
# Cloudera Administrator Training for Apache Hadoop: Hands-On Exercises

**Setup Activity: Configuring Networking** ..... 9
**Hands-On Exercise: Installing Cloudera Manager Server** ..... 17
**Hands-on Exercise: Creating a Hadoop Cluster** ..... 24
**Hands-On Exercise: Working with HDFS** ..... 39
**Hands-On Exercise: Running YARN Applications** ..... 44
**Hands-On Exercise: Explore Hadoop Configurations and Daemon Logs** ..... 56
**Hands-On Exercise: Using Flume to Put Data into HDFS** ..... 62
**Hands-On Exercise: Importing Data with Sqoop** ..... 70
**Hands-On Exercise: Querying HDFS with Hive and Impala** ..... 75
**Hands-On Exercise: Using Hue to Control Hadoop User Access** ..... 86
**Hands-On Exercise: Configuring HDFS High Availability** ..... 99
**Hands-On Exercise: Using the Fair Scheduler** ..... 108
**Hands-On Exercise: Breaking the Cluster** ..... 114
**Hands-On Exercise: Verifying the Cluster's Self-Healing Features** ..... 116
**Hands-On Exercise: Taking HDFS Snapshots** ..... 119

**Hands-On Exercise: Configuring Email Alerts** ..... **121**
**Troubleshooting Challenge: Heap O' Trouble** ..... **123**

# Points to Note During the Exercises

## Directories

The main directory for the exercises is `~/training_materials/admin/`.

Data files used in the exercises are in (`~/training_materials/admin/data`).

## Fewer Step-by-Step Instructions as You Work Through These Exercises

As the exercises progress, and you gain more familiarity with the tools and environment, we may provide fewer step-by-step instructions; as in the real world, we merely give you a requirement and it's up to you to solve the problem! You should feel free to refer to the hints or solutions provided, ask your instructor for assistance, or consult with your fellow students.

## Bonus Exercises

There are additional challenges for some of the hands-on exercises. If you finish the main exercise, please attempt the additional steps.

## Notational Convention

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

```
$ hdfs dfs -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 may be inserted at the end of each line.*

*Dash (-) characters are especially problematic and do not always copy correctly. Be sure to delete any pasted dash characters and key them in manually after pasting in text that contains them.*

Please use caution when copying and pasting when performing the exercises.

**TIP:** If you are using a Mac and you want to copy and paste text from a document on your laptop to a terminal window in the Desktop through the Browser running on Skytap, try this approach:

- `command+C` to copy the text (from the document on your laptop)
- Mouse click in the terminal window in the Gnome Desktop in the browser
- `command+V`, then `command+V` again
- Finally, `shift+control+V` to paste what was copied.

## Catch Up Scripts—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 several exercises, but then you receive an emergency call from work and you must 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 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 five hosts in your cluster is working. If networking has not been configured correctly, you can rerun the `CM_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 executable `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).

The script will further prompt you to specify if you want to run only the steps that simulate the previous exercise, or if you want to completely uninstall and reinstall the cluster and then catch up to the specified exercise. Note that choosing to only complete the previous exercise does not offer as strong an assurance of properly configuring your cluster as a full reset would do. It is however a more expedient option.

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. Once you have answered the initial questions, you do not need to answer any more questions that you may see in the terminal (the scripts answers those questions itself). 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* 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.

Script completion time varies from five minutes to almost an hour depending on how many exercise steps need to be simulated by the script.

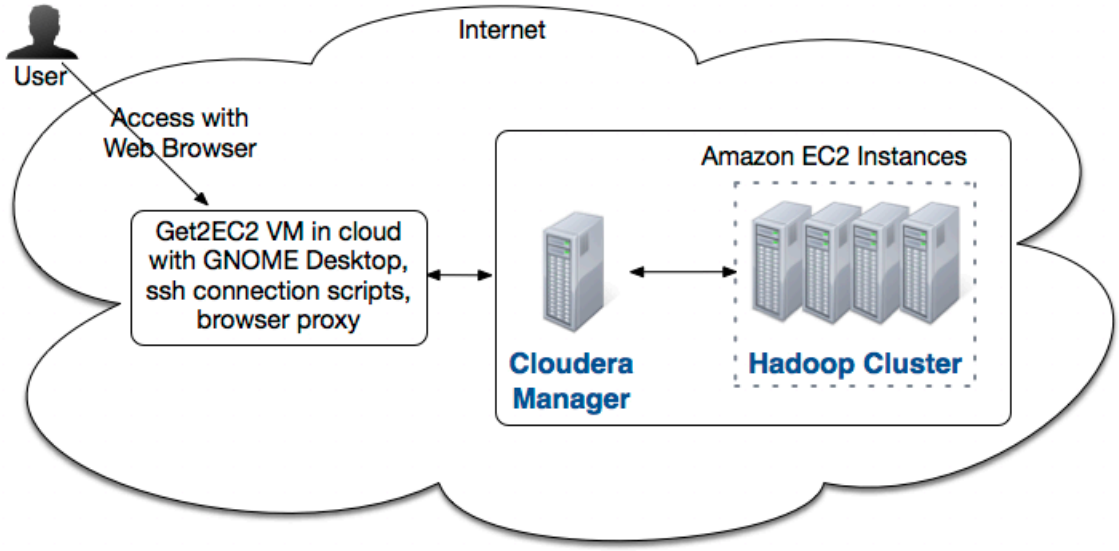
## Training Environment Overview

In this course, you will install Cloudera Manager and CDH (Cloudera Distribution including Apache Hadoop) on five virtual machines (VMs) running in the cloud (for example, on Amazon EC2 or Skytap). These are referred to as the "cluster VMs."

In addition, you will have another VM, called Get2EC2 (or ConnectToCluster), which is typically cloud-hosted with the Desktop accessible via web browser, though it may run locally on your laptop.

The Get2EC2 (or ConnectToCluster) VM provides a Desktop interface, as well as scripts to make SSH connections to the cluster VMs, and a proxy to access the web UIs.

The diagram below shows a typical setup for this course.




- All the VMs use the CentOS 6.6 Linux distribution.
- Use the `training` user account to do your work on any of the VMs. 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.

- The cluster VMs have been configured so that you can make SSH connections to them from the Get2EC2 (or ConnectToCluster) VM without entering a password.

- Please use the hostnames `elephant`, `tiger`, `horse`, `monkey` and `lion` for the five internal hostnames of the cluster VMs. **It is important that you use these five hostnames**, because several scripts you will use in the hands-on exercises, expect these hostnames. The scripts will not work if you use different hostnames.

- If your cluster VMs will run on Skytap, or if you are taking this as an OnDemand course, then you do not need to fill in the table below, because the IP addresses have already been set in the `/etc/hosts` files of your cluster VMs. In this case, you can skip ahead to the "Setup Activity: Configuring Networking" section that follows.

- If you are taking this class with an instructor, and your cluster VMs will run on EC2, your instructor will provide the following details for your five cluster VMs:

  - EC2 *Public* IP addresses—You will run a script that adds the EC2 public IP addresses of your five EC2 instances to the `/etc/hosts` file on your Get2EC2 VM.

  - EC2 *private* IP addresses—You will use these addresses when you run a script that configures the `/etc/hosts` file on each EC2 instance. EC2 private IP addresses start with the number `10`, `172`, or `192.168`

  - Please write out a table like the following with the EC2 instance information:

| Hostname | EC2 Public IP Address | EC2 Private IP Address |
|---|---|---|
| `elephant` | | |
| `tiger` | | |
| `horse` | | |
| `monkey` | | |
| `lion` | | |

# Setup Activity: Configuring Networking

## Task Overview

In this task, you will run scripts to configure networking.

First, you will connect to the `Get2EC2` VM and run a script to configure the `/etc/hosts` file on that VM with the IP addresses and the hostnames of the five cluster VMs.

Next, you will run a script to configure the `/etc/hosts` file on each the cluster VMs, setting the hostnames to `elephant`, `tiger`, `horse`, `monkey` and `lion`.

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 `Get2EC2` (or `ConnectToCluster`) VM.

## Steps to Complete

1. Connect to the `Get2EC2` (or `ConnectToCluster`) VM.

   **If the Get2EC2 (or ConnectToCluster) VM is running in the cloud**:

   - Open a browser (such as Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Internet Explorer 11 or more recent) and go to the "Access URL" provided by your instructor.
   - If the `Get2EC2` (or `ConnectToCluster`) VM is not already in a "Running" state, ask the instructor to start it for you. If you are taking this as an OnDemand course, click the play icon at the top of the screen to start all six VMs.
   - Click on the Desktop thumbnail of the `Get2EC2` (or `ConnectToCluster`) VM to connect to the GNOME Desktop, then proceed to step 2 below.

   **If the Get2EC2 VM will run as a local VM**:

- If you are using VMware Fusion on Mac OS read "Appendix A" and perform any indicated actions.

- 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.

2. Run a script on the `Get2EC2` VM that updates the `/etc/hosts` file with the IP addresses of the VMs where your cluster will be installed.

   **NOTE**: **If you cluster VMs will run on Skytap, skip to step 3**, since the IP addresses are already configured in the `/etc/hosts` file.

   Enter the following command in a terminal window in the `Get2EC2` VM:

   ```
   $ CM_config_local_hosts_file.sh
   ```

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

3. Log in to the `elephant` VM as the `training` user.

   On the `Get2EC2` (or `ConnectToCluster`) VM:

   ```
   $ connect_to_elephant.sh
   ```

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

4. Run a script on `elephant` that updates the `/etc/hosts` files and the `/etc/sysconfig/network` files on the VMs where your cluster will be installed.

**NOTE**: **If you cluster VMs will run on Skytap, skip to step 5**, since the IP address entries in `/etc/hosts` and the hostname entries in `/etc/sysconfig/network` are already configured.

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

```
$ CM_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` each time and then press Enter.

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

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

**Note:** The purpose of exiting and reconnecting to `elephant` is to reset the value of the `$HOSTNAME` variable in the shell after the `CM_config_hosts.sh` script changes the hostname.

If successful, you should now see a `training@elephant:~$` prompt.

6. Start SSH sessions to connect to the other four cluster VMs, logging in as the `training` user.

Open four more terminal windows (or tabs) on your `Get2EC2` (or `ConnectToCluster`) VM, so that a total of five terminal windows are open.

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

```
$ 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`.)

In another terminal window, connect to the `horse` VM.

```
$ connect_to_horse.sh
```

In another terminal window, connect to the `monkey` VM.

```
$ connect_to_monkey.sh
```

In the remaining terminal window, connect to the `lion` VM.

```
$ connect_to_lion.sh
```

7. Verify that you can communicate with all the hosts in your cluster from `elephant` by using the hostnames.

   On **`elephant`**:

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

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

   On **`elephant`**:

```
$ ssh training@tiger ip addr
$ ssh training@horse ip addr
$ ssh training@monkey ip addr
$ ssh training@lion 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 four 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.

**9.** Run the `uname -n` command to verify that all five hostnames have been changed as expected.

On **all five hosts**:

```
$ uname -n
```

**10.** Verify that the value of the `$HOSTNAME` environment variable has been set to the new hostname.

On all five hosts:

```
$ echo $HOSTNAME
```

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

**11.** Start a SOCKS5 proxy server on the `Get2EC2` (or `ConnectToCluster`) VM. The browser on this VM will use the proxy to communicate with your cluster VMs.

Open one more terminal window on the `Get2EC2` (or `ConnectToCluster`) 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.

**12.** Title the proxy terminal window.

Choose Terminal > **Set Title**. Name it "proxy" and click **OK**.

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.

    If when attempting to start the proxy again, you see an error indicating that, "one or more SOCKS5 proxies are already running", note the Process ID(s) returned in the error message, and issue the following command to kill the stale process:

    ```
    $ sudo kill -9 pid
    ```

    where `pid` is the actual process id. Then try executing

    ```
    $ start_SOCKS5_proxy.sh
    ```

    again. This time it should succeed.

## Important Notes About Your Cloud VMs

**If your `Get2EC2` VM is on Skytap and your cluster VMs are on EC2**, the `Get2EC2` VM is configured to *suspend* after a long period without any keyboard or mouse activity. If this occurs (as it likely will after each day of class), your work will not be lost (your EC2 VMs will still be running), but you may need to ask your instructor to un-suspend the `Get2EC2` VM for you. When the `Get2EC2` VM is un-suspended, reestablish SSH sessions with the EC2 instances, and restart the SOCKS5 proxy server. Here are the steps: [1] Open a terminal window (or tab) on the `Get2EC2` (or `ConnectToCluster`) VM, and then execute the appropriate `connect_to` script. [2] Open a separate terminal window, and then execute the `start_SOCKS5_proxy.sh` script.

**If your `Get2EC2` VM is local on VMware, and your cluster VMs are on EC2**, you may want to suspend or stop the `Get2EC2` VM at the end of each day. When the `Get2EC2` VM is un-suspended, reestablish SSH sessions with the EC2 instances, and restart the SOCKS5 proxy server. Here are the steps: [1] Open a terminal window (or tab) on the `Get2EC2` (or `ConnectToCluster`) VM, and then execute the appropriate `connect_to` script. [2] Open a separate terminal window, and then execute the `start_SOCKS5_proxy.sh` script.

**If your `ConnectToCluster` VM is on Skytap and your cluster VMs are also on Skytap**, all six VMs are configured to *suspend* after a long period without any keyboard or mouse activity. If this occurs, your work will not be lost, but you may need to ask your instructor to un-suspend the VMs for you. If you are taking this as an OnDemand course, you can un-suspend any suspended VMs yourself using the play icon (triangle icon) at the top of the browser page that shows the thumbnail views of your VMs. When the VMs are un-suspended, reestablish SSH sessions from the ConnectToCluster VM to the cluster VMs, and restart the SOCKS5 proxy server. Here are the steps: [1] Open a terminal window (or tab) on the `ConnectToCluster` VM, and then execute the appropriate `connect_to` script. [2] Open a separate terminal window, and then execute the `start_SOCKS5_proxy.sh` script.

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

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

# Hands-On Exercise: Installing Cloudera Manager Server

For this installation, you will use Installation Path B to install Cloudera Manager Server on the `lion` machine. Before installing Cloudera Manager, you will configure an external database (MySQL) to be used by Cloudera Manager and CDH, which you will install in the next exercise.

Complete all steps in this exercise on `lion`.

## Verify Existing Environment

1. Verify the Oracle JDK is installed and that JAVA_HOME is defined and referenced in the system PATH.

   **Tip**: use the `connect_to_lion.sh` script to SSH to the lion machine.

   On **lion:**

   ```
   $ java -version
   $ echo $JAVA_HOME
   $ env | grep PATH
   ```

2. Verify Python is installed. It is a requirement for Hue, which you will install later in the course.

   On **lion:**

   ```
   $ sudo rpm -qa | grep python-2.6
   ```

3. Verify Oracle MySQL Server is installed and running on `lion`.

   On **lion:**

```
$ chkconfig | grep mysqld
$ sudo service mysqld status
```

Note that in a true production deployment you would also modify the `/etc/my.cnf` MySQL configurations and move the InnoDB log files to a backup location.

4. Verify the MySQL JDBC Connector is installed. Sqoop (a part of CDH that you will install in this course) does not ship with a JDBC connector, but does require one.

   On **lion:**

```
$ ls -l /usr/share/java
```

## Configure the External Database

In keeping with the approach to installation that is appropriate for a production cluster environment, you will use an external database rather than the embedded PostgreSQL database option.

5. Create the required databases in MySQL.

   On `lion`, view the contents of the `~/training_materials/admin/scripts/`**`mysql-setup.sql`** file (for example, using `cat` or `vi`) and observe the script details. This script creates databases for Cloudera Manager, the Hive Metastore, the Oozie service, the Hue service, the Activity Monitor, and the Reports Manager.

   Note that if you were going to add the Sentry Server CDH service or install Cloudera Navigator, additional databases would also need to be created.

   On **lion:**

```
$ mysql -u root < \
~/training_materials/admin/scripts/mysql-setup.sql
$ mysql -u root
```

Now, at the `mysql>` prompt on `lion`, issue the following commands:

```
show databases;
exit;
```

The `show databases` command should show that the six databases (`amon`, `cmserver`, `hue`, `metastore`, `oozie`, and `rman`) were created.

Note: In a true production deployment, you would also regularly backup your database.

6. Make your MySQL installation secure and set the root user password.

   On **`lion`**:

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] Y
New password: training
Re-enter new password: training
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] Y
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

7. Verify the Cloudera Manager local software repository.

   Your instances contain a local yum repository of Cloudera Manager 5 software to save download time in this course.

   CentOS (and Red Hat) store software repository references in `/etc/yum.repos.d`. Issue the command below to view the settings.

   On **lion**:

```
$ cat /etc/yum.repos.d/cloudera-cm5.repo
```

   View the software packages in a subfolder of the referenced directory.

   On **lion**:

```
$ ls ~/software/cloudera-cm5/RPMS/x86_64
```

Note that these two locations exist on each of the five machines in your environment and have also been made available on HTTP ports 8050 and 8000 respectively via a Linux service. This setup is specific to the course environment and is not required for Cloudera Manager or CDH installations, however it is common to configure a similar setup in secure environments where Cloudera Manager does not have internet access.

If you wanted to install from the online repository, you would create a reference to the Cloudera repository in your `/etc/yum.repos.d` directory.

## Install Cloudera Manager Server

**8.** Install Cloudera Manager Server.

On **`lion`**:

```
$ sudo yum install -y cloudera-manager-daemons \
cloudera-manager-server
```

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

**9.** Set the Cloudera Manager Server service to not start on boot.

On **`lion`**:

```
$ sudo chkconfig cloudera-scm-server off
```

**10.** Run the script to prepare the Cloudera Manager database.

On **`lion`**:

```
$ sudo /usr/share/cmf/schema/scm_prepare_database.sh \
mysql cmserver cmserveruser password
```

After running the command above you should see the message, "All done, your SCM database is configured correctly!"

**11.** Verify the local software repositories.

Run the following two commands to verify the URLs are working.

On **lion:**

```
$ curl lion:8000
$ curl lion:8050/RPMS/x86_64/
```

Each command should show hyperlinks (`<a href ="…"` code) to software repositories. If either command did not successfully contact the web server, discuss with the instructor before continuing.

**12.** Start the Cloudera Manager Server.

On **lion**:

```
$ sudo service cloudera-scm-server start
```

**13.** Review the processes started by the Cloudera Manager Server.

On **lion**:

```
$ top
```

Cloudera Manager Server runs as a java process that you can view by using the `top` Linux utility. Notice the CPU usage is in the 90th percentile or above for a few seconds while the server starts. Once the CPU usage drops, the Cloudera Manager browser interface will become available.

On **lion**:

```
$ ps wax | grep cloudera-scm-server
```

The results of the `ps` command above show that Cloudera Manager Server is using the JDBC MySQL connector to connect to MySQL. It also shows logging configuration and other details.

**14.** Review the Cloudera Manager Server log file.

The path to the log file is `/var/log/cloudera-scm-server/cloudera-scm-server.log`. Note that you must use `sudo` to access Cloudera Manager logs because of restricted permissions on the Cloudera Manager log file directories.

