuration>
  <property>
      <name>fs.defaultFS</name>
      <value>hdfs://orange:8020</value>
  </property>
</configuration>
```

2. Edit etc/hadoop/conf/hdfs-site.xml on orange, adding the following values. Note that although you are adding properties for several different daemons, you will only be starting certain daemons on your machine depending on its role.

   Note that for the dfs.namenode.name.dir property, we are specifying two directories on the same disk. In production of course, these should be on different disks and you should also include an NFS mount.

Name	Value
dfs.namenode.name.dir	file:///disk1/dfs/nn,file:///disk2/dfs/nn
dfs.datanode.data.dir	file:///disk1/dfs/dn,file:///disk2/dfs/dn

   Your file contents for hdfs-site.xml should look like the below:

```
<configuration>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value> file:///disk1/dfs/nn,file:///disk2/dfs/nn</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value> file:///disk1/dfs/dn,file:///disk2/dfs/dn</value>
 </property>
</configuration>
```

3. Edit /etc/hadoop/conf/yarn-site.xml on orange, adding or changing the following values. Be cautious about copy and pasting.

Name	Value
yarn.resourcemanager.hostname	green
yarn.application.classpath	Leave the value specified
yarn.nodemanager.aux-services	mapreduce_shuffle
yarn.nodemanager.local-dirs	file:///disk1/nodemgr/local,file:///disk2/nodemgr/local
yarn.nodemanager.log-dirs	/var/log/hadoop-yarn/containers
yarn.nodemanager.remote-app-log-dir	/var/log/hadoop-yarn/apps
yarn.log-aggregation-enable	true

4. Edit /etc/hadoop/conf/mapred-site.xml on orange, adding the following values.

Name	Value
mapreduce.framework.name	yarn
mapreduce.jobhistory.address	pink:10020
mapreduce.jobhistory.webapp.address	pink:19888
yarn.app.mapreduce.am.staging-dir	/user

5. Because you are running on a Virtual Machine with limited RAM, you will reduce the heap size of the Hadoop daemons. Create a file called /etc/hadoop/conf/hadoop-env.sh on orange and add the following (be wary of copy and pasting):

```
export HADOOP_NAMENODE_OPTS="-Xmx64m"
export HADOOP_SECONDARYNAMENODE_OPTS="-Xmx64m"
export HADOOP_DATANODE_OPTS="-Xmx64m"
export YARN_RESOURCEMANAGER_OPTS="-Xmx64m"
export YARN_NODEMANAGER_OPTS="-Xmx64m"
export HADOOP_JOB_HISTORYSERVER_OPTS="-Xmx64m"
```

6. Before the next step you may need to change some permissions for us to copy things over to the Ubuntu machines.

```
$ sudo gedit /etc/ssh/sshd_config
```

Change the line **PermitRootLogin without-password** to **PermitRootLogin yes**

Save this file then execute the following command.

```
$ sudo service ssh restart
```

It may also be that you have issues with your root password. If that is the case, run the following command to change your root user password on Ubuntu.

```
$ sudo passwd root
```

7. Copy the core-site.xml, hdfs-site.xml, yarn-site.xml, mapred-site.xml and hadoop-env.sh files from orange to blue, green and pink. You can do this by changing the command below as needed. The first time you do this you may need to enter 'yes'. You will also need to enter the root user password each time.

```
scp /etc/hadoop/conf/core-site.xml root@blue:/etc/hadoop/conf/
```

Setting up the File System

In this task you will create the directories specified in the Hadoop configuration and set the ownership.

1. Create the directories specified earlier.
   On orange:

```
$ sudo mkdir –p /disk1/dfs/nn
$ sudo mkdir –p /disk2/dfs/nn
$ sudo mkdir –p /disk1/dfs/dn
$ sudo mkdir –p /disk2/dfs/dn
$ sudo mkdir –p /disk1/nodemgr/local
$ sudo mkdir –p /disk2/nodemgr/local
```

2. Change the ownership of the directories. The hdfs user should own the HDFS directories.
   On orange:

```
$ sudo chown –R hdfs:hadoop /disk1/dfs/nn
$ sudo chown –R hdfs:hadoop /disk2/dfs/nn
$ sudo chown –R hdfs:hadoop /disk1/dfs/dn
$ sudo chown –R hdfs:hadoop /disk2/dfs/dn
```

3. Change directory ownership so that the yarn user owns the NodeManager local directories.
   On orange:

```
$ sudo chown –R yarn:yarn /disk1/nodemgr/local
$ sudo chown –R yarn:yarn /disk2/nodemgr/local
```

4. Run ls commands on the /disk1 and /disk2 directories. Review the output to verify that you have created the right directories and set ownership correctly.

On orange:

