cloudera®

Ask Bigger Questions

Cloudera Administrator Training for Apache Hadoop: Hands-On Exercises

General Notes 3
Preparation: Configuring Networking for Your Cluster 8
Hands-On Exercise: Installing Hadoop 18
Hands-On Exercise: Working With HDFS 23
Hands-On Exercise: Using Flume to Put Data into HDFS 27
Hands-On Exercise: Importing Data With Sqoop 32
Hands-On Exercise: Running a MapReduce Job 37
Hands-On Exercise: Creating a Hadoop Cluster 42
Hands-On Exercise: Querying HDFS With Hive and Cloudera Impala 62
Hands-On Exercise: Using Hue to Control Hadoop User Access 80
Hands-On Exercise: Configuring HDFS High Availability 91
Hands-On Exercise: Managing Jobs 107
Hands-On Exercise: Using the Fair Scheduler 109
Hands-On Exercise: Breaking The Cluster 111
Hands-On Exercise: Verifying The Cluster's Self-Healing Features 113

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Troubleshooting Challenge: Heap O' Trouble.....114

Appendix A: Setting up VMware Fusion on a Mac for the Cloud Training Environment.....117

Appendix B: Setting up VirtualBox for the Cloud Training Environment.....121

cloudera®

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

General Notes

Cloud and Local Training Environments

In this training course, you will use one of the following training environments:

- Cloud training environment: four Amazon EC2 instances plus a VMware or VirtualBox virtual machine (VM)
- Local training environment: four VMware VMs

All of the VMs and EC2 instances use the CentOS 6.3 Linux distribution.

For both training environments:

- Use the `training` user account to do your work
- You should not need to enter passwords for the `training` user
- Should you need superuser access, you can use `sudo` as the `training` user without entering a password. The `training` user has unlimited, passwordless `sudo` privileges

For the cloud training environment:

- You will start the VM, and then use it to connect to your EC2 instances
- The EC2 instances have been configured so that you can connect to them without entering a password
- Your instructor will provide the following details for four EC2 instances per student:
    - EC2 public IP addresses – You will run a script that adds the EC2 public IP addresses of your four EC2 instances to the `/etc/hosts` file on your VM
    - EC2 private IP addresses – You will use these addresses when you run a script that configures the `/etc/hosts` file on your EC2

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

instances. EC2 private IP addresses usually start with the number `10`, `172`, or `192.168`

  - The instructor will not provide EC2 internal host names. Please use the host names `elephant`, `tiger`, `horse`, and `monkey` for the four internal host names. **It is important that you use these four host names**, because several scripts, which expect these four host names, have been provided for your use while you perform exercises. The scripts will not work if you use different host names.

- Please write out a table similar to the following with your EC2 instance information:

Host Name	EC2 Public IP Address	EC2 Private IP Address
`elephant`		
`tiger`		
`horse`		
`monkey`		

For the local training environment:

- The VMs automatically log you in as the `training` user. Should you log out of a GNOME session, CentOS will automatically log you back in as the `training` user

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Notational Convention

In some command-line steps in the exercises, you will see lines like this:

```
$ hadoop fs -put shakespeare  \
/user/training/shakespeare
```

The backslash at the end of the first line signifies that the command is not completed, and continues on the next line. You can enter the code exactly as shown (on two lines), or you can enter it on a single line. If you do the latter, you should *not* type in the backslash.

Copying and Pasting from the Hands-On Exercises Manual

If you wish, you can usually copy commands and strings from this Hands-On Exercises manual and paste them into your terminal sessions. However, please note one important exception:

If you copy and paste table entries that exceed a single line, a space will be inserted at the end of each line.

Please use caution when copying and pasting from table entries when performing the exercises.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Resetting Your Cluster

You can use the `reset_cluster.sh` script to change the state of your cluster so that you can start with a fresh, correct environment for performing any exercise. Use the script in situations such as the following:

- While attempting one of the exercises, you misconfigure your machines so badly that attempting to do the next exercise is no longer possible.
- You have successfully completed a number of exercises, but then you receive an emergency call from work and you have to miss some time in class. When you return, you realize that you have missed two or three exercises. But you want to do the same exercise everyone else is doing so that you can benefit from the post-exercise discussions.

The script works identically in the cloud and local training environments.

The script is destructive: any work that you have done is overwritten when the script runs. If you want to save files before running the script, you can copy the files you want to save to a subdirectory under `/home/training`.

Before you attempt to run the script, verify that networking among the four hosts in your cluster is working. If networking has not been configured correctly, you can rerun the `config_hosts.sh` script to reset the networking configuration prior to running the `reset_cluster.sh` script.

Run the script on `elephant` only. You do not need to change to a directory to run the script; it is in your shell's PATH.

The script starts by prompting you to enter the number of an exercise. Specify the exercise you want to perform *after* the script has run. Then confirm that you want to reset the cluster (thus overwriting any work you have done).

After you have responded to the initial prompts, the script begins by cleaning up your cluster—terminating Hadoop processes, removing Hadoop software, deleting Hadoop-related files, and reverting other changes you might have made to the hosts in your cluster. Please note that as this system cleanup phase is running, you will see errors such as "unrecognized service" and "No packages marked for removal." These errors are expected. They occur because the script attempts to remove *anything*

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

pertinent that might be on your cluster. The number of error messages that appear during this phase of script execution depends on the state the cluster is in when you start the script.

Next, the script simulates steps for each exercise up to the one you want to perform.

In the cloud training environment, script completion time varies from 5 minutes to almost an hour depending on how many exercise steps need to be simulated by the script. In the local training environment, script completion time depends on machine speed.

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Preparation: Configuring Networking for Your Cluster

In this preparatory exercise you will configure networking for your four instances.

Configuring Networking on Your Cluster: Cloud Training Environment

If you are using a local training environment for this course, please skip this task and move on to 'Configuring Networking on Your Cluster: Local Training Environment on page 14.

In this task, you will run scripts to configure networking in a cloud training environment.

First, you will start the local `Get2EC2` VM and run a script to configure the `/etc/hosts` file on that VM with the addresses of the four EC2 instances and the host names `elephant`, `tiger`, `horse`, and `monkey`.

Next, you will run a script to configure the `/etc/hosts` file on the EC2 instances, setting the four host names to `elephant`, `tiger`, `horse`, and `monkey`.

Then you will verify that networking has been set up correctly.

Finally, you will run a script that starts a SOCKS5 proxy server on the local VM.

1. If you are using VMware Fusion on Mac OS to connect to your four EC2 instances, read 'Appendix A, Setting up VMware Fusion on a Mac for the Cloud Training Environment' and perform any indicated actions. After you have done so, continue to the next step.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Start the `Get2EC2` VM.

   You should find the VM on your Desktop, on the C: drive of your machine, or in the location to which you downloaded it. Double-click the icon whose name ends in `.vmx` to launch the VM.

   After a few minutes, the GNOME interface will appear.

   > Note: A VirtualBox VM is available for students running Mac OS who are unable to or prefer not to install VMware Fusion. However, we strongly recommend using the VMware version VM. Use VirtualBox for this course *only* if it is your preferred virtualization environment and if you are knowledgeable enough to be self-sufficient to troubleshoot problems you might run into.
   >
   > If you are using a VirtualBox VM, follow the steps in 'Appendix B, Setting up VirtualBox for the Cloud Training Environment.' When you have completed the steps in the Appendix, continue to the next step.

3. Run a script that modifies the `/etc/hosts` file on your VM by adding your four EC2 instances.

   Enter the following command in the terminal window in the VM:

   ```
   $ config_local_hosts_file.sh
   ```

   The `config_local_hosts_file.sh` script will prompt you to enter the four EC2 public IP addresses for your four EC2 instances. Refer to the table you created when your instructor gave you the EC2 instance information.

4. Log in to the `elephant` EC2 instance as the `training` user.

   ```
   $ connect_to_elephant.sh
   ```

   When prompted to confirm that you want to continue connecting, enter `yes` and then press Enter.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Run the `config_hosts.sh` script on `elephant`. This script sets up the `/etc/hosts` file on `elephant`, copies that file to `tiger`, `horse`, and `monkey`, and then sets the host names for the four hosts to `elephant`, `tiger`, `horse`, and `monkey`.

   Enter the following command on **`elephant`:**

   ```
   $ config_hosts.sh
   ```

   When the script prompts you to enter IP addresses for your EC2 instances, enter the EC2 private IP addresses (which typically start with `10`, `172`, or `192.168`).

   When the script prompts you to confirm that you want to continue connecting, enter `yes` and then press Enter.

6. Terminate and restart the SSH session with `elephant`.

   ```
   $ exit
   $ connect_to_elephant.sh
   ```

   Note: You exit and reconnect to `elephant` to reset the value of the `$HOSTNAME` variable in the shell after the `config_hosts.sh` script changes the host name.

7. Start SSH sessions with your other three EC2 instances, logging in as the `training` user.

   Open three more terminal windows (or tabs) on your VM, so that a total of four terminal windows are open.

   In one of the new terminal windows, connect to the `tiger` EC2 instance.

   ```
   $ connect_to_tiger.sh
   ```

   When the script prompts you to confirm that you want to continue connecting, enter `yes` and then press Enter. (You will also need to do this when you connect to `horse` and `monkey`.)

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

In another terminal window, connect to the `horse` EC2 instance.

```
$ connect_to_horse.sh
```

In the remaining terminal window, connect to the `monkey` EC2 instance.

```
$ connect_to_monkey.sh
```

8. Verify that you can communicate with all the hosts in your cluster from `elephant` by using the host names.

   On **`elephant`**:

```
$ ping elephant
$ ping tiger
$ ping horse
$ ping monkey
```

9. Verify that passwordless SSH works by running the `ip addr` command on `tiger`, `horse`, and `monkey` from a session on `elephant`.

   On **`elephant`**:

```
$ ssh training@tiger ip addr
$ ssh training@horse ip addr
$ ssh training@monkey ip addr
```

   The `ip addr` command shows you IP addresses and properties.

> **Note:** Your environment is set up to allow you to use passwordless SSH to submit commands from a session on `elephant` to the `root` and `training` users on `tiger`, `horse`, and `monkey`, and to allow you to use the `scp` command to copy files from `elephant` to the other three hosts. **Passwordless SSH and `scp` are not required to deploy a Hadoop cluster.** We make these tools available in the classroom environment as a convenience.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

10. Run the `uname -n` command to verify that all four host names have been changed as expected.

    On **all four hosts**:

```
$ uname -n
```

11. Verify that the value of the `$HOSTNAME` environment variable has been set to the new host name.

    On **all four hosts**:

```
$ echo $HOSTNAME
```

    The value `elephant`, `tiger`, `horse`, or `monkey` should appear.

12. Start a SOCKS5 proxy server on the VM. The browser on your VM will use the proxy to communicate with your EC2 instances.

    Open one more terminal window on the local VM – *not* a tab in an existing window – and enter the following command:

```
$ start_SOCKS5_proxy.sh
```

    When the script prompts you to confirm that you want to continue connecting, enter `yes` and then press Enter.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

13. Minimize the terminal window in which you started the SOCKS5 proxy server.

    Important: Do not close this terminal window or exit the SSH session running in it while you are working on exercises. If you accidentally terminate the proxy server, restart it by opening a new terminal window and rerunning the `start_SOCKS5_proxy.sh` script.

At the End of Each Day

At the end of the day, exit all active SSH sessions you have with EC2 instances (including the session running the SOCKS5 proxy server). To exit an SSH session, simply execute the `exit` command.

CAUTION: Do *not* shut down your EC2 instances.

Then suspend the Get2EC2 VM.

When you return in the morning, resume the Get2EC2 VM, reestablish SSH sessions with the EC2 instances, and restart the SOCKS5 proxy server.

To reestablish SSH sessions with the EC2 instances, open a terminal window (or tab), and then execute the appropriate `connect_to` script.

To restart the SOCKS5 proxy server, open a separate terminal window, and then execute the `start_SOCKS5_proxy.sh` script.

This is the end of the setup activity for the cloud training environment.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Configuring Networking on Your Cluster: Local Training Environment

If you are using a cloud training environment for this course, please skip this task.

In this task, you will run scripts to configure networking in a local training environment.

You must start by setting IP addresses for the four hosts. Setting IP addresses is required because the VMware images are initialized with the same static IP address.

Then you will run a script to configure the `/etc/hosts` file, copy it to all four hosts, and set the host names of the four hosts to `elephant`, `tiger`, `horse`, and `monkey`.

Finally, you will verify that networking has been set up correctly.

1. Start the four VMs in your lab environment.

   Double-click on the icon labeled `elephant`. After a few minutes, the GNOME interface will appear. You are now logged in as the `training` user on the `elephant` host.

   After `elephant` has come up, bring up `tiger` by double-clicking the icon labeled `tiger`.

   After `tiger` comes up, bring up `horse`.

   After `horse` comes up, bring up `monkey`.

2. After all four instances have come up, run the `config_ip_addresses.sh` script on `tiger`, `horse`, and `monkey`. **Do *not* run this script on `elephant`.**

   ```
   $ config_ip_addresses.sh
   ```

   When prompted for a host name, enter `tiger`, `horse`, or `monkey`.