On **`lion`**:

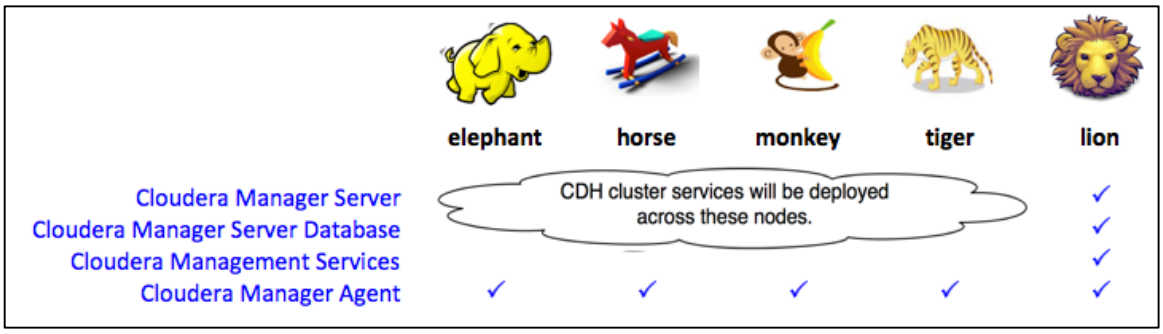
```
$ sudo less /var/log/cloudera-scm-server/cloudera-scm-\
server.log
```

Note: You will login to Cloudera Manager in the next exercise.

**This is the end of the exercise.**

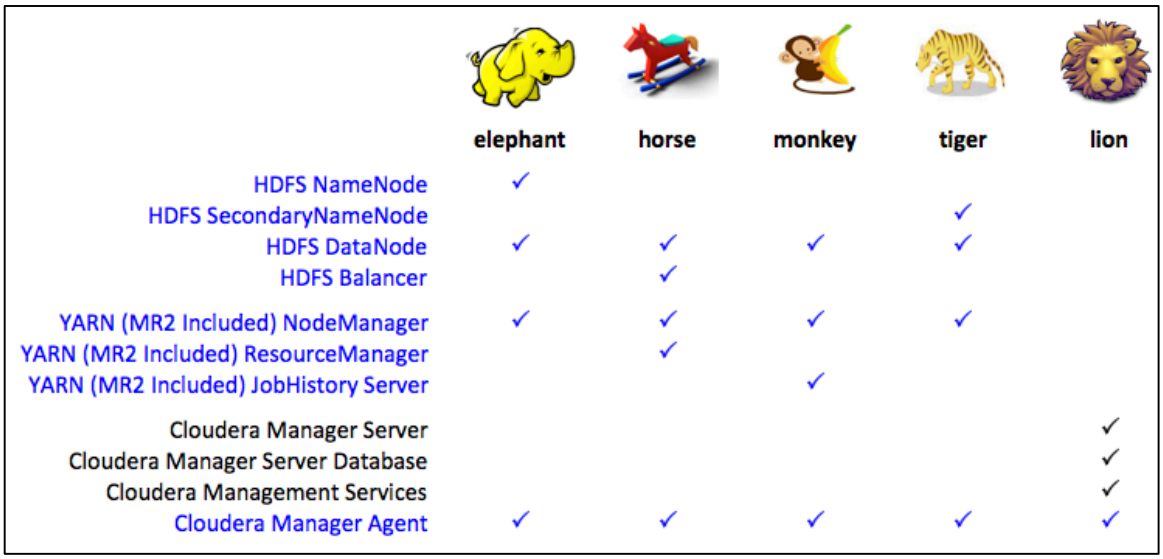
# Hands-on Exercise: Creating a Hadoop Cluster

In this task, you will log in to the Cloudera Manager Admin Console and use the Cloudera Manager services wizard to create a Hadoop cluster. The wizard will prompt you to identify the machines to be managed by Cloudera Manager. It will then install the Cloudera Manager Agent on each machine. At that point, your environment will be ready for the CDH software to be installed.




You will then be prompted to choose which CDH services you want to add in the cluster and to which machines you would like to add each service.

At the end of this exercise, you will have Hadoop daemons deployed across your cluster as depicted here (daemons added in this exercise shown in blue).




In subsequent Exercises, you will add more Hadoop services to your cluster.

Because you have only five hosts to work with, you must run multiple Hadoop daemons on all the hosts except `lion`, which will be used for Cloudera Manager services only. For example, the NameNode, a DataNode and a NodeManager will 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.

After completing the installation steps, you will review a Cloudera Manager Agent log file and review processes running on a machine in the cluster. Finally, there is a section at the end of this exercise that provides a brief tour of the Cloudera Manager Admin UI.

**IMPORTANT**: This exercise builds on the previous one. ***If* you were unable to complete the previous exercise** *or* think you may have made a mistake, run the following command on `elephant` and follow the prompts to prepare for this exercise before continuing (NOTE: ***this script will take several minutes to run*** and an increasingly longer amount of time the further you are into the exercises):

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Login to Cloudera Manager Admin UI

1. Verify the SOCKS5 proxy server you started earlier with the `start_SOCKS5_proxy.sh` script is still running. The proxy server should be running in a separate terminal window on the `Get2EC2` VM.

2. Load the Cloudera Manager Admin Console in browser on the `Get2EC2` VM. The URL is `http://lion:7180`.

   If an "Unable to Connect" message appears, the Cloudera Manager server has not yet fully started. Wait several moments, and then attempt to connect again.

3. Log in with the username `admin`. The password is `admin`.

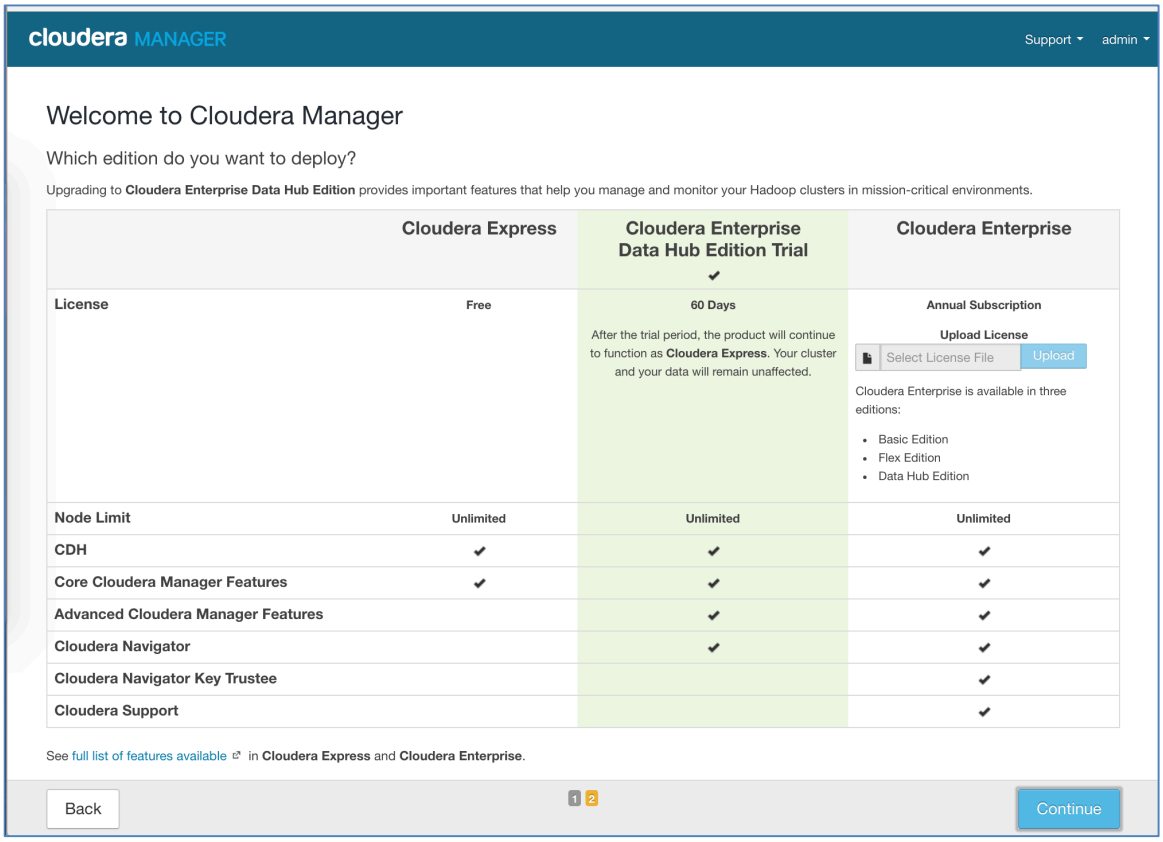
   The "Welcome to Cloudera Manager" page appears with "End User License Terms and Conditions."

Add a checkmark in the box next to “Yes, I accept the End User License Terms and Conditions” at the bottom of the page, and click **Continue**.

4. A “Which edition to you want to deploy?” with a green box highlighting a product edition column should appear.

   Select the “Cloudera Enterprise Data Hub Edition Trial.”

   Click **Continue**.




5. The “Thank you for choosing Cloudera Manager and CDH” page appears.

   Click **Continue**.

## Install Cloudera Manager Agents

6. The “Specify hosts for your CDH cluster installation” page appears.

Type in the names of all five machines: `elephant tiger horse monkey lion`.

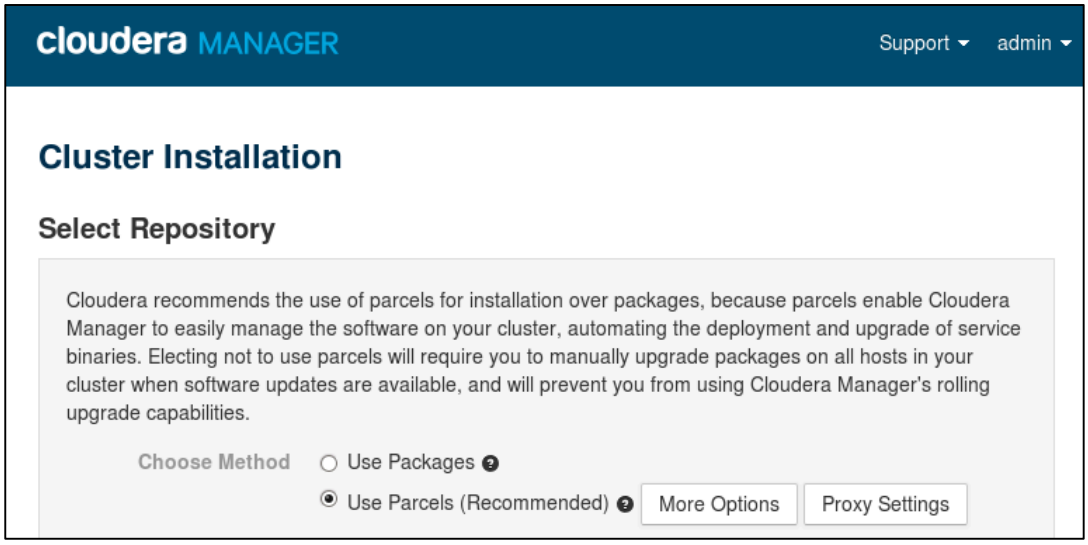
Click **Search**. All five machines should be found. Ensure they are all selected.

| | Expanded Query | Hostname (FQDN) | IP Address | Currently Managed | Result |
|---|---|---|---|---|---|
| ☑ | elephant | elephant | 192.168.123.1 | No | ✓ Host ready |
| ☑ | horse | horse | 192.168.123.3 | No | ✓ Host ready |
| ☑ | lion | lion | 192.168.123.5 | No | ✓ Host ready |
| ☑ | monkey | monkey | 192.168.123.4 | No | ✓ Host ready |
| ☑ | tiger | tiger | 192.168.123.2 | No | ✓ Host ready |

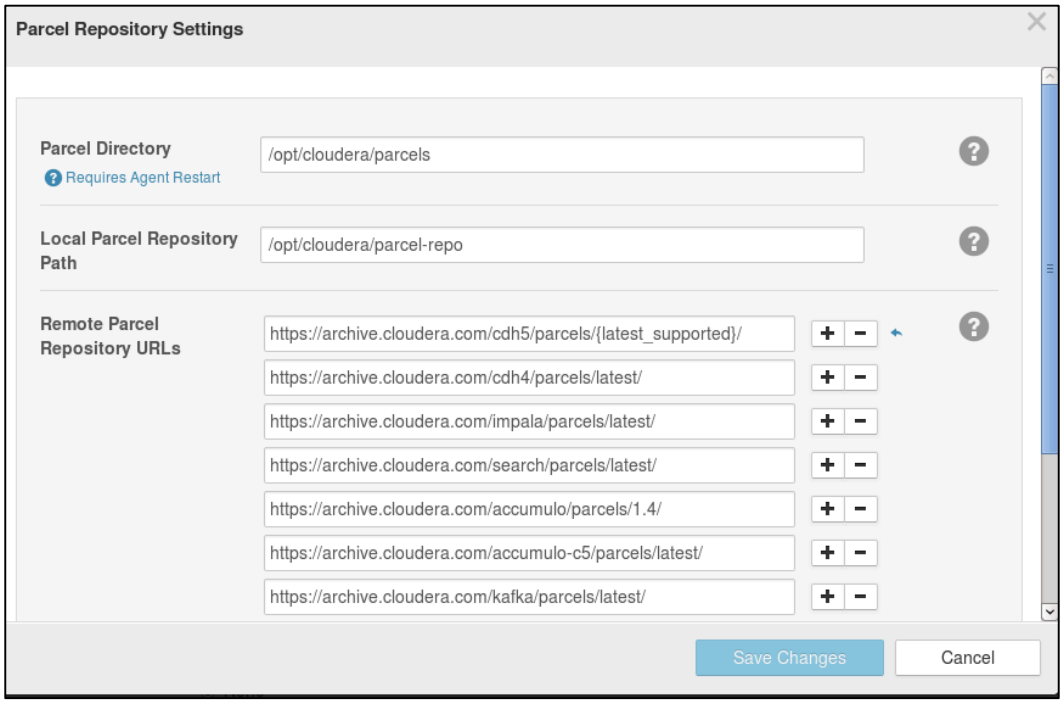
Click **Continue**.

7. The "Cluster Installation – Select Repository" page appears.

In this screen, you will identify the location of the CDH parcel as well as the location of the Cloudera Manager Agent software installer.



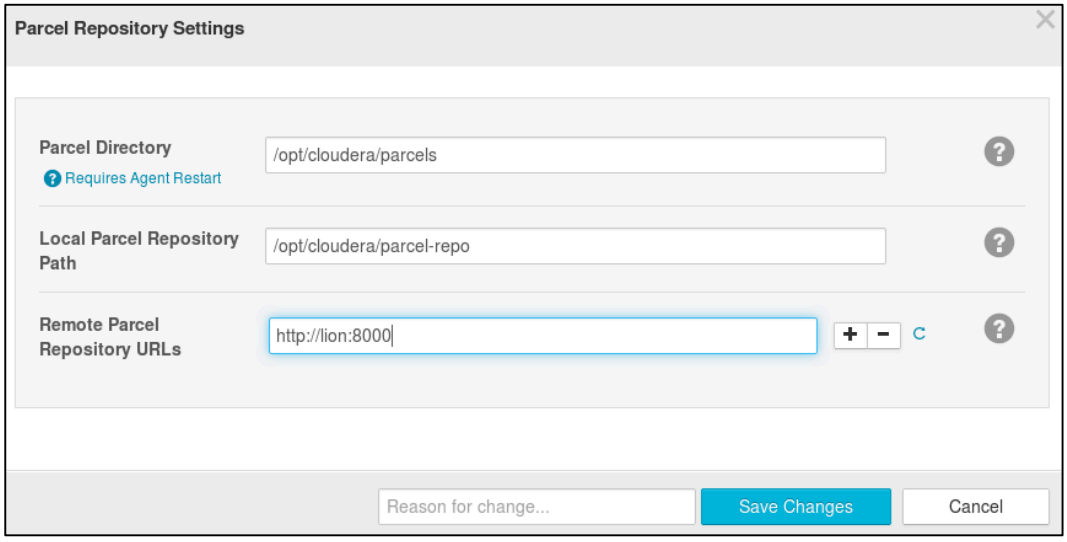

In the "Choose Method" section of the page, where "Use Parcels (Recommended)" is selected, click the **More Options** button.




In the "Remote Parcel Repository URLs" area, click on each of the minus ( **-** ) icons to remove all the current repository references.

Once the existing entries are all removed, click on the plus ( **+** ) icon to add a new Remote Parcel Repository URL.

In the blank field that appears, type in `http://lion:8000`

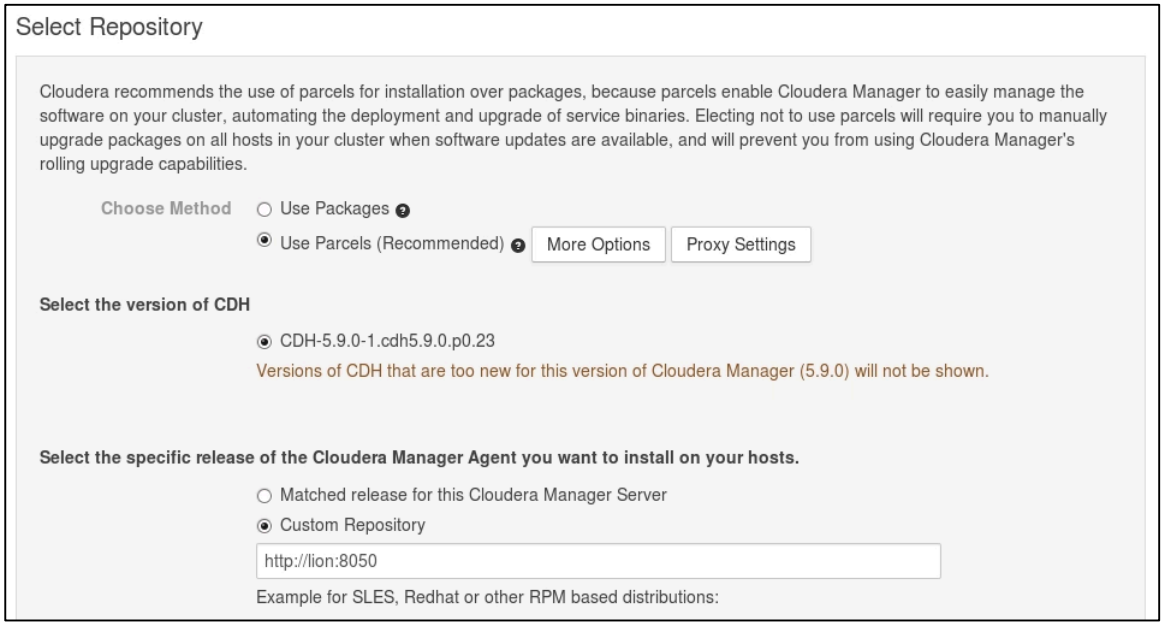


Click **Save Changes** to return to the Select Repository page.

The 'Select Repository' page should now show that a single version of `CDH-5.x.x-x.cdh5.x.x.px.x` (where each x is a number) is available. Select this version of CDH.

Next, in the "Select the specific release of the Cloudera Manager Agent you want to install on your hosts" area choose **Custom Repository**.

