

# Labs: Intro to HDFS/YARN & Apache Spark on CDH 5.2



Lab created on: Dec, 2014

(please send edits and corrections to): sameerf@databricks.com

Estimated lab completion time: **2.5 hours** (spread throughout the day)

License: 

## Objective:

This lab will introduce you to using 3 Hadoop ecosystem components in Cloudera's distribution: HDFS, Spark 1.1.0 and YARN. The lab will first walk you through the Cloudera Manager installation on a VM in AWS, followed by a CDH 5.2 binaries deployment on the same node. Then the lab will introduce students to Hadoop in a DevOps manner: experimenting with the distributed file system, looking at the XML config files, running a batch analytics workload with Spark from disk and from memory, writing some simple scala Spark code, running SQL commands with Spark SQL, breaking things and troubleshooting issues, etc.

**The following high level steps are in the initial part of this lab:**

- Connect via SSH to your Amazon instance
- Install Cloudera Manager and CDH 5.2
- Create a new folder in HDFS and add data files to it
- Start the scala based Spark shell
- Import the fresh data into Spark a RDD
- Persist an RDD to memory
- Write a transformed RDD back into HDFS
- Use the Spark Python shell and inject Spark SQL commands into it

Each student in class will be provided one m3.xlarge EC2 instance running in the us-west-2c (Oregon) data center.

The m3.xlarge instance type comes with 4 vCPUs, 15 GB of RAM and two 40 GB SSDs.

The instructor should have given you the following:

- Public DNS hostname of your specific instance
- .pem and .ppk key pairs to authenticate you to your instance

## Resources to learn more about Spark

### **++ Apache Documentation ++**

Spark homepage @ Apache:

<https://spark.apache.org>

Here are the official Apache docs for Spark 1.1.1:

<https://spark.apache.org/docs/latest/index.html>

Link to user + dev mailing lists:

<https://spark.apache.org/community.html#mailing-lists>

### **++ Developer/API Documentation ++**

Spark Core Scala docs:

<https://spark.apache.org/docs/latest/api/scala/index.html>

Spark Core Java docs:

<https://spark.apache.org/docs/latest/api/java/index.html>

Spark Core Python docs:

<https://spark.apache.org/docs/latest/api/python/index.html>

### **++ Cloudera Documentation ++**

Cloudera's docs for CDH 5.2 includes guides for a quick start, installation, security and HA:

<http://www.cloudera.com/content/cloudera/en/documentation/cdh5/latest/CDH5-Release-Notes/CDH5-Release-Notes.html>

Here is the exact version of all the Apache projects (16+) included in CDH 5.2.1:

[http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh\\_vd\\_cdh\\_package\\_tarball.html](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_vd_cdh_package_tarball.html)

The specific section of Cloudera's docs that refers to Spark Installation and configuration:

[http://www.cloudera.com/content/cloudera/en/documentation/cdh5/latest/CDH5-Installation-Guide/cdh5ig\\_spark\\_installation.html](http://www.cloudera.com/content/cloudera/en/documentation/cdh5/latest/CDH5-Installation-Guide/cdh5ig_spark_installation.html)

### **++ Spark Training Videos ++**

From Spark Summit 2013: <http://spark-summit.org/2013>

From Spark Summit 2014: <http://spark-summit.org/2014>

## ++ Databricks Resources for Spark ++

Databricks will be releasing free videos, docs and labs to learn Spark here:

<http://databricks.com/spark-training-resources>

## ++ Spark Certification ++

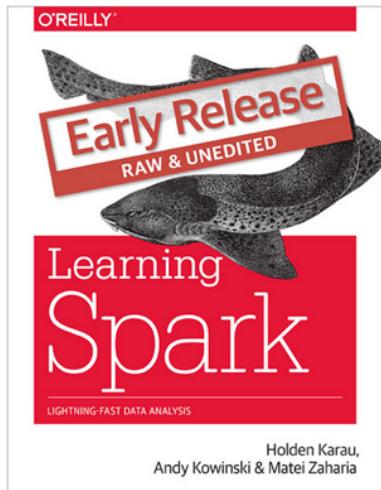
Note, when you're ready to get certified as a Spark Developer, check out the joint Spark certification program between Databricks & O'Reilly:

<http://radar.oreilly.com/2014/09/announcing-spark-certification.html>

[http://www.oreilly.com/data/sparkcert.html?cmp=ex-strata-na-lp-na\\_apache\\_spark\\_certification](http://www.oreilly.com/data/sparkcert.html?cmp=ex-strata-na-lp-na_apache_spark_certification)

## One Special Resource

A good portion of the steps in this lab come from the following book, so it is worth pointing out separately. If you want to dive deeper into Spark after completing this lab, then this is the best technical resource to get started with:



### Learning Spark

Early Release version available now.

<http://shop.oreilly.com/product/0636920028512.do>

## CDH 5.2.1 and Cloudera Manager Information

Cloudera Distribution of Hadoop (CDH) consists of 100% open source Apache Hadoop plus about 20 other open source projects from the Hadoop ecosystem. Cloudera claims that “CDH is thoroughly tested and certified to integrate with the widest range of operating systems, hardware and databases.”

CDH has been around for years, as CDH2, CDH3, CDH4, and now CDH5.

Although CDH is a production-ready distribution of Apache Hadoop, it can be tricky to install, manage and monitor via cmd-line tools. To ease the burden of deploying and managing CDH/Hadoop, Cloudera released Cloudera Manager (CM). There are two types of CM: Free/Express and Enterprise (which comes in basic, flex & data hub editions). CM Express used to be limited to clusters under 50 nodes, but starting with CM 4.5 it can be used with unlimited nodes. Both editions of CM help with deployment of binaries and Hadoop services/configurations (XML files) management. CM Enterprise includes more features like rolling upgrades, LDAP integration, operational reports, automated disaster recovery, data auditing and tech support integration.

To see a detailed comparison between CM Express vs. CM Enterprise, visit this page:  
<http://www.cloudera.com/content/cloudera/en/products-and-services/product-comparison.html>

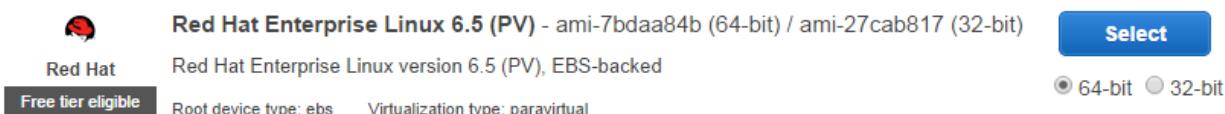
This lab uses Cloudera Manager Express (CM) to push the CDH binaries (packages). In a real, production cluster, Cloudera Manager would typically run on a separate server from the Hadoop cluster. Then the Cloudera Manager software would be used to push the CDH binaries (parcels) externally to a 10, 50 or even 1,000 node Hadoop cluster (with the nodes running various roles like Master daemons or Slave daemons). However, *in this lab*, we will keep things simple and cheap and use the same EC2 node for Cloudera Manager and CDH5.

## Connect to Amazon EC2 Instance

Each student will get 1 virtual server in Amazon use for the duration of the lab. The server has had nothing special done with it (no boot scripts, so silent installs, etc). No one has logged into this 1-node cluster via SSH or a web UI since it launched. You will be the first to do so in the next section. So, after this class is finished, you can launch the same server type in AWS and re-do this entire lab if you wish and then build your skills by trying new things with the environment.

Note that this lab is being released via a Creative Commons license so you are free to print it and share it, as long as it is not for commercial purposes.

This 64-bit AMI was launched in the us-west-2 (Oregon) datacenter to build your EC2-instance (*this AMI is not accessible in other regions*):



Note this AMI is a [ParaVirtualization \(PV\) type](#), which traditionally performs better than Hardware Virtual Machines (HVM).

To launch your specific m3.xlarge node, a 40 GB general purpose SSD (with 30 baseline IOPS / 3000 burst IOPS for 30 minutes) was used. There is only one root storage device configured under /dev/sda1.

Pick your favorite SSH client and connect to the Amazon instance with the connection details below:

Port: 22

Username: ec2-user

Hostname: <public DNS hostname that AWS or the instructor provided you with>

Key Pair: .pem (for OS X) or .ppk (for Windows)

I recommend using **PuTTY** on Windows and **iTerm2** (or Terminal) on OS X.



The general instructions to log in via OS X Terminal are:

Open up your terminal app of choice and type in the following...

Change the permissions of the .pem key file like this: **chmod 400 my-key-pair.pem**

SSH into the VM using this command: **ssh -i Amazon-Private-Key.pem ec2-user@<public hostname of VM>**

Say Yes to this prompt:

The authenticity of host 'ec2-198-51-100-x.compute-1.amazonaws.com (10.254.142.33)' can't be established. RSA key fingerprint is 1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f. Are you sure you want to continue connecting (yes/no)? **yes**

Here are the official details on how to log in via Mac:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/>

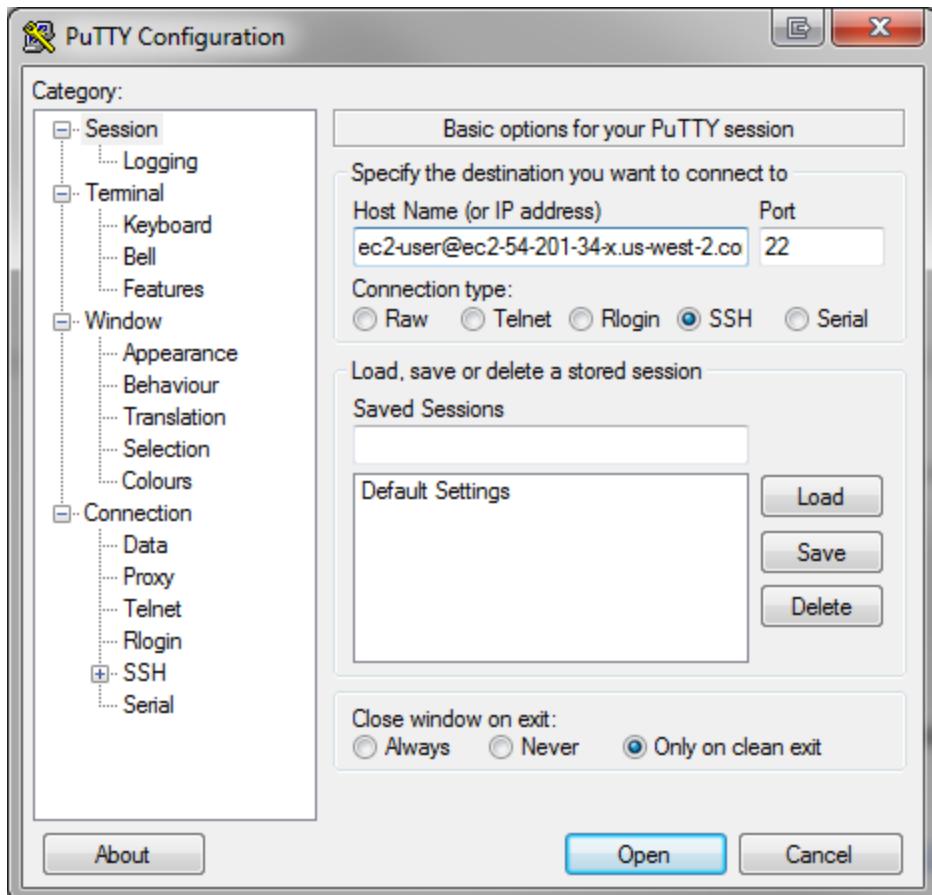
**Connection Settings screenshots for PuTTY (on Windows):**

Download PuTTY.exe from:

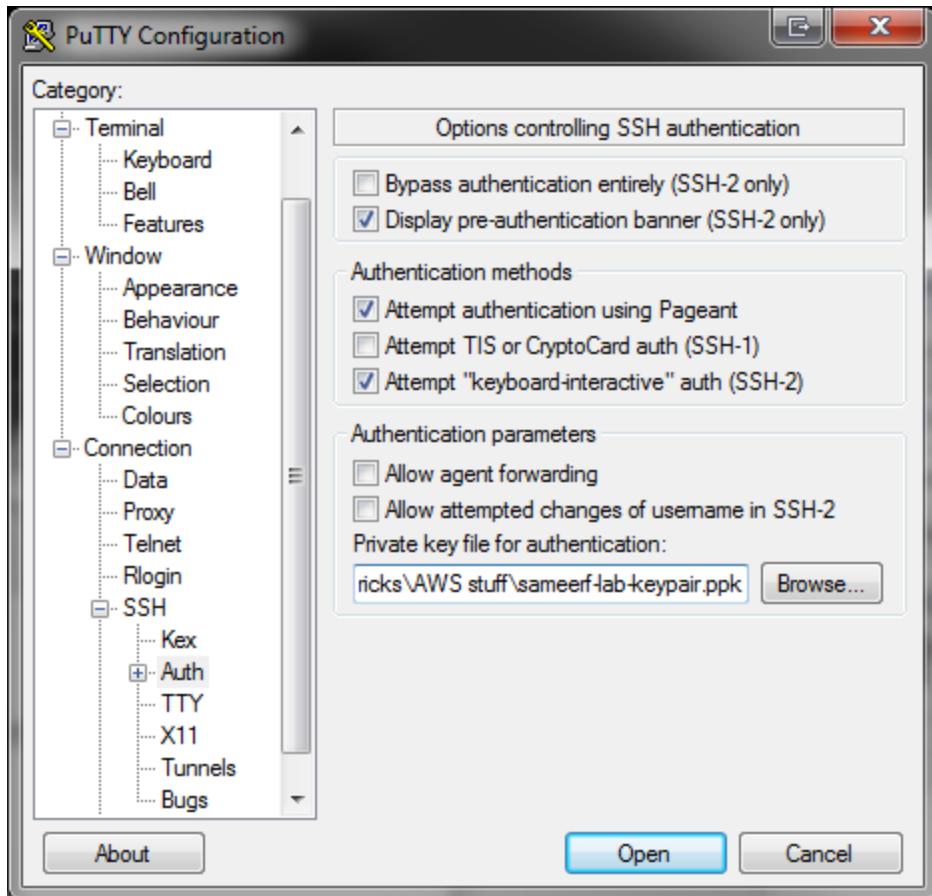
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

There is no installation for PuTTY. You can just run it from the downloaded .exe file.

**Notice that appending 'ec2-user@' to the Host Name will automatically log you into the VM:**



In the left pane, under Connection > SSH > Auth, you can provide the location to the private .ppk key on your Windows machine:



This is what you'll see once you are successfully logged in:

A screenshot of a terminal window titled 'ec2-user@ip-10-0-55-58:~'. The window displays the message 'Using username "ec2-user". Authenticating with public key "imported-openssh-key"' followed by a prompt '[ec2-user@ip-10-0-55-58 ~]\$'. The background of the terminal is black, and the text is white.

## Prepwork + Install Cloudera Manager

Let's begin by checking a few parameters on the EC2 virtual machine and then installing Cloudera Manager (CM).

**First verify that this server has about 15 GB of RAM and only 200 MB or so are currently being used:**

```
[ec2-user@ip-10-0-72-36 ~]$ free -m
      total        used        free      shared      buffers      cached
Mem:       14882         263      14618          0         18          60
-/+ buffers/cache:      184      14697
Swap:          0          0          0
```

**Check what type of file systems are running in the VM:**

```
[ec2-user@ip-10-0-72-36 ~]$ df -Th
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/xvda1      ext3  40G   1.1G   37G   3% /
tmpfs           tmpfs  7.4G    0    7.4G   0% /dev/shm
```

Notice that the main file system is /dev/xvda1, which is of type ext3 and of size 40GB total with 37GB still available. To keep things simple, we will store the HDFS blocks on the same xvda1 partition, even though in production it is best to keep the HDFS blocks on a separate spindle.

**All of the Hadoop daemons require Java JVMs to run. Check if Java's installed:**

```
[ec2-user@ip-10-0-72-36 ~]$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (rhel-2.5.1.2.el6_5-x86_64 u65-b17)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Well, the OpenJDK version of Java is installed, but CDH prefers to run with the Oracle JDK 1.7 version. So, during the CDH 5 install, when we see an option to install Oracle Java SE Development Kit (JDK), we'll accept it.

iptables is an administrative tool/command for IPv4 packet filtering and NAT. It's essentially a firewall mechanism for linux. **Just so we don't have to worry about opening up two dozen specific ports, let's make sure iptables is turned off entirely:**

```
[ec2-user@ip-10-0-60-107 ~]$ chkconfig --list iptables  
iptables      0:off    1:off    2:off    3:off    4:off    5:off    6:off
```

```
[ec2-user@ip-10-0-60-107 ~]$ sudo service iptables status  
iptables: Firewall is not running.
```

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides the mechanism for supporting access control security policies, including United States Department of Defense–style mandatory access controls (MAC). **Having SELinux enabled can complicate things, so if it is enabled, let's turn it off:**

```
[ec2-user@ip-10-0-60-107 ~]$ sestatus  
SELinux status:          enabled  
SELinuxfs mount:         /selinux  
Current mode:            enforcing  
Mode from config file:  enforcing  
Policy version:          24  
Policy from config file: targeted
```

**Turn off SELinux by editing line 7 of the following file by changing the parameter 'enforcing' to 'disabled':**

```
[ec2-user@ip-10-0-60-107 ~]$ sudo vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.  
#       permissive - SELinux prints warnings instead of enforcing.  
#       disabled - No SELinux policy is loaded.  
SELINUX=disabled  
# SELINUXTYPE= can take one of these two values:  
#       targeted - Targeted processes are protected,  
#       mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

**Save and Quit out of the file. We will need to reboot to make the SELinux changes take effect, but first...**

**Check if Memory Swapping is turned on. If it is on, let's turn it off since it's not recommended to use this linux feature with Hadoop:**

```
[ec2-user@ip-10-0-60-107 ~]$ cat /proc/sys/vm/swappiness  
60
```

Swapping is on and set to 60. Turn swapping off by opening the following file and adding the appending the line ‘vm.swappiness = 0’ to the end of the file:

```
[ec2-user@ip-10-0-60-107 ~]$ sudo vi /etc/sysctl.conf  
<beginning of file truncated>  
# Controls the maximum shared segment size, in bytes  
kernel.shmmmax = 68719476736  
  
# Controls the maximum number of shared memory segments, in pages  
kernel.shmall = 4294967296  
  
vm.swappiness = 0
```

**Save and Quit out of the file.**

**Reboot the server to make the SELinux and swap changes take effect:**

```
[ec2-user@ip-10-0-60-107 ~]$ sudo reboot
```

```
Broadcast message from ec2-user@ip-10-0-60-107  
(/dev/pts/1) at 2:18 ...
```

The system is going down for reboot NOW!

**Once you are reconnected, verify SELinux and swapping are disabled:**

```
[ec2-user@ip-10-0-60-107 ~]$ sestatus  
SELinux status: disabled
```

```
[ec2-user@ip-10-0-60-107 ~]$ cat /proc/sys/vm/swappiness  
0
```

**Finally, we are ready to download and install Cloudera Manager:**

```
[ec2-user@ip-10-0-72-36 ~]$ pwd  
/root  
  
[ec2-user@ip-10-0-72-36 ~]$ wget  
http://blueplastic.com/databricks/cloudera-manager-installer.bin  
--2014-10-27 03:27:04--  
http://blueplastic.com/databricks/cloudera-manager-installer.bin  
Resolving blueplastic.com... 74.220.207.68  
Connecting to blueplastic.com|74.220.207.68|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 510569 (499K) [application/octet-stream]  
Saving to: "cloudera-manager-installer.bin"  
  
100%[=====] 510,569      717K/s   in 0.7s  
  
2014-10-27 03:27:05 (717 KB/s) - "cloudera-manager-installer.bin" saved  
[510569/510569]
```

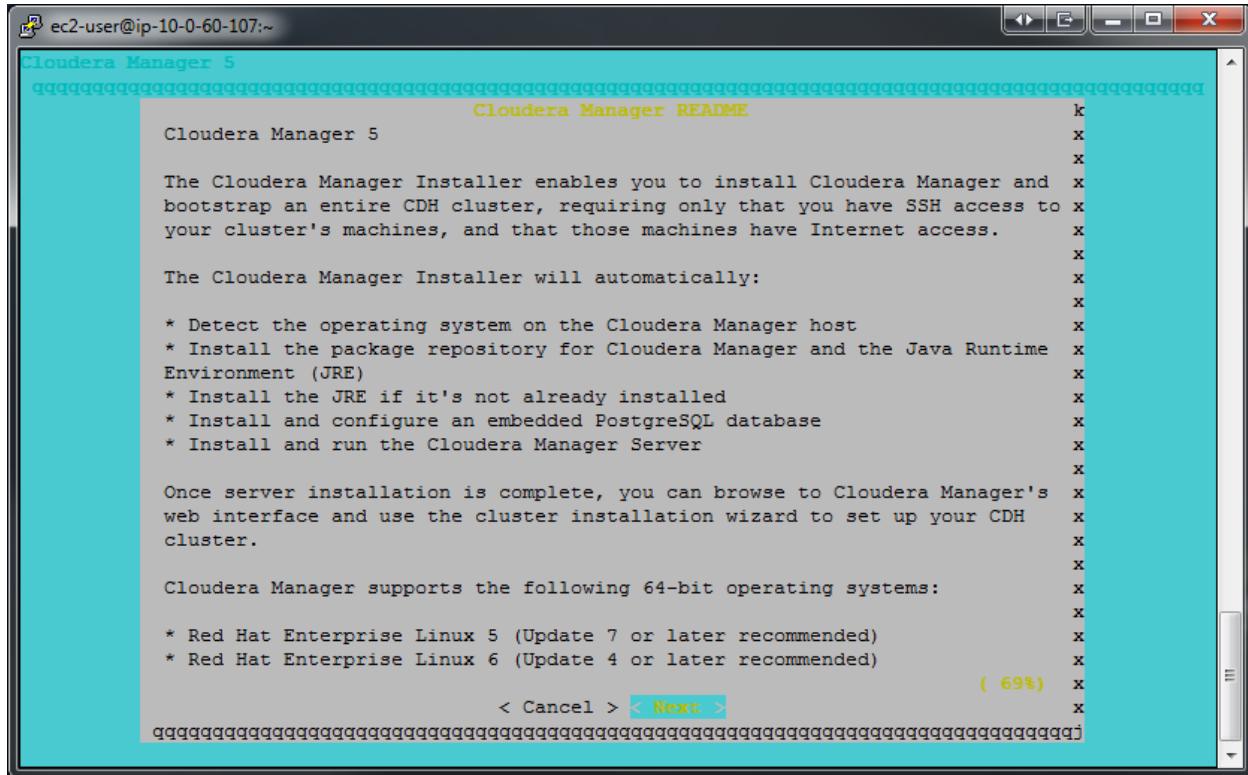
**Set the installer file to have executable permissions:**

```
[ec2-user@ip-10-0-72-36 ~]$ ls -l  
total 504  
-rw-r--r-- 1 root root 510569 Sep 30 21:13 cloudera-manager-installer.bin  
  
[ec2-user@ip-10-0-72-36 ~]$ chmod u+x cloudera-manager-installer.bin  
  
[ec2-user@ip-10-0-72-36 ~]$ ls -l  
total 504  
-rwxr--r-- 1 root root 510569 Sep 30 21:13 cloudera-manager-installer.bin
```