```
$ ls –lR /disk1
$ ls –lR /disk2
```

5. Repeat steps 1 – 4 for each other virtual machine.

Formatting HDFS and Starting the HDFS Daemons

In this task you will format the HDFS file system and start the NameNode and DataNodes. Review the set up diagram at the start of this exercise. Remember, you will be starting a NameNode on orange, a Secondary NameNode on blue and DataNodes on all four hosts.

1. Format the NameNode.
   On orange:

   ```
   $ sudo –u hdfs hdfs namenode –format
   ```

   Note: if you are asked if you want to format the filesystem, enter 'y'

2. Start the NameNode daemon.
   On orange:

   ```
   $ sudo service hadoop-hdfs-namenode start
   ```

3. Run the sudo jps command on orange. You should see a process named NameNode if the daemon started successfully.

4. Run the sudo jps –v command on orange. Locate the –Xmx option in the command output and verify that the maximum heap size of the NameNode is the size that you specified in the hadoop-env.sh file.
   Note: If the –Xmx option is specified multiple times on the command output, the final occurrence is used to determine the heap size.

5. Review the log files for the NameNode from the command line.
   On orange:

   ```
   $ less /var/log/hadoop-hdfs/hadoop-hdfs-namenode-orange.log
   $ less /var/log/hadoop-hdfs/hadoop-hdfs-namenode-orange.out
   ```

   If the daemon started successfully the .out log file will contain information about system resources available to the daemon. Note that error messages might overwrite this information.
   If the daemon did not start correctly, identify the problem in the log file then attempt to correct the problem and start the NameNode again.

6. Start the NameNode web UI using the URL http://orange:50070

7. Select Utilities > Logs
   A list of files in the /var/log/hadoop-hdfs directory should appear.

8. Review the NameNode log files by using the NameNode web UI.

9. Start the Secondary NameNode daemon.
   On blue:

   ```
   $ sudo service hadoop-hdfs-secondarynamenode start
   ```

10. Run the sudo jps command on blue and verify that the Secondary NameNode is running.

11. Run the sudo jps –v command on blue to verify that the maximum heap size of the Secondary NameNode is the size that you configured in the hadoop-env.sh file.

12. Review the Secondary NameNode log files on blue by entering the following on the command line:

    ```
    $ less /var/log/hadoop-hdfs/hadoop-hdfs-secondarynamenode-blue.log
    $ less /var/log/hadoop-hdfs/hadoop-hdfs-secondarynamenode-blue.out
    ```

    If the Secondary NameNode started successfully, the .out log file will contain information about system resources available to the daemon. Error messages may overwrite this.

    If the Secondary NameNode did not start correctly, identify the issue in the log file, correct the issue and start the Secondary NameNode.

13. Start the DataNode daemons.
    Run the following command on all four hosts:

    ```
    $ sudo service hadoop-hdfs-datanode start
    ```

14. Run the sudo jps command on all four hosts to verify the DataNodes are running.

15. Run the sudo jps –v command on all four hosts to check the DataNode heap size is the same as you set in the hadoop-env.sh file.

16. Review all four hosts DataNode log files using either the web UI or the command line. Use some variation on the example commands below.

    ```
    $ less /var/log/hadoop-hdfs/hadoop-hdfs-datanode-blue.log
    $ less /var/log/hadoop-hdfs/hadoop-hdfs-datanode-blue.out
    ```

    URL: http://blue:50075

    If any of the daemons did not start successfully, investigate the issue, correct the problem

and restart the daemon. For the purpose of this project we will have the firewall off which is a common problem where you can't access web UIs. You can either create rules to allow certain access, so give that a go if you have time in your group project.

Creating Directories for YARN and MapReduce in HDFS

Now that you have an operational HDFS file system, you can create a few directories in HDFS that you will need to start the YARN and MapReduce daemons to run MapReduce jobs on your cluster.

You will create:

- /tmp directory
- YARN log directories
- /user directory and a directory for the training user
- /user/history directory

1. Create the needed directories.
   On any host in your cluster enter the following commands:

```
$ sudo –u hdfs hadoop fs –mkdir /tmp
$ sudo –u hdfs hadoop fs –chmod –R 1777 /tmp
$ sudo –u hdfs hadoop fs –mkdir –p /var/log/hadoop-yarn
$ sudo –u hdfs hadoop fs –chown yarn:mapred /var/log/hadoop-yarn
$ sudo –u hdfs hadoop fs –mkdir /user
$ sudo –u hdfs hadoop fs –mkdir /user/<user name>
$ sudo –u hdfs hadoop fs –chown <user name> /user/<user name>
$ sudo –u hdfs hadoop fs –mkdir /user/history
$ sudo –u hdfs hadoop fs –chmod 1777 /user/history
$ sudo –u hdfs hadoop fs –chown mapred:hadoop /user/history
```

Starting the YARN and MapReduce daemons

In this task you will start the YARN and MapReduce daemons and verify that the correct daemons are running on all the hosts in your cluster.

1. Start the ResourceManager daemon.
   On green:

```
$ sudo service hadoop-yarn-resourcemanager start
```

2. Run the sudo jps command on green to verify the presence of the ResourceManager daemon.

3. Run the sudo jps –v command on green to verify the maximum heap size is correct.

4. Review the ResourceManager log files using the command line.
   On green:

```
$ less /var/log/hadoop-yarn/yarn-yarn-resourcemanager-green.log
```

```
$ less /var/log/hadoop-yarn/yarn-yarn-resourcemanager-green.out
```

5. View the ResourceManager web UI using the URL http://green:8088

6. Select “Local logs” under Tools. A list of log files should appear.

7. Review the ResourceManager log files by using the ResourceManager web UI.

8. Start the NodeManagers.
   Run the following command on all four hosts on your cluster.

   ```
   $ sudo service hadoop-yarn-nodemanager start
   ```

9. Run the sudo jps command on all hosts to verify they are running.

10. Run the sudo jps –v to verify the maximum heap size is correct.

11. Review the NodeManager files either using the command line or the web UI. On the web UI, click Nodes under Cluster, then select the number in the Active Nodes column (which should be 4), then select one of the active NodeManagers, then Tools, then Local Logs.

12. Start the JobHistoryServer daemon.
    On pink:

    ```
    $ sudo service hadoop-mapreduce-historyserver start
    ```

13. Run the sudo jps command on pink to verify the JobHistoryServer daemon is there.

14. Run the sudo jps –v command to ensure the heap size is correct.

15. Review the JobHistoryServer log files using the command line.
    On pink:

    ```
    $ less /var/log/hadoop-mapreduce/mapred-mapred-historyserver-pink.out
    ```

    Unlike other Hadoop daemons, the JobHistoryServer does not use the .log file for the significant portion of its output, it uses the .out file.

16. View the JobHistoryServer Web UI using the URL http://pink:19888

17. Select Local Logs under Tools. A list of log files should appear.

18. Review the JobHistoryServer log files using the JobHistoryServer Web UI.

Testing the Cluster

If you have performed all previous steps correctly you should now have a fully operational Hadoop cluster.
In this task you will run some hadoop fs commands and a MapReduce job to verify the cluster is

working correctly. Then you will use the Hadoop Web UI to observe how running the job in a cluster environment differs from pseudo-distributed mode.

1. Upload your shakespeare data to the cluster.
   On orange:

   ```
   $ hadoop fs –mkdir –p shakespeare
   $ hadoop fs –put shakespeare.txt shakespeare
   ```

2. Access the NameNode Web UI.

3. Click Utilities > "Browse the file system", then navigate to the /user/*<user name>*/shakespeare directory and select the shakespeare.txt file.

4. Verify that Hadoop replicated the shakespeare.txt file three times. To see which hosts the file was replicated to, review the information in the Availability field for Block 0.

5. Run a MapReduce job.
   On orange:

   ```
   $ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
   wordcount shakespeare output
   ```

   One the job has run, examine the output to ensure it was successful.

   ```
   $ hadoop fs –ls output
   ```

6. Use the ResourceManager Web UI to determine which hosts the ApplicationMaster, Mapper task and Reducer task ran on.

   Select the Job ID of your MapReduce job (under ID). The Application Overview page appears (the page title appears on the right side of the page). The host on which the ApplicationMaster ran appears in the Node column.

   Then select the "History" link (next to the Tracking URL). The job history for your job appears.

   Click "Map" under Task Type.

   Click the link for the Mapper task. The node on which the Mapper task ran appears in the Node column.

   Return to the page for the word count job, click the Reduce link then drill down to the Reducer task entry. The node on which the Reducer task ran appears in the Node column.

Exercise 3: Querying HDFS with Hive and Impala

In this exercise you will install and configure Apache Hive and Cloudera Impala to query data stored in HDFS.

You will start by configuring and starting a ZooKeeper ensemble. ZooKeeper is required for Hive query concurrency and is a prerequisite for HiveServer2.

Next, you will configure Hive to support concurrent queries and to use a Hive metastore. Then you will initialise the metastore. This guide will take you through setting up a MySQL metastore, however, you are encouraged to look up different databases you can use (this link also explains how to set up a metastore with PostgreSQL and Oracle - http://www.cloudera.com/documentation/cdh/5-1-x/CDH5-Installation-Guide/cdh5ig_hive_metastore_configure.html). You will then install and start the Hive Metastore service.

Next, you will install and configure HiveServer2 on orange so that you can run queries from any system on which the Beeline shell is installed.

Next, you will populate HDFS with data in the movie rating table and run queries against it using the Beeline shell.

Installing, Configuring and Starting a ZooKeeper Ensemble

1. Install ZooKeeper. For now we will only be doing this on 2 of our machines, though this usually isn't recommended. Generally 3 or 5 zookeeper hosts should be running, as they work by communicating by majority vote. When you set up your full team's cluster you should have 5 zookeeper hosts running so bear this in mind.
   On orange and green:

   ```
   $ sudo yum --assumeyes install zookeeper-server
   ```

2. Initialise the three ZooKeeper servers with a unique ID.
   On orange:

   ```
   $ sudo service zookeeper-server init --myid 1
   ```

   On green:

   ```
   $ sudo service zookeeper-server init --myid 2
   ```

3. Using sudo, edit the ZooKeeper configuration file at the path /etc/zookeeper/conf/zoo.cfg on orange. Append the following lines to the end of the file:

   ```
   server.1=orange:2888:3888
   server.2=green:2888:3888
   ```

4. Using sudo, create a configuration file for Java options at the path /etc/zookeeper/conf/java.env on orange. The file should have a single line as follows (be wary of copy and pasting):

```
export JVMFLAGS=”-Xmx64m”
```

5. Copy the ZooKeeper configuration files to the other host that will run ZooKeeper servers.
   On orange:

```
$ cd /etc/zookeeper/conf
$ scp zoo.cfg root@green:/etc/zookeeper/conf/
$ scp java.env root@green:/etc/zookeeper/conf/
```

6. Start the two ZooKeeper servers.
   On orange and green:

```
$ sudo service zookeeper-server start
```

7. Run the sudo jps –v command to verify that the ZooKeeper processes are up and running on orange and green. The process name is QuorumPeerMain and check that the maximum heap size is set to 64MB.

Installing and Configuring Hive

By default, Hive is configured to use an Apache Derby metastore. In this task you will configure Hive to use a MySQL metastore that you will create later. You will also configure Hive to support concurrent queries. You will create a scratch directory in HDFS that can be used by all users when running Hive queries. Note that the Impala servers will share the Hive metastore.

1. Install Hive.
   On orange:

```
$ sudo yum --assumeyes install hive
```

2. Before we go any further, we’re also going to ensure we have MySQL installed.

```
$ sudo yum install mysql-server

$ sudo service mysqld start

$ sudo yum install mysql-connector-java

$ /usr/bin/mysql_secure_installation

$ sudo /sbin/chkconfig mysqld on
```

3. Using sudo, modify the /etc/hive/conf/hive-site.xml file on orange as follows. Be wary of copy and pasting.

Name	Value
javax.jdo.option.ConnectionURL	jdbc:mysql://orange:3306/metastore
javax.jdo.option.ConnectionDriverName	com.mysql.jdbc.Driver
javax.jdo.option.ConnectionUserName	hive
javax.jdo.option.ConnectionPassword	hive
datanucleus.autoCreateSchema	false
datanucleus.fixedDatastore	true
datanucleus.autoStartMechanism	SchemaTable
hive.metastore.uris	thrift://***<orange_IP_address>***:9083 (See note after table as to how to find this)
hive.support.concurrency	True
hive.zookeeper.quorum	orange,green

Note: The hive.metastore.uris configuration parameter takes an IP address of a fully qualified host name (FQHN) for its value. To get the IP address of orange follow the instructions below. Alternatively, use the ifconfig command.

a. Open the /etc/hosts file