- In the blank field that appears, type in `http://lion:8050`




**IMPORTANT**: carefully verify that the details in your browser match the settings shown in the screenshot above before proceeding. If an incorrect repository is chosen, it can take several minutes to reset your environment correctly.

Click **Continue**.

**8.** The "Cluster Installation – JDK Installation Options" page appears.

A supported version of the Oracle JDK is already installed.

Verify the box is *unchecked* so that the installation of JDK will be skipped.

Click **Continue**.

9. The “Cluster Installation – Enable Single User Mode” page appears.

   Keep the default setting (Single User Mode *not* enabled).

   Click **Continue**.

10. The “Cluster Installation – Provide SSH login credentials” page appears.

    Keep “Login To All Hosts As” set to root.

    For “Authentication Method: choose “All hosts accept same private key”

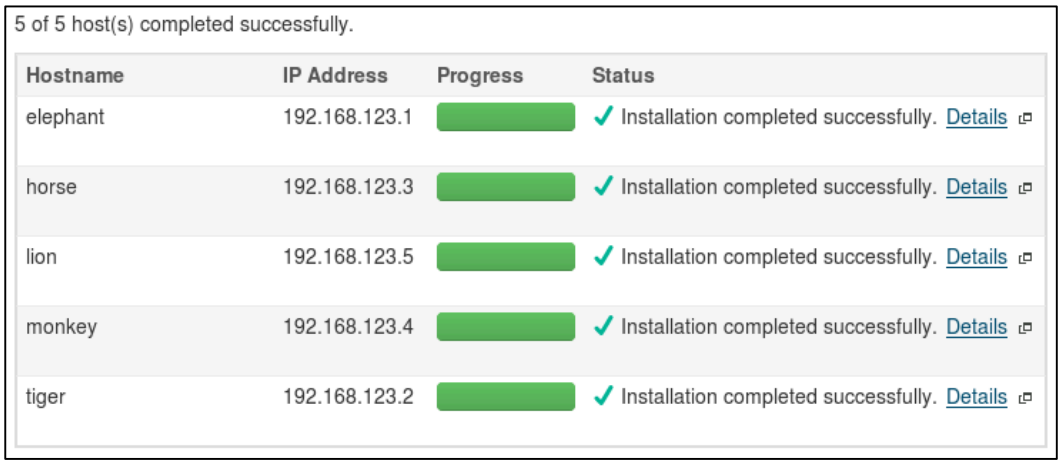
    Click the **Browse** button. In the “Places” column choose **training**. Then, in the “Name” area, right-click (or on a Mac Ctrl-click or two finger-tap on the trackpad), and select **Show Hidden Files**. Finally, click into the `.ssh` directory, select the **`id_rsa`** file and click **Open**.

    Leave the passphrase fields blank fields and click **Continue**.

    When prompted to continue with no passphrase click **OK**.

11. The “Cluster Installation – Installation in progress” page appears.

    Cloudera Manager installs the Cloudera Manager Agent on each machine.




Once the Cloudera Manager Agent installations successfully complete on all machines, click **Continue**.

## Install Hadoop Cluster

**12.** The “Cluster Installation – Installing Selected Parcels” page appears.

The CDH parcel is downloaded, distributed, and activated on all hosts in the cluster.

The “distributing” action may take a few minutes since this involves copying the large CDH parcel file from the lion machine to all the other machines in the cluster.

When it completes, click **Continue**.

**13.** The “Cluster Installation – Inspect hosts for correctness” page appears.

After a moment, output from the Host Inspector appears.

Click **Finish**.

**14.** The “Cluster Setup—Choose the CDH 5 services that you want to install on your cluster” page appears.

Click **Custom Services**.

A table appears with a list of CDH service types.

**15.** Select the **HDFS** and **YARN (MR2 Included)** service types.

You will add additional services in later exercises.

Click **Continue**.

**16.** The “Cluster Setup—Customize Role Assignments” page appears.

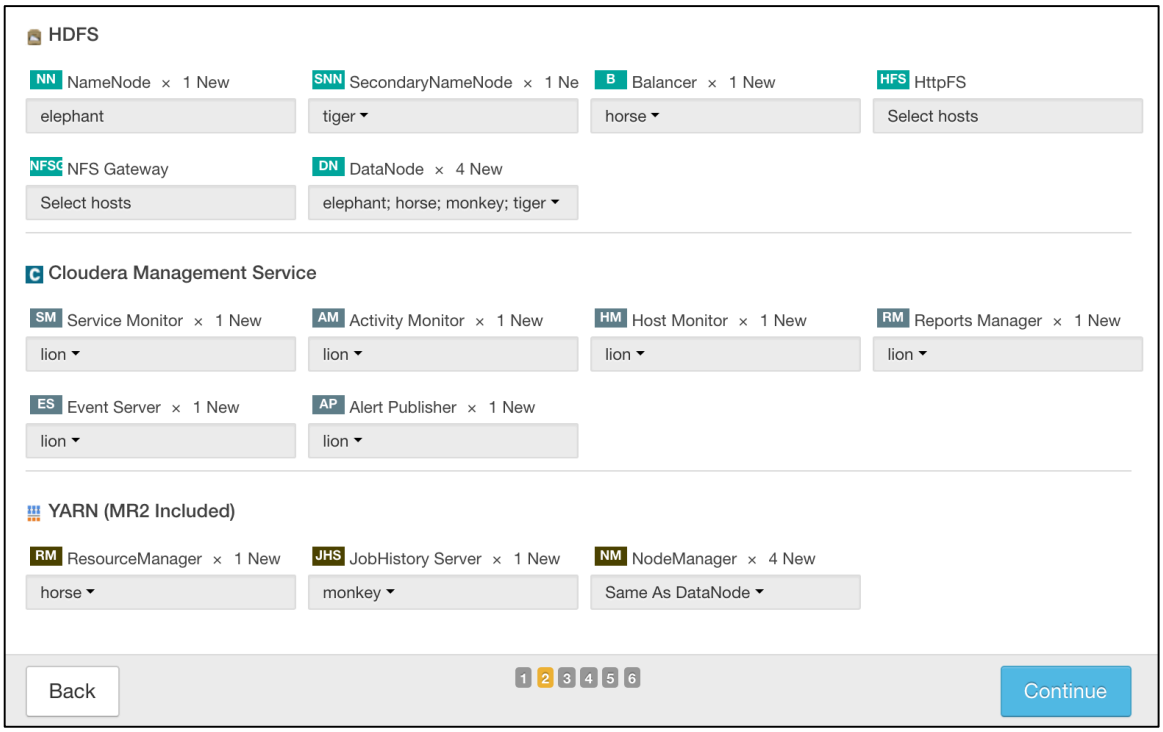
Assign the following roles.

| Role | Node(s) |
|---|---|
| **HDFS** | |
| NameNode | `elephant` |
| SecondaryNameNode | `tiger` |
| Balancer | `horse` |
| HttpFS | Do not specify any hosts |
| NFS Gateway | Do not specify any hosts |
| DataNode | `elephant`, `horse`, `monkey`, `tiger`, **but *not* `lion`**; the order in which the hosts are specified is not significant |
| **Cloudera Management Service** | |
| Service Monitor | `lion` |
| Activity Monitor | `lion` |
| Host Monitor | `lion` |
| Reports Manager | `lion` |
| Event Server | `lion` |
| Alert Publisher | `lion` |
| **YARN (MR2 Included)** | |
| ResourceManager | `horse` |
| JobHistoryServer | `monkey` |
| NodeManager | Same as DataNode (`elephant`, `tiger`, `horse`, `monkey`, but *not* `lion`) |

***To assign a role***, click the fields with one or more hostnames in them. For example, the field under SecondaryNameNode might initially have the value `elephant`. To change the role assignment for the SecondaryNameNode to `tiger`, click the field under SecondaryNameNode. A list of hosts appears. Select `tiger`, and then click OK. The field under SecondaryNameNode now has the value `tiger` in it.

When you have finished assigning roles, compare your role assignments to the role assignments in the screen shot below.

Verify that your role assignments are correct.




When you are certain that you have the correct role assignments, click **Continue**.

**17.** The "Cluster Setup – Database Setup" page appears.

Fill in the details as show here.

| Database Hostname | Database Type | Database Name | Username | Password |
|---|---|---|---|---|
| `lion` | `MySQL` | `amon` | `amonuser` | `password` |
| `lion` | `MySQL` | `rman` | `rmanuser` | `password` |

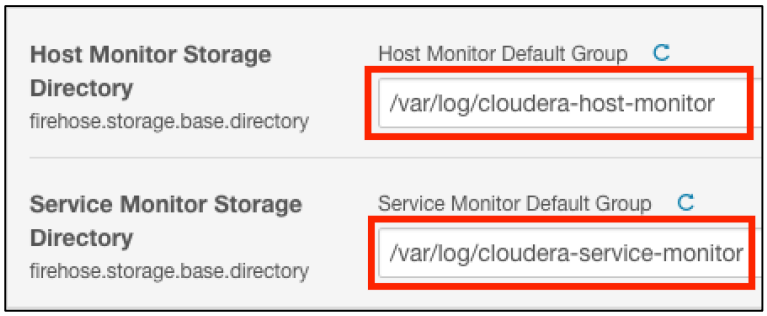
Click **Test Connection** to verify that Cloudera Manager can connect to the MySQL databases you created in an earlier Exercise in this course.

After you have verified that all connections are successful, click **Continue**.

**18.** The "Cluster Setup – Review Changes" page appears.

Specify the following values (change lib to log):

- Host Monitor Storage Directory: `/var/log/cloudera-host-monitor`
- Service Monitor Storage Directory: `/var/log/cloudera-service-monitor`




Click **Continue**.

**19.** The "Cluster Setup- First Run Command" page appears.

Progress messages appear while cluster services are being created and starting.

The following is a summary of what Cloudera Manager's Cluster Setup wizard does for you:

- Formats the HDFS NameNode
- Starts the HDFS service
- Creates a /tmp directory in HDFS
- Creates a MR2 Job History directory in HDFS
- Creates a YARN NodeManager remote application log directory in HDFS
- Starts the YARN service
- Starts the Cloudera Manager Service service
- Deploys all client configurations

When all the cluster services have started, click **Continue**.

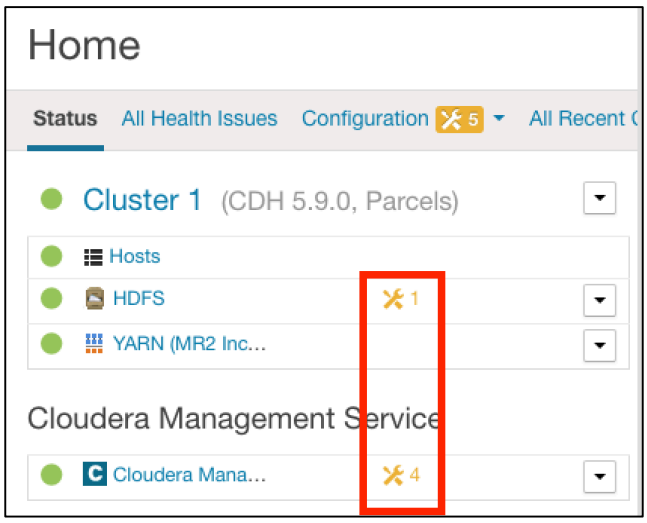
**20.** The "Cluster Setup – Congratulations!" page appears.

The page indicates that services have been added and are now configured and running. Click **Finish**.

The Cloudera Manager Home page appears. Cluster installation is now complete.

**21.** A note regarding the configuration warnings.

The configuration warnings—highlighted in red in the screen shot below—are expected, and indicate that although the services are in good health, they are operating in low memory conditions. In a production deployment, you would want to ensure these warnings were addressed. However, in our training environment these warnings can be safely ignored.




## Examine Running Processes on a Cluster Node

**22.** Review operating system services on a host in the cluster.

On **`elephant`**:

```
$ chkconfig | grep cloudera
$ sudo service cloudera-scm-agent status
```

You have already added YARN (Including MR2) and HDFS services, yet the only service registered with init scripts at the operating system level is the cloudera-scm-agent service.

With Cloudera Manager managed clusters, the `cloudera-manager-agent` service on each cluster node manages starting and stopping the deployed Hadoop daemons.

**23.** Review the running Java processes on a host in the cluster.

On **elephant**:

```
$ sudo jps
```

The Hadoop daemons run as Java processes. You should see the NodeManager, NameNode, and DataNode processes running.

Examine the details of one of the running CDH daemons.

```
$ sudo ps -ef | grep NAMENODE
```

You will see details about the process.

If you grep for DATANODE or NODEMANAGER you will see details on those processes. Grep for "cloudera" to see all CDH and Cloudera Manager processes currently running on the machine.

## Testing Your Hadoop Installation

We will now test the Hadoop installation by uploading some data. The `hdfs dfs` command you use in this step will be explored in greater detail in the next exercise.

**24.** Upload some data to the cluster.

The commands below have you unzip a file on the local drive and then upload it to HDFS' `/tmp` directory.

On **elephant**:

```
$ cd ~/training_materials/admin/data
$ gunzip shakespeare.txt.gz
$ hdfs dfs -put shakespeare.txt /tmp
```

**25.** Verify that the file is now in HDFS.

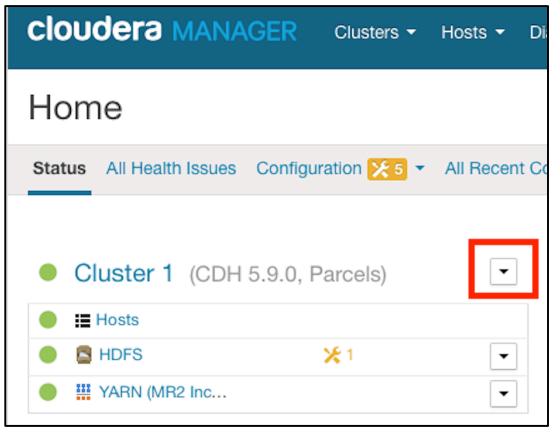
In Cloudera Manager choose Clusters > **HDFS**. Then click on **File Browser**.

Browse into `tmp` and confirm that `shakespeare.txt` appears.

## A Brief Tour of Cloudera Manager

Now that you have the Cloudera Manager managed cluster running, let's briefly explore a few areas in the Admin console.

On the Home page, you will see the cluster named Cluster 1 that you created.




**26.** Click on the drop-down menu to the right of Cluster 1 to see many of the actions you can perform on the cluster, such as Add Service, Stop, and Restart.

**27.** Choose **Hosts** > **All Hosts** to view the current status of each managed host in the cluster. Clicking on the > icon in the Roles column for a host will reveal which roles are deployed on that host.

| Status | Name | IP | Roles | Last Heartbeat | Load Average | Disk Usage | Physical Memory | Swap Space |
|---|---|---|---|---|---|---|---|---|
| ● | elephant | 10.0.0.103 | 3 Role(s): HDFS DataNode, HDFS NameNode, YARN (MR2 Included) NodeManager | 5.95s ago | 0.15 0.04 0.01 | 10.7 GiB / 31.6 GiB | 1.4 GiB / 7.8 GiB | 0 B / 2 GiB |
| ● | horse | 10.0.0.106 | 4 Role(s) | 6.34s ago | 0.00 0.00 0.00 | 10.7 GiB / 31.6 GiB | 1.3 GiB / 7.8 GiB | 0 B / 2 GiB |

In the exercises that follow, you will discover many other areas of Cloudera Manager that will prove useful for administering your Hadoop cluster(s).

**This is the end of the exercise.**

# 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.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

Perform all Command Line steps in this exercise on **`elephant`**.

## Confirm HDFS Processes and Settings

1. Confirm that all HDFS processes are running.

   From the Cloudera Manager Home page click on **HDFS** and then click into the **Instances** page.

   Notice that the three daemons that support HDFS—the `NameNode`, `SecondaryNameNode`, and `DataNode` daemons—are running. In fact, there are DataNodes running on four hosts.

2. Determine the current HDFS replication factor.

   From the Cloudera Manager Home page click into the Search box and type "`replication`".

   Choose **HDFS: Replication Factor** which should be one of the search results.

   You are taken to the HDFS Configuration page where you will find the `dfs.replication` setting that has the default value of `3`.

3. Similarly, use the search box in the HDFS configuration page, and search for "block size" to discover the HDFS Block Size setting which defaults to 128 MiB.

## Add Directories and Files to HDFS

4. As the hdfs user, create a home directory for the training user on HDFS and give the `training` user ownership of it.

   On **elephant**:

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

5. Create an HDFS directory called `/user/training/weblog`, in which you will store the access log.

   On **elephant**:

```
$ hdfs dfs -mkdir weblog
```

6. Extract the `access_log.gz` file and upload it to HDFS.

   On **elephant**:

```
$ cd ~/training_materials/admin/data
$ gunzip access_log.gz
$ hdfs dfs -put access_log weblog
```

   The put command uploaded the file to HDFS. Since the file size is 504MB, HDFS will have split it into multiple blocks. Let's explore how the file was stored.

7. Run the `hdfs dfs -ls` command to review the file's permissions in HDFS.

   On **elephant**:

```
$ hdfs dfs -ls /user/training/weblog
```

## Analyze File Storage in HDFS and On Disk

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

a. From the Cloudera Manager Clusters menu, choose **HDFS** for your cluster.

b. Click on the **NameNode Web UI** link. This will open the NameNode Web UI at `http://elephant:50070`.

c. Choose Utilities > **Browse the file system** then navigate into the `/user/training/weblog` directory.

d. Notice that the permissions shown for the `access_log` file are identical to the permissions you observed when you ran the `hdfs dfs -ls` command.

e. Now select the `access_log` file in the NameNode Web UI file browser to bring up the "File information – access_log" window.

   Notice that the Block Information drop-down list has four entries, Block 0, Block 1, Block 2, and Block 3. This makes sense because as you discovered, the HDFS Block Size on your cluster is 128 MB. The extracted `access_log` file is 559 MB.

   Blocks 0, 1, and 2 all show a size of 134217728 (or 128MB) in accordance with the size specified in the HDFS block size setting you observed earlier in this exercise. Block 3 is smaller than the others as you can observe in the "Size" details if you choose it.

   Notice also that each block is available on three different hosts in the cluster. This is what you would expect since three is the current (and default) replication factor in HDFS. Also, notice that each block may be replicated to different data nodes than the other blocks that make up the file.

f. Choose **Block 0** and write down the value that appears in the Size field. If Block 0 is not replicated on `elephant`, then chose another block that is replicated on `elephant`.

_____

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.

**9.** Locate the HDFS block on the `elephant` Linux file system.

a. In Cloudera Manager's HDFS Configuration page, conduct a search for "Data Directory". You will see that the Data Node Data Directory is `/dfs/dn`.

b. Let's find the blocks stored in that directory.
On **`elephant`**:

```
$ sudo find /dfs/dn -name '*BLKID*' -ls
```

where BLKID is the actual Block ID you copied from the NameNode Web UI. Be sure to keep the * characters in each side of the BLKID in your find command.

c. Verify that two files with the Block ID you copied appear in the `find` command output—one file with an extension, `.meta`, and another file without this extension. The `.meta` file is a metadata file and contains checksums for sections of the block.

d. Verify in the results of the find command output that the size of the file containing the HDFS block is exactly the size that was reported in the NameNode Web UI.