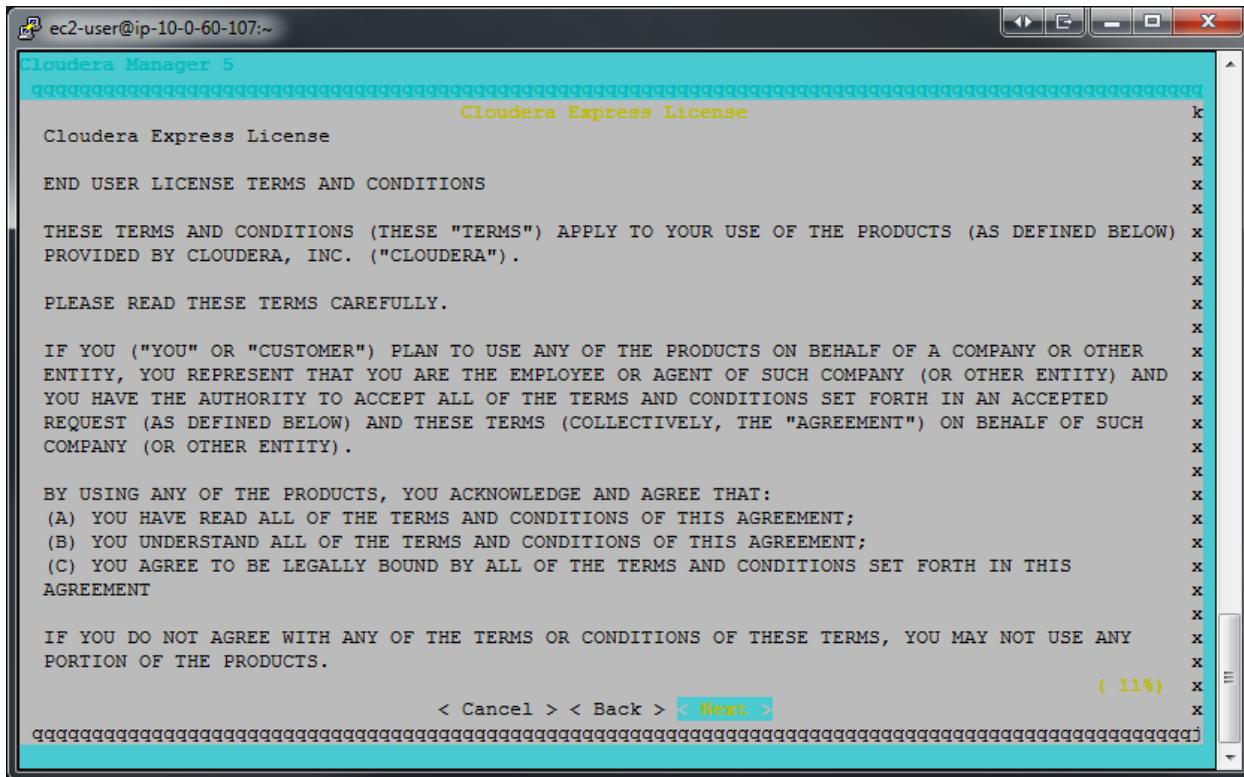
**Run the Cloudera Manager installer:**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo ./cloudera-manager-installer.bin
```

Press **Enter** to choose Next for the Readme file:



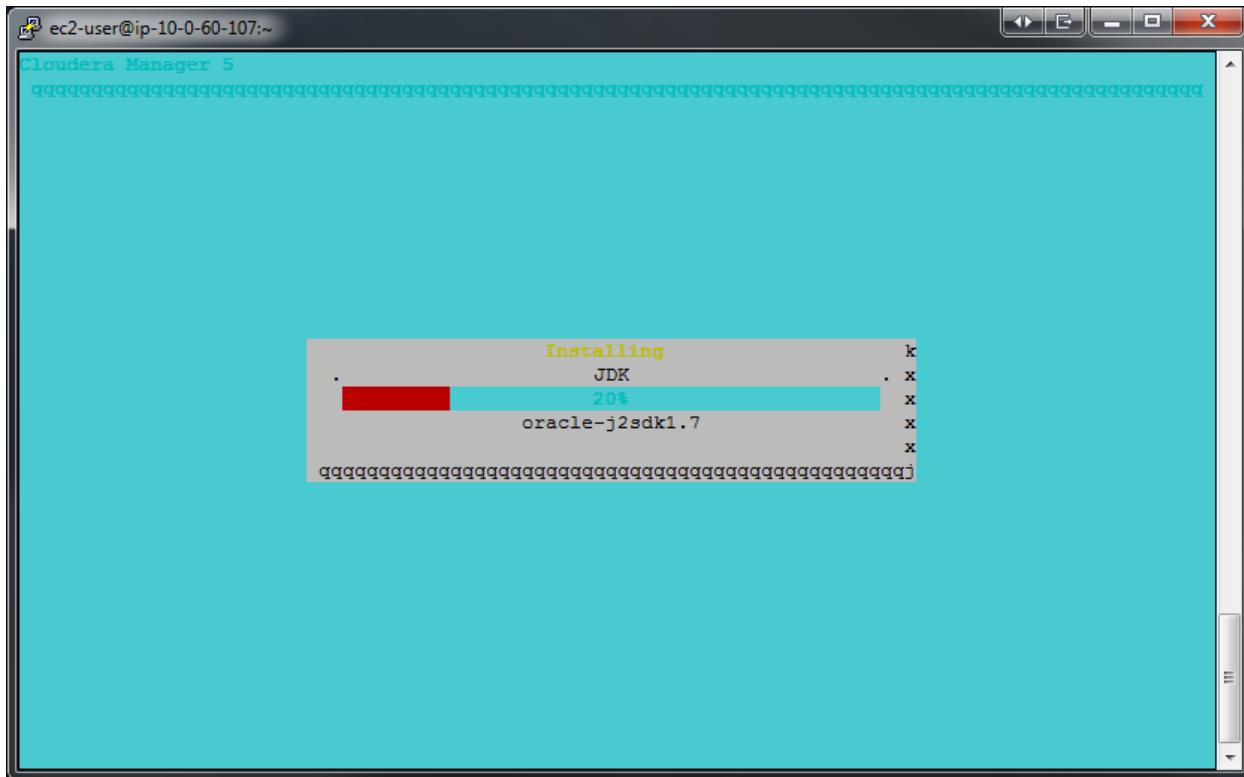
Then you will see this, simply hit enter to choose **Next** again:



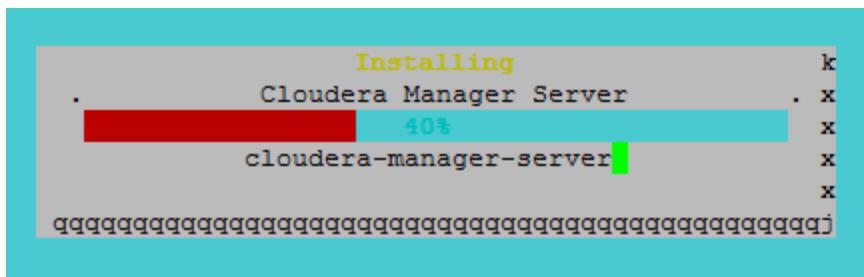
Now use the right arrow key to choose **Yes** for the license agreement.

The Oracle Binary Code License Agreement appears now. Press **Enter** to choose **Next** and then choose **Yes** on the 'Accept this license' screen.

The Oracle JDK installation will begin:

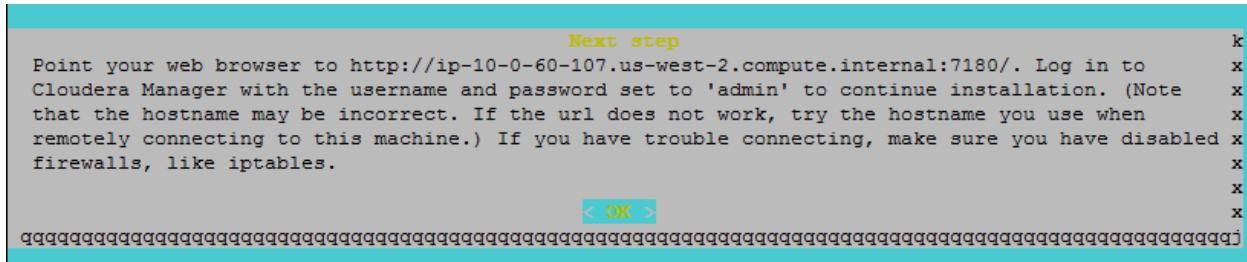


Followed by the CM installation:

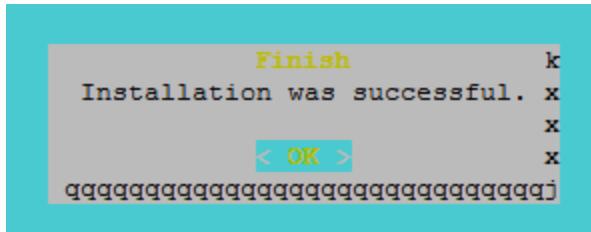


The installer will also install an embedded PostgreSQL database server.

When the installation is finished, you will see this prompt. **Note that the URL is an internal hostname, so you will not be able to use it to access the GUI for Cloudera Manager (instead you'll have to use the *public* hostname). Hit **Enter** to return to the linux cmd prompt:**



**Choose **OK** on the final installation screen:**



You should now be returned to the linux cmd prompt.

## Connect to Cloudera Manager Admin Console to install CDH

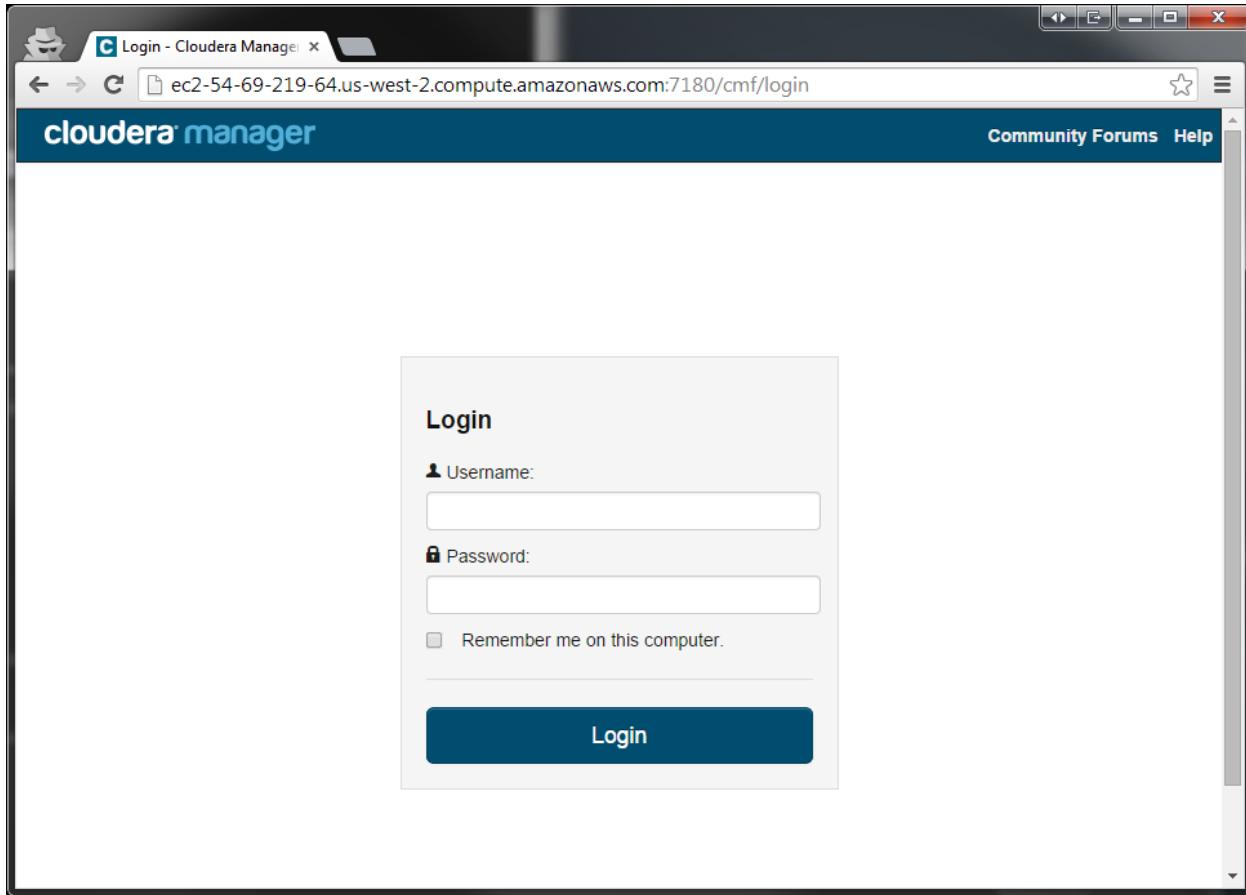
So far, only Cloudera Manager has been installed on the Amazon VM. CDH5 has not been installed yet. Time to do that now.

*Note, once you start this section, it is very important not to close your browser tab or to click 'back' in the browser while the installation is happening.*

Wait for about 45 seconds for the Cloudera Manager web site service to start up and then...

**From Chrome or Firefox, go to the server URL for Cloudera Manager:**

**<http://<your public hostname or ip>:7180>**



Log in to Cloudera Manager using the default credentials:

Username: **admin**

Password: **admin**

Put a tick mark next to "**Remember me on this computer**".

The Cloudera Manager welcome screen and edition selector now appears. **Click on Cloudera Express and then Continue at the bottom of the screen:**

The screenshot shows the 'Welcome to Cloudera Manager' page. At the top, it says 'Welcome to Cloudera Manager. Which edition do you want to deploy?'. Below that, it says 'Upgrading to Cloudera Enterprise Data Hub Edition provides important features that help you manage and monitor your Hadoop clusters in mission-critical environments.' A red arrow points to the 'Cloudera Express' column in a table.

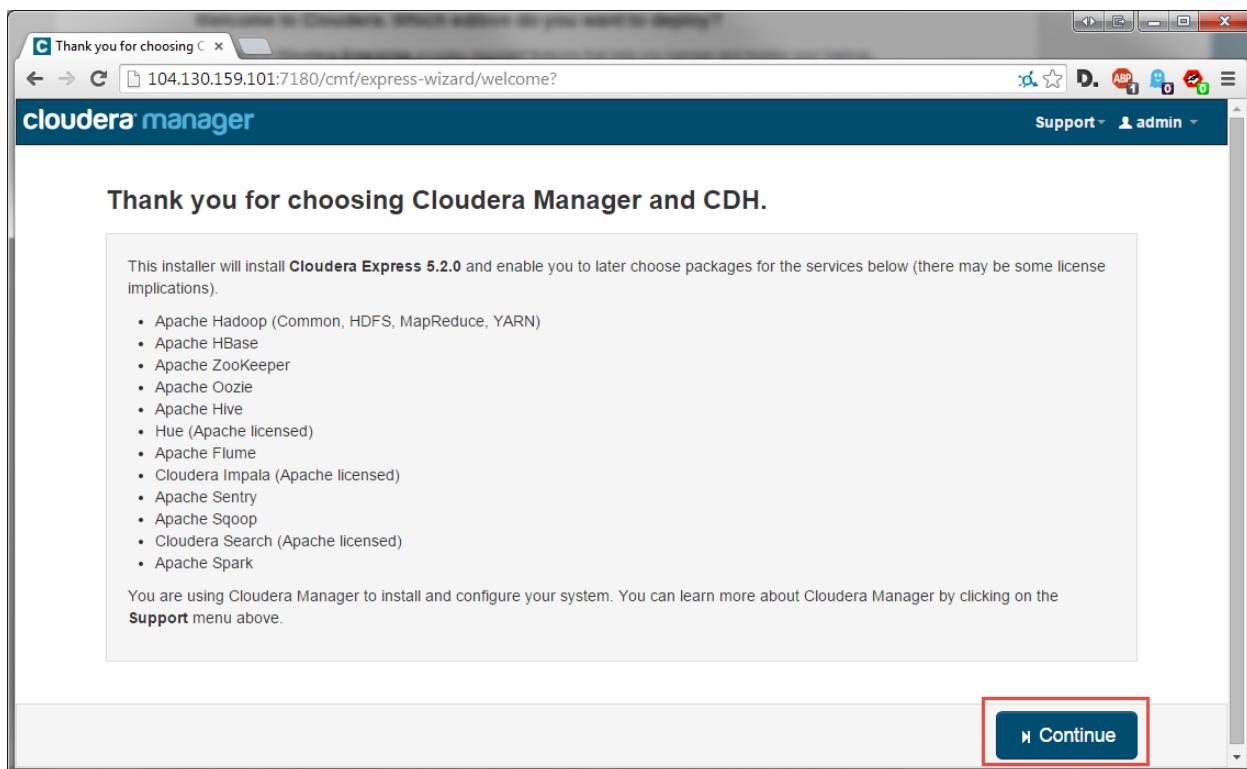
	Cloudera Express	Cloudera Enterprise Data Hub Edition Trial	Cloudera Enterprise
License	Free	60 Days After the trial period, the product will continue to function as <b>Cloudera Express</b> . Your cluster and your data will remain unaffected.	Annual Subscription <b>Upload License</b> Cloudera Enterprise is available in three editions: <ul style="list-style-type: none"><li>Basic Edition</li><li>Flex Edition</li><li>Data Hub Edition</li></ul>
Node Limit	Unlimited	Unlimited	Unlimited
CDH	✓	✓	✓
Core Cloudera Manager Features	✓	✓	✓
Advanced Cloudera Manager Features		✓	✓
Cloudera Navigator		✓	✓
Cloudera Support			✓

For full list of features available in **Cloudera Express** and **Cloudera Enterprise**, [click here](#).

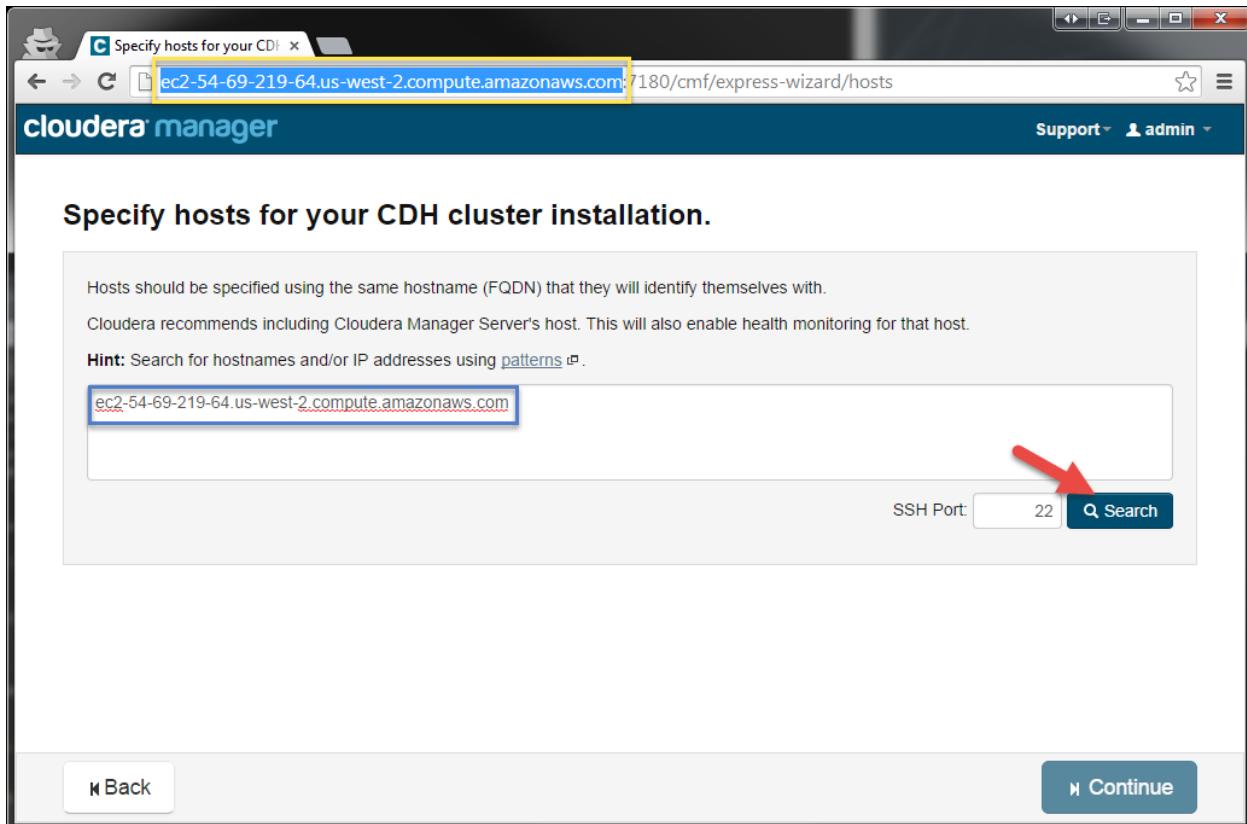
**Continue**

Note, we will not be using the Cloudera Enterprise features in this lab, so please don't select the Enterprise version as it may add more memory overhead to the install.

On the Thank you screen, just click **Continue**:



Type in your public IP and click **Search**. You can copy + paste the IP from the URL:



Your local Virtual Machine should now be found and automatically checked. **Click Continue:**

The screenshot shows a web browser window for Cloudera Manager. The URL is 104.130.159.101:7180/cmftexpress-wizard/hosts. The page title is "Specify hosts for your CDH cluster installation". It displays a table of scanned hosts:

	Expanded Query	Hostname (FQDN)	IP Address	Currently Managed	Result
<input checked="" type="checkbox"/>	104.130.159.101	cdh5-cm-vm01	104.130.159.101	No	<span style="color: green;">✓</span> Host ready: 1 ms response time.

A red arrow points to the "Continue" button at the bottom right of the form.

On the Cluster Installation screen, verify that your settings are the same as the screenshot below and click **Continue**. Do NOT install Accumulo, or the Sqoop connectors at this time as they will not be used in this lab and will consume extra resources on the VM. This lab was created using CDH 5.2.1 and it is highly recommended that you stick with this version of this lab, so you have a consistent environment as the instructor and other students.

## Cluster Installation

### Select Repository

Cloudera recommends the use of parcels for installation over packages, because parcels enable Cloudera Manager to easily manage the software on your cluster, automating the deployment and upgrade of service binaries. Electing not to use parcels will require you to manually upgrade packages on all hosts in your cluster when software updates are available, and will prevent you from using Cloudera Manager's rolling upgrade capabilities.

**Choose Method**  Use Packages [?](#)  Use Parcels (Recommended) [?](#) [More Options](#)

**Select the version of CDH**

CDH-5.2.1-1.cdh5.2.1.p0.12  
 CDH-4.7.0-1.cdh4.7.0.p0.40

**Additional Parcels**

ACCUMULO-1.6.0-1.cdh5.1.0.p0.51  
 ACCUMULO-1.4.4-1.cdh4.5.0.p0.65  
 None

KEYTRUSTEE-3.7.0-0.cdh5.2.0.p6  
 None

SQOOP\_NETEZZA\_CONNECTOR-1.2c5  
 None

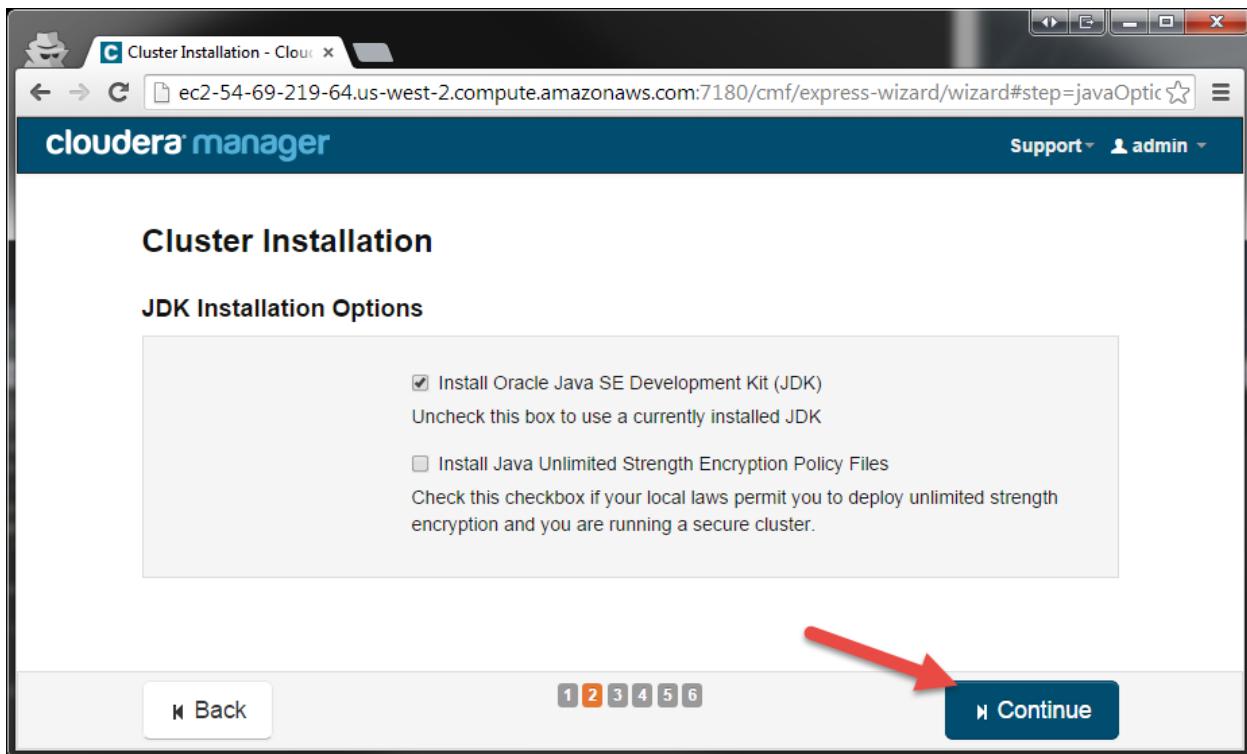
SQOOP\_TERADATA\_CONNECTOR-1.2c5  
 None