```
$ cat /etc/hosts
```

b. Look for a line similar to the following:

192.168.123.1 orange

c. In this example the value for hive.metastore.uris would be thrift://192.168.123.1:9083

4. Create the Hive warehouse directory in HDFS.
On any host:

```
$ sudo –u hdfs hadoop fs –mkdir –p /user/hive/warehouse
$ sudo –u hdfs hadoop fs –chmod 1777 /user/hive
```

5. Make the MySQL connector jar file accessible to Hive.
On orange:

```
$ sudo ln –s /usr/share/java/mysql-connector-java.jar /usr/lib/hive/lib
```

Initialising the Hive Metastore in MySQL

Hive and Impala can both make use of a single, common Hive metastore. You will create a MySQL database for the metastore then configure Hive and Impala to use it for storing table definitions.

6. Start a MySQL session and create the user hive and assign a password and privileges to this user.
   On orange

   ```
   $ mysql –u root –p
   mysql> CREATE DATABASE metastore;
   mysql> USE metastore;
   mysql> CREATE USER 'hive'@'orange' IDENTIFIED BY 'hive';
   mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'hive'@'orange';
   mysql> GRANT SELECT, INSERT, UPDATE, DELETE, LOCK TABLES, EXECUTE,
   CREATE, ALTER, INDEX ON metastore.* TO 'hive'@'orange';
   mysql> FLUSH PRIVILEGES;
   ```

7. Open a new terminal window on orange. Run a command to initialise the Hive metastore in the new terminal window.
   On orange:

   ```
   $ sudo /usr/lib/hive/bin/schematool –dbType mysql –initSchema
   ```

8. Return to the terminal window in which your MySQL session is running. Remove the ALTER and CREATE privileges from the user hiveuser.

   ```
   mysql> REVOKE ALTER, CREATE ON metastore.* FROM 'hive'@'orange';
   ```

9. Run the SHOW tables command in your MySQL session to verify that you have created the metastore correctly.

   ```
   mysql> SHOW tables;
   ```

   A list of the Hive metastore tables should appear.

10. Quit your MySQL session.

    ```
    mysql> quit;
    ```

Installing and Starting the Hive Metastore Service

The Hive Metastore is a Thrift server that Hive and Impala daemons use to access the Hive metastore.

1. Install the Hive Metastore service.
   On orange:

   ```
   $ sudo yum install --assumeyes hive-metastore
   ```

2. Start the service.
   On orange:

   ```
   $ sudo service hive-metastore start
   ```

3. Run the sudo jps command on orange. Verify that the RunJar process is active, then run the ps –ef | grep RunJar command on orange (The RunJar process runs the Hive Metastore server).

Installing, Configuring and Starting HiveServer2

In this task you will install HiveServer2, make some configurations, start HiveServer2 and run a quick test to validate it.

1. Install HiveServer2
   On orange:

   ```
   $ sudo yum install --assumeyes hive-server2
   ```

2. Add the following line to the end of the /etc/default/hive-server2 file on orange. This configuration ensure that HiveServer2 uses the correct version of MapReduce when you run Hive queries.

   ```
   export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
   ```

3. Start HiveServer2.
   On orange:

   ```
   $ sudo service hive-server2 start
   ```

4. Verify that you can run a Hive command from the Beeline shell.
   On orange:

   ```
   $ beeline –u jdbc:hive2://orange:10000/default –n training
   > SHOW tables;
   ```

   No tables should appear because you haven't define any Hive tables yet, however, you should not see any error messages.

5. Exit the Beeline shell.

   ```
   > !quit
   ```

Modifying the HDFS Configuration for Impala

In this task you will configure some HDFS parameters that dramatically improve Impala performance by letting Impala (and other applications) read DataNode files directly.

1. Add four properties to the /etc/hadoop/conf/hdfs-site.xml file on orange as follows.

Name	Value
dfs.client.read.shortcircuit	true
dfs.domain.socket.path	/var/run/hadoop-hdfs/dn._PORT
dfs.client.file-block-storage-locations.timeout.millis	10000
dfs.datanode.hdfs-blocks-metadata.enabled	true

   Note: Be sure to enter the value for the dfs.domain.socket.path configuration parameter exactly as it appears.

2. Copy the hdfs-site.xml file from orange to blue, green and pink.

3. Restart the DataNodes on orange, blue, green and pink.
   On all four hosts:

```
$ sudo service hadoop-hdfs-datanode restart
```

4. Review the DataNode logs and run sudo jps to ensure the DataNodes are up and running on each host.

Installing, Configuring and Starting the Impala Daemons

In this task you will install the Impala State Store Server on green, the Impala Catalog Server on green, and the Impala Server on all four hosts. Then you will configure Impala and start the Impala server daemons.

1. Install the Impala State Store Server.
   On green:

```
$ sudo yum install --assumeyes impala-state-store
```

2. Install the Impala Catalog Server.
   On green:

```
$ sudo yum install --assumeyes impala-catalog
```

3. Install the Impala Server.
   On all four hosts:

```
$ sudo yum install --assumeyes impala-server
```

```
$ sudo apt-get install impala-server
```

4. Configure the Impala State Store Server and Impala Catalog Server locations.
   On orange, open the /etc/default/impala file (using sudo) and change the values of the IMPALA_CATALOG_SERVICE_HOST and IMPALA_STATE_STORE_HOST variables from 127.0.0.1 to green. The rest of your document should look like the text below, DO NOT COPY & PASTE:

```
IMPALA_STATE_STORE_ARGS=" \
 -log_dir=${IMPALA_LOG_DIR} \
 -catalog_service_host=${IMPALA_CATALOG_SERVICE_HOST} \
 -state_store_port=${IMPALA_STATE_STORE_PORT} \
 -state_store_host=${IMPALA_STATE_STORE_HOST}"