   To verify that the `config_ip_addresses.sh` script has run correctly, review the IP address that appears in the terminal window after the script has run. The following IP addresses should appear:

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

- On `tiger`: `192.168.123.2`
- On `horse`: `192.168.123.3`
- On `monkey`: `192.168.123.4`

If your results differ, consult your instructor.

3. Run the `config_hosts.sh` script to configure the `/etc/hosts` file, copy the `/etc/hosts` file to `tiger`, `horse`, and `monkey`, and then set the host name on `elephant`, `tiger`, `horse`, and `monkey`.

   Enter the following command on **`elephant`**:

```
$ config_hosts.sh
```

   Several times during script execution, you will be prompted to confirm that you want to continue connecting. Enter `yes` and press Enter in response to the prompts.

4. Verify that you can communicate with all the hosts in your cluster from `elephant` by using the host names.

   On **`elephant`**:

```
$ ping elephant
$ ping tiger
$ ping horse
$ ping monkey
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Verify that passwordless SSH works by running the `ip addr` command on `tiger`, `horse`, and `monkey` from a session on `elephant`.

   On **`elephant`**:

```
$ ssh training@tiger ip addr
$ ssh training@horse ip addr
$ ssh training@monkey ip addr
```

   The `ip addr` command shows you IP addresses and properties.

> **Note:** Your environment is set up to allow you to use passwordless SSH to submit commands from a session on `elephant` to the `root` and `training` users on `tiger`, `horse`, and `monkey`, and to allow you to use the `scp` command to copy files from `elephant` to the other three hosts. **Passwordless SSH and `scp` are not required to deploy a Hadoop cluster.** We make these tools available in the classroom environment as a convenience.

6. Run the `uname -n` command to verify that all four host names have been changed as expected.

   On **all four hosts**:

```
$ uname -n
```

7. Restart your login session on all four hosts.

   Log out of GNOME on all four hosts using the "System > Log Out training" menu option. After you log out, you are automatically logged back in.

   Note: Logging out and logging back in sets the `$HOSTNAME` environment variable to the new host name—`elephant`, `tiger`, `horse`, or `monkey`. Because several Hadoop scripts make use of this environment variable, it is important that you take this action.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

8. Verify that the value of the `$HOSTNAME` environment variable has been set to the new host name.

   On **all four hosts**:

```
$ echo $HOSTNAME
```

   The value `elephant`, `tiger`, `horse`, or `monkey` should appear.

> **At the End of Each Day**
>
> At the end of the day, suspend your four VMs.
>
> When you return in the morning, resume your four VMs.

This is the end of the setup activity for the local training environment.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Installing Hadoop

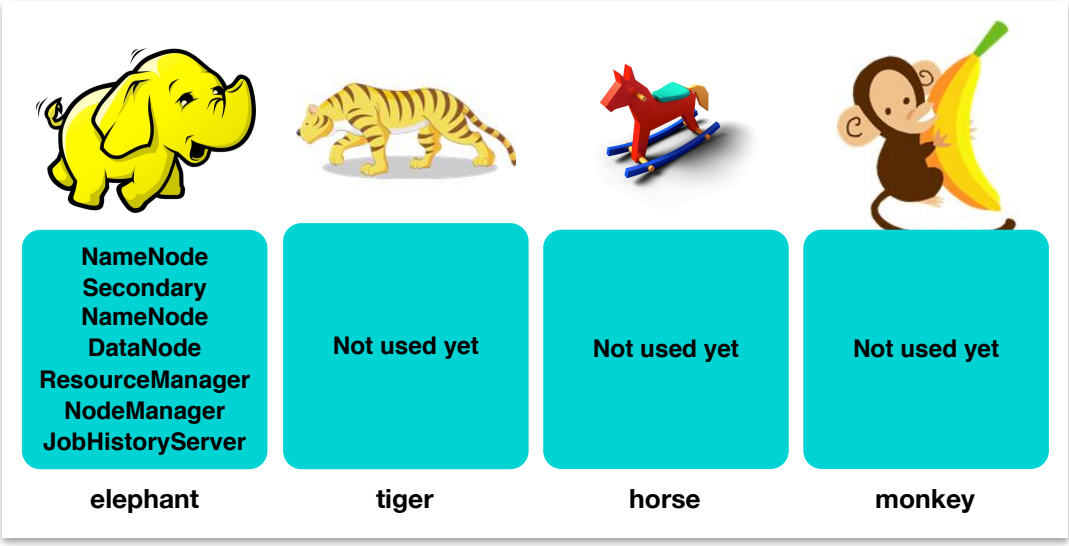
In this exercise, you will install and test Hadoop in pseudo-distributed mode using the CDH RPMs. The purpose of this exercise is to quickly create a working environment in which you can familiarize yourself with Hadoop. In later chapters, we will explain in detail all the commands you run in this exercise.

Installing Hadoop in Pseudo-Distributed Mode From CDH

Your instances contain a local yum repository of CDH5. We have created the local repository to save download time; typically, you would use an online repository.

You will initially install CDH in *pseudo-distributed* mode. This is a way of installing a Hadoop cluster on a single machine; the machine runs all six Hadoop daemons (NameNode, Secondary NameNode, DataNode, ResourceManager, NodeManager, and JobHistoryServer). With a single DataNode, the replication factor is set to 1.

At the end of this exercise, you will have Hadoop daemons deployed in a single host as depicted here:



cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

1. Install a pseudo-distributed Hadoop installation on `elephant`. From the terminal window or SSH session with `elephant`, enter the following:

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

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.

Note: The `--assumeyes` option assumes an answer of **`y`** in response to an expected confirmation prompt.

2. Run a script to change the host name in the Hadoop configuration files from `localhost` to `elephant`.

On **`elephant`**:

```
$ use_host_name.sh
```

Note: This script is specific to the training environment and is *not* part of CDH.

Note: Pseudo-distributed installation configures the host name in Hadoop configuration files as `localhost`. When running Hadoop in a cloud environment, this causes a small issue with Hadoop's Web UI. To fix the issue, simply change `localhost` to a resolvable host name – in this case, `elephant`.

3. Format the NameNode.

On **`elephant`**:

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

Note: The NameNode metadata is stored in `/var/lib/hadoop-hdfs/cache/hdfs/dfs/name`.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. CDH uses the Linux *alternatives* system, which is a way of retaining multiple configuration settings. You will find the configuration files in `/etc/hadoop/conf`, which is a symbolic link from `/etc/alternatives/hadoop-conf`.

   Change directory to `/etc/hadoop/conf` and inspect the `*-site.xml` files.

5. Start Hadoop's HDFS daemons.

   On **elephant**:

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

6. You now need to create the `staging` directory in HDFS. YARN and MapReduce will need this directory.

   On **elephant**:

```
$ 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 **elephant**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

8. Start the YARN and MapReduce daemons.

   On **elephant**:

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

9. Check to ensure all six daemons are running.

   On **elephant**:

```
$ sudo jps
```

   You should see the six Hadoop daemons running. If you do not, ask your instructor for assistance.

10. Finally, create a user directory for the user `training`.

    On **elephant**:

```
$ sudo -u hdfs hadoop fs -mkdir -p /user/training
$ sudo -u hdfs hadoop fs -chown training /user/training
```

Testing Your Hadoop Installation

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

1. Change directories to the `training_materials/admin/data` directory and unzip the `shakespeare.txt.gz` file. Check the file to ensure it has been extracted correctly.

   On **elephant**:

```
$ cd ~/training_materials/admin/data
$ gunzip shakespeare.txt.gz
$ tail shakespeare.txt
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Create a directory in HDFS and upload the file to that directory. Check to make sure the file is in place.

   On **elephant**:

```
$ hadoop fs -mkdir input
$ hadoop fs -put shakespeare.txt input
$ hadoop fs -ls /user
$ hadoop fs -ls input
$ hadoop fs -tail input/shakespeare.txt
```

This is the end of the exercise.

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Working With HDFS

In this Hands-On Exercise you will copy a large Web server log file into HDFS and explore the results in the Hadoop NameNode Web UI and the Linux file system.

Perform all steps in this exercise on `elephant`.

1. List the Java processes that are running on `elephant`.

   On **`elephant`**:

   ```
   $ sudo jps
   ```

   Three daemons that support HDFS – the `NameNode`, `SecondaryNameNode`, and `DataNode` daemons – are running. (The `ResourceManager`, `NodeManager`, and `JobHistoryServer` daemons that support YARN and MapReduce are also running.)

2. Create an HDFS directory called `weblog`, in which you will store the access log.

   On **`elephant`**:

   ```
   $ hadoop fs -mkdir weblog
   ```

3. Extract the `access_log.gz` file and upload it in a single step, then run the `hadoop fs -ls` command to review the file permissions in HDFS.

   On **`elephant`**:

   ```
   $ cd ~/training_materials/admin/data
   $ gunzip -c access_log.gz \
   | hadoop fs -put - weblog/access_log
   ```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. Run the `hadoop fs -ls` command to review the file's permissions in HDFS.

   On **`elephant`**:

```
$ hadoop fs -ls /user/training/weblog
```

5. Locate information about the `access_log` file in the NameNode Web UI.

   a. Start Firefox.

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

   c. Click Utilities > "Browse the file system," then navigate to the `/user/training/weblog` directory.

   d. Verify that the permissions for the `access_log` file are identical to the permissions you observed when you ran the `hadoop fs -ls` command.

   e. Now select the `access_log` 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—`elephant`—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 this course, the replication number will be 3, and you will see three hosts listed under Availability.

   f. Choose Block 0 and write down the value that appears in the Size field.

      _____

      You will need this value when you examine the Linux file that contains this block.

   g. Select the value of the Block ID field and copy it (Edit menu > Copy). You will need this value for the next step in this exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. Locate the HDFS block in the Linux file system.

   a. In your terminal session on `elephant`, 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.

   On **`elephant`**:

   ```
   $ 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.

7. 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.

8. Starting any Linux editor with `sudo`, open the file containing the HDFS block. Verify that the first few lines of the file match the first chunk of the `access_log` file content.

   Note: You must start your editor with `sudo` because you are logged in to Linux as the `training` user, and this user does not have privileges to access the Linux file that contains the HDFS block.

   You can review the `access_log` file content as follows:

   ```
   $ hadoop fs -cat weblog/access_log | less
   ```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

9. Remove the `weblog` directory, which contains the Web server log file, from HDFS.

   On **`elephant`**:

```
$ hadoop fs -rm -r weblog
```

This is the end of the exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Using Flume to Put Data into HDFS

In this Hands-On Exercise you will use Flume to import dynamically-generated data into HDFS. A very common use case for Flume is to collect access logs from a large number of Web servers on your network; we will simulate a simple version of this in the following exercise.

Perform all steps in this exercise on `elephant`.

Installing Flume

1. Install Flume.

   On **`elephant`**:

   ```
   $ sudo yum install --assumeyes flume-ng
   ```

2. Using `sudo`, create a new file on `elephant` named `/etc/hadoop/conf/flume-conf.properties` by copying the text below and inserting it into the file, and then saving the file.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

```
tail1.sources = src1
tail1.channels = ch1
tail1.sinks = sink1

tail1.sources.src1.type = exec
tail1.sources.src1.command = tail -F /tmp/access_log
tail1.sources.src1.channels = ch1

tail1.channels.ch1.type = memory
tail1.channels.ch1.capacity = 500

tail1.sinks.sink1.type = avro
tail1.sinks.sink1.hostname = localhost
tail1.sinks.sink1.port = 6000
tail1.sinks.sink1.batch-size = 1
tail1.sinks.sink1.channel = ch1
##
collector1.sources = src1
collector1.channels = ch1
collector1.sinks = sink1

collector1.sources.src1.type = avro
collector1.sources.src1.bind = localhost
collector1.sources.src1.port = 6000
collector1.sources.src1.channels = ch1

collector1.channels.ch1.type = memory
collector1.channels.ch1.capacity = 500

collector1.sinks.sink1.type = hdfs
collector1.sinks.sink1.hdfs.path = flume/collector1
collector1.sinks.sink1.hdfs.filePrefix = access_log
collector1.sinks.sink1.channel = ch1
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

3. Create the `flume/collector1` directory in HDFS to store the files.

   On **`elephant`**:

```
$ hadoop fs -mkdir -p flume/collector1
```

4. Start four terminal windows on `elephant` (or start four additional SSH sessions to `elephant`).

Generating the Logs

1. In the first terminal window, run the `accesslog-gen.bash` shell script, which simulates a Web server creating log files. This shell script also rotates the log files regularly.

```
$ accesslog-gen.bash /tmp/access_log
```

Note: The `accesslog-gen.bash` script is specific to the training environment and is *not* part of CDH.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Wait for a minute or so, and then in the second terminal window, verify that the log file has been created. Notice that the log file is rotated periodically.

```
$ ls -l /tmp/access*
-rw-rw-r-- 1 training training  498 Nov 15 15:12
/tmp/access_log
-rw-rw-r-- 1 training training  997 Nov 15 15:12
/tmp/access_log.0
-rw-rw-r-- 1 training training 1005 Nov 15 15:11
/tmp/access_log.1
-rw-rw-r-- 1 training training  994 Nov 15 15:11
/tmp/access_log.2
-rw-rw-r-- 1 training training 1003 Nov 15 15:11
/tmp/access_log.3
-rw-rw-r-- 1 training training 1003 Nov 15 15:11
/tmp/access_log.4
-rw-rw-r-- 1 training training  984 Nov 15 15:11
/tmp/access_log.5
```

Starting the Collector

1. Start collector number 1 in the second terminal window.