**Select the specific release of the Cloudera Manager Agent you want to install on your hosts.**

Matched release for this Cloudera Manager Server  
 Custom Repository

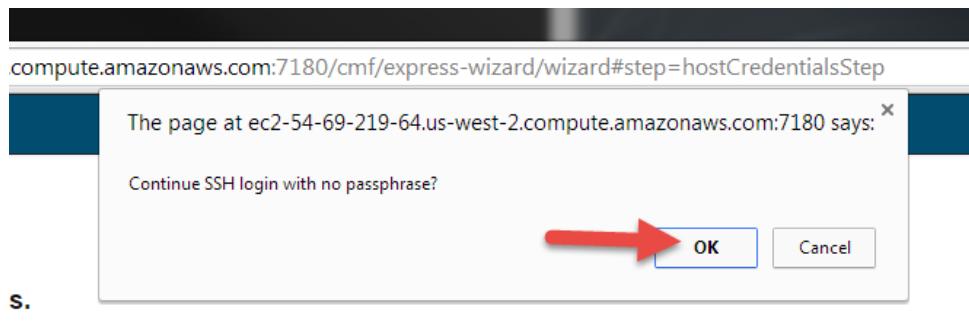
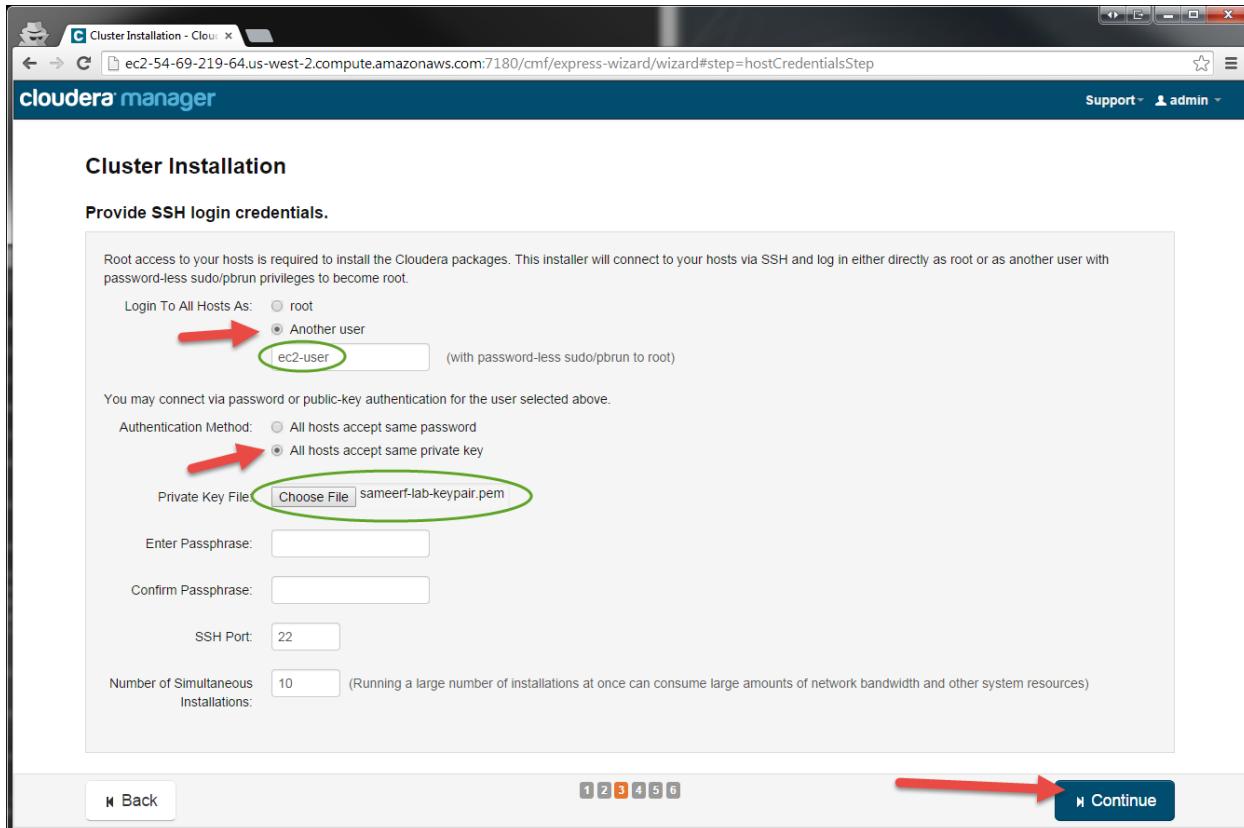
 **Continue**

The next screen gives you the option to install Oracle JDK. Since the Oracle JDK is the most widely deployed and tested JDK for Hadoop, we'll also run with the Oracle version. **Just click Continue:**



On the ‘SSH login credentials’ page, choose to:

- **Login To All Hosts As: Another user** (type in this username: **ec2-user**)
- **Authentication Method: All hosts accept same private key** (then click **Choose File** and select the .pem file the instructor gave you)
- In the pop-up to ‘Continue SSH login with no passphrase?’, just click **OK**



The cluster installation will now kick off and run for ~1 minute:

### Cluster Installation

Installation in progress.

0 of 1 host(s) completed successfully. [Abort Installation](#)

Hostname	IP Address	Progress	Status	
ip-10-0-60-107.us-west-2.compute.internal	10.0.60.107	<div style="width: 100%;"> </div>	Refreshing package metadata...	<a href="#">Details</a>

When it is finished, click **Continue**:

The screenshot shows the Cloudera Manager Cluster Installation interface. The title bar reads "Cluster Installation - Cloud". The URL in the address bar is "ec2-54-69-219-64.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/wizard#step=installStep". The top navigation bar includes "Support", "admin", and a user icon. The main content area has a header "Cluster Installation" and a message "Installation completed successfully." Below this is a progress bar that is fully green. A table displays the host status: "1 of 1 host(s) completed successfully." with one row for "ip-10-0-60-107.us-west-2.compute.internal" (IP 10.0.60.107) which is marked as "Installation completed successfully.". At the bottom, there are navigation buttons "Back" and "Continue", with a red arrow pointing to the "Continue" button. A footer navigation bar shows steps 1 through 6.

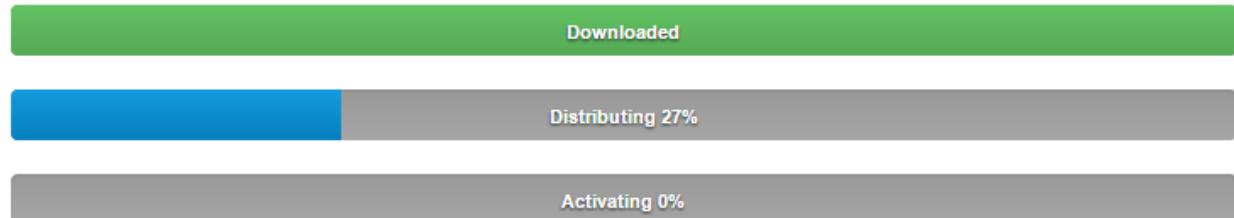
The Parcels installation will now start (note this part will take about 2 - 3 minutes):

## Cluster Installation

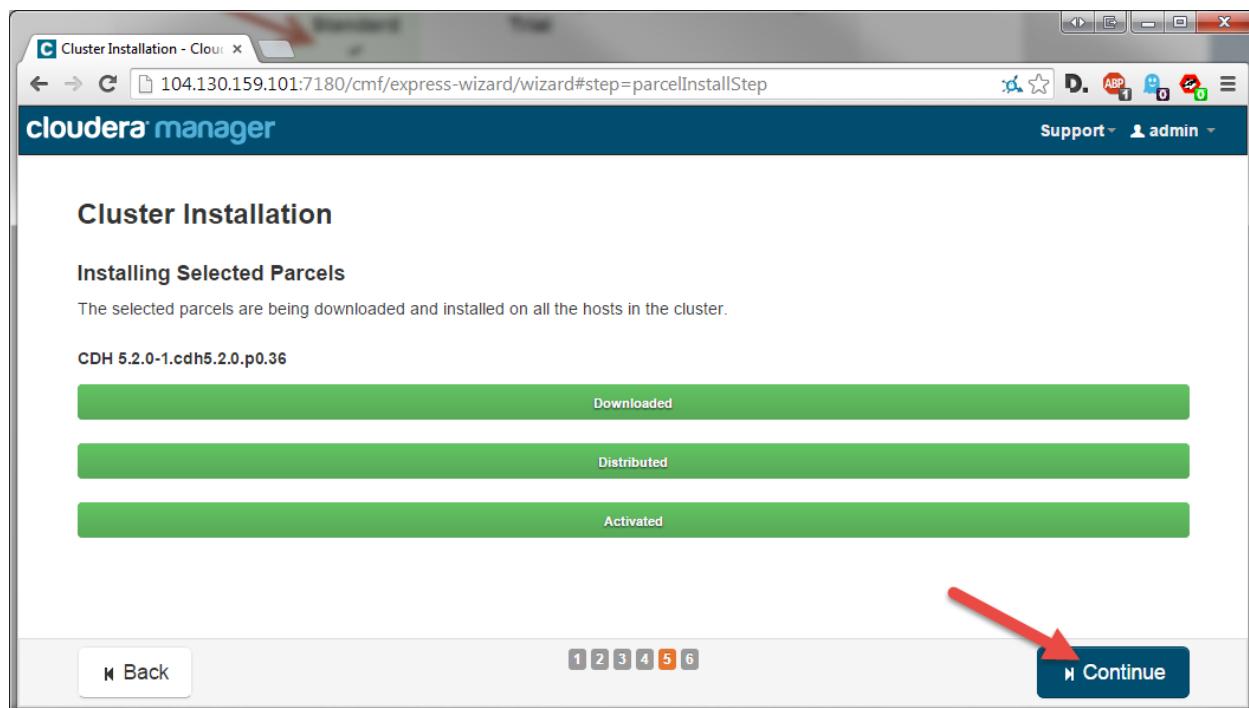
### Installing Selected Parcels

The selected parcels are being downloaded and installed on all the hosts in the cluster.

CDH 5.2.1-1.cdh5.2.1.p0.12



When the Parcels installation is complete, click **Continue** to proceed:



The host inspector will automatically run. It should find nothing wrong and automatically take you to the following page of Validations:

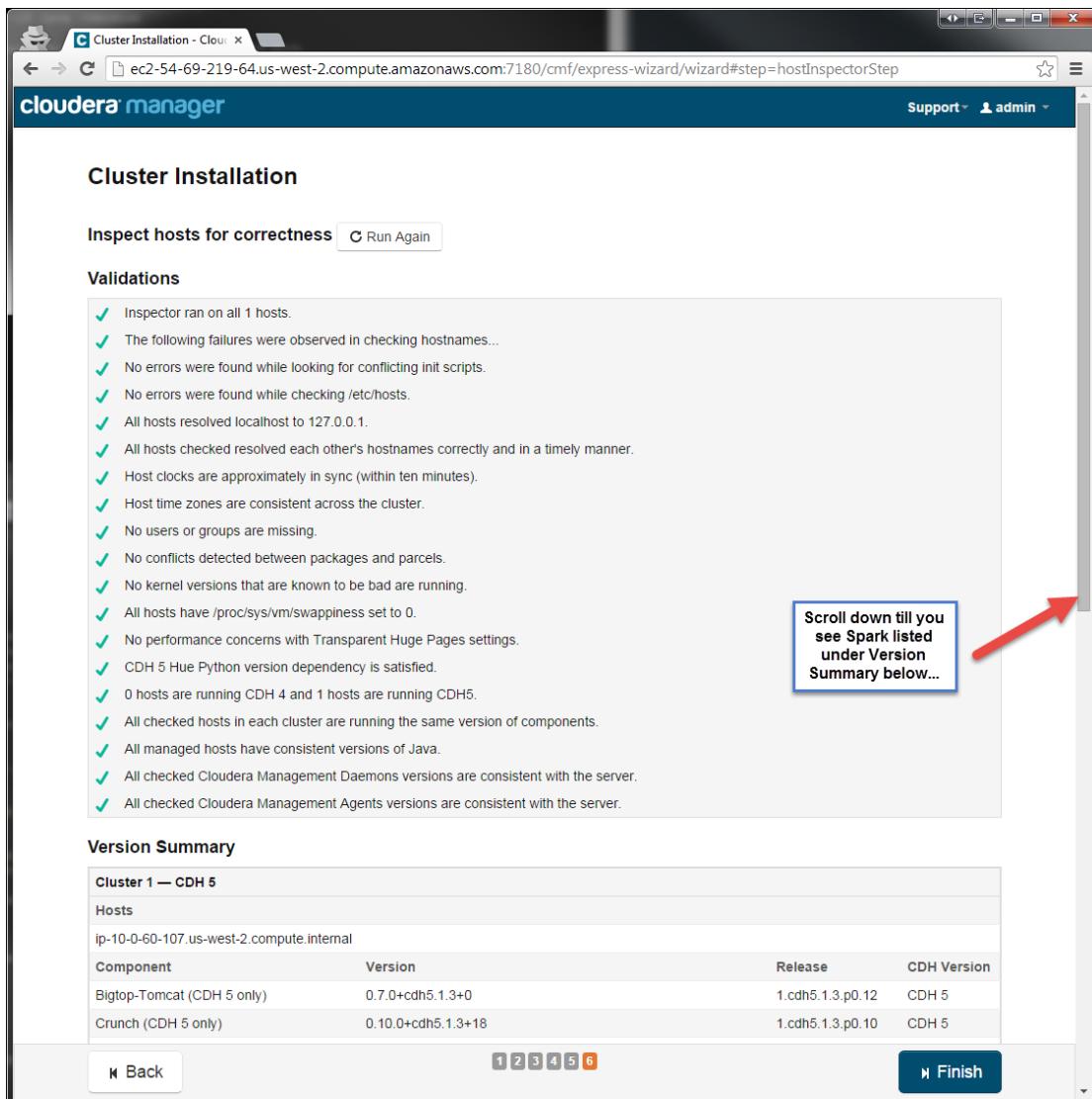
## Cluster Installation

### Inspect hosts for correctness

Inspecting hosts... This could take a minute. 

[Skip Host Inspector](#)

**Scroll down till you see the different versions of the various Apache projects listed under Version Summary below:**



The screenshot shows the 'Cluster Installation' step in the Cloudera Manager wizard. The 'Host Inspector' step has completed successfully, displaying a list of validation items with green checkmarks. A red arrow points from a callout box to the 'Version Summary' section, which is currently empty. The 'Finish' button at the bottom right is visible.

**Validations**

- ✓ Inspector ran on all 1 hosts.
- ✓ The following failures were observed in checking hostnames...
- ✓ No errors were found while looking for conflicting init scripts.
- ✓ No errors were found while checking /etc/hosts.
- ✓ All hosts resolved localhost to 127.0.0.1.
- ✓ All hosts checked resolved each other's hostnames correctly and in a timely manner.
- ✓ Host clocks are approximately in sync (within ten minutes).
- ✓ Host time zones are consistent across the cluster.
- ✓ No users or groups are missing.
- ✓ No conflicts detected between packages and parcels.
- ✓ No kernel versions that are known to be bad are running.
- ✓ All hosts have /proc/sys/vm/swappiness set to 0.
- ✓ No performance concerns with Transparent Huge Pages settings.
- ✓ CDH 5 Hue Python version dependency is satisfied.
- ✓ 0 hosts are running CDH 4 and 1 hosts are running CDH5.
- ✓ All checked hosts in each cluster are running the same version of components.
- ✓ All managed hosts have consistent versions of Java.
- ✓ All checked Cloudera Management Daemons versions are consistent with the server.
- ✓ All checked Cloudera Management Agents versions are consistent with the server.

**Version Summary**

Cluster 1 — CDH 5			
Hosts	Component	Version	Release
ip-10-0-60-107.us-west-2.compute.internal	Bigtop-Tomcat (CDH 5 only)	0.7.0+cdh5.1.3+0	1.cdh5.1.3.p0.12
	Crunch (CDH 5 only)	0.10.0+cdh5.1.3+18	1.cdh5.1.3.p0.10

1 2 3 4 5 6

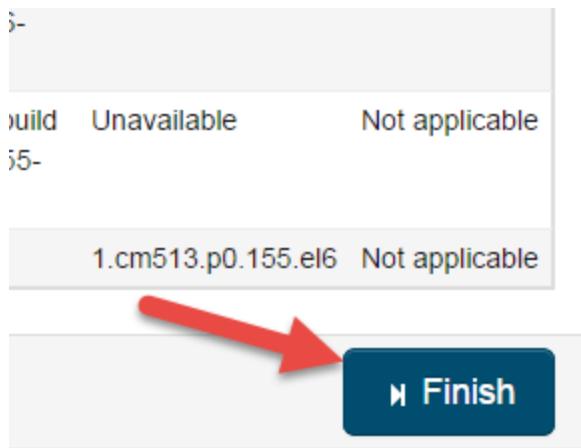
[Back](#) [Finish](#)

**Notice that we are running Spark 1.1.0:**

Solr	4.4.0+cdh5.2.1+284	1.cdh5.2.1.p0.15	CDH 5
spark	1.1.0+cdh5.2.1+63	1.cdh5.2.1.p0.9	CDH 5
Sqoop2	1.99.3+cdh5.2.1+32	1.cdh5.2.1.p0.7	CDH 5

**CDH 5.2 also comes with HDFS 2.5 and YARN 2.5.**

**Click Finish:**



We are almost finished with the CDH installation wizard. Now we have to choose which of the CDH 5 servers we want to start. **Click on ‘Custom Services’ and only select the items selected in the screenshot below. Then click Continue.**

**Custom Services**

Choose your own services. Services required by chosen services will automatically be included. Flume can be added after your initial cluster has been set up.

Service Type	Description
<input type="checkbox"/> HBase	Apache HBase provides random, real-time, read/write access to large data sets (requires HDFS and ZooKeeper).
<input checked="" type="checkbox"/>  HDFS	Apache Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute hosts throughout a cluster to enable reliable, extremely rapid computations.