IMPALA_CATALOG_ARGS=" \
 -log_dir=${IMPALA_LOG_DIR} \
 -catalog_service_host=${IMPALA_CATALOG_SERVICE_HOST} \
 -state_store_port=${IMPALA_STATE_STORE_PORT} \
 -state_store_host=${IMPALA_STATE_STORE_HOST}"

IMPALA_SERVER_ARGS=" \
   -log_dir=${IMPALA_LOG_DIR} \
   -catalog_service_host=${IMPALA_CATALOG_SERVICE_HOST} \
   -state_store_port=${IMPALA_STATE_STORE_PORT} \
   -use_statestore \
   -state_store_host=${IMPALA_STATE_STORE_HOST} \
   -be_port=${IMPALA_BACKEND_PORT}"

export ENABLE_CORE_DUMPS=false
```

5. Copy the Hive configuration on orange to the Impala configuration directory.
   On orange:

```
$ cd /etc/hive/conf
$ sudo cp hive-site.xml /etc/impala/conf
```

6. Copy the Hadoop configuration files required by Impala from orange to the Impala configuration dictionary.
   On orange:

```
$ cd /etc/hadoop/conf
$ sudo cp core-site.xml /etc/impala/conf
$ sudo cp hdfs-site.xml /etc/impala/conf
$ sudo cp log4j.properties /etc/impala/conf
```

7. Copy the Impala configuration files and the /etc/default/impala file from orange to the other three hosts.

8. Start the Impala State Store daemon.
   On green:

   ```
   $ sudo service impala-state-store start
   ```

   Note: If you run the sudo jps command here, you will not see the Impala State Store daemon because it is not a Java process. We will use a different command to verify the Impala processes are running later.

9. Start the Impala Catalog Server daemon.
   On green:

   ```
   $ sudo service impala-catalog start
   ```

   If you are having issues at this point, check your /etc/default/impala file.

10. Start the Impala Server daemons.
    On all four hosts:

    ```
    $ sudo service impala-server start
    ```

11. Locate the Impala log file in the /var/log/impala directory and review the log file to verify that the Impala daemons started without any problems. Empty log files indicate that no problems have occurred.

12. Verify the Impala daemons are running.
    On all four hosts:

    ```
    $ ps –ef | grep impala
    ```

    You should see the impalad daemon running on all four hosts. On green you should also see the statestored and catalogd daemons.

13. Start a browser and navigate to http://orange:25000 to review the Impala status pages.

Running Hive Queries

In this task you will populate HDFS with data from the movierating table using Sqoop. Then you will define the movierating table in Hive and run a simple query against the table. There are a few things we should check before we get started though…

1. Before now we've been happily continuing on without worrying about whether our nodes have synced times. For this we use the ntp daemon, and you will need to make sure these are synced up when you are creating any kind of cluster.

2. You've probably already got ntp installed, but just in case you don't, use the following install commands:

```
$ sudo yum install ntp

$ sudo apt-get install ntp
```

3. Open the file at /etc/ntp.conf file. You will see some lines that look like the following below. Make sure every node in your cluster is accessing the same ntp server, then save and exit.

```
server 0.centos.pool.ntp.org
server 1.centos.pool.ntp.org
server 2.centos.pool.ntp.org
…
```

4. You could optionally configure the ntp service to run at reboot on the centos machines.

```
$ sudo chkconfig ntpd on
```

5. Perform the corresponding commands on each machine:

```
$ sudo service ntpd start                    //centos
$ sudo service ntp start                     //Ubuntu

$ sudo ntpdate –u <ntp_server_address>

$ sudo hwclock --systohc
```

6. You should also ensure you have a user in MySQL that can access your databases from any location for the following steps. You could specify each address you will allow, but the below command will grant access to all tables in all databases to the hive user from any IP. Use this if necessary but consider the security implications and how you could make your server more secure. You do need to perform both of these commands.

```
mysql> GRANT ALL ON *.* to hive@localhost IDENTIFIED BY ‘hive’;
mysql> GRANT ALL ON *.* to hive@’%’ IDENTIFIED BY ‘hive’;
```

7. Before we start using Sqoop you should import the movielens database into MySQL. The quickest way to do this is to create a database for movielens then run the import script. You will need to have the movielens_import.sql script and the necessary files for it to load into MySQL. Read the import script so you understand it, then run it in the MySQL terminal to load it. Make sure you have given your hive user access to the database.

```
mysql> CREATE DATABASE movies;
mysql> USE movies;
mysql> \. movielens_import.sql
mysql> grant all on movies.* to hive@’%’;
```

8. Install Sqoop and create a link to the MySQL JDBC driver.

```
$ sudo yum install –y sqoop
```

```
$ sudo ln –s /usr/share/java/mysql-connector-java.jar /usr/lib/sqoop/lib/
```

Use the 'sqoop help' command for information on sqoop!

9. There are a number of sqoop commands you can use for exploring your data such as sqoop list-databases, and sqoop list-tables.

```
$ sqoop list-databases --connect jdbc:mysql://orange --username <user_name>
--password <password>

$ sqoop list-tables --connect jdbc:mysql://orange/movies --username
<user_name> --password <password>
```

10. Once the movielens database is in MySQL, import the ratings table into HDFS. The ratings table does not have a primary key, only foreign keys, and so we need to specify how the map jobs that import the data will split the work by using the --split-by option.
On orange:

```
$ sqoop import --connect jdbc:mysql://orange/movies --table ratings --fields-
terminated-by '\t' --username <user_name> --password <password> --split-
by=itemID
```

11. Review the data loaded into HDFS.
On any host:

```
$ hadoop fs –ls ratings

$ hadoop fs –cat ratings/part-m-00000
```

12. Start the Beeline shell and connect to HiveServer2.
On orange:

```
$ beeline –u jdbc:hive2://orange:10000 –n hive
```

13. Define the movierating table in Hive. Note that this is just creating an external table, which is actually outside the Hive warehouse, we are just directing Hive to where it is.