```
$ flume-ng agent --conf /etc/hadoop/conf/ \
--conf-file /etc/hadoop/conf/flume-conf.properties \
--name collector1
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Starting the Source

1. In the third terminal window, start the source, which will tail the `/tmp/access_log` file.

```
$ flume-ng agent \
--conf-file /etc/hadoop/conf/flume-conf.properties \
--name tail1
```

Viewing Data in HDFS

1. In the fourth terminal window, run the `hadoop fs` command to see the logs being written to HDFS.

```
$ hadoop fs -ls flume/collector1
```

2. Stop the log generator by hitting Ctrl-C in the first terminal window.

3. Stop the collector by hitting Ctrl-C in the second terminal window.

4. Stop the source by hitting Ctrl-C in the third terminal window.

5. Remove the generated access log files to clear up space on your virtual machine.

   On **`elephant`**:

```
$ rm -rf /tmp/access_log*
```

This is the end of the Exercise.

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Importing Data With Sqoop

For this exercise you will import data from a relational database using Sqoop. The data you load here will be used in a subsequent exercise.

Consider the MySQL database `movielens`, derived from the MovieLens project from University of Minnesota. (See note at the end of this exercise.) The database consists of several related tables, but we will import only two of these: `movie`, which contains about 3,900 movies; and `movierating`, which has about 1,000,000 ratings of those movies.

Perform all steps in this exercise on `elephant`.

Reviewing the Database Tables

First, review the database tables to be loaded into Hadoop.

1. Log on to MySQL.

   On **`elephant`**:

```
$ mysql --user=training --password=training movielens
```

2. Review the structure and contents of the `movie` table.

   On **`elephant`**:

```
mysql> DESCRIBE movie;
. . .
mysql> SELECT * FROM movie LIMIT 5;
```

3. Note the column names for the table.

   _____

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. Review the structure and contents of the `movierating` table.

   On **elephant**:

```
mysql> DESCRIBE movierating;
. . .
mysql> SELECT * FROM movierating LIMIT 5;
```

5. Note these column names.

   _____

6. Exit MySQL.

   On **elephant**:

```
mysql> quit;
```

Installing Sqoop

1. Install Sqoop.

   On **elephant**:

```
$ sudo yum install --assumeyes sqoop
```

2. Using `sudo`, create a link to the MySQL JDBC driver in the `/usr/lib/sqoop/lib` directory.

   On **elephant**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Importing with Sqoop

You invoke Sqoop on the command line to perform several commands. With it you can connect to your database server to list the databases (schemas) to which you have access, and list the tables available for loading. For database access, you provide a connect string to identify the server, and your username and password.

1. Show the commands available in Sqoop.

   On **elephant**:

```
$ sqoop help
```

2. List the databases (schemas) in your database server.

   On **elephant**:

```
$ sqoop list-databases \
--connect jdbc:mysql://localhost \
--username training --password training
```

   (Note: Instead of entering `--password training` on your command line, you may prefer to enter `-P`, and let Sqoop prompt you for the password, which is then not visible when you type it.)

3. List the tables in the `movielens` database.

   On **elephant**:

```
$ sqoop list-tables \
  --connect jdbc:mysql://localhost/movielens \
  --username training --password training
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. Import the `movie` table into Hadoop.

   On **`elephant`**:

```
$ sqoop import \
--connect jdbc:mysql://localhost/movielens \
--table movie --fields-terminated-by '\t' \
--username training --password training
```

   The `--fields-terminated-by '\t'` option separates the fields in the HDFS file with the tab character, which is sometimes useful if users will be working with Hive and Pig.

   Warnings that packages such as hbase, hive-hcatalog, and accumulo are not installed are expected. It is not a problem that these packages are not installed on your system.

5. Verify that the command has worked.

   On **`elephant`**:

```
$ hadoop fs -ls movie
$ hadoop fs -tail movie/part-m-00000
```

6. Import the `movierating` table into Hadoop using the command in step 4 as an example.

   Verify that the `movierating` table was imported using the command in step 5 as an example.

This is the end of the Exercise

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Note:

This exercise uses the MovieLens data set, or subsets thereof. This data is freely available for academic purposes, and is used and distributed by Cloudera with the express permission of the UMN GroupLens Research Group. If you would like to use this data for your own research purposes, you are free to do so, as long as you cite the GroupLens Research Group in any resulting publications. If you would like to use this data for commercial purposes, you must obtain explicit permission. You may find the full dataset, as well as detailed license terms, at http://www.grouplens.org/node/73

cloudera®

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Running a MapReduce Job

In this exercise you will run a MapReduce job on the pseudo-distributed Hadoop cluster you set up in the previous exercise.

Perform all steps in this exercise on `elephant`.

Introduction

The code and configuration needed to run a Hadoop job is typically packaged into a Java Archive (JAR) file. MapReduce jobs packaged this way can be submitted to the cluster with the `hadoop jar` command, which takes a minimum of two arguments. The first argument is the path to the JAR file itself, while the second is the name of the MapReduce program within the JAR to run (since a JAR may contain several distinct programs). The program itself may accept any number of additional parameters, such as the location of input and output data, depending on its requirements.

Submitting a MapReduce Job to Your Cluster

We will now test the Hadoop installation by running a sample Hadoop job that ships with the Hadoop source code. This job is WordCount, the classic MapReduce program we discussed in class. We'll run the WordCount program against the Shakespeare data added to HDFS in a previous exercise.

1. Use the `hadoop fs -ls` command to ensure that the files added in the previous lab are still there.

   On **`elephant`**:

```
$ hadoop fs -ls input
$ hadoop fs -tail input/shakespeare.txt
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Since the code for the job we want to execute is in a Java Archive (JAR) file, we'll use the `hadoop jar` command to submit it to the cluster. Like many MapReduce programs, WordCount accepts two additional arguments: the HDFS directory path containing input and the HDFS directory path into which output should be placed. Therefore, we can run the WordCount program with the following command.

   On **elephant**:

```
$ hadoop jar /usr/lib/hadoop-mapreduce/\
hadoop-mapreduce-examples.jar wordcount input counts
```

Verifying Job Output

1. Once the job has completed you can then inspect the output by listing the contents of the output (`counts`) directory.

   On **elephant**:

```
$ hadoop fs -ls counts
```

2. This directory should show all job output. Job output will include a `_SUCCESS` flag and a file created by the single Reducer that ran. You can view the output of that file by using the `hadoop fs -cat` command.

   On **elephant**:

```
$ hadoop fs -cat counts/part-r-00000
```

Reviewing the Job in the ResourceManager Web UI

1. Start the ResourceManager Web UI in Firefox by entering the following URL: `http://elephant:8088`

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Click “FINISHED” (under Applications).

   If you have followed the exercise instructions exactly as written, you should see three entries for completed jobs:

   - The `movie.jar` job
   - The `movierating.jar` job
   - The `word count` job

   You ran the `movie.jar` and `movierating.jar` jobs in the Sqoop exercise. You ran the `word count` job in this exercise.

   Note: More or fewer jobs might appear in the ResourceManager Web UI. There will be an entry for each completed MapReduce job that you have run on your lab system up to this point.

3. Determine the log level for Reducer tasks for the `word count` job.

   a. Click “History” (in the Tracking UI column).

   b. Click “Configuration” (under Job).

      Twenty entries from the job configuration that was in effect when the `word count` job ran appear.

   c. In the Search field, enter `log.level`.

   d. Locate the `mapreduce.reduce.log.level` property. Its value should be `INFO`.

4. Locate the task logs for the `word count` Mapper and Reducer tasks.

   a. Click “Overview” (under Job) to return to the page for the `word count` job.

   b. Click “Map” (under Task Type).

   c. Click the link for the Mapper task.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

d. Click "logs."

   The Mapper task log appears.

e. Return to the page for the `word count` job and locate the task log for the Reducer task. Observe the amount of output in the Reducer task log.

5. Remove the `counts` directory from HDFS and rerun the WordCount program.

On **`elephant`**:

```
$ hadoop fs -rm -r counts
$ hadoop jar /usr/lib/hadoop-mapreduce/\
hadoop-mapreduce-examples.jar wordcount \
-D mapreduce.reduce.log.level=DEBUG input counts
```

You must delete the `counts` directory before running the WordCount program a second time because MapReduce will not run if you specify an output path which already exists.

MapReduce programs that have been coded to take advantage of the Hadoop ToolRunner allow you to pass several types of arguments to Hadoop, including run-time configuration parameters. The `hadoop jar` command shown above sets the log level for reduce tasks to `DEBUG`.

When your job is running, look for a line in standard output similar to the following:

```
14/02/01 05:47:16 INFO mapreduce.Job: Running job:
job_1391249780844_0004
```

You should be able to locate this job in the ResourceManager Web UI.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. In the ResourceManager Web UI, locate the entry for the job that you just ran. Verify the following after the job finishes:

    - The value of the `mapreduce.reduce.log.level` configuration attribute is `DEBUG` in the job configuration.
    - The number of records written to the Reducer task log for this job contains DEBUG log records and is significantly larger than the number of records written to the Reducer task log during the previous WordCount job execution.

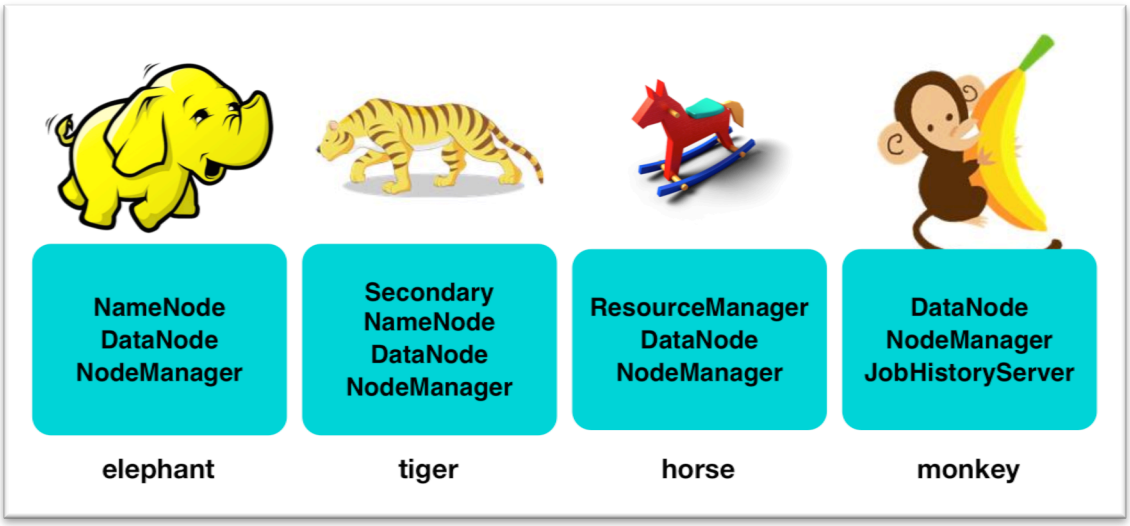
This is the end of the exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Creating a Hadoop Cluster

In this exercise, you will create and test a fully distributed Hadoop cluster. You have been assigned a set of four hosts, but up until this exercise, you have only used one of the hosts – `elephant`. In this exercise, you will stop the daemons currently running on `elephant`, install Hadoop software on the other three hosts, and then configure a Hadoop cluster that uses all four hosts.

At the end of this exercise, you will have daemons deployed on your four hosts as follows:



Because you have four hosts to work with, you will have to run multiple Hadoop daemons on all the hosts. For example, the NameNode, a DataNode and a NodeManager all run on `elephant`. Having a very limited number of hosts is not unusual when deploying Hadoop for a proof-of-concept project. Please follow the best practices in the Planning Your Hadoop Cluster chapter when sizing and deploying your production clusters.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing Hadoop Software on Your Cluster

In this task, you will remove the Hadoop SecondaryNameNode, ResourceManager, and JobHistoryServer from `elephant` and install Hadoop software on `tiger`, `horse`, and `monkey`.

When you installed Hadoop in pseudo-distributed mode on `elephant` in a previous exercise, you installed the NameNode, SecondaryNameNode, DataNode, ResourceManager, NodeManager, and JobHistoryServer on that host. Because the SecondaryNameNode is going to run on `tiger`, the ResourceManager is going to run on `horse`, and the JobHistoryServer is going to run on `monkey`, you can remove these three components from `elephant` now.