<input type="checkbox"/> Hive	Hive is a data warehouse system that offers a SQL-like language called HiveQL.
<input type="checkbox"/> Hue	Hue is a graphical user interface to work with Cloudera's Distribution Including Apache Hadoop (requires HDFS, MapReduce, and Hive).
<input type="checkbox"/> Impala	Impala provides a real-time SQL query interface for data stored in HDFS and HBase. Impala requires Hive service and shares Hive Metastore with Hue.
<input type="checkbox"/> Isilon	EMC Isilon is a distributed filesystem.
<input type="checkbox"/> Key-Value Store Indexer	Key-Value Store Indexer listens for changes in data inside tables contained in HBase and indexes them using Solr.
<input type="checkbox"/> MapReduce	Apache Hadoop MapReduce supports distributed computing on large data sets across your cluster (requires HDFS). YARN (MapReduce 2 Included) is recommended instead. MapReduce is included for backward compatibility.
<input type="checkbox"/> Oozie	Oozie is a workflow coordination service to manage data processing jobs on your cluster.
<input type="checkbox"/> Solr	Solr is a distributed service for indexing and searching data stored in HDFS.
<input checked="" type="checkbox"/>  Spark	Apache Spark is an open source cluster computing system. This service runs Spark as an application on YARN.
<input type="checkbox"/> Sqoop 2	Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. The version supported by Cloudera Manager is <b>Sqoop 2</b> .
<input checked="" type="checkbox"/>  YARN (MR2 Included)	Apache Hadoop MapReduce 2.0 (MRv2), or YARN, is a data computation framework that supports MapReduce applications (requires HDFS).
<input type="checkbox"/> ZooKeeper	Apache ZooKeeper is a centralized service for maintaining and synchronizing configuration data.

**Back** **Continue**

1 2 3 4 5 6

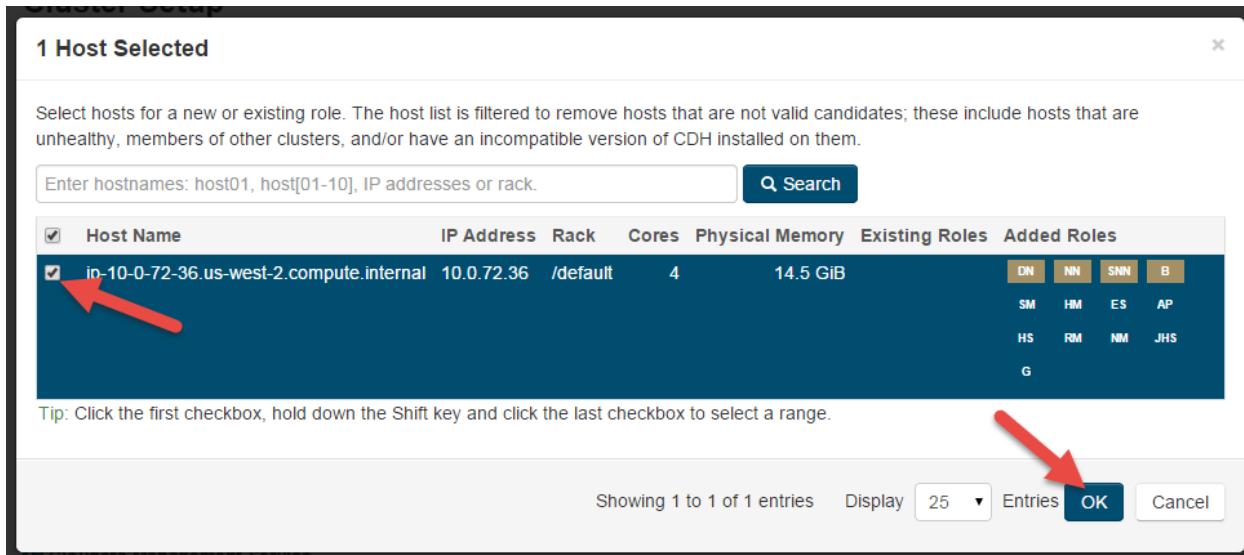
On the Customize Role Assignments page, we have to make 1 change. For the Spark service, we will want to Select a host for the Gateway. **Click Select a host under Spark's Gateway Server:**

The screenshot shows the 'Cluster Setup - Cloudera' interface on a web browser. The title bar says 'cloudera manager'. The main section is titled 'Cluster Setup' and 'Customize Role Assignments'. It lists various services and their configuration options:

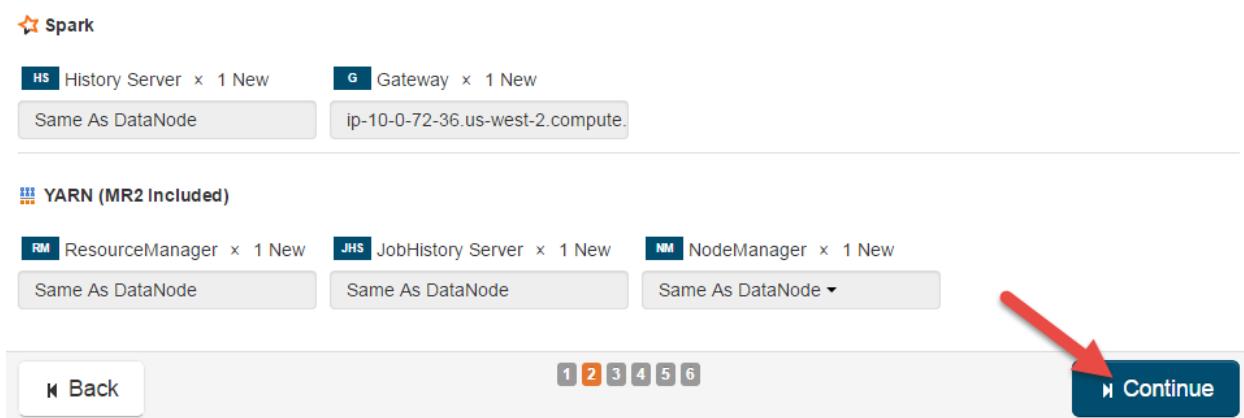
- HDFS:** NameNode (1 New), SecondaryNameNode (1 New), Balancer (1 New), HttpFS. Buttons: Same As DataNode, Same As DataNode, Same As DataNode, Select hosts.
- NFSG:** NFS Gateway. Buttons: Select hosts, ip-10-0-72-36.us-west-2.compute.
- Cloudera Management Service:** Service Monitor (1 New), Activity Monitor, Host Monitor (1 New), Event Server (1 New). Buttons: Same As DataNode, Select a host, Same As DataNode, Same As DataNode.
- AP:** Alert Publisher (1 New). Button: Same As DataNode.
- Spark:** History Server (1 New), Gateway. Buttons: Same As DataNode, Select hosts.
- YARN (MR2 Included):** ResourceManager (1 New), JobHistory Server (1 New), NodeManager (1 New). Buttons: Same As DataNode, Same As DataNode, Same As DataNode.

At the bottom, there are navigation buttons: 'Back', a page number indicator (1 2 3 4 5 6), and 'Continue'.

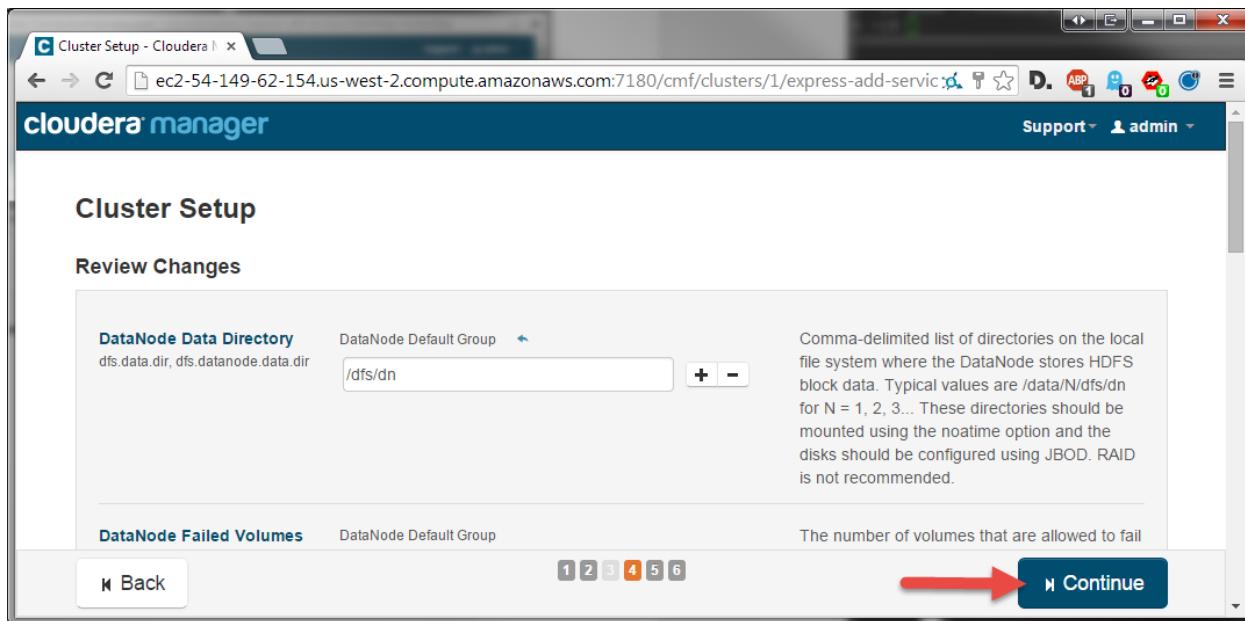
When a pop-up appears, click on the only Host Name that you see and click OK:



Almost done. Click Continue:



**There's no need to make changes on this page** (but feel free to review the settings if you'd like as an FYI). Click **Continue**:



The Cluster Setup will run through 12 steps as seen here:

## Cluster Setup

**Progress**

Command	Context	Status	Started at	Ended at
First Run		In Progress	Dec 3, 2014 4:03:39 PM EST	

**Command Progress**

Completed 1 of 12 steps.

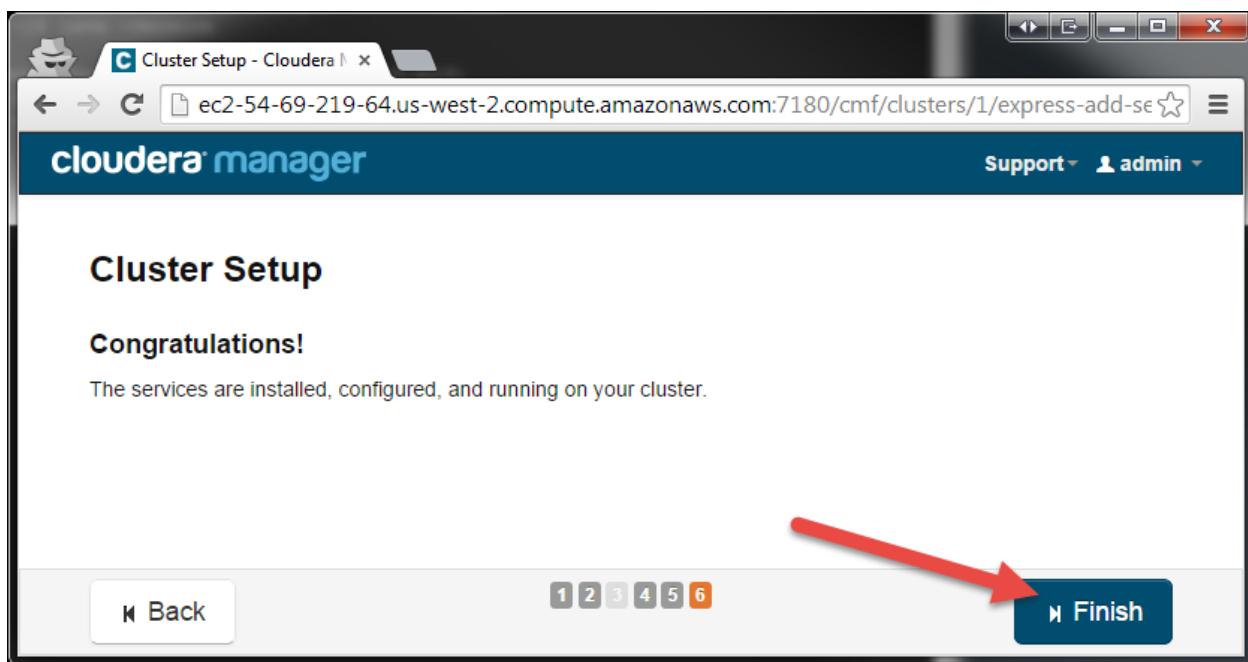
✓ Checking if the name directories of the NameNode are empty. Formatting HDFS only if empty. Successfully formatted NameNode. <a href="#">Details ↗</a>
Starting HDFS Service <a href="#">Details ↗</a>
Creating HDFS /tmp directory
Creating MR2 job history directory
Creating NodeManager remote application log directory
Starting YARN (MR2 Included) Service
Execute command CreateSparkUserDirCommand on service Spark
Execute command CreateSparkHistoryDirCommand on service Spark

[Back](#) 1 2 3 4 5 6 [Continue](#)

Click **Continue** when the 12 steps are finished:



You are now finished with the CDH 5 installation and services startup! Click **Finish**:



**When you see the Cloudera Manager GUI, there will be some warnings and critical health issues displayed in red. We will address these in the next section.**

The screenshot shows the Cloudera Manager Home page. On the left, a sidebar lists 'Cluster 1' (CDH 5.2.1, Parcels) with components: Hosts (red), HDFS (red), Spark (green), and YARN (red). Below this is the 'Cloudera Management Service' section with 'Cloudera Manager' listed. The main area features four charts: 'Cluster CPU' (Host CPU Usage Across Hosts 24.2%), 'Cluster Disk IO' (bytes / second), 'Cluster Network IO' (bytes / second), and 'HDFS IO' (bytes / second). All four charts display 'NO DATA'.

**After a few seconds the graphs will start to become populated:**

The screenshot shows the same Cloudera Manager Home page after a few seconds. The charts are now populated with data. The 'Cluster CPU' chart shows 'Host CPU Usage Across Hosts 6.1%'. The 'Cluster Disk IO' chart shows 'Total Disk Byt... 28.2K/s' and 'Total Disk Byt... 54.9K/s'. The 'Cluster Network IO' chart shows 'Total Bytes Re... 1.3K/s' and 'Total Bytes Tra... 9.1K/s'. The 'HDFS IO' chart shows 'Total Bytes Read... 1b/s' and 'Total Bytes W... 0.95b/s'.

## Post-install work & exploring Cloudera Manager

Now that the CDH installation is finished, run the jps command to see which Java daemons have started up:

```
[ec2-user@ip-10-0-72-36 ~]$ sudo /usr/java/jdk1.7.0_67-cloudera/bin/jps  
8653 AlertPublisher  
8445 HistoryServer  
9345 Jps  
7334 NodeManager  
7377 JobHistoryServer  
8591 Main  
6455 DataNode  
7562 ResourceManager  
1649 Main  
8608 EventCatcherService  
8630 Main  
6414 NameNode  
6507 SecondaryNameNode
```

We will study these later on in the class.

Let's begin exploring Cloudera Manager by reviewing which services have been started for us by Cloudera Manager:

**Cluster 1 (CDH 5.2.1, Parcels)**

Hosts	
HDFS	! 2 ✖ 2
Spark	
YARN (MR2 Inc.)	! 1

**Cloudera Management Service**

Cloudera Mana...	✖ 4
------------------	-----

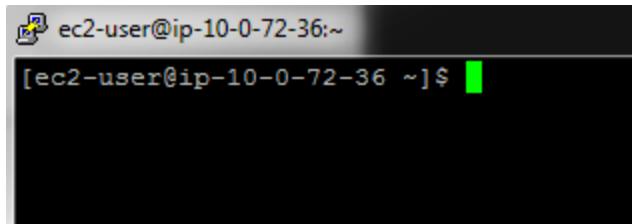
Note above that all services have actually started for us, including HDFS. HDFS & YARN are in a warning state, though, so we'll address that in the HDFS lab.

## HDFS Command Line

This 30 min lab introduces you to the HDFS command line. We will:

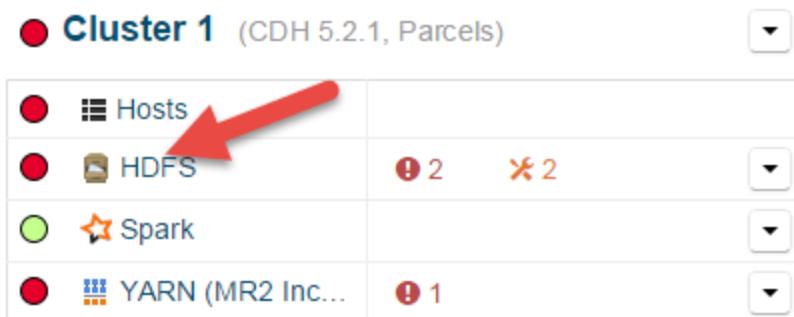
- change the replication factor in HDFS
- create directories in HDFS
- move files from the Linux file system to HDFS
- run a file system health check and look at the Block Scanner report
- explore the HDFS web GUIs

Following the end of the last lab, you should be at the EC2 VM command prompt as user 'ec2-user':