```
> CREATE EXTERNAL TABLE movierating
> (userid INT, movieid STRING, rating TINYINT, timestamp INT)
> ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
> LOCATION '/user/<user_name>/ratings';
```

14. Verify that you created the movierating table in the Hive metastore.
On orange:

```
> SHOW tables;
```

You should see an entry for the movierating table.

15. Run a simple test query that counts the number of rows in the movieratings table.
    On orange:

    ```
    > SELECT COUNT(*) FROM movierating;
    ```

    Make a note of the amount of time the query takes to execute when running in Hive.

16. Terminate the Beeline shell.
    On orange:

    ```
    > !quit
    ```

Running Impala Queries

In this task you will install the Impala shell, then run the same query from the previous question in Impala and compare the performance.

1. Install the Impala shell.
   On orange:

   ```
   $ sudo yum install -y impala-shell
   ```

2. Start the Impala shell.
   On orange:

   ```
   $ impala-shell
   ```

3. Connect to the Impala server running on green.

   ```
   > CONNECT green;
   ```

   Note: You can connect to any of your four Impala servers to run queries, we are using green as an example.

4. Because you defined a new table from outside of Impala, you must now refresh the Hive metadata in Impala.

   ```
   > INVALIDATE METADATA;
   ```

5. In Impala, run the same query against the movierating table that you ran in Hive.

   ```
   > SELECT COUNT(*) FROM movierating;
   ```

   Compare the amount of time it took to run the query in Impala to the amount of time it took in Hive.

Exit the Impala shell.