Then you will install the following Hadoop daemons on your other three hosts:

- SecondaryNameNode – On `tiger`
- DataNode – On `tiger`, `horse`, and `monkey`
- ResourceManager – On `horse`
- NodeManager – On `tiger`, `horse`, and `monkey`
- JobHistoryServer – On `monkey`

1. First, stop the existing pseudo-distributed cluster running on `elephant`.

   Enter the following commands on **`elephant`**:

```
$ 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
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Remove any log files that were created while the pseudo-distributed cluster was running.

   On **elephant**:

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

   Removing the log files that were created when you were running Hadoop on `elephant` 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 SecondaryNameNode, ResourceManager, and JobHistoryServer from `elephant`. Removing these components also removes the pseudo-distributed mode configuration.

   On **elephant**:

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

4. Install the SecondaryNameNode on `tiger`.

   Run the following command on **tiger**:

```
$ sudo yum install --assumeyes \
hadoop-hdfs-secondarynamenode
```

5. Install the DataNode on `tiger`, `horse`, and `monkey`.

   Run the following command on **tiger**, **horse**, **and monkey:**

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. Install the ResourceManager on `horse`.

   Run the following command on **`horse`:**

```
$ sudo yum install --assumeyes \
hadoop-yarn-resourcemanager
```

7. Install the NodeManager on `tiger`, `horse`, and `monkey`.

   Run the following command on **`tiger`**, **`horse`**, **and `monkey`:**

```
$ sudo yum install --assumeyes hadoop-yarn-nodemanager
```

8. Install the `hadoop-mapreduce` package on `tiger`, `horse`, and `monkey`. This package contains the `mapreduce_shuffle` auxiliary service, which is needed for MapReduce jobs.

   Run the following command on **`tiger`**, **`horse`**, **and `monkey`:**

```
$ sudo yum install --assumeyes hadoop-mapreduce
```

9. Install the JobHistoryServer on `monkey`.

   Run the following command on **`monkey`:**

```
$ sudo yum install --assumeyes \
hadoop-mapreduce-historyserver
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Modifying The Hadoop Configuration Files

In this task, you will edit four Hadoop configuration files on `elephant`: `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 `tiger`, `horse`, and `monkey`. Throughout these exercises, you will synchronize changes to the Hadoop configuration across all the hosts in your cluster.

Note: When editing the configuration files, you will need to use `sudo`. For example:

```
$ sudo vi core-site.xml
```

Also note: when you add configuration properties to the configuration files, the format is:

```
<property>
     <name>property_name</name>
     <value>property_value</value>
</property>
<property>
     <name>another_property_name</name>
     <value>another_value</value>
</property>
...
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

1. On `elephant`, change directories to the directory that contains stub configuration files.

   On **`elephant`**:

```
$ cd ~/training_materials/admin/stubs
```

2. Copy stub configuration files to the configuration directory. Using the stub files will save you some typing.

   On **`elephant`**:

```
$ sudo cp core-site.xml /etc/hadoop/conf/
$ sudo cp hdfs-site.xml /etc/hadoop/conf/
$ sudo cp yarn-site.xml /etc/hadoop/conf/
$ sudo cp mapred-site.xml /etc/hadoop/conf/
```

3. Edit `/etc/hadoop/conf/core-site.xml` **on `elephant`**, adding the following property and value between the `<configuration>...</configuration>` tags. Edit property names and values inside the `<name>` and `<value>` tags, and don't forget to use `sudo` when you edit Hadoop configuration files.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. Edit `/etc/hadoop/conf/hdfs-site.xml` on **`elephant`**, 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 (NameNode, ResourceManager, and so on).

   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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Edit `/etc/hadoop/conf/yarn-site.xml` on **`elephant`**, adding the following values.

   Caution: Do not copy and paste table entries that exceed a single line—a space will be inserted at the end of each line.

Name	Value
yarn.resourcemanager.hostname	horse
yarn.application.classpath	Leave the value specified in the stub file
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

6. Edit `/etc/hadoop/conf/mapred-site.xml` on **`elephant`**, adding the following values.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

7. Because we are running on a Virtual Machine with very limited RAM, we will reduce the heap size of the Hadoop daemons. Create a file called `/etc/hadoop/conf/hadoop-env.sh` on **`elephant`** and add the following lines:

```
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"
```

8. Review the `copy_configuration.sh` script in the `~/training_materials/admin/scripts` directory. This script, which is specific to this class and not part of CDH, copies the `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, `mapred-site.xml`, and `hadoop-env.sh` files from `elephant` to `tiger`, `horse`, and `monkey`.

9. Run the `copy_configuration.sh` script. You do not need to change to a directory before running this script because it is in your shell's PATH.

   On **`elephant`**:

```
$ copy_configuration.sh
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Setting Up The File System

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

1. Create the directories specified in the properties above.

   On **elephant**:

```
$ 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 **elephant**:

```
$ 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 **elephant**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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 **`elephant`**:

```
$ ls -lR /disk1
$ ls -lR /disk2
```

5. Run the `set_up_directories.sh` script to create the same directories and set the same file permissions on `tiger`, `horse`, and `monkey`.

   On **`elephant`**:

```
$ set_up_directories.sh
```

   Note: This script is specific to the training environment and is *not* part of CDH.

Formatting HDFS and Starting the HDFS Daemons

In this task, you will format the HDFS file system and start the NameNode and Data Nodes.

Review the deployment diagram at the beginning of this exercise. As shown in the diagram, you will be starting a NameNode on `elephant`, a SecondaryNameNode on `tiger`, and DataNodes on all four hosts in your cluster.

1. Format the NameNode.

   On **`elephant`**:

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

   (Note: if you are asked whether you want to reformat the filesystem, answer 'Y'.)

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Start the NameNode daemon.

   On **elephant**:

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

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

4. Run the `sudo jps -v` command on `elephant`. 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.

   On **elephant**:

```
$ sudo jps -v
```

   Note: If the -Xmx option is specified multiple times on the `java` command line, the final occurrence is used to determine the heap size.

5. Review the log files for the NameNode from the command line.

   On **elephant**:

```
$ less /var/log/hadoop-hdfs/\
hadoop-hdfs-namenode-elephant.log
$ less /var/log/hadoop-hdfs/\
hadoop-hdfs-namenode-elephant.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 successfully, identify the problem in the log file. Then correct the problem and attempt to start the NameNode again.

6. Start the NameNode Web UI using the URL http://elephant:50070.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

7. Select Utilities > Logs.

   A list of files in the `/var/log/hadoop-hdfs` directory appears.

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

9. Start the SecondaryNameNode daemon.

   On **`tiger`**:

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

10. Run the `sudo jps` command on `tiger` and verify that SecondaryNameNode is running.

11. Run the `sudo jps -v` command on `tiger` to verify that the maximum heap size of the SecondaryNameNode is the size that you configured in the `hadoop-env.sh` file.

12. Review the SecondaryNameNode log files on `tiger` using the command line.

    Enter the following commands on **`tiger`**:

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

    If the SecondaryNameNode 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 SecondaryNameNode did not start successfully, identify the problem in the log file. Then correct the problem and attempt to start the SecondaryNameNode again.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

13. Start the DataNode daemons.

    Run the following command **on all four hosts in your cluster:**

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

14. Run the `sudo jps` command on all four hosts and verify that DataNodes are running.

15. Run the `sudo jps -v` command on all four hosts to verify that the maximum heap size of the DataNode is the size that you configured in the `hadoop-env.sh` file.

16. Review the DataNode log files on `elephant` using either the command line or the NameNode Web UI.

    If the DataNode 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 DataNode did not start successfully, correct the problem and restart the DataNode.

17. Review the DataNode log files on **`tiger`**. You can either use the command line or the DataNode Web UI.

    To use the command line, enter the following commands on **`tiger`**:

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

    To use the DataNode Web UI to view a DataNode log file, navigate to http://tiger:50075, then select DataNode Logs.

    If the daemon did not start successfully, identify the problem in the log file. Then correct the problem and attempt to start the DataNode again.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

18. Review the DataNode log files on **horse** and **monkey**.

    If any of the daemons did not start successfully, correct the problem and restart the daemon.

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 and to run MapReduce jobs in your cluster.

You will create:

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

1. Create the needed directories.

   On any host in your cluster:

```
$ 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/training
$ sudo -u hdfs hadoop fs -chown training /user/training
$ 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
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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 **horse**:

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

2. Run the `sudo jps` command on `horse` and verify the presence of the ResourceManager daemon.

3. Run the `sudo jps -v` command on `horse` to verify that the maximum heap size of the ResourceManager is the size that you configured in the `hadoop-env.sh` file.

4. Review the ResourceManager log files using the command line.

   On **horse**:

   ```
   $ less /var/log/hadoop-yarn/\
   yarn-yarn-resourcemanager-horse.log
   $ less /var/log/hadoop-yarn/\
   yarn-yarn-resourcemanager-horse.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.

5. View the ResourceManager Web UI using the URL `http://horse:8088`.

6. Select "Local logs" (under Tools).

   A list of log files appears.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

8. Start the NodeManagers.

   Run the following command **on all four hosts in your cluster:**

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

9. Run the `sudo jps` command and review the NodeManager log files on all four hosts to verify that NodeManager daemons are running. If they are not, find and fix any problems and restart the daemons.

10. Run the `sudo jps -v` command on all four hosts to verify that the maximum heap size of the NodeManager is the size that you configured in the `hadoop-env.sh` file.

11. Review the NodeManager log files either by using the command line or the ResourceManager Web UI. If you are using the Web UI, click Nodes (under Cluster) on the left side of the page, and then select the number in the Active Nodes column (which should be 4), then select one of the active NodeManagers, then select Tools, then select Local Logs.

12. Start the JobHistoryServer daemon

    On **`monkey`**:

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

13. Run the `sudo jps` command on `monkey` and verify the presence of the JobHistoryServer daemon.

14. Run the `sudo jps -v` command on `monkey` to verify that the maximum heap size of the JobHistoryServer is the size that you configured in the `hadoop-env.sh` file.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

15. Review the JobHistoryServer log files using the command line.

    On **monkey**:

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

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

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

17. Select "Local logs" (under Tools).

    A list of log files appears.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Testing Your Cluster

If you have performed all the previous steps in the exercise 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 that the cluster is working correctly. Then you will use the Hadoop Web UIs to observe how running the test job in a cluster environment differs from running in a pseudo-distributed cluster.

1. Upload some data to the cluster.

   On **`elephant`**:

   ```
   $ cd ~/training_materials/admin/data
   $ hadoop fs -mkdir -p elephant/shakespeare
   $ hadoop fs -put shakespeare.txt elephant/shakespeare
   ```

2. Access the NameNode Web UI.

3. Click Utilities > "Browse the file system", navigate to the `/user/training/elephant/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 **`elephant`**:

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

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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 in a small font 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 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.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Querying HDFS With Hive and Cloudera Impala

In this exercise, you will install and configure 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 a working ZooKeeper ensemble is a prerequisite for HiveServer2.

Next, you will configure Hive to support concurrent queries and to use a Hive metastore. Then you will initialize the metastore in the existing MySQL database on `elephant` (which already contains the `movie` and `movierating` tables). You will then install and start the Hive Metastore service.

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

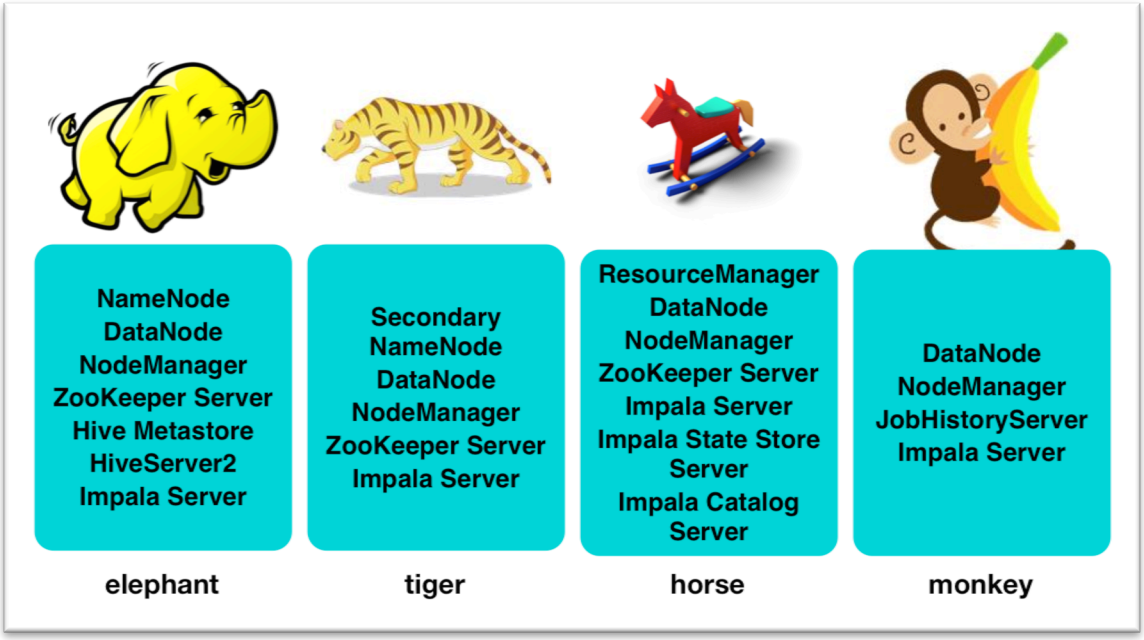
Then you will modify the HDFS configuration for Impala, restart the DataNodes, and install, configure, and start the Impala daemons.

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

Finally, you will install the Impala shell on `elephant`, run Impala queries, and compare the amount of time it took to run the query in Hive to the amount of time it took to run the query in Impala.

At the end of this exercise, you should have daemons deployed on your four hosts as follows:

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.



Installing, Configuring and Starting a ZooKeeper Ensemble

A working ZooKeeper ensemble is required to support concurrency for Hive queries when using HiveServer2. In this task, you will install, configure and start the ZooKeeper ensemble that will support HiveServer2.

1. Install ZooKeeper.

   On **`elephant`**, **`tiger`**, and **`horse`**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Initialize the three ZooKeeper servers with a unique ID.

   On **elephant**:

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

   On **tiger**:

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

   On **horse**:

```
$ sudo service zookeeper-server init --myid 3
```

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

```
server.1=elephant:2888:3888
server.2=tiger:2888:3888
server.3=horse:2888:3888
```

4. Using `sudo`, create a configuration file for Java options at the path `/etc/zookeeper/conf/java.env` on **elephant.** The file should have a single line, as follows:

```
export JVMFLAGS="-Xmx64m"
```

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Copy the Zookeeper configuration files to the other hosts that will run Zookeeper servers: `horse` and `tiger`.

   On **`elephant`**:

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

6. Start the three Zookeeper servers.

   On **`elephant`**, **`tiger`**, and **`horse`**:

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

7. Run the `sudo jps -v` command to verify that the ZooKeeper processes are up and running on `elephant`, `tiger`, and `horse`. The process name is `QuorumPeerMain`. Verify that the maximum heap size is set to 64MB.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing and Configuring Hive

In this task, you will install Hive and configure Hive for HiveServer2 and a centralized Hive Metastore.

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 in the next task. You will also configure Hive to support concurrent queries, which is required when you use HiveServer2.

Then you will create a scratch directory in HDFS that can be used by all users when running Hive queries.

Note that the Impala servers that you will install later in this exercise will share the Hive metastore.

1. Install Hive.

On **elephant**:

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

2. Copy the Hive stub configuration file to the Hive configuration directory. Using this stub file will save you some typing.

On **elephant**:

```
$ cd ~/training_materials/admin/stubs
$ sudo cp hive-site.xml /etc/hive/conf
$ cd /etc/hive/conf
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

3. Using `sudo`, modify the `/etc/hive/conf/hive-site.xml` file on **`elephant`** as follows.

   Caution: Do not copy and paste table entries that exceed a single line—a space will be inserted at the end of each line.

Name	Value
javax.jdo.option.ConnectionURL	jdbc:mysql://elephant:3306/metastore
javax.jdo.option.ConnectionDriverName	com.mysql.jdbc.Driver
javax.jdo.option.ConnectionUserName	hiveuser
javax.jdo.option.ConnectionPassword	password
datanucleus.autoCreateSchema	false
datanucleus.fixedDatastore	true
datanucleus.autoStartMechanism	SchemaTable
hive.metastore.uris	thrift://*<elephant_IP_address>*:9083 (See note on the next page about how to obtain the IP address)
hive.support.concurrency	true
hive.zookeeper.quorum	elephant,tiger,horse

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Note: The `hive.metastore.uris` configuration parameter takes an IP address or a fully qualified host name (FQHN) for its value. Because you are not using FQHNs in your cluster, you must specify the IP address of `elephant` as part of the value for `hive.metastore.uris`.

You can obtain the IP address from the `/etc/hosts` file:

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

Look for a line similar to the following:

```
192.168.123.1 elephant
```

In this example, the IP address is `192.168.123.1`, and 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 in your cluster:

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

5. Make the MySQL connector jar file accessible to Hive.

   On **`elephant`**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Initializing the Hive Metastore in MySQL

In this task, you will create a MySQL database for the Hive metastore.

Hive and Impala can both make use of a single, common Hive metastore. You will configure Hive and Impala to use a common metastore for storing table definitions.

1. Start a MySQL session and create the user `hiveuser` and assign a password and privileges to this user.

   On **elephant**:

```
$ mysql -u root -h localhost
> CREATE DATABASE metastore;
> USE metastore;
> CREATE USER 'hiveuser'@'%' IDENTIFIED BY 'password';
> REVOKE ALL PRIVILEGES, GRANT OPTION FROM \
'hiveuser'@'%';
> GRANT SELECT,INSERT,UPDATE,DELETE,LOCK TABLES, \
EXECUTE,CREATE,ALTER ON metastore.* \
TO 'hiveuser'@'%';
```

2. Open a new terminal window on `elephant`. Run a command to initialize the Hive metastore in the new terminal window.

   On **elephant:**

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

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

```
> REVOKE ALTER,CREATE \
ON metastore.* FROM 'hiveuser'@'%';
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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

```
> SHOW tables;
```

   A list of the Hive metastore tables should appear.

5. Quit your MySQL session.

```
> quit;
```

Installing and Starting the Hive Metastore Service

In this task, you will install and start the Hive Metastore service.

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

1. Install the Hive Metastore service.

   On **`elephant`**:

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

2. Start the service.

   On **`elephant`**:

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

3. Run the `sudo jps` command on `elephant`.

   Verify that the `RunJar` process is active, then run the `ps -ef | grep RunJar` command on `elephant`. (The `RunJar` process runs the Hive Metastore server.)

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing, Configuring, and Starting HiveServer2

In this task, you will install HiveServer2. Then you will make one additional configuration change required for HiveServer2. Then you will start HiveServer2.

At the end of this task, you will run a quick test to validate the installation.

1. Install HiveServer2.

   On **elephant:**

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

2. Add the following line to the end of the `/etc/default/hive-server2` file on **elephant**. This configuration ensures 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 **elephant**:

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

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

   On **elephant**:

   ```
   $ beeline -u jdbc:hive2://elephant:10000/default \
   -n training
   > SHOW tables;
   ```

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Exit the Beeline shell.

```
> !quit
```

Modifying the HDFS Configuration for Impala

In this task, you will configure several 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 **`elephant`** 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 in the table.

2. Copy the `hdfs-site.xml` file from `elephant` to `tiger`, `horse`, and `monkey`. If you like, you can use the `copy_configuration.sh` script that you used in the previous exercise.

   Note: Remember, the `copy_configuration.sh` script is in your shell's PATH. Therefore, you do not need to change directories to run the script.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

3. Restart the DataNodes on `elephant`, `tiger`, `horse`, and `monkey`.

   On **all four hosts in your cluster**:

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

4. Review the DataNode logs and run the `sudo jps` command on all four hosts in your cluster to verify that the DataNode is up and running.

Installing, Configuring, and Starting the Impala Daemons

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

1. Install the Impala State Store Server.

   On **`horse`**:

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

2. Install the Impala Catalog Server.

   On **`horse`**:

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

3. Install the Impala Server.

   On **all four hosts in your cluster**:

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

4. Configure the Impala State Store Server and Impala Catalog Server locations.

   On **`elephant`**, 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 `horse`.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. Copy the Hive configuration on `elephant` to the Impala configuration directory.

   On **`elephant`**:

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

6. Copy Hadoop configuration files required by Impala from `elephant` to the Impala configuration directory.

   On **`elephant`**:

```
$ 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. Run a script that copies the Impala configuration files from `elephant` to the other three hosts.

   On **`elephant`**:

```
$ copy_impala_configuration.sh
```

   Note: This script is specific to the training environment and is *not* part of CDH.

   This script copies the files in the `/etc/impala/conf` directory on `elephant` and the `/etc/default/impala` file to the other three hosts.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

8. Start the Impala State Store Server daemon.

   On **horse**:

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

   Note: If you run the `sudo jps` command now, you will *not* see the Impala State Store daemon, because it is not a Java process. After you have finished starting all the Impala servers, you will run the `ps` command to verify that the processes are active.

9. Start the Impala Catalog Server daemon.

   On **horse**:

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

10. Start the Impala Server daemons.

    On **all four hosts in your cluster**:

    ```
    $ 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. Run the `ps -ef | grep impala` command on all four hosts in your cluster. You should see the `impalad` daemon running on all four hosts. On `horse`, you should also see the `statestored` and `catalogd` daemons.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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.

Note: You already imported the `movierating` table in the Importing Data With Sqoop exercise. But you reformatted HDFS storage during the Creating a Hadoop Cluster, after the initial Sqoop import. Therefore, the `movierating` table that you imported earlier is no longer in HDFS.

1. Import the `movierating` table from MySQL.

   On **`elephant`**:

```
$ sqoop import \
--connect jdbc:mysql://localhost/movielens \
--table movierating --fields-terminated-by '\t' \
--username training --password training
```

2. Review the data loaded into HDFS.

   On any host in your cluster:

```
$ hadoop fs -cat movierating/part-m-00000 | head
```

3. Start the Beeline shell and connect to HiveServer2.

   On **`elephant`**:

```
$ beeline -u jdbc:hive2://elephant:10000 -n training
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

4. Define the `movierating` table in Hive.

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

5. Verify that you created the `movierating` table in the Hive metastore.

   On **elephant**:

```
> SHOW tables;
```

   You should see an entry for the `movierating` table.

6. Run a simple test query that counts the number of rows in the `movieratings` table.

   On **elephant**:

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

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

7. Terminate the Beeline shell.

   On **elephant**:

```
> !quit
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Running Impala Queries

In this task, you will install the Impala shell. Then you will run the same query in Impala that you ran in Hive and compare performance.

1. Install the Impala shell.

   On **elephant**:

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

2. Start the Impala shell.

   On **elephant**:

   ```
   $ impala-shell
   ```

3. Connect to the Impala server running on `horse`.

   ```
   > CONNECT horse;
   ```

   Note: You can connect to any of your four Impala servers to run queries. We use `horse` here as an example. If you like, connect to any of the other servers.

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.

cloudera
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. Exit the Impala shell.

```
> quit;
```

This is the end of the Exercise.

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

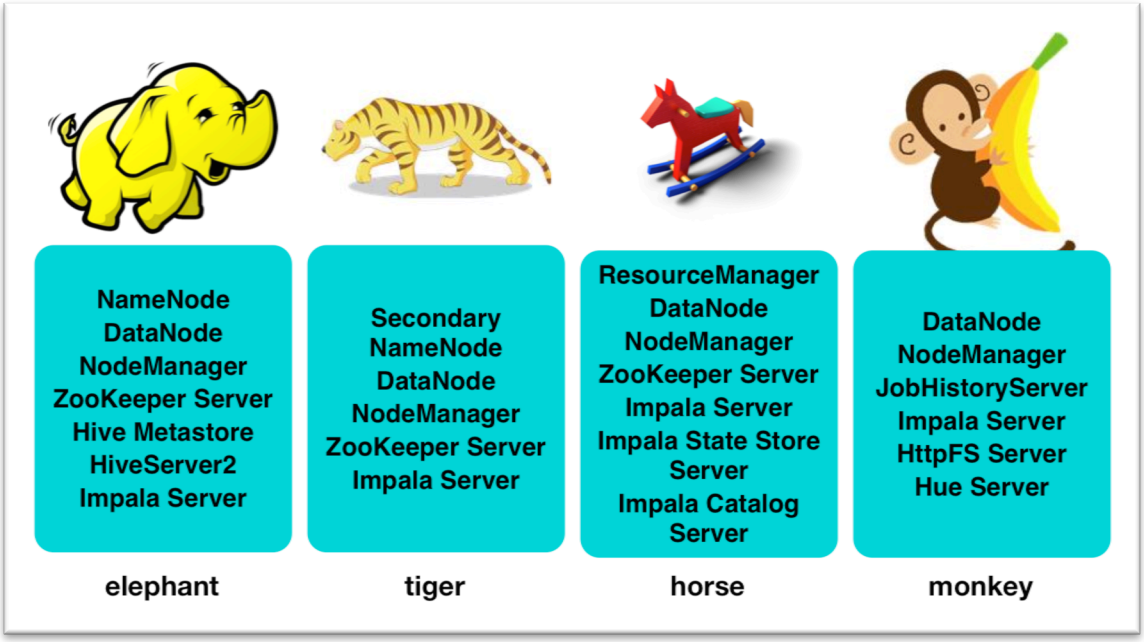
Hands-On Exercise: Using Hue to Control Hadoop User Access

In this exercise, you will configure a Hue environment that provides business analysts with the following capabilities:

- Submitting Pig, Hive, and Impala queries
- Managing definitions in the Hive Metastore
- Browsing the HDFS file system
- Browsing Hadoop jobs

Users will be able to access their environments by using a Web browser, eliminating the need for administrators to install Hadoop client environments on the analysts' systems.

At the end of this exercise, you should have daemons deployed on your four hosts as follows:



The Hue server is deployed on `monkey`. The HttpFS server on `monkey` supports the Hue File Browser application.

cloudera
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing HttpFS and Configuring a Proxy User for HttpFS

In this task, you will install HttpFS. Hue's HDFS browser uses HttpFS to access HDFS in an HA deployment.

Then you will configure Hadoop to allow the user `httpfs` – the user running the HttpFS server – to access HDFS on behalf of other users as a *proxy user*. You must then restart the NameNode for this configuration change to take effect.

Note: In HDFS HA deployments, Hue's file browser can only use HttpFS to access HDFS. In non-HA HDFS deployments, Hue's file browser can use either HttpFS or WebHDFS to access HDFS.