```
ec2-user@ip-10-0-72-36:~ [ec2-user@ip-10-0-72-36 ~]$
```

HDFS occasionally shows as in bad health b/c of under-replicated blocks. You can see this error if you click on the HDFS service and on the resulting page look under the Health Tests panel:



### Cloudera Management Service



Cloudera Mana...	Red (4 errors)	▼
------------------	----------------	---

## Health Tests [Expand All](#)

▼ ● 3 bad.	
● NameNode summary: ip-10-0-72-36.us-west-2.compute.internal (Availability: Active, Health: Bad). This health test reflects the health of the active NameNode.	<a href="#">Details</a>
● 1 under replicated blocks in the cluster. 1 total blocks in the cluster. Percentage under replicated blocks: 100.00%. Critical threshold: 40.00%.	<a href="#">Details</a>
● Healthy DataNode: 0. Concerning DataNode: 0. Total DataNode: 1. Percent healthy: 0.00%. Percent healthy or concerning: 0.00%. Critical threshold: 90.00%.	<a href="#">Details</a>
► ○ 4 good.	

But just ignore this issue for now, we'll revisit it later.

First, we will do a few housekeeping tasks to get our 1-node cluster ready for business.

The default replication factor for HDFS is 3 and Cloudera Manager sets this as the default even though we have a 1-node cluster. Let's change the replication factor down to 1, since that makes more sense for our setup.

Note, that the replication factor change from 3 down to 1 will not affect the pre-existing files in HDFS, but only any new files that are added. The CDH5 install added about 11 files to HDFS and those files will remain with replication factor 3. We will change the replication factor for THOSE files later.

Under the HDFS service page, click on Configuration:

The screenshot shows the Cloudera Manager interface for Cluster 1. The top navigation bar includes Home, Clusters, Hosts, Diagnostics, Audits, Charts, Administration, and a search bar. Below the navigation is a timeline showing data from 30 minutes preceding October 27, 2014, at 4:35 AM UTC. The main content area is titled 'HDFS' and contains tabs for Status, Instances, Configuration (which is highlighted with a red arrow), Commands, Audits, and Charts. The 'HDFS Summary' section displays configured capacity (5.8 GB/35.4 GB) and quick links to NameNode Web UI (Active). The 'Status Summary' section shows SecondaryNameNode, NameNode, and Balancer status. To the right are two charts: 'HDFS Capacity' and 'Total Bytes Read Across DataNodes'. The 'HDFS Capacity' chart shows a blue bar for Configured Capacity (35.4G) and a green bar for Non-HDFS Used (5.7G) against a background of 18.6G.

Here you will see 1 validation check and 1 validation warning.

Click on the validation warning:

The screenshot shows the 'Configuration' page for Cluster 1. The top navigation bar includes Status, Instances, Configuration (selected), Commands, Audits, and Charts. The main content area is titled 'Configuration' and features a search bar. A yellow banner at the top states '⚠ 1 validation warning below.' with a red arrow pointing to it. Below the banner is a table with three columns: Category, Property, and Value. The 'Category' column has two items: 'Service-Wide' and 'Balancer Default Group'. The 'Property' column lists 'ZooKeeper Service' for both categories. The 'Value' column shows 'ZooKeeper' for Service-Wide and 'none' for Balancer Default Group. Radio buttons are used to select the values.

Category	Property	Value
▶ Service-Wide	ZooKeeper Service	<input checked="" type="radio"/> ZooKeeper <input type="radio"/> none
▶ Balancer Default Group		

You'll see that Cloudera Manager recommends at least 1 GB for the HDFS NameNode JVM, but b/c of low memory conditions, the NN has been restricted to using only 391 MB or so. This is actually fine for our lab's purposes. Notice in the warning (in the screenshot) that 1 GB of RAM supports 1 million HDFS blocks. We will not be even close to this in our lab.

Configuration

Switch to the new layout

**⚠ 1 validation warning below. (Remove Filter)**

Category	Property	Value	Description
NameNode Default Group / Resource Management	Java Heap Size of Namenode in Bytes	391 MiB	Maximum size in bytes for the Java Process heap memory. Passed to Java -Xmx. Java Heap Size of Namenode in Bytes is recommended to be at least 1GB for every million HDFS blocks. Suggested minimum value: 1073741824

Save Changes

Just so we're aware, take a look at how much free memory (in GB) and disk space is left on the server after the CM + CDH installations:

**Switch to the CMD line and:**