```
> quit;
```

Exercise 4: Configuring HDFS High Availability

In this exercise you will reconfigure HDFS for high availability, eliminating the NameNode as a single point of failure.

You will start by shutting down unnecessary servers that you will no longer be using. You will then modify two of the Hadoop configuration files and install three journal nodes on your system.

An ensemble of ZooKeeper nodes is required for HDFS high availability. You will use the ZooKeeper ensemble you created earlier.

Before you start the NameNodes you must initialise the shared edits directory for your existing NameNode and bootstrap the new standby NameNode. Finally, you will start the NameNodes then install and start the two ZooKeeper Failover Controllers. After the Failover Controllers have been started, you will test your configuration by intentionally bringing one of the servers down.

The NameNodes on orange and blue will be deployed in an HA configuration. The JournalNodes on orange and green support the HA configuration. The ZooKeeper Failover Controllers on orange and blue provide automatic failover capability. The ZooKeeper servers on orange and green support the ZooKeeper Failover Controllers.

Bringing Down Unneeded Servers

You will no longer use Hive, Impala or Hue for the remaining administration exercises so we'll shut down the servers associated with them to improve performance. Of course, you may want to bring them back up again later.

1. Shut down the Hive Metastore service.
   On orange:

   ```
   $ sudo service hive-metastore stop
   ```

2. Shut down HiveServer2.
   On orange:

   ```
   $ sudo service hive-server2 stop
   ```

3. Shut down the Impala Servers.
   On all four hosts:

   ```
   $ sudo service impala-server stop
   ```

4. Shut down the Impala Catalog Server.
   On green:

   ```
   $ sudo service impala-catalog stop
   ```

5. Shut down the Impala State Store Server.
   On green:

   ```
   $ sudo service impala-state-store stop
   ```

Bringing Down the Existing HDFS Configuration

Before you reconfigure the cluster, you will save off your Hadoop configuration and then stop all of the HDFS daemons on your cluster:

- The NameNode on orange
- The Secondary NameNode on green
- DataNodes on orange, blue, green and pink

Note that you do not need to stop the YARN and MapReduce daemons while configuring the HDFS high availability.

1. Back up your cluster's current configuration so that it is available in case you run into problems during the exercise.
   On orange:

   ```
   $ mkdir /home/<user_name>/backup_config
   $ cp /etc/hadoop/conf/* /home/<user_name>/backup_config
   ```

2. Stop the Secondary NameNode.
   On blue:

   ```
   $ sudo service hadoop-hdfs-secondarynamenode stop
   ```

3. Stop the NameNode.
   On orange:

   ```
   $ sudo service hadoop-hdfs-namenode stop
   ```

4. Stop the DataNodes.
   On all four hosts:

   ```
   $ sudo service hadoop-hdfs-datanode stop
   ```

Modifying the Hadoop Configuration

In this task you will modify some configuration parameters to support the new configuration.

1. On orange, modify the following configuration parameter in the /etc/hadoop/conf/core-site.xml file.

Name	Value
fs.defaultFS	hdfs://mycluster

   You will define mycluster in the next step when you configure hdfs-site.xml.

   Then add the following configuration parameter to the core-site.xml file.

Name	Value

ha.zookeeper.quorum	orange:2181,green:2181

2. Add configuration parameters for HDFS HA to /etc/hadoop/conf/hdfs-site.xml file on orange.

Name	Value
dfs.nameservices	mycluster
dfs.ha.namenodes.mycluster	nn1,nn2
dfs.namenode.rpc-address.mycluster.nn1	orange:8020
dfs.namenode.rpc-address.mycluster.nn2	green:8020
dfs.namenode.http-address.mycluster.nn1	orange:50070
dfs.namenode.http-address.mycluster.nn2	green:50070
dfs.namenode.shared.edits.dir	qjournal://orange:8485; green:8485/mycluster
dfs.journalnode.edits.dir	/disk1/dfs/jn
dfs.client.failover.proxy.provider.mycluster	org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
dfs.ha.fencing.methods	shell(/bin/true)
dfs.ha.automatic-failover.enabled	true

Note: No fencing will be performed during failover as part of specifying shell(true). This configuration is suitable for quick deployment or proof-of-concept. Optionally, after you have high availability working, you could change the configuration to use sshfence as the fencing method, generate an SSH key, and configure its location.

3. Copy the modified Hadoop configuration to the other nodes in your cluster.

Installing JournalNode Software and Starting the JournalNodes

Now you will install the HDFS JournalNode software on the hosts that will serve JournalNodes: orange and green. Then you will start the JournalNodes.

1. Install the JournalNode software.
   On orange and green:

   ```
   $ sudo yum install -y hadoop-hdfs-journalnode
   ```

2. Create the shared edits directory on all three hosts that will run JournalNodes. You configured this directory with the dfs.journalnode.edits.dir parameter in the hdfs-site.xml file.
   On orange and green:

   ```
   $ sudo mkdir –p /disk1/dfs/jn
   ```

3. Change the ownership of the shared edits directory on all three JournalNodes to user hdfs and group hadoop.
   On orange and green:

   ```
   $ sudo chown –R hdfs:hadoop /disk1/dfs/jn
   ```

4. Reduce the heap size of the JournalNodes by adding the following line to etc/hadoop/conf/hadoop-env.sh on orange:

   ```
   export HADOOP_JOURNALNODE_OPTS=”-Xmx64m”
   ```

5. Copy the Hadoop configuration from orange to the other three nodes in your cluster.

6. Start the JournalNodes.
   On orange and green:

   ```
   $ sudo service hadoop-hdfs-journalnode start
   ```

7. Run the sudo jps command to verify that the JournalNode process is up and running on orange and green.

8. Review the JournalNode log files on orange and green to determine if problems occurred during startup.

   NOTE: Similarly with the Zookeeper daemons, we should really have an odd number of JournalNodes, so when you set up your group cluster you should initialise more.

Initilialising the Shared Edits Directory and Starting the First NameNode

When converting a non-HA deployment to have high availability, you must initialise the shared edits directory with the edits data from the local NameNode edits directories. After you have initilialised the shared edits directory, you will start the nn1 NameNode on orange.

Note: the dfs.ha.namenodes.mycluster configuration parameter defines the two NameNodes in your cluster as nn1 and nn2. The dfs.namenode.rpc-address.mycluster.nn1 parameter specifies that the nn1 NameNode will run on orange.

1. Initialise the shared edits directory.
   On orange:

   ```
   $ sudo –u hdfs hdfs namenode –initializeSharedEdits
   ```

2. Start the nn1 NameNode.
   On orange:

   ```
   $ sudo service hadoop-hdfs-namenode start
   ```

3. Run the sudo jps command and review the NameNode log files on orange and verify that the NameNode started correctly.

4. Verify that the service state of the nn1 NameNode running on orange is Standby.
   On orange:

   ```
   $ sudo –u hdfs hdfs haadmin –getServiceState nn1
   ```

   The service state of this NameNode will transition to Active after you start the ZooKeeper Failover Controllers.

5. Attempt to list files in the /user/<user_name>/ directory.
   On any host in your cluster:

   ```
   $ hadoop fs –ls
   ```

   You are not able to access HDFS because no NameNodes are in the Active status. Enter Ctrl+C to stop the HDFS operation.

Installing, Bootstrapping and Starting the Standby NameNode

Before you can start the standby NameNode on green you need to install the software and copy the contents of the Active NameNode's metadata directories.

1. Install NameNode software on the host that will run the nn2 NameNode.
   On green:

   ```
   $ sudo yum install hadoop-hdfs-namenode
   ```

2. Prepare the nn2 NameNode by bootstrapping it.
   On green:

   ```
   $ sudo –u hdfs hdfs namenode –bootstrapStandby
   ```

3. Start the nn2 NameNode.
   On green:

   ```
   $ sudo service hadoop-hdfs-namenode start
   ```

4. Run the sudo jps command and review the NameNode log file on blue to verify the NameNode started correctly.

5. Verify that the service state of the nn2 NameNode is Standby.
   On any host:

   ```
   $ sudo –u hdfs hdfs haadmin –getServiceState nn2
   ```

6. Attempt to list files in the /user/training/orange directory.