**10.** Start 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.

```
$ sudo head /dfs/dn/path/to/block
```

**Note:** Replace `/path/to/block` in the command above with the actual path to the block as shown in the results of the find command you ran in the previous step.

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

```
$ hdfs dfs -cat weblog/access_log | head -
```

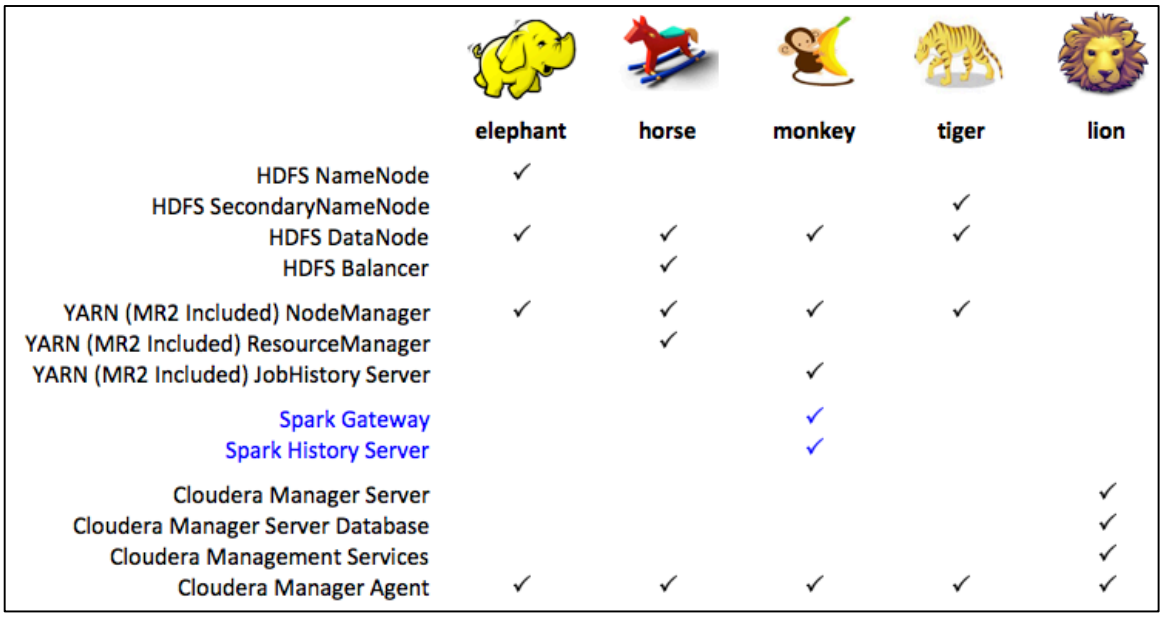
The results returned by the last two commands should match exactly.

**This is the end of the exercise.**

# Hands-On Exercise: Running YARN Applications

In this exercise, you will run two YARN applications on your cluster. The first application is a MapReduce job. The second is an Apache Spark application. You will add the Spark service on your cluster before running the Spark application.

After completing this exercise, your cluster will have the following components installed (items installed in this exercise highlighted in blue):



| | elephant | horse | monkey | tiger | lion |
|---|---|---|---|---|---|
| HDFS NameNode | ✓ | | | | |
| HDFS SecondaryNameNode | | | | ✓ | |
| HDFS DataNode | ✓ | ✓ | ✓ | ✓ | |
| HDFS Balancer | | ✓ | | | |
| YARN (MR2 Included) NodeManager | ✓ | ✓ | ✓ | ✓ | |
| YARN (MR2 Included) ResourceManager | | ✓ | | | |
| YARN (MR2 Included) JobHistory Server | | | ✓ | | |
| Spark Gateway | | | ✓ | | |
| Spark History Server | | | ✓ | | |
| Cloudera Manager Server | | | | | ✓ |
| Cloudera Manager Server Database | | | | | ✓ |
| Cloudera Management Services | | | | | ✓ |
| Cloudera Manager Agent | ✓ | ✓ | ✓ | ✓ | ✓ |

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

Perform all steps in this exercise on **`elephant`**.

## Submitting a MapReduce Application to Your Cluster

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

1. Since the code for the application 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 /opt/cloudera/parcels/CDH/lib/hadoop-\
mapreduce/hadoop-mapreduce-examples.jar wordcount \
/tmp/shakespeare.txt counts
```

## Verifying MapReduce Job Output

2. Once the program has completed you can inspect the output by listing the contents of the output (`counts`) directory.

   On **elephant**:

```
$ hdfs dfs -ls counts
```

3. This directory should show all the data output for the job. Job output will include a `_SUCCESS` flag and one file created by each Reducer that ran. You can view the output by using the `hdfs dfs -cat` command.

   On **elephant**:

```
$ hdfs dfs -cat counts/part-r-00000
```

## Review the MapReduce Application Details and Logs

In this task, you will start by looking at details in the YARN Applications area of Cloudera Manager. The Application Details link in Cloudera Manager will then take you to the HistoryServer Web UI at `http://monkey:19888`.

As you go through the steps below, see if you can reconstruct what occurred where when you ran the MapReduce job, by creating a chart like the one below.

| | **Node(s)** |
|---|---|
| **Application Master** | |
| **Map Task(s)** | |
| **Reduce Task(s)** | |

4. Locate your MapReduce application in Cloudera Manager.

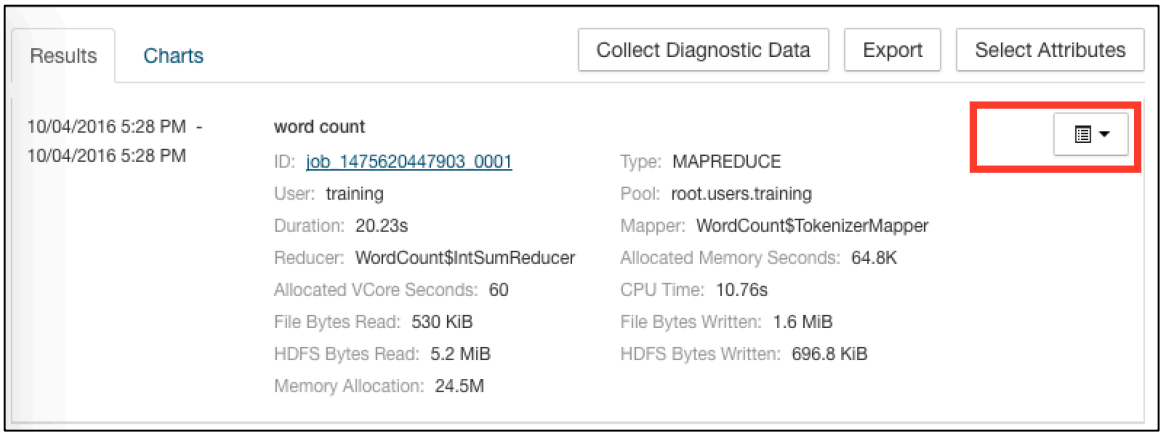
   In Cloudera Manager, choose Clusters > **YARN Applications**.

   In the Results tab that displays, you should see the MapReduce application that just ran.

   **Note:** There will be an entry for each completed MapReduce job that you have run on your cluster within the timeframe of your search. The default search is for applications run within the last 30 minutes.

5. Access the HistoryServer Web UI to discover where the Application Master ran.

   From the drop-down menu for the "word count" application, choose **Application Details**.




This action will open a page in the HistoryServer Web UI with details about the job.

6. Locate where the ApplicationMaster ran and view the log.

   Notice the ApplicationMaster section shows which cluster node ran the MapReduce Application Master. Click the "logs" link to view the ApplicationMaster log. Click the browser's back button when done viewing the log.

   Notice also the number of map and reduce tasks that ran to complete the word count job. The number of reducers run by the job should correspond to the number of `part-r-#####` files you saw when you ran the `hdfs dfs -ls` command earlier. There are no `part-m-#####` files because the job ran at least one reducer.

7. Locate where the mapper task ran and view the log.

   From the HistoryServer Web UI's **Job** menu choose **Map tasks**.

   From the Map Tasks table, click on the link in the Name column for the task.

   The Attempts table displays. Notice the "Node" column shows you where the map task attempt ran.

   Click the **logs** link and review the contents of the mapper task log. When done, click the browser back button to return to the previous page.

8. Locate where the reduce tasks ran and view the logs.

From the HistoryServer Web UI's Job menu choose **Reduce tasks**.

From the Reduce Tasks table, click on the link in the Name column for one of the tasks.

The Attempts table displays. Notice the "Node" column shows you where this reducer task ran.

Click the **logs** link and review the contents of the log. Observe the amount of output in the Reducer task log. When done, click the browser back button to return to the previous page.

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

   Expand the **Job** menu and choose **Configuration**.

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

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

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

   Note: INFO is default value for the "JobHistory Server Logging Threshold" which can be found in the Cloudera Manager YARN Configuration page for your cluster.

## Run the MapReduce Application with a Custom Log Level Setting

10. Remove the `counts` directory from HDFS and rerun the WordCount program, this time passing it a log level argument.

    On **`elephant`**:

```
$ hdfs dfs -rm -r counts
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-\
mapreduce/hadoop-mapreduce-examples.jar wordcount \
-D mapreduce.reduce.log.level=DEBUG \
/tmp/shakespeare.txt counts
```

**Note**: 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 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`.

**Note:** The `-D` option, as used in the `hadoop jar` command above, allows you override a default property setting by specifying the property and the value you want to assign.

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

```
16/10/19 16:50:02 INFO mapreduce.Job: Running job:
job_1476916718448_0002
```

**11.** After the job completes, locate and view one of the reducer logs.

From Cloudera Manager's YARN Applications page for your cluster, locate the entry for the application that you just ran.

Click on the ID link, and use the information available under the Job menu's "Configuration" and "Reduce tasks" links to verify the following:

- The value of the `mapreduce.reduce.log.level` configuration attribute is `DEBUG`.
- The Reducer task's logs for this job contain DEBUG log records and the logs are larger than the number of records written to the Reducer task's logs during the previous WordCount job execution.

**12.** Verify the results of the word count job were written to HDFS using any of the following three methods.

Option 1: In Cloudera Manager, browse to the **HDFS** page for your cluster, then choose **File Browser**. Drill down into `/user/training/counts`.

Option 2: Access the HDFS NameNode Web UI at `http://elephant:50070`. Choose Utilities > **Browse the file system**, and navigate to the `/user/training/counts` directory.

Option 3: on any machine in the cluster that has the DataNode installed (`elephant`, `tiger`, `monkey`, or `horse`) run the following command in a terminal:

```
$ hdfs dfs -tail counts/part-r-00000
```

## Add the Apache Spark Service

In this task, you will add the Spark service to your cluster using Cloudera Manager. You will then run a Spark application.

**13.** In Cloudera Manager, navigate to the Home page.

**14.** Select the **Add Service** option from the drop-down menu for Cluster 1.

The “Add Service to Cluster 1” wizard appears.

15. Select **Spark** and click **Continue**.

16. The “Customize Role Assignments for Spark” page appears.

    Specify host assignments as follows:

    - History Server: `monkey` only
    - Gateway: `monkey` only

    Click **Continue**.

17. Progress messages appear on the “Add Spark Service to Cluster 1” page.

    When the adding of the service has completed, click **Continue**.

18. The Congratulations page appears.

    Click **Finish**.

19. The Home page appears.

    A status indicator shows you that the `Spark` service is in good health.

## Adjust Spark Memory Settings

20. In Cloudera Manager, navigate to the **YARN Configuration** page.

21. Search for `yarn.scheduler.maximum-allocation-mb`. Change the container memory to `1.5GB` and click **Save Changes**.

22. Also, search for and change `yarn.nodemanager.resource.memory-mb` for both the “NodeManager Default Group” and “NodeManager Group 1” to `1.5GB` and **Save Changes**.

23. Notice (just to the right “Actions” menu), the icons indicating “Stale Configuration – restart needed” and “Stale Configuration – client configuration redeployment needed” are displayed.

24. Click on the "Stale Configuration – restart needed" icon.

    The "State Configurations" page displays.

    Cloudera Manager shows the changes that will be made in the YARN configuration files. Note that in addition to the changes you just manually configured, there are some additional configuration changes that will be propagated as a result of the fact that you added the Spark service.

    Click on **Restart Stale Services**.

25. In the "Restart Stale Services" page, ensure "Re-deploy client configuration" is selected, and click **Restart Now**.

    When the restart has completed, click **Finish**.

    Note: you may notice a new configuration warning that appears next to "Hosts" on the Cloudera Manager home page. If you look into it, Cloudera Manager indicates that memory is overcommitted on four of the hosts. These configuration issues, along with the other configuration warnings that appeared after the initial cluster installation, would need to be addressed in a true production cluster, however they can be safely ignored in the classroom environment.

## Run Spark as a YARN Application

You should complete these steps on `monkey` since monkey is where you just added the Spark gateway role.

26. Start the Spark shell and connect to the yarn-client spark context on `monkey`.

    Recall that the Spark Gateway service was installed on `monkey` so the Spark shell should be run from `monkey`.

    On **`monkey`**:

```
$ spark-shell --master yarn-client
```

The Scala Spark shell will launch. You should eventually see the message, "SQL context available as sqlContext." If necessary, click the `Enter` key on your keyboard to see the `scala>` prompt.

**27.** Type in the commands below to run a word count application using the shakespeare.txt file that is already in HDFS.

This accomplishes something very similar to the job you ran in the MapReduce exercise, but this time the computational framework being used is Spark.

```
scala> val file = sc.textFile(
"hdfs://elephant:8020/tmp/shakespeare.txt")
scala> val counts = file.flatMap(line => line.split(
" ")).map(word => (word, 1)).reduceByKey(
_ + _).sortByKey()
scala> counts.saveAsTextFile(
"hdfs://elephant:8020/tmp/sparkcount")
scala> sc.stop()
scala> sys.exit()
```

**28.** View the application results written to HDFS by Spark.

On **elephant**:

```
$ hdfs dfs -cat /tmp/sparkcount/part-00000 | less
```

## Review Application Details in the Spark History Server

View the Spark application details in Cloudera Manager and the Spark History Server Web UI.

**29.** In Cloudera Manager, go to the **YARN Applications** page for your cluster.

You will see a “Spark shell” application.

30. Click on the link in the ID field.

    A page in the YARN Resource Manager Web UI opens with details about the application.

31. Click on the **History** link.

    A Spark Jobs page in the Spark History Server Web UI opens.

    Notice that this Spark Application consisted of two jobs that are now completed.

32. In the Completed Jobs area, click on the “**sortByKey...**” link for the first job that ran (Job Id 0).

    Notice that this first job consisted of two stages.

33. Click on the “**map at...** “ link for the first stage (Stage 0).

    The “Details for Stage 0” page appears.

    Here you can see that there were two tasks in this stage and you can see on which executor and on which host each task ran. You can also see tasks details such as duration, input data size, the amount of data written to disk during shuffle operations.

34. Click on the **Executors** tab to see a summary of all the executors used by the Spark application.

## Review the Spark Application Logs

35. Access the Spark application logs from the command line.

    First locate the application ID.

    On **`elephant`**:

```
$ yarn application -list -appStates FINISHED
```

Copy the application ID for the Spark application returned by the command above.

Now run this command (where appId is the actual application ID).

On **elephant**:

```
$ yarn logs -applicationId appId | less
```

Scroll through the logs returned by the shell. Notice that the logs for all the containers that ran the Spark executors are included in the results.

These Spark application logs are stored in HDFS under `/user/spark/\ applicationHistory`.

**This is the end of the exercise.**

# Hands-On Exercise: Explore Hadoop Configurations and Daemon Logs

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Exploring Hadoop Configuration Settings

In this task, you will explore some of the Hadoop configuration files auto-generated by Cloudera Manager.

1. Go to the directory that contains the Hadoop configuration for Cloudera Manager-managed daemons running on `elephant` and then view the contents.

   On **`elephant`**:

   ```
   $ cd /var/run/cloudera-scm-agent/process
   $ sudo tree
   ```

   Notice how there are separate directories for each role instance running on `elephant`: DataNode, NameNode, and NodeManager. Notice also that some files, such as `hdfs-site.xml` and `core-site.xml`, exist in more than one daemon's configuration directory.

2. Compare the first 20 lines of the NameNode's copy of `hdfs-site.xml` to the NodeManager's copy of the same file.

   On **`elephant`**:

```
$ sudo ls | grep NODE
$ sudo head -20 nn-hdfs-NAMENODE/core-site.xml
$ sudo head -20 nn-yarn-NODEMANAGER/core-site.xml
```

In the commands above, replace each *nn* with the actual numbers generated by Cloudera Manager. In the third command above, choose the *nn*-yarn-NODEMANAGER directory with the highest *nn* value. There are two directories for yarn-NODEMANAGER because you make a settings change in the previous exercise and the agent retained a copy of the previous revision.

The entries in these `core-site.xml` files reflect the settings configured in Cloudera Manager for these role instances. Some of the settings reflect choices you made when you ran the Cloudera Manager installation wizard. Other entries are optimal initial or default values chosen by Cloudera Manager.

3. Make a configuration change in Cloudera Manager.

   In Cloudera Manager, browse to the **HDFS Configuration** page for your cluster.

   Conduct a search for the word "trash". The NameNode Default Group "Filesystem Trash Interval" setting appears.

   Double-click into the Value area where it currently reads "`1 day(s)`".

   Change this value to `2 day(s)`. Click **Save Changes**.

   Notice the  "Stale Configuration – Restart needed" icon that appears on the screen. Click on it.

   The "Stale Configurations" screen appears, showing the changes to core-site.xml that will be made.

   Click **Restart Stale Services**. The "Restart Stale Services" screen appears.

   Click **Restart Now**. The "Restart Stale Services" screen appears. Wait for the child commands to complete successfully. Click **Finish**.

4. Return to the `elephant` terminal window and list the contents of the `/var/run/cloudera-scm-agent/process` directory.