1. Install HttpFS.

   On **monkey**:

   ```
   $ sudo yum install --assumeyes hadoop-httpfs
   ```

2. Start the HttpFS server.

   On **monkey**:

   ```
   $ sudo service hadoop-httpfs start
   ```

3. Use the `curl` command to run the HttpFS `GETHOMEDIRECTORY` operation, verifying HttpFS availability.

   On any host in your cluster:

   ```
   $ curl -s "http://monkey:14000/webhdfs/v1\
   ?op=GETHOMEDIRECTORY&user.name=keri" \
   | python -m json.tool
   ```

   Even though there is no user named `keri` on your system, the `curl` command should succeed, indicating successful installation and startup of HttpFS.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Note: The `GETHOMEDIRECTORY` operation simply returns the string `/user/`, followed by the name of the user you specify in the command. The operation does not access HDFS. Therefore, running this command is an easy way to verify HttpFS operation.

Note: The HttpFS REST API returns JSON objects. Piping the JSON objects to `python -m json.tool` makes the objects easier to read in standard output.

4. Try to run the HttpFS `LISTSTATUS` operation. This operation accesses HDFS.

   On any host in your cluster:

```
$ curl -s "http://monkey:14000/webhdfs/v1/\
user/training?op=LISTSTATUS&user.name=training" \
| python -m json.tool
```

   The HttpFS `LISTSTATUS` operation will fail because the `httpfs` user running the HttpFS server does not have privileges to submit an HDFS request on behalf of the `training` user.

5. Allow the `httpfs` user to act as a proxy user and submit requests on behalf of any other user when submitting requests from `monkey`. Add the following two properties to the `core-site.xml` file **on `elephant`**:

Name	Value
hadoop.proxyuser.httpfs.hosts	monkey
hadoop.proxyuser.httpfs.groups	*

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. Copy the modified `core-site.xml` file from `elephant` to the Hadoop configuration on `tiger`, `horse`, and `monkey` by using the `copy_configuration.sh` script.

7. Restart the NameNode.

   On **`elephant`**:

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

8. Review the NameNode logs and run the `sudo jps` command on `elephant` to verify that the NameNode is up and running.

9. Try to run the HttpFS `LISTSTATUS` operation again.

   On any host in your cluster:

```
$ curl -s "http://monkey:14000/webhdfs/v1/\
user/training?op=LISTSTATUS&user.name=training" \
| python -m json.tool
```

   If you configured the `core-site.xml` file correctly, the `curl` command should work this time, returning a JSON object containing the statuses of all the subdirectories under the `/user/training` path.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing, Configuring, and Starting Hue

Next, you will install Hue on `monkey`. Then you will configure the location of the HttpFS server and the YARN ResourceManager in the Hue configuration and add a couple of users that you can use when you test Hue later in the exercise.

You will also configure Hive so that Hue can run Hive and Impala queries.

At the end of this task, you will start the Hue server.

1. Install the Hue server on `monkey`.

On **`monkey`**:

```
$ sudo yum install --assumeyes hue
```

2. Using `sudo`, edit the `/etc/hue/conf/hue.ini` file on `monkey` and make the changes indicated in the table. To change properties that are present in the hue.ini file but provided as comments, you can simply uncomment them.

Section	Key	Value
[desktop]	secret_key	Any arbitrary sequence of characters
[hadoop] / [[hdfs_clusters]] / [[[default]]]	webhdfs_url	http://monkey:14000/webhdfs/v1/
[hadoop] / [[yarn_clusters]] / [[[default]]]	resourcemanager_host resourcemanager_api_url proxy_api_url history_server_api_url	horse http://horse:8088 http://horse:8088 http://monkey:19888
[beeswax]	hive_server_host	elephant
[impala]	server_host	horse

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

3. Make the MySQL connector library accessible to the Hive installation on `monkey` so that Hive can access the metastore, which is a MySQL database.

   On **monkey**:

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

4. Copy the Hive configuration from `elephant` to `monkey`.

   On **elephant**:

```
$ cd /etc/hive/conf
$ scp hive-site.xml root@monkey:/etc/hive/conf
```

5. Start the Hue server.

   On **monkey**:

```
$ sudo service hue start
```

6. Run the `ps -ef | grep python | grep hue` command on `monkey` and verify that processes running the Hue Cherry Py server and the Hue supervisor are present.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Exploring the Hue User Interface

In this task, you will log in to the Hue UI as an administrative user and briefly explore the following applications: Hue Home page, Hive UI, Cloudera Impala Query UI, Pig Editor, File Browser, Metastore Manager, Job Browser, Hue Shell, User Admin, and Help.

You will also explore how the Hue UI reports misconfiguration and determine why you cannot use the Job Designer and Oozie Editor/Dashboard applications.

1. Submit a Hadoop WordCount job. You will browse output from this job in Hue.

   On **`elephant`**:

   ```
   $ hadoop jar /usr/lib/hadoop-mapreduce/\
   hadoop-mapreduce-examples.jar \
   wordcount elephant/shakespeare test_output_for_hue
   ```

2. If necessary, start the Firefox browser. Maximize the browser window to give Hue enough space to display as many options as possible on its top menu.

3. View the Hue UI. The URL is http://monkey:8888.

4. Authenticate as the `admin` user, with the password `training`.

   The first time that you authenticate to Hue, the administrative user is created, and the Quick Start Wizard appears. Explore the information in the Quick Start Wizard if you like. Do not install any example applications or add any new users at this time.

5. Access the Hue logs. The URL is http://monkey:8888/logs.

6. Click the Home icon:

   The My Documents page appears.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

7. If you completed the Querying HDFS With Hive and Cloudera Impala exercise, start the Hive Query Editor by selecting Query Editors > Hive.

   Enter the following query to verify that Hive is working in Hue:

```
SHOW tables;
```

   Click Execute. The result of the query should be the `movierating` table.

   Enter another query to count the number of records in the `movierating` table:

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

   The query should run successfully.

8. If you completed the Querying HDFS With Hive and Cloudera Impala exercise, start the Impala Query Editor by selecting Query Editors > Impala.

   Enter the following query to verify that Impala is working in Hue:

```
SHOW tables;
```

   Click Execute. The result of the query should be the `movierating` table.

   Enter another query to count the number of records in the `movierating` table:

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

   The query should run successfully—and significantly faster than when you ran the same query with Hive.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

9. Start the Pig UI by selecting Query Editors > Pig.

   The Pig Query Editor appears.

   You can edit and save Pig scripts using Hue's Pig Query Editor in your current Hue deployment.

   Note: Running scripts from the Pig Editor requires Oozie, which you have not installed in your cluster.

10. Select the File Browser option from the top-level menu.

    The Hue File Browser application appears.

    Browse the HDFS file system. If you wish, execute some `hadoop fs` commands from the command line to verify that you obtain the same results from the command line and the Hue File Browser.

11. In the File Browser, navigate to the `/user/admin` path.

    You never explicitly created this path. On the first Hue login, Hue creates a superuser – in this case, the `admin` user – and an HDFS path for that user – in this case, the `/user/admin` path.

12. In the File Browser, navigate to the `/user/training/test_output_for_hue` directory—the output directory of the WordCount job that you ran before starting the Hue UI.

13. Click the entry for the `part-r-00000` file – the output file from the WordCount job.

    A read-only editor showing the contents of the `part-r-00000` file appears.

14. Start the Metastore Manager by selecting Data Browsers > Metastore Tables.

    The Metastore Manager appears with an entry for the `movierating` table.

    Select the entry for the `movierating` table.

    The schema for the Hive `movierating` table, which you created in the Hive exercise, appears in the Metastore Manager.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

15. Select the Job Browser option from the top-level menu.

    An entry for the Hive job that you ran earlier appears in the Hue Job Browser.

    Specify `training` in the Username field.

    An entry for the WordCount job you ran at the beginning of this task appears.

    Browse the job details including the logs. If you wish, access the ResourceManager Web UI, locate the entry for the same job, and compare the information in the Hue Job Browser with the information in the ResourceManager Web UI.

16. Start the User Management Tool by selecting admin > Manage users.

    A screen appears that lets you define Hue users and groups.

    Notice that the automatically created entry for the `admin` user is present.

    You will create another user and a group in the next task.

17. Click the Help icon—a question mark on the top-level menu.

    Hue UI user documentation appears.

18. Click the About Hue icon (at the very left of the top-level menu).

    The Quick Start Wizard appears.

    Note: Oozie, Pig, and Spark are not installed on your cluster. Therefore, the Hue applications that rely on these software components are not operational.

19. Select Configuration (to the right of the Quick Start option).

    Locate the eight parameters that you configured in the `hue.ini` file: `hive_server_host` (beeswax section), `secret_key` (desktop section), `webhdfs_url` (hadoop section), `resourcemanager_host` (hadoop section), `proxy_api_url` (hadoop section), `resourcemanager_api_url` (hadoop section), and `history_server_api_url` (hadoop section), and `server_host` (impala section).

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Setting up the Hue Environment for Business Analysts

You have been given a requirement to set up a Hue environment for business analysts. The environment will allow analysts to submit Hive and Impala queries, edit and save Pig queries, browse HDFS, manage table definitions in the Hive Metastore, and browse Hadoop jobs. Analysts who have this environment will not need Hadoop installed on their systems. Instead, they will access all the Hadoop functionality that they need through a Web browser.

You will use the User Admin application to set up the analysts' Hue environment.

1. Verify that you are still logged in to Hue as the `admin` user.

2. Activate the Hue User Management tool by selecting admin > Manage Users.

3. Select Groups.

4. Add the `analysts` group to Hue. Users in this group should be able to access the About, Hive Query Editor (referred to internally as Beeswax), File Browser, Help, Impala Query Editor, Job Browser, Metastore Manager, and Pig Editor applications.

5. Select Users.

6. Add a Hue `user` named `fred` with the password `training`. Make `fred` a member of the `analysts` group. Make sure that `fred` is *not* a member of the `default` group.

7. Sign out of the Hue UI. The Sign Out icon is at the far right side of the top-level menu.

8. Log back in to the Hue UI as user `fred` with password `training`.

   Verify that in the session for `fred`, only the Hue applications configured for the `analysts` group appear.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Configuring HDFS High Availability

In this exercise, you will reconfigure HDFS, eliminating the NameNode as a single point of failure for your Hadoop cluster.

You will start by shutting down servers that you will no longer use in this exercise or other subsequent exercises.

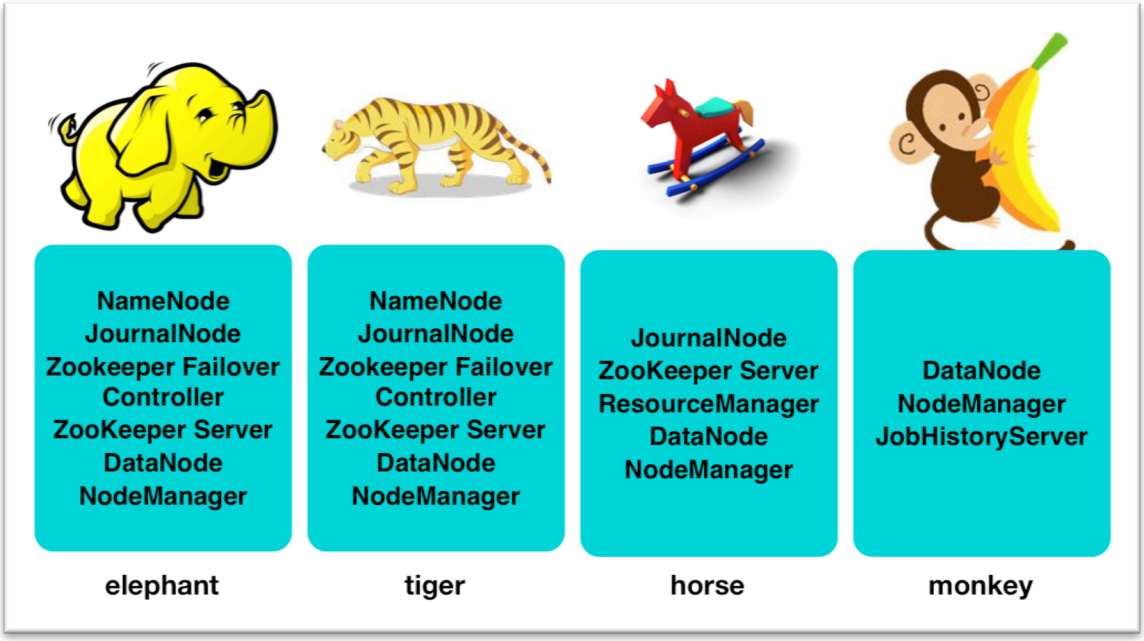
Then you will modify two of the Hadoop configuration files: `core-site.xml` and `hdfs-site.xml`. Next, you will install and start three journal nodes on your systems.

An ensemble of ZooKeeper nodes is required for HDFS high availability (HA). You will use the ZooKeeper ensemble that you created earlier, when you deployed HiveServer2.

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

At the end of this exercise, you should have daemons deployed and running on your four hosts as follows:



The NameNodes on `elephant` and `tiger` are deployed in an HA configuration. The JournalNodes on `elephant`, `tiger`, and `horse` support the HA configuration. The ZooKeeper Failover Controllers on `elephant` and `tiger` provide automatic failover capability. The ZooKeeper servers on `elephant`, `tiger`, and `horse` support the ZooKeeper Failover Controllers.