   You are still unable to access HDFS because no NameNodes are in the Active status.
   Enter Ctrl+C to stop the HDFS operation.

Installing, Formatting, and Starting the ZooKeeper Failover Controllers

You have already put a couple of pieces in place to prepare for installation of the ZooKeeper Failover Controller. Now it is time to install the Failover Controller software, format a znode in ZooKeeper that the controller will use to keep track of the Active and Standby nodes, and start the Failover Controller. For this instance we will only be setting up one, again, this is not recommended and you should set up more in your group cluster.

1. Install the ZooKeeper Failover Controller software.
   On orange and green:

   ```
   $ sudo yum install -y hadoop-hdfs-zkfc
   ```

2. Reduce the heap size of the ZooKeeper Failover Controllers by adding the following line to /etc/hadoop/conf/hadoop-env.sh on orange:

   ```
   export HADOOP_ZKFC_OPTS=”-Xmx64m”
   ```

3. Copy the Hadoop configuration from orange to the other three nodes in your cluster.

4. Format the znode in ZooKeeper that keeps track of the Active and Standby nodes.
   On orange and green:

   ```
   $ sudo –u hdfs hdfs zkfc –formatZK
   ```

5. Start the ZooKeeper Failover Controllers.
   On orange and green:

   ```
   $ sudo service hadoop-hdfs-zkfc start
   ```

6. Run the sudo jps command and review the ZooKeeper Failover Controller log files to verify that the ZooKeeper Failover Controllers on orange and blue started correctly.

7. Verify that the service state of the nn1 NameNode on orange is Active and that the service state of the nn2 NameNode on blue is Standby.
   On any host:

   ```
   $ sudo –u hdfs hdfs haadmin –getServiceState nn1
   $ sudo –u hdfs hdfs haadmin –getServiceState nn2
   ```

8. Attempt to list files in the /user/<user_name> directory in HDFS.
   You should not be able to access this directory!

Starting the DataNodes and Restarting the MapReduce Daemons

Before you can test the deployment you will need to start the DataNodes. You will also need to restart the ResourceManager and NodeManager daemons as these communicate with HDFS and will need the updated configuration.

1. Start the DataNodes.
   On all four hosts:

   ```
   $ sudo service hadoop-hdfs-datanode start
   ```

2. Review the DataNode log files and verify that the DataNodes started correctly.

3. Restart the ResourceManager.
   On green:

   ```
   $ sudo service hadoop-yarn-resourcemanager restart
   ```

4. Restart the NodeManagers.
   On all four hosts:

   ```
   $ sudo service hadoop-yarn-nodemanager restart
   ```

5. Restart the JobHistoryServer.
   On pink:

   ```
   $ sudo service hadoop-mapreduce-historyserver restart
   ```

6. Run the sudo jps command and review the ResourceManager and NodeManager log files and verify that they started correctly.

Testing the HDFS High Availability Deployment

You are now ready to test your high availability deployment.

1. Verify that the Hadoop processes running on your hosts are expected with sudo jps.

   orange – NameNode, JournalNode, DFSZKFailoverController, QuorumPeerMain, DataNode, NodeManager

   blue – DataNode, NodeManager

   green – JournalNode, QuorumPeerMain, DataNode, ResourceManager, NodeManager, NameNode

   pink – DataNode, NodeManager, JobHistoryServer

2. View the NameNode Web UI on orange. The first line should be “NameNode ‘orange:8020’ (active)”

3. View the NameNode Web UI on green. The first line should be “NameNode ‘green:8020’ (standby)”

4. Bring down the nn1 NameNode.
   On orange:

   ```
   $ sudo service hadoop-hdfs-namenode stop
   ```

5. Using the hdfs haadmin –getServiceState command, verify that the service state of the nn2 NameNode is now Active, and that you can no longer query the service state of the nn1 NameNode.

6. Bring the nn1 NameNode back up.
   On orange:

   ```
   $ sudo service hadoop-hdfs-namenode start
   ```

7. Verify that the service state of the nn2 NameNode is still Active and that now the service state of the nn1 NameNode is Standby.

8. Bring down the nn2 NameNode.
   On green:

   ```
   $ sudo service hadoop-hdfs-namenode stop
   ```

9. Verify that the service state of the nn1 NameNode is now Active, and that you can no longer query the service state of the nn2 NameNode.

10. Bring the nn2 NameNode back up.
    On green:

    ```
    $ sudo service hadoop-hdfs-namenode start
    ```

11. Verify that the service state of the nn1 NameNode is still Active, and that now the service state of the nn2 NameNode is Standby.

Exercise 5: Breaking & Healing The Cluster

Undertake this exercise when uploading a large part of the data for your project work if you want to observe Hadoop's features.

1. As the file is being uploaded to HDFS, launch a web browser and view the NameNode Web UI. Using File Browser to look at the /user/training/orange directory in HDFS, observe the size and name of the file as it is being uploaded, refreshing the page periodically.

2. Locate a block that has been replicated on orange. In the NameNode Web UI, navigate to the /user/training/orange/access_log file and select the file for the File Information window to appear.
   Find a block with a replica on orange. Once you have found one, note the block ID. We will revisit this block later.

3. Now, intentionally cause a failure and observe what happens.
   Stop the DataNode on orange:

   ```
   $ sudo service hadoop-hdfs-datanode stop
   ```

4. Visit the NameNode;s Web UI and click on 'Live Nodes'. Refresh the browser a few times and notice that the 'Last Contact' value for the orange DataNote keeps increasing.

5. Run the hdfs fsck / command to see that the NameNode currently thinks there are no problems.
   On any host:

   ```
   $ sudo –u hdfs hdfs fsck /
   ```

6. View the NameNode Web UI and confirm that you now have one 'dead node'.

7. View the location of the block from the file you investigated in the previous exercise. Notice that Hadoop has automatically re-replicated the data to another host to retain the three-time replications.

8. Run the hdfs fsck command to observe that the filesystem is still healthy.
   On any host:

   ```
   $ sudo –u hdfs hdfs fsck /
   ```

9. Run the hdfs dfsadmin –report command to see that one dead DataNode is now reported.
   On any host:

   ```
   $ sudo –u hdfs hdfs dfsadmin –report
   ```

10. Restart the DataNode on orange to bring your cluster back to full strength.

    ```
    $ sudo service hadoop-hdfs-datanode start
    ```

Challenges

These are to be undertaken within your group and are purposefully left without instructions. It is up to you to use the knowledge you have gained so far and investigate the solutions within your team. The first task **must** be undertaken first but the others may be completed in any order.

When you are creating your cluster and installing software, document your process. This will form part of your final report for Big Data, and it is important to make notes as you go to use later. It is encouraged that you start writing up the administrative side of the report now as you go through it and it is fresh in your mind.

1. Link together the virtual machines on each member's computer to create one larger cluster. You will need to rearrange what is installed where and make sure it is functioning as one cluster, not many. For example, you should have only one active namenode and one standby namenode within your whole cluster. Consider having a node that is only running the namenode so it has enough resources. Try implementing 'rack awareness' in your cluster, for example, have each physical machine act as a rack, that way if a single machine loses power for whatever reason, you've definitely not lost any data!

2. Integrate Greenplum into your cluster.

3. You looked at a range of SQL and NoSQL databases earlier in the course. Try connecting MySQL, Cassandra, HBase, and MongoDB into your cluster.

4. Later in the course you will be using a variety of other technologies. Install and prepare Flume, Pig, and Pentaho to be used on your cluster. Feel free to add others that you find interesting and want to try out!

5. Set up Hue for use on your cluster.

6. Read through the *-site.xml files in greater detail, and consider what values you think would work well, or investigate what others in the field suggest. Use this when building your cluster. Also consider checking out the different scheduling tools (e.g. Fair Scheduler) and see what would work well for your setup.