```
$ sudo ls -l /var/run/cloudera-scm-agent/process
```

Notice now how there are now two directories each for NameNode, DataNode, and NodeManager. The old settings have been retained, however the new settings have also been deployed and will now be used. The directory with the higher number in the name is the newer one.

5. Find the difference between the old NameNode `core-site.xml` file and the new one.

   On **`elephant`**:

```
$ sudo diff -C 2 nn-hdfs-NAMENODE/core-site.xml nn-hd\
fs-NAMENODE/core-site.xml
```

   The *nn* values above should be replaced with the actual numbers with which the configuration directories are named.

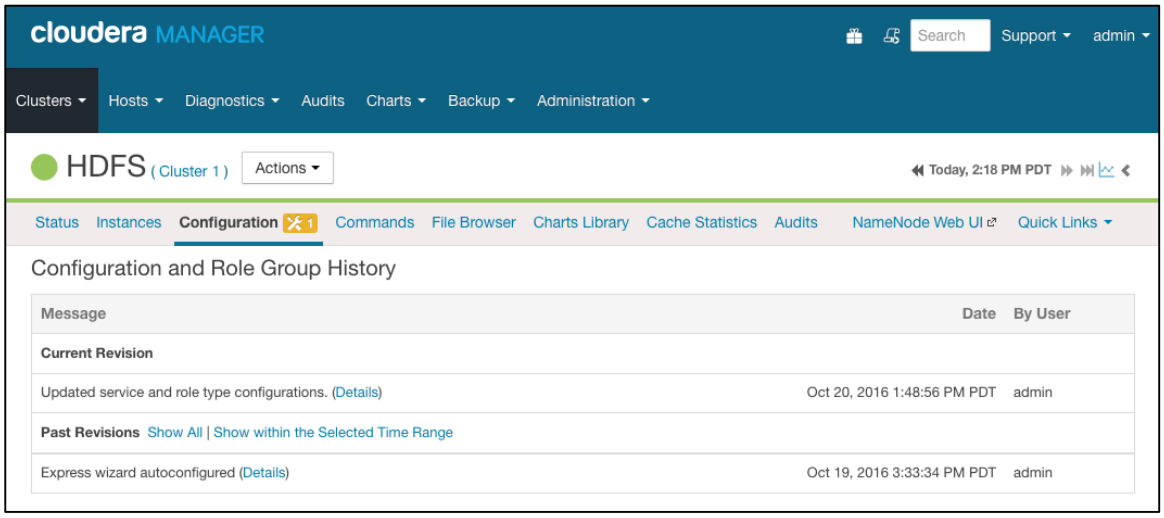
   You should see that the `fs.trash.interval` property value change has been deployed to the new NameNode configuration file.

6. Revert the configuration change and restart the cluster.

   In Cloudera Manager, go to the **HDFS Configuration** page for your cluster.

   Click the **History and Rollback** button.

   The "Configuration and Role Group History" page displays.




Under Current Revision click **Details**.

The "Revision Details" screen displays. Notice the Filesystem Trash Interval property value that you just modified is listed. Choose **Revert Configuration Changes**.

A message appears indicating that the revision was reverted. Click onto the **HDFS Status** page and notice the "Stale Configuration – Restart needed" icon.

Click on the icon and follow the steps to restart the stale services.

## Examining Hadoop Daemon Log Files

In the previous Exercise, you reviewed the application logs from MapReduce and Spark running as YARN applications. Here you will review Hadoop daemon log files, including the HDFS NameNode and YARN ResourceManager log files.

With Cloudera Manager, Hadoop daemons generate a `.log.out` file, a standard error log (`stderr.log`), and a standard output log (`stdout.log`).

In this task, you will examine Hadoop daemon log files using the NameNode Web UI, Cloudera Manager, the ResourceManager Web UI and the NodeManager Web UI.

7. View the NameNode log file using the NameNode Web UI.

   From the Cloudera Manager HDFS page, click the **NameNode Web UI** link.

From the NameNode Web UI, select **Utilities** > **Logs**.

The list of directories and files in the `/var/log/hadoop-hdfs` directory on `elephant` appears.

Open the NameNode log file and review the file.

8. Access the daemon logs directly from Cloudera Manager.

   In Cloudera Manager, choose **Hosts** > **All Hosts** from the top navigation bar.

   Select **elephant** and then choose the **Processes** tab.

   Locate the row for the NameNode and click **Full log file**.

   The Log Details page opens at the tail end of the NameNode log file. Scroll up to view earlier log messages.

   **Note:** If the log file is very large, and you want to see messages near the top, scrolling in the Cloudera Manager UI will be slow. Other tools provide quick access to the entire log file.

   Click **Download Full Log** to download the entire log.

9. Review the NameNode daemons standard error and standard output logs using Cloudera Manager.

   Return to the Processes page for `elephant`.

   Click the **Stdout** link for the NameNode instance. The standard output log appears. Review the file, then return to the Processes page.

   Click the **Stderr** link for the NameNode instance. The standard error log appears. Review the file.

   Note: if you want to locate these log files on disk, they can be found on `elephant` in the `/var/log/hadoop-hdfs` and `/var/run/cloudera-scm-agent/process/nn-hdfs-NAMENODE/logs` directories.

10. Using Cloudera Manager, review recent entries in the SecondaryNameNode logs.

    To find the log, go to the **HDFS Instances** page for your cluster, then locate the SecondaryNameNode role type and click the **tiger** link in the Host column.

In the Status page for the `tiger` host, scroll down to the Roles area and click **Role Log File** in the SecondaryNameNode column.

**11.** Access the ResourceManager log file using the ResourceManager Web UI.

Navigate to the **ResourceManager Web UI** (from the Cloudera Manager YARN page's Web UI menu or by specifying the URL `http://horse:8088` in your browser).

Choose **Nodes** from the **Cluster** menu on the left side of the page.

Click the **horse:8042** link to be taken to the NodeManager Web UI.

Expand the **Tools** menu on the left side of the page.

Click **Local logs**.

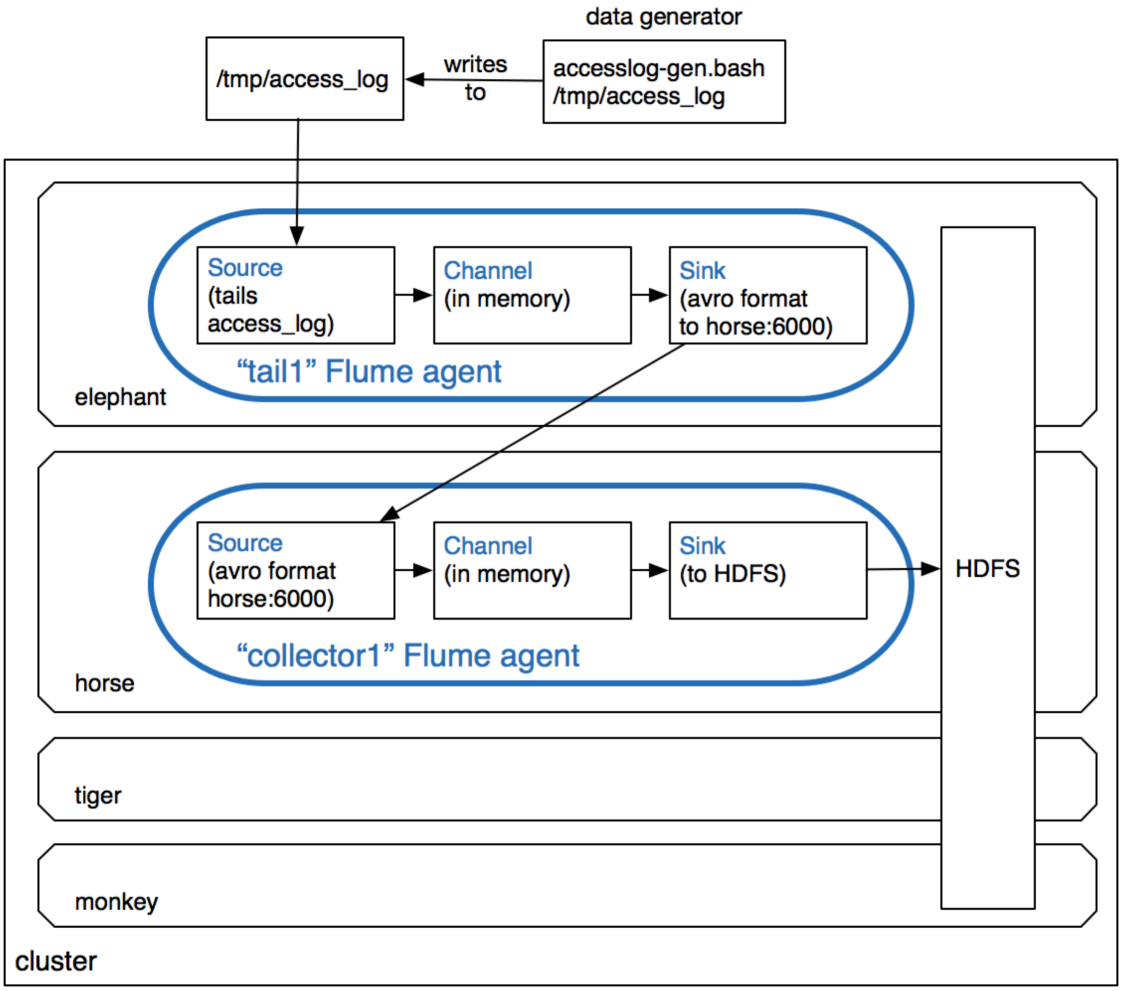
Finally, click the entry for the ResourceManager log file and review the file.

**This is the end of the Exercise.**

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

In this 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 many Web servers on your network; we will simulate a simple version of this in the following exercise.

The diagram below shows the data flow that will occur once you complete the Exercise.



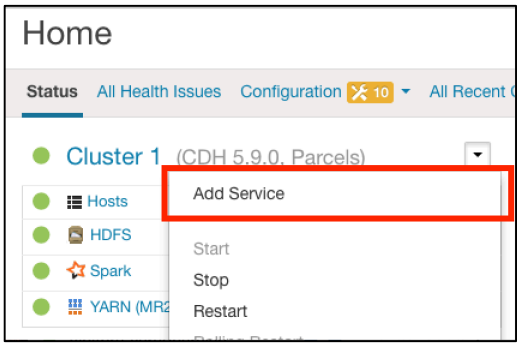
IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Adding the Flume Service

1. Add the Flume service on `elephant` and `horse`.

   From the Cloudera Manager Home page, click the down arrow next to your cluster and choose **Add Service**.




   Select **Flume** and click **Continue**.

   Note that HDFS is a dependency, however the HDFS service is already added.

   Use the **Select hosts** button to add the Flume Agent on both `elephant` and `horse` then click **OK** and **Continue**. At the Congratulations screen click **Finish**.

2. Update the configuration for the Flume agent on `elephant`.

   In Cloudera Manager, navigate to the **Flume Instances** page.

   Select the **Agent** that resides on `elephant`, then click **Configuration**.

   Delete the default contents of the two properties listed in the table below entirely and replace with the lines shown below.

Note: The Configuration File lines are also available in `~/training_materials/admin/scripts/flume-tail1.txt`

*Tip*: You can expand the Configuration File text area to make it easier to edit in by dragging out the bottom right corner of the text box.

| **Property** | **Value** |
|---|---|
| Agent Name | `tail1` |
| Configuration File | `tail1.sources = src1`<br>`tail1.channels = ch1`<br>`tail1.sinks = sink1`<br><br>`tail1.sources.src1.type = exec`<br>`tail1.sources.src1.command = tail -F /tmp/access_log`<br>`tail1.sources.src1.channels = ch1`<br><br>`tail1.channels.ch1.type = memory`<br>`tail1.channels.ch1.capacity = 500`<br><br>`tail1.sinks.sink1.type = avro`<br>`tail1.sinks.sink1.hostname = horse`<br>`tail1.sinks.sink1.port = 6000`<br>`tail1.sinks.sink1.batch-size = 1`<br>`tail1.sinks.sink1.channel = ch1` |

**Tip**: Ensure that each property defined in the configuration file text box is on a single and separate line.

Click **Save Changes**.

3. Update the configuration for the Flume agent on `horse`.

   Return to the Cloudera Manager **Flume Instances** page.

   Select the **Agent** that resides on `horse`, then click **Configuration**.

   Delete the default contents of the two properties listed in the table below entirely and replace with the lines shown below.

   Note: The Configuration File lines are also available in `~/training_materials/admin/scripts/flume-collector1.txt`

| Property | Value |
|---|---|
| Agent Name | `collector1` |
| Configuration File | `collector1.sources = src1`<br>`collector1.channels = ch1`<br>`collector1.sinks = sink1`<br><br>`collector1.sources.src1.type = avro`<br>`collector1.sources.src1.bind = horse`<br>`collector1.sources.src1.port = 6000`<br>`collector1.sources.src1.channels = ch1`<br><br>`collector1.channels.ch1.type = memory`<br>`collector1.channels.ch1.capacity = 500`<br><br>`collector1.sinks.sink1.type = hdfs`<br>`collector1.sinks.sink1.hdfs.path =`<br>`hdfs://elephant/user/flume/collector1`<br>`collector1.sinks.sink1.hdfs.filePrefix = access_log`<br>`collector1.sinks.sink1.channel = ch1` |

Click **Save changes**.

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

   On **elephant**:

```
$ sudo -u hdfs hdfs dfs -mkdir -p \
/user/flume/collector1
$ sudo -u hdfs hdfs dfs -chown -R flume /user/flume
```

## Starting the Data Generator

5. Open a new terminal window on `elephant` (or an ssh connection to `elephant`). In this 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.

   On **elephant**:

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

   **Note:** The `accesslog-gen.sh` script is specific to the training environment and is *not* part of CDH. Leave the script running for now.

6. Open a second new terminal window on `elephant` (or an ssh connection to `elephant`). Verify that the log file has been created. Notice that the log file is rotated periodically.

   On **elephant**:

```
$ 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
```

## Starting the Flume Collector Agent

Here you start the Flume agent that will insert the data into HDFS. This agent receives data from the source Flume agent.

7. Start the collector1 Flume Agent on `horse`.

   In Cloudera Manager, go to the **Flume Instances** page. Tick the checkbox next to the **Agent** hosted on `horse`.

   From the **Actions for Selected (1)** menu choose **Start**. In the window that appears, click **Start**.

   In the "Command Details: Start" screen wait for confirmation that the agent started successfully. Click **Close**.

## Starting the Flume Source Agent

Here you start the agent that reads the source log files and passes the data along to the collector agent you have already started.

8. Start the tail1 Flume Agent on `elephant`.

   From the Cloudera Manager **Flume Instances** page, tick the checkbox next to the Agent hosted on `elephant`.

   From the **Actions for Selected (1)** menu choose **Start**. In the window that appears, click **Start**.

   In the "Command Details: Start" screen wait for confirmation that the agent started successfully. Click **Close**.

## Viewing Data in HDFS

9. Confirm the data is being written into HDFS.

   In Cloudera Manager browse to the **HDFS** page for your cluster and click on **File Browser**.

Drill down into `/user/flume/collector1`. You should see many `access_log` files.

**10.** View Metric Details.

Return to the Flume page and click on the **Metric Details** tab. Here you can see details related to the Channels, Sinks, and Sources of your running Flume agents. If you are interested, an explanation of the metrics available is at `http://tiny.cloudera.com/flumemetrics`.

## Increase the File Size in HDFS (Optional)

These two steps are optional, but may be of interest to some students.

**11.** Edit the **Collector1** agent configuration settings on `horse` by adding these three additional name value pairs:

```
collector1.sinks.sink1.hdfs.rollSize = 2048
collector1.sinks.sink1.hdfs.rollCount = 100
collector1.sinks.sink1.hdfs.rollInterval = 60
```

Click **Save Changes**.

**12.** From the Flume Status page, click on the  "**Stale Configuration – refresh needed**" icon and follow the prompts to refresh the cluster.

**13.** Execute `hdfs dfs -ls /user/flume/collector1` in a terminal window and note the larger file size of the more recent content posted by Flume to HDFS. You may need to wait a few moments before new files are written.

## Viewing the Logs

**14.** Check the log files to see messages.

In Cloudera Manager choose **Diagnostics** > **Logs**.

Click **Select Sources** and configure as follows:

- Uncheck all sources except Flume

- Set the Minimum Log Level to **INFO**
- Leave the timeframe of your search set to 30 minutes

Click **Search**.

Browse through the logged actions from both Flume agents.

## Cleaning Up

**15.** Stop the log generator by hitting `Ctrl+C` in the first terminal window.

**16.** Stop both Flume agents from the **Flume Instances** page in Cloudera Manager.

**17.** Remove the generated access log files from the `/tmp` directory to clear up space on your virtual machine.

On **`elephant`**:

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

**This is the end of the Exercise.**

# 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.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

**Perform all steps done in a terminal 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 -ptraining movielens
   ```

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

   On **`elephant`**:

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

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

   On **`elephant`**:

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

4. Exit MySQL.

   On **`elephant`**:

```
mysql> quit;
```

## Adding the Sqoop 1 Client

5. Add the Sqoop 1 Client gateway on `elephant`.

   From the Home page in Cloudera Manager, click the down arrow icon next to Cluster 1 and choose **Add Service**.

   Select **Sqoop 1 Client** and click **Continue**.

   In the "Custom Role Assignments" page, click in the **Select hosts** box to add the Gateway on `elephant`. Click **Continue**.

   The "Add Sqoop 1 Client Service to Cluster 1" page appears. Once the client configuration has been deployed successfully, click **Continue**. At the "Congratulations" screen click **Finish**.

6. Using `sudo`, create a symlink to the MySQL JDBC driver.

   On **`elephant`**:

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

Now run the command below to confirm the symlink was properly created.

On **elephant**:

```
$ readlink /opt/cloudera/parcels/CDH/lib/\
sqoop/lib/mysql-connector-java.jar
```

If the symlink was properly defined, the command should return the `/usr/share/java/mysql-connector-java.jar` path.

## 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.

7. Show the commands available in Sqoop.

   On **elephant**:

   ```
   $ sqoop help
   ```

   You can safely ignore the warning that Accumulo does not exist since this course does not use Accumulo.

8. 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.)

**9.** List the tables in the `movielens` database.

On **`elephant`**:

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

**10.** 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.

Notice how the INFO messages that appear show that a MapReduce job consisting of four map tasks was completed.

**11.** Verify that the command has worked.

On **elephant**:

```
$ hdfs dfs -ls movie
$ hdfs dfs -tail movie/part-m-00000
```

12. Import the `movierating` table into Hadoop using the command in step 10 as an example.

    Verify that the `movierating` table was imported by using the command in step 11 above as an example, or by using the Cloudera Manager HDFS page's File Browser.

13. Optionally observe the results in Cloudera Manager's YARN Applications page.

    Navigate to the **YARN Applications** page.

    Notice the last two YARN applications that ran (`movie.jar` and `movierating.jar`).

    Explore the job details for either or both of these jobs.

## This is the end of the Exercise

**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].

# Hands-On Exercise: Querying HDFS with Hive and Impala

In this exercise, you will add Hive and Apache Impala (incubating) services to your cluster, enabling you to query data stored in HDFS.