Bringing Down Unneeded Servers

In this task, you will shut down unneeded servers that you started in previous exercises.

You will no longer use Hive, Impala, or Hue for the remaining exercises. Any servers associated with these components can be shut down now to improve your cluster's performance.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

1. Shut down the Hive Metastore service.

   On **elephant**:

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

2. Shut down HiveServer2.

   On **elephant**:

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

3. Shut down the Impala Servers.

   On **all four hosts in your cluster**:

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

4. Shut down the Impala Catalog Server.

   On **horse**:

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

5. Shut down the Impala State Store Server.

   On **horse**:

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

6. Shut down the HttpFS Server.

   On **monkey**:

```
$ sudo service hadoop-httpfs stop
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

7. Shut down the Hue Server.

   On **monkey**:

```
$ sudo service hue stop
```

Bringing Down the Existing HDFS Configuration

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

- The NameNode on `elephant`
- The SecondaryNameNode on `tiger`
- DataNodes on `elephant`, `tiger`, `horse`, and `monkey`

You do not need to stop the YARN and MapReduce daemons while configuring HDFS HA.

1. Back up your cluster's current configuration so that it will be available in case you run into problems while performing this exercise.

   On **elephant**:

```
$ mkdir /home/training/backup_config
$ cp /etc/hadoop/conf/* /home/training/backup_config
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Stop the SecondaryNameNode.

   On **`tiger`**:

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

3. Stop the NameNode.

   On **`elephant`**:

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

4. Stop the DataNodes.

   On **all four hosts in your cluster**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Modifying the Hadoop Configuration

In this task, you will modify parameters in the `core-site.xml` and `hdfs-site.xml` files to support the following configuration:

- NameNodes running on `elephant` and `tiger`
- JournalNodes running on `elephant`, `tiger`, and `horse`
- Automatic failover supported by a ZooKeeper ensemble running on `elephant`, `tiger`, and `horse`

You will update the two Hadoop configuration files on `elephant`, and then copy the files to the other three systems.

1. On **`elephant`**, modify the following configuration parameter in the `/etc/hadoop/conf/core-site.xml` file:

Name	New Value
fs.defaultFS	hdfs://mycluster

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

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

Name	Value
ha.zookeeper.quorum	elephant:2181,tiger:2181,horse:2181

cloudera
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

2. Add configuration parameters for HDFS HA to the `/etc/hadoop/conf/hdfs-site.xml` file on **`elephant`**.

   You can edit `/etc/hadoop/conf/hdfs-site.xml` and add the properties manually, or, to save typing, you can copy the properties from `~/training_materials/admin/stubs/hdfs-ha-properties.xml` and paste them into `/etc/hadoop/conf/hdfs-site.xml`.

Name	Value
dfs.nameservices	mycluster
dfs.ha.namenodes.mycluster	nn1,nn2
dfs.namenode.rpc-address.mycluster.nn1	elephant:8020
dfs.namenode.rpc-address.mycluster.nn2	tiger:8020
dfs.namenode.http-address.mycluster.nn1	elephant:50070
dfs.namenode.http-address.mycluster.nn2	tiger:50070
dfs.namenode.shared.edits.dir	qjournal://elephant:8485;tiger:8485;horse: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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Note: As a result of specifying `shell(true)` as the fencing method, *no* fencing will be performed during failover. This configuration is suitable for a quick deployment or proof-of-concept. As an optional exercise that you could perform after you have your HA deployment working, you could change the fencing configuration to use `sshfence` as the fencing method, generate an SSH key, and configure its location as described in the course material.

3. On **elephant**, run the `copy_configuration.sh` script to copy the modified Hadoop configuration to the other nodes in your cluster.

Installing JournalNode Software and Starting the JournalNodes

Next, you will install the HDFS JournalNode software on the three hosts that will serve JournalNodes: `elephant`, `tiger`, and `horse`. Then you will start the JournalNodes.

1. Install the JournalNode software.

   On **elephant**, **tiger**, and **horse**:

```
$ sudo yum install --assumeyes 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 **elephant**, **tiger**, and **horse**:

```
$ 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 **elephant**, **tiger**, and **horse**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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

```
export HADOOP_JOURNALNODE_OPTS="-Xmx64m"
```

5. Copy the Hadoop configuration from `elephant` to the other three nodes in your cluster using the `copy_configuration.sh` script.

6. Start the three JournalNodes.

   On **elephant**, **tiger**, and **horse**:

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

7. Run the `sudo jps` command to verify that the `JournalNode` process is up and running on `elephant`, `tiger`, and `horse`.

8. Review the JournalNode log files on `elephant`, `tiger`, and `horse` to determine if problems occurred during JournalNode startup.

Initializing the Shared Edits Directory and Starting the First NameNode

When you convert a deployment that does not have high availability to a high availability deployment – as you are doing on your cluster – you must initialize the shared edits directory with the edits data from the local NameNode edits directories.

After you have initialized the shared edits directory, you will start the `nn1` NameNode on `elephant`.

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` configuration parameter specifies that the `nn1` NameNode will run on `elephant`; the `dfs.namenode.rpc-address.mycluster.nn2` configuration parameter specifies that the `nn2` NameNode will run on `tiger`.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

1. Initialize the shared edits directory.

   On **elephant**:

```
$ sudo -u hdfs hdfs namenode -initializeSharedEdits
```

2. Start the `nn1` NameNode.

   On **elephant**:

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

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

4. Verify that the service state of the `nn1` NameNode running on `elephant` is Standby.

   On **elephant**:

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

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

   Note: You configured the `nn1` NameNode as the NameNode running on `elephant` with the `dfs.namenode.rpc-address.mycluster.nn1` and `dfs.namenode.http-address.mycluster.nn1` configuration parameters in the `hdfs-site.xml` file.

5. Attempt to list files in the `/user/training/elephant` directory.

   On any host in your cluster:

```
$ hadoop fs -ls /user/training/elephant
```

   You are not able to access HDFS because no NameNodes are in Active status.

   Enter Cntl+C to stop the HDFS operation.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Installing, Bootstrapping and Starting the Standby NameNode

Before you can start the standby NameNode on `tiger`, you must install NameNode software and copy the contents of the active NameNode's metadata directories, including the namespace information and most recent checkpoint to the standby NameNode.

1. Install NameNode software on the host that will run the `nn2` NameNode.

   On **`tiger`**:

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

2. Prepare the `nn2` NameNode by bootstrapping it.

   On **`tiger`:**

   ```
   $ sudo -u hdfs hdfs namenode -bootstrapStandby
   ```

3. Start the `nn2` NameNode.

   On **`tiger`:**

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

4. Run the `sudo jps` command and review the NameNode log file on `tiger` and verify that the NameNode started correctly.

5. Verify that the service state of the `nn2` NameNode is Standby.

   On **any host in your cluster:**

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

6. Attempt to list files in the `/user/training/elephant` directory.

   You are still unable to access HDFS because no NameNodes are in Active status.

   Enter Cntl+C to stop the HDFS operation.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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 Controllers:

- You configured a ZooKeeper ensemble and started three ZooKeeper nodes.
- You configured the ZooKeeper ensemble for HDFS HA in the `core-site.xml` file with the `ha.zookeeper.quorum` configuration parameter.
- You set the `dfs.ha.automatic-failover.enabled` parameter to `true` in the `hdfs-site.xml` file.

Now you are ready to install the Failover Controller software, format a znode in ZooKeeper that the controllers will use to keep track of the active and standby nodes, and start the failover controllers.

1. Install the ZooKeeper Failover Controller software.

   On **`elephant`** and **`tiger`**:

   ```
   $ sudo yum install --assumeyes 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 **`elephant`:**

   ```
   export HADOOP_ZKFC_OPTS="-Xmx64m"
   ```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

3. Copy the Hadoop configuration from `elephant` to the other three nodes in your cluster using the `copy_configuration.sh` script.

4. Format the znode in ZooKeeper that keeps track of the Active and Standby nodes.

   On **`elephant` or `tiger`**:

```
$ sudo -u hdfs hdfs zkfc -formatZK
```

5. Start the ZooKeeper Failover Controllers.

   On **`elephant`** and **`tiger`**:

```
$ 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 `elephant` and `tiger` started correctly.

7. Verify that the service state of the `nn1` NameNode on `elephant` is Active and that the service state of the `nn2` NameNode on `tiger` is Standby.

   On **any host in your cluster**:

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

8. Attempt to list files in the `/user/training/elephant` directory.

   You should now be able to access this directory in HDFS.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Starting the DataNodes and Restarting the MapReduce Daemons

Before you can test the high availability deployment, you will start the four DataNodes in your cluster.

You will also restart the ResourceManager and NodeManager daemons. These daemons communicate with HDFS, so they need to obtain the updated HDFS configuration.

1. Start the DataNodes.

   On **all four hosts in your cluster**:

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

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

3. Restart the ResourceManager.

   On **`horse`**:

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

4. Restart the NodeManagers.

   On **all four hosts in your cluster**:

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

5. Restart the JobHistoryServer.

   On **`monkey`**:

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

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

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Testing the HDFS High Availability Deployment

You are now ready to verify that NameNode high availability works on your cluster.

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

   On **all four hosts in your cluster**:

   ```
   $ sudo jps
   ```

   The following processes should be running:

   On `elephant` – `NameNode`, `JournalNode`, `DFSZKFailoverController`, `QuorumPeerMain`, `DataNode`, `NodeManager`

   On `tiger` – `NameNode`, `JournalNode`, `DFSZKFailoverController`, `QuorumPeerMain`, `DataNode`, `NodeManager`

   On `horse` – `JournalNode`, `QuorumPeerMain`, `DataNode`, `ResourceManager`, `NodeManager`

   On `monkey` – `DataNode`, `NodeManager`, `JobHistoryServer`

   Note: The `QuorumPeerMain` process runs the ZooKeeper server.

2. View the NameNode Web UI on `elephant`.

   The first line that appears in the Web UI should be "NameNode 'elephant:8020' (active)."

3. View the NameNode Web UI on `tiger`.

   The first line that appears in the Web UI should be "NameNode 'tiger:8020' (standby)."

4. Bring down the `nn1` NameNode.

   On **`elephant`**:

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


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

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 **`elephant`**:

```
$ 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 **`tiger`**:

```
$ 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 **`tiger`**:

```
$ 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.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Managing Jobs

In this exercise, you will manage jobs using Hadoop's command line tools. We will use a simple 'sleep' job for demonstration purposes: the job allows you to specify the number of Mappers and Reducers you want to run; each will run for a specified number of milliseconds.

1. Open three new terminal windows (or start three new SSH sessions) on **`elephant`.**

2. In the first terminal window, type the following command to run a Hadoop job from `elephant` **but do not press Enter**:

```
$ hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=elephant \
-m 10 -r 10 -mt 20000 -rt 20000
```

3. In the second terminal window, type the following command to run a Hadoop job from `tiger` **but do not press Enter**:

```
$ ssh training@tiger hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=tiger \
-m 10 -r 10 -mt 20000 -rt 20000
```

4. Now press Enter in the first and second terminal window to run the jobs on `elephant` and `tiger` concurrently.

5. In the third terminal window, list the jobs with the following command:

```
$ yarn application -list
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. Take a look at more detailed status for a job. Copy one of the application IDs and issue the following command:

```
$ yarn application -status <application_id>
```

7. Kill both jobs.

```
$ yarn application -kill <application_id>
```

8. Close the three new terminal windows or SSH sessions that you started at the beginning of this exercise.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Using the Fair Scheduler

In this exercise, you will submit some jobs to the cluster and observe the behavior of the Fair Scheduler.

1. Navigate to http://horse:8088/cluster/scheduler to view the Fair Scheduler Administration UI.

2. Open four new terminal windows (or start four new SSH sessions) on **`elephant`**.

   You will run four MapReduce jobs simultaneously in the four windows. Each of the jobs will specify a different pool name.

   Note: Remember, we use the terms *pool* and *queue* interchangeably.