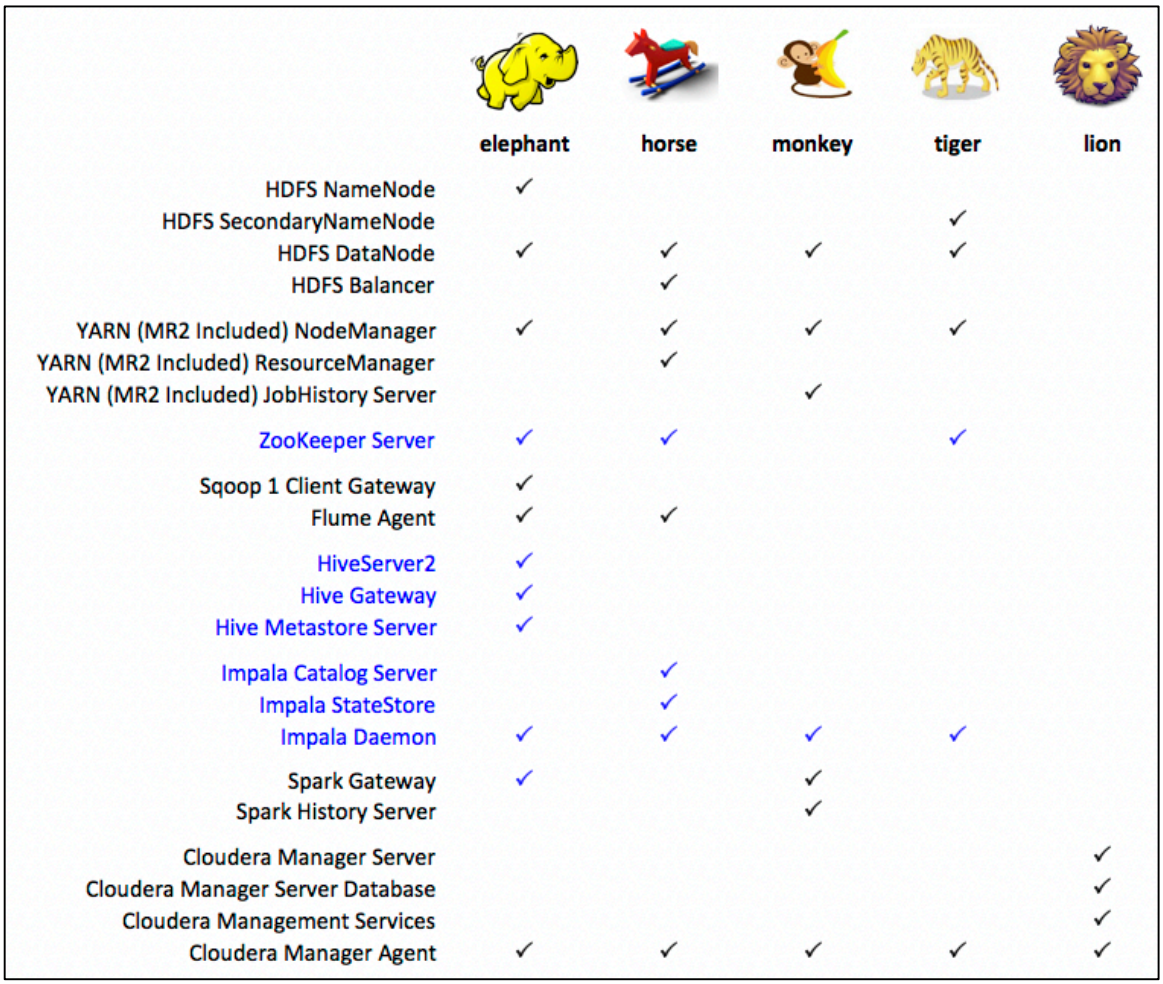
You will start by adding the ZooKeeper service to your cluster. ZooKeeper is a prerequisite for HiveServer2, which you will deploy when you add the Hive service.

Then you will add the Hive service, including a Hive Metastore Server and HiveServer2, to your Hadoop cluster, and configure the service.

Next, you will add the Impala service to your cluster and configure the service.

Then you will populate HDFS with data from the `movierating` table and run queries against it using both Hive and Impala.

At the end of this exercise, you should have daemons deployed on your five hosts as follows (new services added in this exercise are highlighted in blue):



| | elephant | horse | monkey | tiger | lion |
|---|---|---|---|---|---|
| HDFS NameNode | ✓ | | | | |
| HDFS SecondaryNameNode | | | | ✓ | |
| HDFS DataNode | ✓ | ✓ | ✓ | ✓ | |
| HDFS Balancer | | ✓ | | | |
| YARN (MR2 Included) NodeManager | ✓ | ✓ | ✓ | ✓ | |
| YARN (MR2 Included) ResourceManager | | ✓ | | | |
| YARN (MR2 Included) JobHistory Server | | | ✓ | | |
| ZooKeeper Server | ✓ | ✓ | | ✓ | |
| Sqoop 1 Client Gateway | ✓ | | | | |
| Flume Agent | ✓ | ✓ | | | |
| HiveServer2 | ✓ | | | | |
| Hive Gateway | ✓ | | | | |
| Hive Metastore Server | ✓ | | | | |
| Impala Catalog Server | | ✓ | | | |
| Impala StateStore | | ✓ | | | |
| Impala Daemon | ✓ | ✓ | ✓ | ✓ | |
| Spark Gateway | ✓ | | ✓ | | |
| Spark History Server | | | ✓ | | |
| Cloudera Manager Server | | | | | ✓ |
| Cloudera Manager Server Database | | | | | ✓ |
| Cloudera Management Services | | | | | ✓ |
| Cloudera Manager Agent | ✓ | ✓ | ✓ | ✓ | ✓ |

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Adding the ZooKeeper Service

In this task, you will add a ZooKeeper service to your cluster. A running ZooKeeper service is a prerequisite for many other services so you will add this service now.

When you add additional services later in the class, you may notice the Exercise instructions have you select ZooKeeper as part of the set of dependencies for the new service to use.

1. From the Cloudera Manager Home page, select the **Add Service** menu option from the drop-down menu to the right of Cluster 1.

   The Add Service wizard appears.

2. Select **ZooKeeper** and click **Continue**.

   The Customize Role Assignments page appears.

3. Specify the following host assignments:

   - Server: `elephant`, `horse`, and `tiger` but ***not*** `lion` or `monkey`

   Click **OK** and then click **Continue**.

4. The Review Changes page appears.

   Review the default values specified on this page and click **Continue**.

5. Progress messages appear on the "First Run Command" page.

   When the adding of the service has completed, click **Continue**.

6. The Congratulations page appears.

   Click **Finish**.

7. The Cloudera Manager Home page appears.

   The `ZooKeeper` service now appears in the list of services.

   A health issue icon may appear next to the new ZooKeeper service temporarily, however this should go away momentarily and the status should change to Good Health.

## Adding the Hive Service to Your Cluster

In this task, you will add the Hive service to your cluster using Cloudera Manager.

You will configure the Hive Metastore to use the MySQL `metastore` database and to have a HiveServer2 instance.

Hive and Impala can both make use of a single, common Hive metastore. Recall that you created a few databases by running a SQL script prior to installing your CDH cluster. One of the databases you created is named metastore, which will be used by Hive and Impala as a common metastore for storing table definitions.

At the end of the task, you will run a simple Hive command to verify that the Hive service has been added.

8. From the Cloudera Manager Home page, select the **Add Service** option for Cluster 1.

   The Add Service wizard appears.

9. Select **Hive** and click **Continue**.

   The "Select the set of dependencies for your new service" page appears.

10. Select the row containing the `hdfs`, `yarn`, and `zookeeper` services (but *not* the Spark service), then click **Continue**.

    The Customize Role Assignments page appears.

11. Specify host assignments as follows:

    - Gateway: `elephant` only
    - Hive Metastore Server: `elephant` only
    - WebHCat Server: Do not select any hosts
    - HiveServer2: `elephant` only

    Verify that you have selected *only* `elephant` *and not any additional hosts* for the Gateway, Hive Metastore Server, and HiveServer2 roles.

    Click **Continue**.

12. The “Database Setup” page appears.

    Specify values for the database as follows:

    - Database Hostname: `lion`
    - Database Type: `MySQL`
    - Database Name: `metastore`
    - User Name: `hiveuser`
    - Password: `password`

    Click **Test Connection** and verify that connection to the MySQL database is successful (indicated by the “Continue” button turning blue and becoming active).

    Click **Continue**.

13. The Review Changes page appears.

    Review the default values specified on this page and click **Continue**.

14. Progress messages appear on the First Run Command page.

    When the adding of the service has completed, click **Continue**.

15. The Congratulations page appears.

    Click **Finish**.

16. The Home page appears.

    A status indicator shows you that the `Hive` service is in good health.

    The Hive service shows two configuration issues, related to Spark on YARN Service and Spark Executor Cores, however these can be safely ignored on the training cluster.

17. Install the Spark client configuration on `elephant`.

    Previously you ran the Spark shell on `monkey`. Here you will add the Spark Gateway role on `elephant` so that you can run the Spark shell from `elephant`.

    Navigate to the **Spark Instances** page for your cluster and click **Add Role Instances**.

Add the Gateway role to `elephant` and click **Continue**.

Click on the "Stale configuration: Client configuration redeployment needed" icon.

In the Stale Configurations page click **Deploy Client Configuration**.

Wait for the commands to complete, then click **Finish**.

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

On **`elephant`**:

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

If you see a warning about hbase-prefix-tree, you can safely ignore it.

In the Beeline shell, type the following command:

```
> SHOW TABLES;
```

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

**19.** Exit the Beeline shell.

```
> !quit
```

## Adding the Impala Service

In this task, you will add the Impala service to your cluster.

Cloudera Manager will automatically configure Impala to use the Hive Metastore service that you created earlier in this exercise.

**20.** From the Cloudera Manager Home page, select the **Add Service** option for Cluster 1.

The Add Service wizard appears.

21. Select **Impala** and click **Continue**.

22. Specify host assignments as follows:

    - Impala Catalog Server: `horse`
    - Impala StateStore: `horse`
    - Impala Daemon: `elephant`, `horse`, `monkey`, and `tiger`

    Click **Continue**.

23. The Review Changes page appears.

    Review the default values specified for the Impala Daemon Scratch Directories on this page and click **Continue**.

24. Progress messages appear on the First Run Command page.

    When the adding of the service has completed, click **Continue**.

25. The Congratulations page appears.

    Click **Finish**.

26. Restart the HDFS service.

    After adding Impala, on the Cloudera Manager home page you will notice that the HDFS service has stale configurations as indicated by the  icon that appears.

    Click on the "**Stale Configuration – Restart needed**" icon. The "Stale Configurations" page appears.

    Click **Restart Stale Services**.

    Check the box to "Re-deploy client configuration", then click **Restart Now**.

    When the action completes click **Finish**.

27. Confirm the Impala services started.

    Browse to the **Impala** page for your cluster and click on **Instances**.

You should see that the Impala Catalog Server, Impala Daemon, and Impala StateStore services have all started successfully.

## Running Hive Queries

In this task, you will define the `movierating` table in Hive and run a simple query against the table. Then you will change Hive's execution engine from MapReduce to Spark and run the same query.

**Note:** You already imported the `movierating` table into HDFS in the Importing Data with Sqoop exercise.

**28.** Review the `movierating` table data imported into HDFS during the Sqoop exercise.

On **`elephant`**:

```
$ hdfs dfs -cat movierating/part-m-00000 | head
```

**29.** Start the Beeline shell and connect to HiveServer2.

On **`elephant`**:

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

**30.** 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';
```

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

On **`elephant`**:

```
> SHOW TABLES;
```

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

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

On **`elephant`**:

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

Browse to Cloudera Manager's **YARN Applications** page to view the MapReduce job that was run when you executed the Hive query. In the **Results** tab, make note of the amount of time the query takes to execute when running in Hive.

## Running Hive on Spark

**33.** Set Spark as the execution engine to be used by Hive for this Beeline session and run the same query again.

Still in the Beeline shell on **`elephant`**:

```
> set hive.execution.engine;
```

The command above should show that the current execution engine is MapReduce (mr).

```
> set hive.execution.engine=spark;
```

The above command changes the execution engine for Hive to Spark.

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

Observe the results in the Beeline shell.

In Cloudera Manager, navigate to the **YARN Applications** page. Notice the **Hive on Spark** application. Assuming you have not yet exited the Beeline shell, it will still be running.

**34.** Terminate the Beeline shell.

On **elephant**:

```
> !quit
```

## Running Impala Queries

In this task, you will run Impala queries against the `movierating` table you defined earlier.

**35.** Start the Impala shell.

On **elephant**:

```
$ impala-shell
```

**36.** Connect to the Impala Catalog Server running on `horse`.

```
> CONNECT horse;
```

**37.** Since you defined a new table after starting the Impala server on `horse`, you must now refresh that server's copy of the Hive metadata.

```
> INVALIDATE METADATA;
```

**38.** 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 on MapReduce and Hive on Spark.

**39.** Exit the Impala shell.

```
> quit;
```

**40.** Explore Impala queries in Cloudera Manager

In Cloudera Manager, from the **Clusters** menu, choose **Impala Queries**.

Notice that both queries you ran from the impala-shell appear in the Results panel.

For the `SELECT` query that you ran, choose **Query Details** from the drop-down menu on the right. Browse through the query details noting the information about the query that is available to you.

**This is the end of the Exercise.**

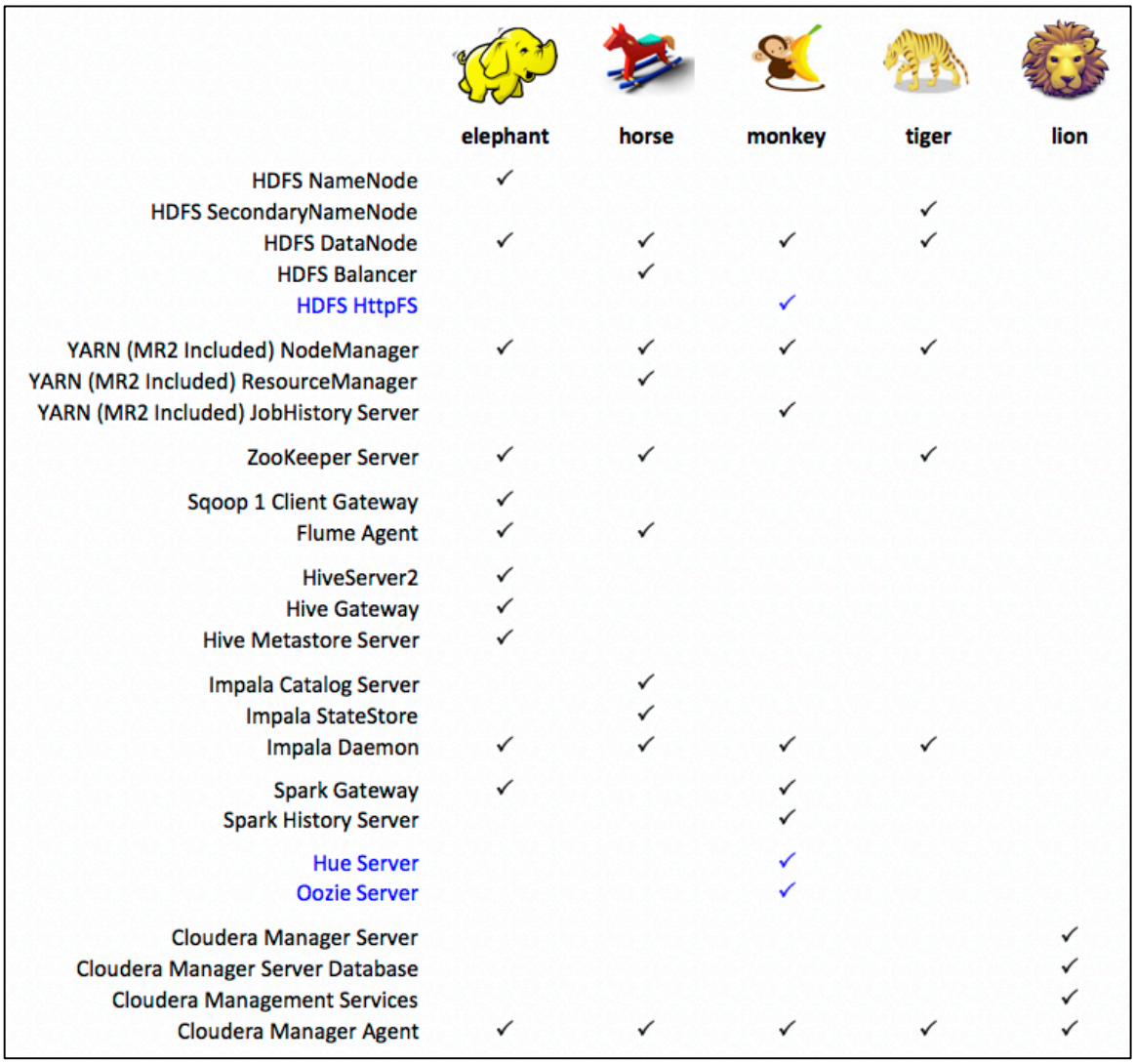
# Hands-On Exercise: Using Hue to Control Hadoop User Access

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 YARN applications

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 five hosts as follows (new daemons shown in blue):



| | elephant | horse | monkey | tiger | lion |
|---|---|---|---|---|---|
| HDFS NameNode | ✓ | | | | |
| HDFS SecondaryNameNode | | | | ✓ | |
| HDFS DataNode | ✓ | ✓ | ✓ | ✓ | |
| HDFS Balancer | | ✓ | | | |
| HDFS HttpFS | | | ✓ | | |
| YARN (MR2 Included) NodeManager | ✓ | ✓ | ✓ | ✓ | |
| YARN (MR2 Included) ResourceManager | | ✓ | | | |
| YARN (MR2 Included) JobHistory Server | | | ✓ | | |
| ZooKeeper Server | ✓ | ✓ | | ✓ | |
| Sqoop 1 Client Gateway | ✓ | | | | |
| Flume Agent | ✓ | ✓ | | | |
| HiveServer2 | ✓ | | | | |
| Hive Gateway | ✓ | | | | |
| Hive Metastore Server | ✓ | | | | |
| Impala Catalog Server | | ✓ | | | |
| Impala StateStore | | ✓ | | | |
| Impala Daemon | ✓ | ✓ | ✓ | ✓ | |
| Spark Gateway | ✓ | | ✓ | | |
| Spark History Server | | | ✓ | | |
| Hue Server | | | ✓ | | |
| Oozie Server | | | ✓ | | |
| Cloudera Manager Server | | | | | ✓ |
| Cloudera Manager Server Database | | | | | ✓ |
| Cloudera Management Services | | | | | ✓ |
| Cloudera Manager Agent | ✓ | ✓ | ✓ | ✓ | ✓ |

The Hue server will be deployed on `monkey`. The HttpFS and Oozie servers on `monkey` will support several Hue applications.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Adding an HttpFS Role Instance to the `hdfs` Service

In this task, you will use the Cloudera Manager wizard to add an HttpFS role instance to the `hdfs` service. The HttpFS role instance will reside on `monkey`. After adding the role instance, you will run a `curl` command from the command line to verify that HttpFS works correctly.

1. In Cloudera Manager, navigate to the **HDFS Instances** page.

2. Click **Add Role Instances**.

   The "Add Role Instances to HDFS – Customize Role Assignments" page appears.

3. For HttpFS, specify `monkey`.

   Click **Continue**.

4. The `hdfs` Role Instances page reappears. The `HttpFS (monkey)` role instance now appears in the list of role instances.

   Notice that the status for this role instance is Stopped.

5. Start the HttpFS role instance.

   In the **HDFS Instances** page, check the box next to **HttpFs** and from the **Actions for Selected (1)** menu choose **Start**.

   In the Start dialog window click **Start**. When the Command Details window shows that the command completed successfully click **Close**.

6. To verify HttpFS operation, run the HttpFS `LISTSTATUS` operation to examine the content in the `/user/training` directory in HDFS.

   On **`elephant`**:

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

   **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.

## Adding the Oozie Service

In this task, you will add the Oozie service to your cluster. With Cloudera Manager, Oozie is a prerequisite for adding the Hue service. We won't use this service in this exercise, but feel free to explore the Oozie service on your own if you like.

You will configure the Oozie instance to reside on `monkey`.

7. In Cloudera Manager, navigate to the Home page.

8. Select the **Add Service** option for Cluster 1.

   The Add Service Wizard appears.

9. Select **Oozie** and click **Continue**.

   The "Select the set of dependencies for your new service" page appears.

10. Select the row containing the `hdfs`, `yarn`, and `zookeeper` services, then click **Continue**.

    The Customize Role Assignments page appears.

11. Specify host assignments as follows:

    - Oozie Server: `monkey`

    Click **Continue**.

12. The Database Setup page appears. Specify the settings as shown here:
    - Database Host Name: `lion`
    - Database Type: `MySQL`
    - Database Name: `oozie`
    - Username: `oozieuser`
    - Password: `password`

    Click "**Test Connection**" and verify the connection is successful. Click **Continue**.

13. The Review Changes page appears.

    Review the default values specified on this page and click **Continue**.

14. Progress messages appear on the First Run Command page.

    When the adding of the service has completed, click **Continue**.

15. The Congratulations page appears.

    Click **Finish**.

16. The Home page appears.

    A status indicator shows you that the Oozie service is in good health.

## Adding the Hue Service

In this task, you will add a Hue service to your cluster, configuring the Hue instance to run on `monkey`.

17. From the Cloudera Manager Home page, select the **Add Service** option for Cluster 1.

    The Add Service wizard appears.

18. Select **Hue** and click **Continue**.

The “Select the set of dependencies for your new Hue” page appears.

19. Select the row containing the `hdfs`, `hive`, `impala`, `oozie`, `yarn`, and `zookeeper` services, then click **Continue**.

    The Customize Role Assignments page appears.

20. Specify host assignments as follows:

    - Hue Server: `monkey`

    Click **Continue**.

21. The Database Setup page appears. Fill in the details as shown here:
    - Database Host Name: `lion`
    - Database Type: `MySQL`
    - Database Name: `hue`
    - Username: `hueuser`
    - Password: `password`

22. Click **Test Connection** and verify the connection is successful. Click **Continue**.

23. Progress messages appear on the “First Run Command” page.

    When the adding of the service has completed, click **Continue**.

24. The Congratulations page appears.

    Click **Finish**.

25. The Home page appears.

    A status indicator shows you that the Hue service is in good health.

**26.** Submit a Hadoop WordCount job so that there will be a MapReduce job entry that you can browse in Hue after you start the Hue UI.

On **elephant**:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-
mapreduce/hadoop-mapreduce-\
examples.jar wordcount /tmp/shakespeare.txt test_output
```

## 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, Impala Query UI, Pig Editor, File Browser, Metastore Manager, Job Browser, Hue Shell, User Admin, and Help.

### Logging into Hue

**27.** Maximize the browser window to give Hue enough space to display as many options as possible on its top menu.

**28.** View the Hue UI.

Access the **Hue Web UI** from the Cloudera Manager **Hue** page for your cluster or just browse to `http://monkey:8888`

**29.** Login to Hue.

Note: that as the message box in the browser indicates, as the first person to login to this Hue service, you are actually defining the Hue superuser credentials in this step.

Type in `admin` as the user, with the password `training`, then click **Create Account**.

**30.** The Quick Start Wizard displays.

Click the **Home** icon.

At the "Did you know?" dialog, click **Got it!**

The "My documents" page appears.

## Access Hive Using Hue

**31.** If you completed the "Querying HDFS with Hive and 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 the **Execute** icon (play button). 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;
```

Give it some time to complete. The UI will first show the Log tab contents, then it should eventually show the Results tab. The query should run successfully.

## Access Impala Using Hue

**32.** If you completed the "Querying HDFS with Hive and 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 the **Execute** icon. The result of the query should show 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.

## Access the Metastore Using Hue

**33.** Click on the **Metastore Manager** link.

The Metastore Manager appears showing the `movierating` table.

In the 'TABLES' area, in the Table Name column, click on the **movierating** table link.

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

Notice the icons in the top right of the screen that provide options such as import data, browse data, and view file location.

## Access Pig Using Hue

**34.** Start the Pig UI by selecting **Query Editors** > **Pig**.

The Pig Editor appears.

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

## Access HDFS Using Hue

**35.** In Hue, click on the **HDFS Browser** (document icon) towards the top right of the Hue UI.

This opens the File Browser application.

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

Note the **Upload** menu as well. You could upload files to HDFS through Hue.

There are also **Actions** available giving the Hue user options to Rename, Move, Download or change permissions on HDFS files (assuming the user has the permissions to do so).

36. In the **File Browser**, navigate to the `/user/admin` directory.

    On the first Hue login, Hue created a superuser—in this case, the `admin` user — and an HDFS path for that user—in this case, the `/user/admin` path.

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

38. 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.

## Browse YARN Applications Using Hue

39. In Hue, select the **Job Browser** (list icon) option.

    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 MapReduce word count job you ran in the previous step appears with the status "SUCCEEDED."

    Browse the completed "word count" job details by clicking on the link in the ID column and then looking through the details in the **Attempts**, **Tasks**, **Metadata**, and **Counters** tabs.

If you are interested, look in Cloudera Manager's **YARN Applications** page for your cluster, locate the entry for the same `word count` job, and follow the **Application Details** link which takes you to the HistoryServer Web UI. Compare the details you find there with the information available in the Hue Job Browser.

## Browse Users, Documentation, Settings, and Logs of Hue

**40.** Access the Hue User Admin tool by selecting the **Administration** menu (cog and wheels icon) and then choosing **Manage Users**.

The User Admin screen appears where you can define Hue users and groups and set permissions.

Notice the automatically created entry for the `admin` user.

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

**41.** Click the **Documentation** (question mark) icon:

Hue UI user documentation appears.

**42.** Click the **About Hue** icon (to the left of the Home icon).

The Quick Start Wizard's Check Configuration tab shows "All OK. Configuration check passed."

Click into the **Configuration** tab (to the right of the About Hue link) to examine Hue's configuration.

Click the **Server Logs** tab to examine Hue's logs.

## Setting up the Hue Environment for Business Analysts

Consider a scenario where 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.

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

**44.** Activate the Hue User Management tool by selecting **Administration** > **Manage Users**.

**45.** Select **Groups**.

**46.** Click **Add group** and name the new group `analysts`.

Configure the permissions by selecting the ones listed below:

- `about.access`
- `beeswax.access`
- `filebrowser.access`
- `help.access`
- `impala.access`
- `jobbrowser.access`
- `metastore.write`
- `metastore.access`
- `pig.access`

Click **Add group**.

**47.** Select **Users**.

**48.** Click **Add user.**

In the **Step 1: Credentials** tab, create a user named `fred` with the password `training`. Click **Next**.

In the **Step 2: Profile and Groups** tab, make `fred` a member of the `analysts` group. However, make sure that `fred` is *not* a member of the `default` group.

Click **Add User**.

**49.** Sign out of the Hue UI (using the arrow icon in the top right of the screen).

**50.** 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. For example, the Administration menu does not allow fred to manage users. Fred also has no access to the Workflows, Search, and Security menus that are available to the admin user.

**This is the end of the Exercise.**

# 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 modifying the `hdfs` service's configuration to enable HDFS high availability.

You will then shut down services that you will no longer use in this exercise or other subsequent exercises.

Next, you will enable automatic failover for the NameNode. Automatic failover uses the ZooKeeper service that you added to your cluster in an earlier exercise. ZooKeeper is a prerequisite for HDFS HA automatic failover.

Once you have enabled HDFS high availability with automatic failover, you will intentionally bring one of the servers down as a test. HDFS services should still be available, but it will be served by a NameNode running on a different host.

At the end of this exercise, you should have daemons deployed and running on your five hosts as follows (new daemons shown in blue):

| |  elephant |  horse |  monkey |  tiger |  lion |
|---|---|---|---|---|---|
| HDFS NameNode | ✓ | | | | |
| ~~HDFS SecondaryNameNode~~ | | | | removed | |
| HDFS Standby NameNode | | | | ✓ | |
| JournalNode | ✓ | ✓ | | ✓ | |
| Failover Controller | ✓ | | | ✓ | |
| HDFS DataNode | ✓ | ✓ | ✓ | ✓ | |
| HDFS Balancer | | ✓ | | | |
| HDFS HttpFS | | | ✓ | | |
| YARN (MR2 Included) NodeManager | ✓ | ✓ | ✓ | ✓ | |
| YARN (MR2 Included) ResourceManager | | ✓ | | | |
| YARN (MR2 Included) JobHistory Server | | | ✓ | | |
| ZooKeeper Server | ✓ | ✓ | | ✓ | |
| Sqoop 1 Client Gateway | ✓ | | | | |
| Flume Agent | stopped | stopped | | | |
| HiveServer2 | stopped | | | | |
| Hive Gateway | stopped | | | | |
| Hive Metastore Server | stopped | | | | |
| Impala Catalog Server | | stopped | | | |
| Impala StateStore | | stopped | | | |
| Impala Daemon | stopped | stopped | stopped | stopped | |
| Spark Gateway | stopped | | stopped | | |
| Spark History Server | | | stopped | | |
| Hue Server | | | stopped | | |
| Oozie Server | | | stopped | | |
| Cloudera Manager Server | | | | | ✓ |
| Cloudera Manager Server Database | | | | | ✓ |
| Cloudera Management Services | | | | | ✓ |
| Cloudera Manager Agent | ✓ | ✓ | ✓ | ✓ | ✓ |

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Bringing Down Unneeded Services

Since you will no longer use Hue, Oozie, Impala, or Hive for the remaining exercises, you can stop their services to improve your cluster's performance.

1. In Cloudera Manager, navigate to the **Home** page.

2. Stop the **Hue** service as follows:

   In the row for the **Hue** service, chose **Stop** from the drop-down menu.

   Click **Stop** in the confirmation window.

   Click **Close** after messages in the Stop Command page indicate that the Hue service has stopped.

   The Home page reappears. The status of the `Hue` service should have changed to Stopped.

3. Using steps like the ones you followed to stop the Hue service, stop the **Oozie** service.

4. Stop the **Impala** service.

5. Stop the **Hive** service.

6. Stop the **Flume** service if it is running.

Verify that the only services that are still up and running on your cluster are the **Hosts**, **HDFS**, **Spark**, **YARN (MR2 Included)**, **ZooKeeper**, and **Cloudera Manager Service** services. All these services should have good health.

## Enabling HDFS High Availability

In this task, you will configure your Hadoop configuration to use HDFS high availability.

7. In Cloudera Manager, browse to the **HDFS Instances** page.

   Observe that the HDFS service comprises the following role instances:

   - Balancer on `horse`
   - DataNodes on `horse`, `tiger`, `elephant`, and `monkey`
   - An HttpFS server on `monkey`
   - The active NameNode on `elephant`
   - The SecondaryNameNode on `tiger`

   The list of role instances will change after you enable high availability.

8. From the **Actions** menu, choose **Enable High Availability**.

9. The "Getting Started" page appears.

   Change the Nameservice Name to `mycluster`.

   Click **Continue**.

10. The "Assign Roles" page appears.

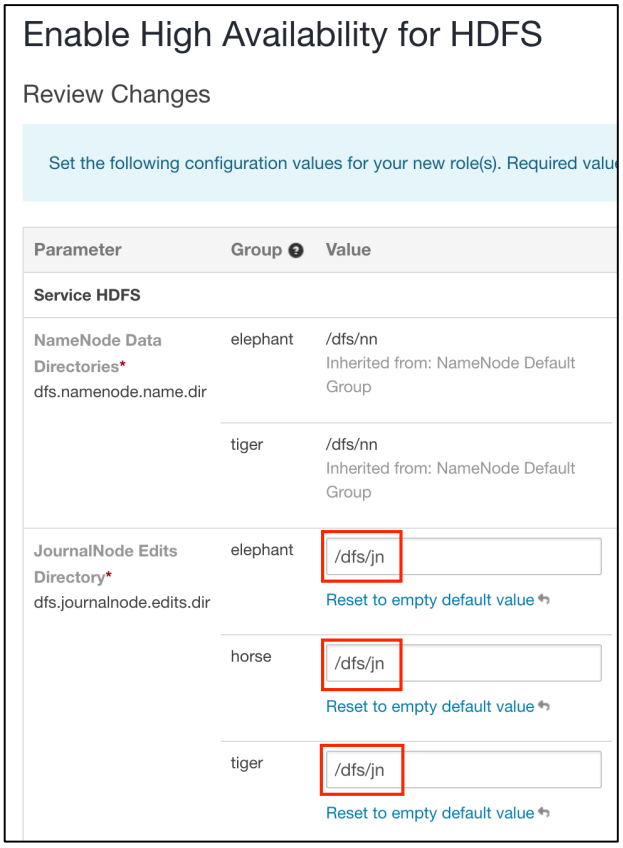
    Specify the following:

    - NameNode Hosts
      - `elephant(Current)`
      - `tiger`
    - JournalNode Hosts
      - `elephant, horse, tiger`

    Click **Continue**.

**11.** The “Review Changes” page appears.

Specify the value `/dfs/jn` **in all three JournalNode Edits Directory fields**.




Click **Continue**.

The “Enable High Availability Command” page appears. The messages shown in the screen shot below appear as Cloudera Manager enables HDFS high availability.

Note: Formatting the name directories of the current NameNode will fail. As described in the Cloudera Manager interface, this is expected.

- Check that name directories for the new Standby NameNode either do not exist or are writable and empty. Can optionally clear directories.
  `Process host-validate-writable-empty-dirs (id=107) on host tiger (id=5) exited with 0 and expected 0`
- Check that edits directories for the nameservice either do not exist or are writable and empty. Can optionally clear directories.
  `Successfully completed 3 steps.`
- Stop hdfs and its dependent services
  `All services successfully stopped.`
- Creating roles to enable High Availability.
  `Successfully added new JournalNode to HDFS on tiger.`
- Deleting the SecondaryNameNode role. The checkpoint directories of the SecondaryNameNode will not be deleted.
  `Successfully deleted role.`
- Configuring NameNodes and the HDFS service to enable High Availability.
  `Successfully updated config value.`
- Initializing High Availability state in ZooKeeper.
  `Successfully initialized High Availability state in ZooKeeper from Failover Controller (elephant)`
- Starting the JournalNodes
  `Successfully completed 3 steps.`
- Formatting the name directories of the current NameNode. If the name directories are not empty, this is expected to fail.
  `Failed to format NameNode.`

  Role Log

  Full log file
- Initializing shared edits directory of NameNodes.
  `Successfully initialized shared edits directory of the NameNode`
- Starting the NameNode that will be transitioned to active mode NameNode (elephant).
  `Successfully started process.`
- Waiting for the Active NameNode to start up.
  `NameNode started responding to RPCs successfully.`
- Bootstrapping Standby NameNode by initializing its name directories.
  `Successfully bootstrapped Standby NameNode NameNode (tiger).`
- Starting Standby NameNode
  `Successfully started process.`
- Starting the Failover Controller on the host of the Active NameNode.
  `Successfully started process.`
- Starting the Failover Controller on the host of the Standby NameNode.
  `Successfully started process.`
- Waiting for the Standby NameNode to start up.
  `NameNode started responding to RPCs successfully.`
- Creating HDFS /tmp directory if not already created.
  `Command (428) has completed successfully`
- Start hdfs and its dependent services
  `All services successfully started.`
- Deploying configurations for clients of services in this cluster.
  `Successfully deployed all client configurations.`

When the process of enabling high availability has finished, click **Continue**.

An informational message appears informing you of post-setup steps regarding the Hive Metastore. You will not perform the post-setup steps because you will not be using Hive for any remaining exercises.

Click **Finish**.

12. The HDFS service's Status page appears. Click into the **Instances** page.

    Observe that the HDFS service now comprises the following role instances:
    - Balancer on `horse`
    - DataNodes on `tiger`, `elephant`, `monkey`, and `horse`
    - Failover Controllers on `tiger` and `elephant`
    - An HttpFS server on `monkey`
    - JournalNodes on `elephant`, `tiger`, and `horse`
    - The active NameNode on `elephant`
    - The standby NameNode on `tiger`
    - No SecondaryNameNode

## Verifying Automatic NameNode Failover

In this task, you will restart the active NameNode, bringing it down and then up. The standby NameNode will transition to the active state when the active NameNode goes down, and the formerly active NameNode will transition to the standby state.

Then you will restart the new active NameNode to restore the original states of the two NameNodes.

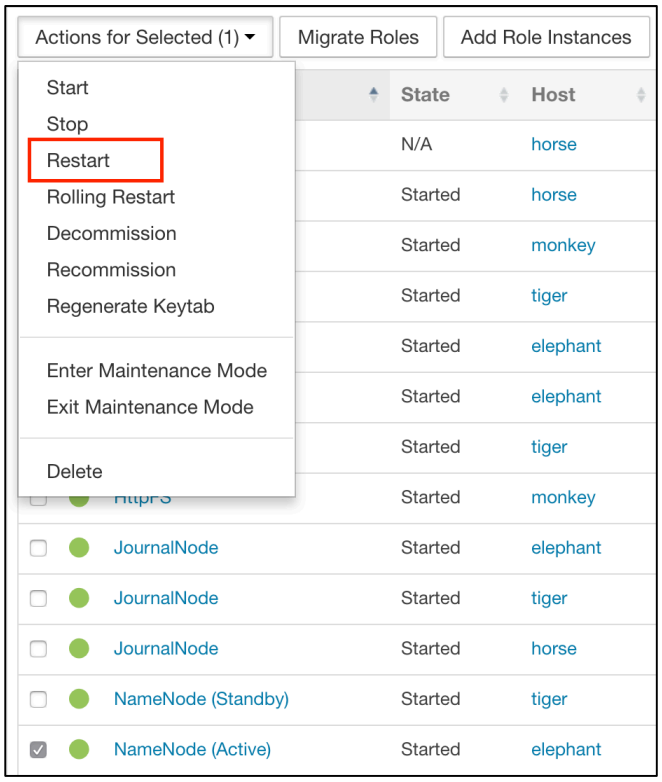
13. From the **HDFS Instances** page, click on **Federation and High Availability** and observe that the state of one of the NameNodes is active and the state of the other NameNode is standby.

14. Click the **Instances** link again to return to the previous page view.

15. Select the check box to the left of the entry for the active NameNode.

16. Choose **Actions for Selected (1)** > **Restart**.

    Click **Restart** to confirm that you want to restart the instance




17. Wait for the restart operation to complete. When it has successfully completed click **Close**.

    Verify that the states of the NameNodes have changed—the NameNode that was originally active (`elephant`) is now the standby, and the NameNode that was originally the standby (`tiger`) is now active. If the Cloudera Manager UI does not immediately reflect this change, give it a few seconds and it will.

18. Go to **Diagnostics** > **Events** and notice the many recent event entries related to the restarting of the NameNode.

19. Back in the **HDFS Instances** tab, restart the NameNode that is currently the active NameNode.

    After the restart has completed, verify that the states of the NameNodes have again changed.

**This is the end of the Exercise.**

# 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.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

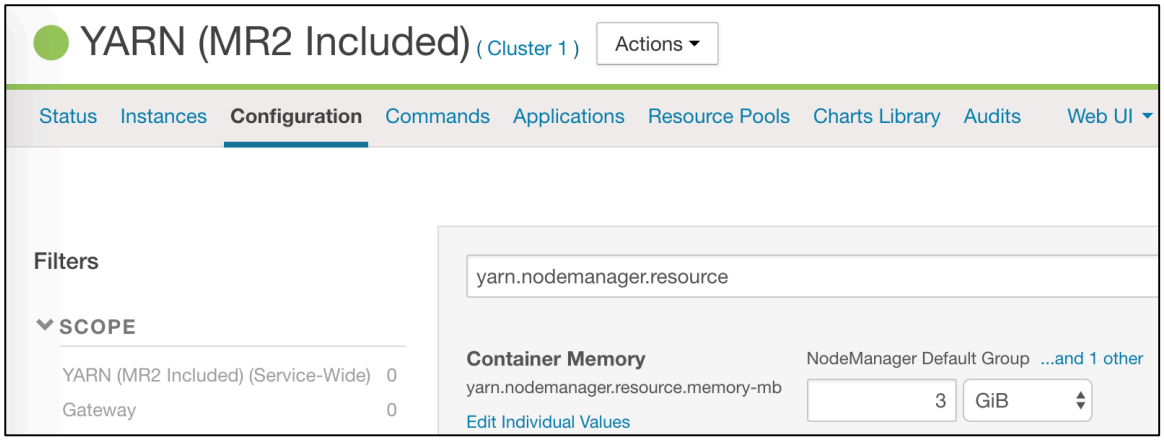
1. Adjust YARN memory setting and restart the cluster.

   To demonstrate the Fair Scheduler, you will need to increase the amount of memory that YARN containers are permitted to use on your cluster.

   In Cloudera Manager, go to the **YARN Configuration** page.

   In the search box, search for **`yarn.nodemanager.resource.memory-mb`**

   Change the value of this parameter to `3GB` in both Role Groups where it appears (by default the UI shows one textbox to edit both Role Groups).




   Click **Save Changes**.

Click on the "**Stale Configuration – Restart needed**" icon and follow the steps to restart the cluster. In the Review Changes screen, *uncheck* "**Rolling Restart**" and *check* "**Re-deploy client configuration**."

When the restart has completed, click **Finish**.

2. Analyze the script you will run in this exercise.

   To make it easier to start and stop MapReduce jobs during this exercise, a script has been provided. View the script to gain an understanding of what it does.

   On **elephant**:

```
$ cd ~/training_materials/admin/scripts
$ cat pools.sh
```

   The script will start or stop a job in the pool you specify. It takes two parameters. The first is the pool name and the second is the action to take (start or stop). Each job it will run will be relatively long running and consist of 10 mappers and 10 reducers.

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

3. Start three Hadoop jobs, each in a different pool.

   On **elephant**:

```
$ ./pools.sh pool1 start
$ ./pools.sh pool2 start
$ ./pools.sh pool3 start
```

   It is recommended that you attempt to go through this exercise by only starting jobs when prompted in the instructions. However, depending on how quickly you complete the steps, a job may have completed earlier than the instructions anticipated. Therefore, please note that at any time during this exercise you can start additional jobs in any pool using any of the three commands you ran in this step.

4. Verify the jobs started.

   In Cloudera Manager, browse to **Clusters** > **YARN Applications**.

   You should notice that the three jobs have started. If the jobs do not yet display in the page, wait a moment and then refresh the page.

   Note: Cloudera Manager does refresh pages automatically, however in this exercise you may find it useful to refresh the pages manually to more quickly observe the latest status of pools and jobs running in the pools.

   Leave this browser tab open.

5. Observe the status of the pools.

   Open another browser tab, and in Cloudera Manager, browse to the **YARN** service's **Resource Pools** page.

   Analyze the details in the "Resource Pools Usage" table.

   If `pool1`, `pool2`, and `pool3` do not yet display, refresh the browser tab.

   The table displays the pools in the cluster. The pools you submitted jobs to should have pending containers and allocated containers.

   The table also shows the amount of memory and vcores that have been allocated to each pool.

6. Analyze the Per Pool charts.

   On the same page in Cloudera Manager, notice the "Per Pool Shares" charts. There is another chart for "Per Pool Allocations (Allocated Memory)" and another for "Per Pool Allocations (Allocated VCores)."

   If nothing is displaying in these charts yet, wait a moment and they will. Optionally refresh the browser page.

   Leave this browser tab open.

7. Start another job in another pool.

   In the same shell session on `elephant`:

```
$ ./pools.sh pool4 start
```

8. Back in Cloudera Manager, observe the resource allocation effect of starting a new job in a new pool.

   Occasionally refresh the **YARN Resource Pools** page.

   Some pools may be initially over their fair share because the first jobs to run will take all available cluster resources.

   However, 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.

   Tip: Mouse over any one of the "Per Pool" charts and then click the double-arrow icon to expand the chart size.

9. Conduct further experiments.

   Stop the job running in pool1.

   ```
   $ ./pools.sh pool1 stop
   ```

   Wait a minute or two, then observe the results in the charts on the Dynamic Resource Pools page.

   Start a second job in pool3.

   ```
   $ ./pools.sh pool3 start
   ```

   Again, observe the results.

10. Configure a Dynamic Resource Pool for `pool2`.

    Browse to the **Clusters** > **Dynamic Resource Pool Configuration** page.

    From the **Resource Pools** tab, click on **Create Resource Pool**.

    Set the Resource Pool Name to `pool2`.

    In the **Configuration Sets** tab, configure the following settings:

- Weight: **2**
- Virtual Cores Min: **1**
- Virtual Cores Max: **2**
- Memory Min: **2400**
- Memory Max: **5000**

In the **Scheduling Policy** tab, keep **DRF** as the scheduling policy.

Click **Create** to create the new resource pool.

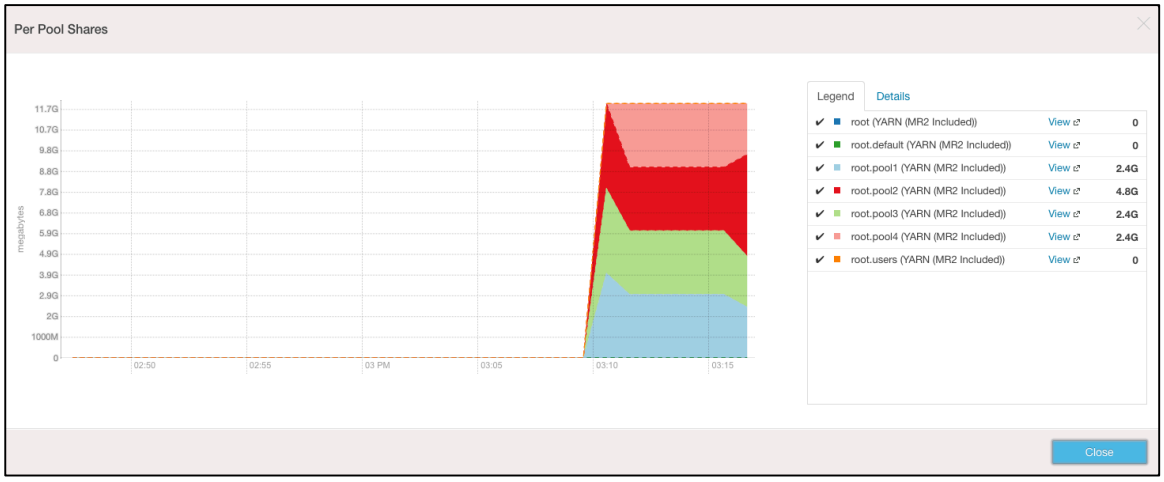
**11.** Observe the effect of the new Dynamic Resource Pool on the resource allocations within the cluster.

In the **YARN Applications** page, check how many jobs are still running and in which pools they are running.

Use the `pools.sh` script to start or stop jobs so that there is one job running in each of the four pools.

Return to the **YARN Resource Pools** page to observe the affect of the pool settings you defined.

As you continue to observe the "Per Pool Shares – Fair Share Memory" chart, you should soon see that pool2 is given a greater share of resources.




**12.** Clean up.

When you are done observing the behavior of the Fair Scheduler, stop all running jobs either by using the `pools.sh` script or kill the applications from the **YARN Applications** page in Cloudera Manager.

**This is the end of the Exercise.**

# Hands-On Exercise: Breaking the Cluster

In this exercise, you will see what happens during failures of portions of the Hadoop cluster.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

1. Verify the existence of a large file in HDFS.

   In a previous exercise, you placed the weblog files in HDFS. Verify the files are there.

   On **elephant**:

   ```
   $ hdfs dfs -ls weblog
   ```

   *Only if you do not see the access_log file* in HDFS, place it there now.

   On **elephant**:

   ```
   $ cd ~/training_materials/admin/data
   $ hdfs dfs -mkdir weblog
   $ gunzip -c access_log.gz \
   | hdfs dfs -put - weblog/access_log
   ```

2. Locate a block that has been replicated on `elephant` as follows:

   In the **NameNode Web UI** for your active NameNode, navigate to the `/user/training/weblog/access_log` 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.

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.

   Using Cloudera Manager, from the **HDFS Instances** page, stop the DataNode running on `elephant`.

4. Visit the **NameNode Web UI** again and click on **Datanodes**. Refresh the browser several times and notice that the "Last contact" value for the `elephant` DataNode stays the same, while the "Last contact" values for the other DataNodes continue to update with more recent timestamps.

5. Run the HDFS file system consistency check to see that the NameNode currently thinks there are no problems.

   On **`elephant`**:

```
$ sudo -u hdfs hdfs fsck /
```

6. Wait for at least ten minutes to pass before starting the next exercise.

   (optional) in any terminal:

```
$ sleep 600 && echo "10 minutes have passed."
```

**This is the end of the Exercise.**

# Hands-On Exercise: Verifying the Cluster's Self-Healing Features

In this exercise, you will see what has happened to the data on the dead DataNode.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

1. In the **NameNode Web UI**, click on **Datanodes** 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. In Cloudera Manager, navigate to the **HDFS Charts Library** page and click on the **Blocks and Files** chart filter.

   Scroll down and view the charts which show evidence of what occurred. For example, view the "Under-replicated Blocks" chart, the "Pending Replication Blocks" chart, and the "Scheduled Replication Blocks" chart.

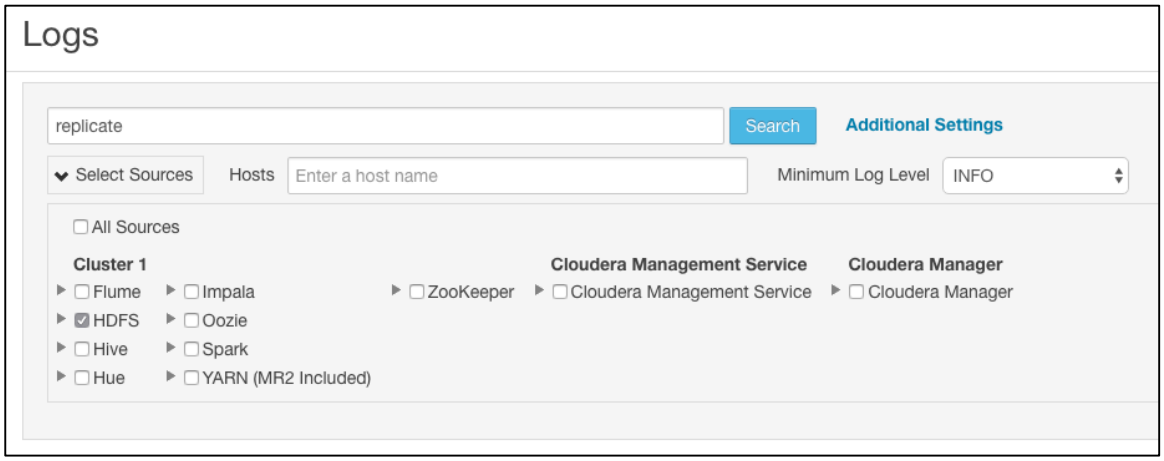
   Note the spike in activity that occurred after the DataNode went down.

4. View the audit and log trails in Cloudera Manager.

   In Cloudera Manager, click on **Audits**.

   Note the timestamp for when the HDFS service was stopped.

   Choose **Diagnostics** > **Logs**, select sources **HDFS** only, set the Minimum Log Level to **INFO**, and enter the search term "`replicate`".




Click **Search**.

Scroll down to the log entries that occurred just over 10 minutes after the DataNode was stopped. Notice the log messages related blocks being replicated. You can also find these log entries by searching for "blockstatechange".

5. Run the `hdfs fsck` command again to observe that the filesystem is still healthy.

   On **elephant**:

```
$ sudo -u hdfs hdfs fsck /
```

6. Run the `hdfs dfsadmin -report` command to see that one dead DataNode is now reported.

   On **elephant**:

```
$ sudo -u hdfs hdfs dfsadmin -report
```

7. Use Cloudera Manager to restart the DataNode on `elephant`, bringing your cluster back to full strength.

8. Run the `hdfs fsck` command again to observe the temporary over replication of blocks.

   On **elephant**:

```
$ sudo -u hdfs hdfs fsck /
```

Note that the over replication situation will resolve itself (if it has not already) now that the previously unavailable DataNode is once again running.

If the command above did not show any over replicated blocks, go to **Diagnostics** > **Logs** in Cloudera Manager and search the **HDFS** source for "`replica`". You should find evidence of the temporary over-replication in the log entries corresponding to the time range just after the DataNode was started again.

**This is the end of the Exercise.**

# Hands-On Exercise: Taking HDFS Snapshots

In this exercise, you will enable HDFS snapshots on a directory and then practice restoring data from a snapshot.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

1. Enable snapshots on a directory in HDFS.

   In Cloudera Manager, go to the **HDFS** page for your cluster and click **File Browser**.

   Browse to `/user/training`, then click **Enable Snapshots**.

   In the "Enable Snapshots" window, keep the Snapshottable Path set to `/user/training` and click **Enable Snapshots**.

   The command completes. Notice in the message displayed in the Details panel showing a command that could be run in a terminal window to accomplish the same action using the `hdfs dfsadmin` tool.

   Click **Close**. Notice that there is now a "Take Snapshot" button.

2. Take a snapshot.

   Still in the Cloudera Manager **File Browser** at `/user/training`, Click **Take Snapshot**. Give it the name `snap1` and click **OK**.

   After the snapshot completes click **Close**.

   The snapshot section should now show your "snap1" listing.

3. Delete data from `/user/training` then restore data from the snapshot.

   Now let's see what happens if we delete some data.

On **elephant**:

```
$ hdfs dfs -rm -r weblog
$ hdfs dfs -ls /user/training
```

The second command should show that the weblog directory is now gone.

However, your weblog data is still available, which you can see by running the commands here:

```
$ hdfs dfs -ls /user/training/.snapshot/snap1
$ hdfs dfs -tail .snapshot/snap1/weblog/access_log
```

Restore a copy of the weblog directory to the original location and then verify it is back in place.

```
$ hdfs dfs -cp .snapshot/snap1/weblog weblog
$ hdfs dfs -ls /user/training
```

**This is the end of the Exercise.**

# Hands-On Exercise: Configuring Email Alerts

In this exercise, you will configure Cloudera Manager to use an email server to send alerts.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

1. Configure Cloudera Manager to send email alerts using the email server on `lion`.

   In Cloudera Manager, choose **Clusters** > **Cloudera Management Service**.

   Click on **Configuration** and then choose the **Alert Publisher** filter.

   Confirm the **Alerts: Enable Email Alerts** property is *checked*.

   Configure the following:
   - Alerts: Mail Server Username: **training**
   - Alerts: Mail Server Password: **training**
   - Alerts: Mail Message Recipients: **training@localhost**
   - Alerts: Mail Message Format: **text**

   Save the changes.

2. **Restart** the Cloudera Management Service.

3. Send a test alert from Cloudera Manager.

   In Cloudera Manager, go to **Administration** > **Alerts**. You should see that the recipient(s) of alerts is now set to `training@localhost`.

   Click on the **Send Test Alert** link at the top of the page.

4. Confirm emails are being received from Cloudera Manager.

The postfix email server is running on `lion`. Here you can use the mail command line client to access the training user's inbox.

On **`lion`**:

```
$ mail
```

The "Test Alert" email should show as new (N).

At the `&` prompt, type in the number that appears to the right of the `N` and hit the `Enter` key so you can read the email.

After you are done reading the email, type `q` and again hit the `Enter` key to exit the mail client.

**This is the end of the Exercise.**

# 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.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

## Recreating the Problem

1. Confirm files in HDFS.

   On **elephant**:

   ```
   $ hdfs dfs -ls /tmp/shakespeare.txt
   ```

   The command above should show that shakespeare.txt is in HDFS.

   *Only if shakespeare.txt was not found*, run these commands to place the file in HDFS.

   On **elephant**:

```
$ cd ~/training_materials/admin/data
$ gunzip shakespeare.txt.gz
$ hdfs dfs -put shakespeare.txt /tmp
```

On **elephant**:

```
$ hdfs dfs -ls weblog/access_log
```

The command above should confirm that the access_log file exists.

*Only if* `access_log` *was not found,* run these commands to place the file in HDFS.

On **elephant**:

```
$ cd ~/training_materials/admin/data
$ hdfs dfs -mkdir weblog
$ gunzip -c access_log.gz \
| hdfs dfs -put - weblog/access_log
```

2. Run the Heap of Trouble program.

   On **elephant**:

```
$ cd ~/training_materials/admin/java
$ hadoop jar EvilJobs.jar HeapOfTrouble \
/tmp/shakespeare.txt heapOfTrouble
```

## Attacking the Problem

The primary goal of this troubleshooting exercise 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 can.

**Do not turn to the next page unless you are ready for some hints.**

## 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 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.**

# 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 must 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 act 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 this case you might encounter terminal freezes as described above.

To configure VMware Fusion to use a fixed IP address, perform the following steps:

**3.** Start VMware Fusion.

4. Create an entry in the Virtual Machines list for the `Get2EC2` VM. To create the entry, drag the `Cloudera-Training-Get2EC2-VM-1.2.vmx` file to the Virtual Machines list.

   You should see the `Cloudera-Training-Get2EC2-VM-1.2` entry in the Virtual Machines list. We will refer to the `Cloudera-Training-Get2EC2-VM-1.2` VM as the `Get2EC2` VM.

5. Make sure the `Get2EC2` VM is powered down.

6. 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.

7. Click the **Settings** icon in the VMware Fusion Toolbar (or select **Virtual Machines** > **Settings**).

8. Click **Network Adapter**.

9. Click the arrow next to **Advanced options**.

   The MAC Address field appears.

10. If the MAC Address field is empty, click **Generate** to generate a MAC address for the `Get2EC2` VM.

11. 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.

12. Open the following file on your Mac using superuser (`sudo`) privileges:

    ```
    /Library/Preferences/VMware\ Fusion/vmnet8/dhcpd.conf
    ```

    Look for the `range` statement. It should have a range of IP addresses. For example:

```
range 172.16.73.128 172.16.73.254;
```

13. 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`.

14. 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.

15. Save and close the `dhcpd.conf` file.

16. Run the following commands from a terminal window on your Mac:

```
$ sudo /Applications/VMware\ Fusion.app/Contents/\
Library/vmnet-cli --stop
$ sudo /Applications/VMware\ Fusion.app/Contents/\
Library/vmnet-cli --start
```

17. Start the `Get2EC2` VM.

18. After the VM has come up, run the following command in a Linux terminal window:

```
$ ip addr
```

Verify that the IP address that appears is the IP address that you specified in the `dhcpd.conf` file.

**This is the end of this Appendix.**