3. In the first terminal window, type the following command to run a Hadoop job from `elephant` **but do not press Enter**:

```
$ hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=elephant \
-D mapreduce.job.queuename=elephant \
-m 10 -r 10 -mt 20000 -rt 20000
```

4. In the second terminal window, type the following command to run a Hadoop job from `tiger` **but do not press Enter**:

```
$ ssh training@tiger hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=tiger \
-D mapreduce.job.queuename=tiger \
-m 10 -r 10 -mt 20000 -rt 20000
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

5. In the third terminal window, type the following command to run a Hadoop job from `horse` **but do not press Enter**:

```
$ ssh training@horse hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=horse \
-D mapreduce.job.queuename=horse \
-m 10 -r 10 -mt 20000 -rt 20000
```

6. In the fourth terminal window, type the following command to run a Hadoop job from `monkey` **but do not press Enter**:

```
$ ssh training@monkey hadoop jar \
~/training_materials/admin/java/sleep.jar SleepJob \
-D mapreduce.job.name=monkey \
-D mapreduce.job.queuename=monkey \
-m 10 -r 10 -mt 20000 -rt 20000
```

7. Now press Enter in all four terminal windows to run the jobs on `elephant`, `tiger`, `horse`, and `monkey` concurrently.

8. Using the Fair Scheduler Administration UI, observe as the pools—for example, `root.monkey`—begin running Map tasks. Some pools will be over their fair share because the first jobs to run will take all available cluster resources as they start.

   Over time, notice that the jobs running over fair share begin to shed resources, which are reallocated to other pools to approach fair share allocation for all pools.

9. Close the four new terminal windows or SSH sessions that you that you started earlier in this exercise.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Breaking The Cluster

In this exercise, you will see what happens during failures of portions of the Hadoop cluster.

1. Load a large file into HDFS.

   Unzip the file `access_log.gz` and place it in your subdirectory on the cluster. This is a 504MB file, and so will be split into multiple blocks.

   On **`elephant`**:

```
$ cd ~/training_materials/admin/data
$ gunzip -c access_log.gz \
| hadoop fs -put - elephant/access_log
```

   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/elephant` directory in HDFS, observe the size and name of the file as it is being uploaded. Refresh the page periodically.

   Discussion point: What do you notice about the size and name of the file being displayed as it is uploaded?

2. Locate a block that has been replicated on `elephant` as follows:

   In the NameNode Web UI, navigate to the `/user/training/elephant/access_log` file and select the file. The File Information window appears.

   Locate the Availability section in the File Information window for Block 0. You should see three hosts on which Block 0 is available. If one of the replicas is on `elephant`, note its Block ID. You will need to refer to the Block ID in the next exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

If none of Block 0's replicas are on `elephant`, view the replication information for other blocks in the file until you locate a block that has been replicated on `elephant`. Once you have located a block that has been replicated on `elephant`, note its block ID.

We will revisit this block when the NameNode recognizes that one of the DataNodes is a 'dead node' (after 10 minutes).

3. Now, intentionally cause a failure and observe what happens.

   Stop the DataNode on **`elephant`:**

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

4. Visit the NameNode's Web UI and click on 'Live Nodes'. Refresh the browser several times and notice that the 'Last Contact' value for the `elephant` DataNode keeps increasing.

5. Run the `hdfs fsck /` command to see that the NameNode currently thinks there are no problems.

   On any host in your cluster:

```
$ sudo -u hdfs hdfs fsck /
```

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Hands-On Exercise: Verifying The Cluster's Self-Healing Features

In this portion of the exercise, you will see what has happened to the data on the dead DataNode.

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

2. View the location of the block from the `access_log` file you investigated in the previous exercise. Notice that Hadoop has automatically re-replicated the data to another host to retain three-fold replication.

3. Run the `hdfs fsck` command to observe that the filesystem is still healthy.

   On any host in your cluster:

   ```
   $ sudo -u hdfs hdfs fsck /
   ```

4. Run the `hdfs dfsadmin -report` command to see that one dead DataNode is now reported.

   On any host in your cluster:

   ```
   $ sudo -u hdfs hdfs dfsadmin -report
   ```

5. Restart the DataNode on **`elephant`** to bring your cluster back to full strength.

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

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Troubleshooting Challenge: Heap O' Trouble

It's 8:30 AM and you are enjoying your first cup of coffee. Keri, who is making the transition from writing RDBMS stored procedures to coding Java MapReduce, shows up in your doorway before you're even halfway through that first cup.

"I just tried to run a MapReduce job and I got an out of memory exception. I heard that there was 32GB on those new machines you bought. But when I run this stupid job, I keep getting out of memory errors. Isn't 32GB enough memory? If I don't fix this thing, I'm going to be in a heap of trouble. I told my manager I was 99% complete with my project but now I'm not even sure if I can do what I want to do with Hadoop."

Put down your coffee and see if you can help Keri get her job running.

Recreating the Problem

1. Run the following commands on **`elephant`**:

```
$ cd ~/training_materials/admin/java
$ hadoop jar EvilJobs.jar HeapOfTrouble \
elephant/shakespeare/shakespeare.txt heapOfTrouble
```


Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Attacking the Problem

The primary goal of this and all the other troubleshooting exercises is to start to become more comfortable analyzing problem scenarios by using Hadoop's log files and Web UIs. **Although you might be able to determine the source of the problem and fix it, doing so successfully is not the primary goal here.**

Take as many actions as you can think of to troubleshoot this problem. Please write down the actions that you take while performing this challenge so that you can share them with other members of the class when you discuss this exercise later.

Fix the problem if you are able to.

Do not turn to the next page unless you are ready for some hints.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Some Questions to Ask While Troubleshooting a Problem

This list of questions provides some steps that you could follow while troubleshooting a Hadoop problem. All of the steps do not necessarily apply to all Hadoop issues, but this list is a good place to start.

- What is there that is different in the environment that was not there before the problem started occurring?
- Is there a pattern to the failure? Is it repeatable?
- If a specific job seems to be the cause of the problem, locate the task logs for the job, including the ApplicationMaster logs, and review them. Does anything stand out?
- Are there any unexpected messages in the NameNode, ResourceManager, and NodeManager logs?
- How is the health of your cluster?
  - Is there adequate disk space?
  - More specifically, does the `/var/log` directory have adequate disk space?
  - Might this be a swapping issue?
  - Is network utilization extraordinarily high?
  - Is CPU utilization extraordinarily high?
- Can you correlate this event with any of the issues?
- If it seems like a Hadoop MapReduce job is the cause of the problem, is it possible to get the source code for the job?
- Does searching the Web for the error provide any useful hints?

Fixing the Problem

If you have time and are able to, fix the problem so that Keri can run her job.

Post-Exercise Discussion

After some time has passed, your instructor will ask you to stop troubleshooting and will lead the class in a discussion of troubleshooting techniques.

This is the end of the Exercise.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Appendix A: Setting up VMware Fusion on a Mac for the Cloud Training Environment

When performing the Hands-On Exercises for this course, you use a small CentOS virtual machine called `Get2EC2`. This VM is configured to use NAT networking. You connect to Amazon EC2 instances from the guest OS by starting SSH sessions. The `Get2EC2` VM is supported for VMware or VirtualBox.

VMware Fusion, like other hypervisors, runs an internal DHCP server for NAT-ted guests that assigns IP addresses to the guests. From time to time, the internal DHCP server releases and renews the guests' leases. Unfortunately, the internal DHCP server in VMware Fusion does not always assign the same IP address to a guest that it had prior to the release and renew, and the `Get2EC2` VM's IP address changes.

Changing the IP address results in problems for active SSH sessions. Sometimes the terminal window in which the client is running will freeze up, becoming unresponsive to mouse and keyboard input, and no longer displaying standard output. At other times, sessions will be shut down with a Broken Pipe error. If this happens, you will have to re-open any failed sessions.

If you are using VMware Fusion on a Mac to perform the Hands-On Exercises for this course, you need to decide whether you would prefer to take action or do nothing:

- If you have administrator privileges on your Mac, you can configure VMware Fusion to use a fixed IP address. The instructions for configuring VMware Fusion to use a fixed IP address appear below.
- If you have VirtualBox installed, you can use the VirtualBox `Get2EC2` VM. instead of VMware Fusion.
- You can do nothing, in which case you might encounter terminal freezes as described above.

To configure VMware Fusion to use a fixed IP address, perform the following steps:

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

1. Start VMware Fusion.

2. Create an entry in the Virtual Machines list for the `Get2EC2` VM. To create the entry, drag the `Cloudera-Training-Get2EC2-VM-1.0.vmx` file to the Virtual Machines list.

   You should see the `Cloudera-Training-Get2EC2-VM-1.0` entry in the Virtual Machines list. We will refer to the `Cloudera-Training-Get2EC2-VM-1.0` VM as the `Get2EC2` VM.

3. Make sure the `Get2EC2` VM is powered down.

4. Click once on the `Get2EC2` VM entry in the Virtual Machines list to select the VM.

   Note: If you accidentally double-click the entry, you start the VM. Before you proceed to the next step, power down the VM.

5. Click the Settings icon in the VMware Fusion Toolbar (or select Virtual Machines > Settings).

6. Click Network Adapter.

7. Click Advanced Options.

   The MAC Address field appears.

8. If the MAC Address field is empty, click Generate to generate a MAC address for the `Get2EC2` VM.

9. Copy the MAC address and paste it into a file where you can access it later. You will need to use the MAC address in a subsequent step.

10. Open the following file on your Mac using superuser (`sudo`) privileges:

    - VMware Fusion 4 and higher: `/Library/Preferences/VMware Fusion/vmnet8/dhcpd.conf`
    - VMware Fusion 3: `/Library/Application Support/VMware Fusion/vmnet8/dhcpd.conf`

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Look for the `range` statement. It should have a range of IP addresses. For example:

```
range 172.16.73.128 172.16.73.254;
```

11. Choose an IP address for the `Get2EC2` VM. The IP address should have the first three tuples of the IP addresses in the `range` statement, but the fourth tuple should be outside of the addresses in the `range` statement. Given the example of the `range` statement in the previous step, you would choose an IP address that starts with `172.16.73` and ends with a number lower than `128` (but not `0`, `1`, or `2` – those numbers are reserved for other purposes).

    For example, `172.16.73.10`.

12. Add four lines to the bottom of the `dhcpd.conf` file as follows:

```
host Get2EC2 {
    hardware ethernet <MAC_Addresss>;
    fixed-address  <IP_Address>;
}
```

    Replace `<MAC_Address>` with the MAC address you generated in an earlier step.

    Replace `<IP_Address>` with the IP address you chose in the previous step.

    Be sure to include the semicolons after the MAC and IP addresses as shown in the example.

13. Save and close the `dhcpd.conf` file.

cloudera®
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

14. Run the following commands from a terminal window on your Mac:

    For VMware Fusion 4 or higher:

```
$ sudo /Applications/VMware\ Fusion.app/Contents/\
Library/vmnet-cli --stop
$ sudo /Applications/VMware\ Fusion.app/Contents/\
Library/vmnet-cli --start
```

    For VMware Fusion 3:

```
$ sudo /Library/Application Support/VMware\ Fusion/\
boot.sh --restart
```

15. Start the `Get2EC2` VM.

16. After the VM has come up, run the following command in a terminal window:

```
$ ip addr
```

    Verify that the IP address that appears is the IP address that you specified in the `dhcpd.conf` file.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

Appendix B: Setting up VirtualBox for the Cloud Training Environment

Follow these steps to set up VirtualBox for the cloud training environment if you do not want to install VMware Fusion on your Mac.

VMware Fusion is our preferred hypervisor for students running this course on Mac OS. Please use VMware Fusion if possible. Use VirtualBox for this course *only* if it is your preferred virtualization environment and if you are knowledgeable enough to be self-sufficient to troubleshoot problems you might run into.

This setup activity comprises:

- Creating the `GetEC2` VM
- Powering up the VM
- Installing VirtualBox Guest Additions on the VM

This setup requires VirtualBox version 4 or higher.

1. Get the `.vmdk` file for the class from your instructor and copy it onto the system on which you will be doing the Hands-On Exercises.
2. Start VirtualBox.
3. Select Machine > New.

   The Name and Operating System dialog box appears.
4. In the Name and Operating System dialog box, specify `Get2EC2` as the Name, `Linux` as the Type, and `Red Hat` (*not* Red Hat 64-bit) as the Version. Click Continue.

   The Memory Size dialog box appears.
5. In the Memory Size dialog box, accept the default memory size of 512 MB and click Continue.

   The Hard Drive dialog box appears.

cloudera
Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

6. In the Hard Drive dialog box, select "Use an existing virtual hard drive file."

7. In the field under the "Use an existing virtual hard drive file" selection, navigate to the `.vmdk` file for the class and click Open.

8. Click Create.

   The Oracle VM VirtualBox Manager dialog box appears. The `Get2EC2` VM appears on the left side of the dialog box, with the status Powered Off.

9. Click Start to start the `Get2EC2` VM.

   The VM starts up. After startup is complete, the GNOME interface appears. You are automatically logged in as the `training` user.

   Now you are ready to install VirtualBox Guest Additions on the VM.

   Note: The version of VirtualBox and Guest Additions must be the same. You must install Guest Additions now to guarantee compatibility between your version of VirtualBox and Guest Additions.

10. Select Devices > Install Guest Additions.

    After several seconds, a dialog box appears prompting you to select how you want to install the version of `VBOXADDITIONS` on your system. Verify that Open Autorun Prompt is selected as the Action, then click OK.

11. Another dialog box appears prompting you to confirm you want to run the Guest Additions installer. Click Run.

12. The Authenticate dialog box appears, prompting you to enter the `root` user's password. Specify `training` and click Authenticate.

13. Messages appear in the terminal window while VirtualBox is building and installing the Guest Additions.

    When installation is complete, the message, "Press Return to close this window" appears in the terminal window.

14. Press Return.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.

15. Select System > “Log Out training” to log out of your GNOME session.

    After you have logged out, you are automatically logged back in as the `training` user.

You have completed the VirtualBox setup. Please return to the next step in “Configuring Networking on Your Cluster: Cloud Training Environment” and continue the setup activity for the cloud training environment.

Copyright © 2010-2014 Cloudera, Inc. All rights reserved.
Not to be reproduced without prior written consent.