```
[ec2-user@ip-10-0-72-36 ~]$ free -g
              total        used        free      shared  buffers   cached
Mem:          14           10          4          0          0          6
-/+ buffers/cache:       3           11
Swap:          0           0           0
```

```
[ec2-user@ip-10-0-72-36 ~]$ df -Th
Filesystem  Type  Size  Used Avail Use% Mounted on
/dev/xvda1  ext4  40G  8.5G  29G  23% /
tmpfs       tmpfs  7.3G    0  7.3G   0% /dev/shm
cm_processes tmpfs  7.3G  2.9M  7.3G   1%
```

**Switch back to the web UI and:**

Click **Remove Filter** on the validation warning:

The screenshot shows the HDFS Configuration page. At the top, there are tabs: HDFS, Status, Instances, Configuration (which is selected), and Commands. Below the tabs is a search bar containing "filterWARNING" with a clear button (an 'x'). A red arrow points to this clear button. Underneath the search bar, a yellow header bar displays the message "⚠ 1 validation warning below. (Remove Filter)". The main table has columns: Category, Property, and Value. One row is visible: "NameNode Default Group / Java Heap Size of" with a value of "2".

In the left pane, expand **Service-Wide** and click on **Replication**:

The screenshot shows the Configuration page. At the top, there are tabs: HDFS, Status, Instances, Configuration (selected), Commands, Audits, and Charts. Below the tabs is a search bar with a clear button. To the right of the search bar are buttons for Role Groups and a message icon. A red arrow points to the "Role Groups" button. The left sidebar contains a tree view with categories: Service-Wide (expanded), Advanced, Cloudera Navigator, High Availability, Logs, Monitoring, Performance, Ports and Addresses, Proxy, Replication (highlighted with a red arrow), and Security. The main table has columns: Category, Property, Value, and Description. It lists three properties under the Replication category: "Replication Factor" (dfs.replication) with a value of "3" (circled in green), "Minimal Block Replication" (dfs.replication.min, dfs.namenode.replication.min) with a value of "1", and "Maximal Block Replication" (dfs.replication.max) with a value of "512".

You'll notice that the `dfs.replication` is set to 3. But since we have only one node, we need to reduce this to 1. Click on 3 and change the value to 1.

below.			
	Property	Value	Description
	Replication Factor dfs.replication	<input type="text" value="1"/> <a href="#">Reset to the default value: 3</a>	Default block replication. The number of replications to make when the file is created. The default value is used if a replication number is not specified.
	Minimal Block Replication dfs.replication.min,	1 default value	The minimal block replication.

Then click on **Save Changes** at the top of the screen:

The screenshot shows the Cloudera Manager interface. At the top, there is a navigation bar with tabs like 'Administration', a notification icon with '7' notifications, a search bar, and a user 'admin'. Below the navigation bar is a toolbar with a red circular icon and an 'Actions' dropdown. The main content area has a pink header bar with a red circular icon and the text 'Switch to the new layout'. Below this is a table with columns 'Role Groups' and 'Notes'. A red arrow points to a blue 'Save Changes' button. The table rows contain configuration details for 'dfs.replication' and 'dfs.replication.min'. The bottom part of the screen shows a detailed description of the 'dfs.replication' setting.

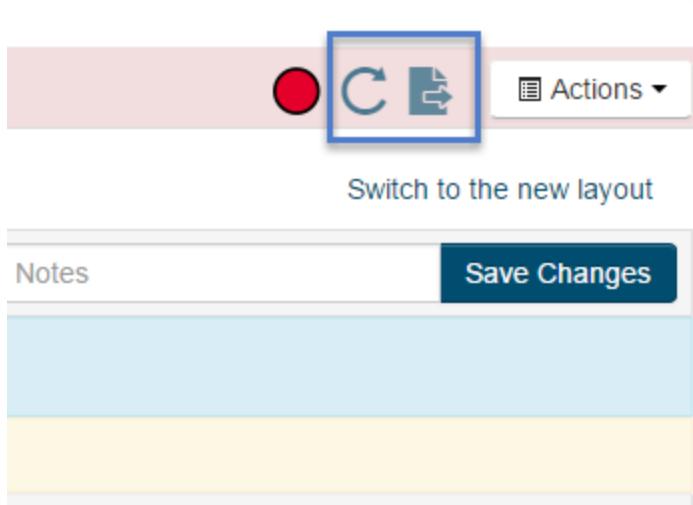
Role Groups	Notes
1	Default block replication. The number of replications to make when the file is created. The default value is used if a replication number is not specified. <a href="#">value: 3</a>

Saving the changes only applies it to Cloudera Manager's settings (stored in PostgreSQL) but not to the underlying Hadoop XML configuration files. We have to Deploy Client Configuration for the entire cluster and restart the HDFS service... then the new setting will take effect.

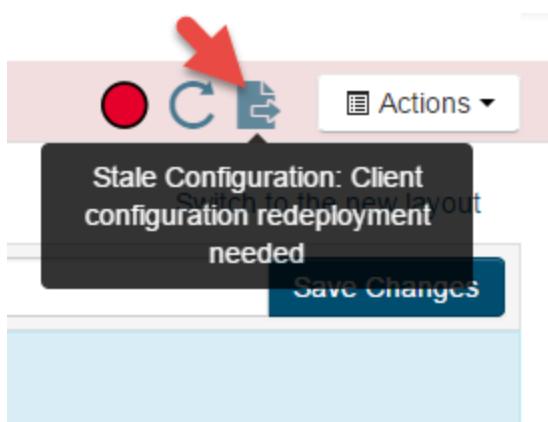
According to Cloudera: “Although Cloudera Manager will deploy client configuration files automatically in many cases, if you have modified the configurations for a service, you may need to redeploy those configuration files.”

Later in this lab, we will open the hdfs-site.xml file to verify that the replication factor is indeed set to 1.

Notice that a couple of new icons have appeared to deploy client configuration and restart the HDFS service:



**Click the Deploy Client Configuration button:**



On the next page, click **Restart Cluster**:

cloudera manager

Home Clusters Hosts Diagnostics Audits Charts Administration

## Cluster 1 Stale Configurations

After reviewing changes, invoke the [Restart Cluster](#) wizard to propagate changes to all roles, redeploy client configurations, and restart the cluster.

### Review Changes

Filter by All Files ▾ HDFS ▾ All Roles ▾ [Remove Filter](#)

File: hadoop-conf/hdfs-site.xml

```
... @@ -23,9 +23,9 @@
23   23     <value>cdh5-cm-vm01:50070</value>
24   24   </property>
25   25   <property>
26   26     <name>dfs.replication</name>
27 - 27     <value>3</value>
27 + 27     <value>1</value>
28   28   </property>
29   29   <property>
30   30     <name>dfs.blocksize</name>
31   31     <value>134217728</value>
```

File: hdfs-site.xml

```
... @@ -19,9 +19,9 @@
19   19     <value>supergroup</value>
20   20   </property>
21   21   <property>
```

Click Restart Now:

Cloudera Manager

Home Clusters Hosts Diagnostics Audits Charts Administration

## Cluster 1 Stale Configurations

### Restart Cluster

Rolling Restart Not Available  
All services running with outdated configurations in the cluster and their dependencies will be restarted.

Client Configuration  Re-deploy client configuration

Back 1 2 3 4 Restart Now

You will now see the Progress screen. This will take about 2 - 3 mins:

### Cluster 1 Stale Configurations

Progress

Command	Context	Status	Started at	Ended at
Restart	Cluster 1	In Progress	Oct 27, 2014 5:16:51 AM UTC	

Child Commands

( All) ( Failed Only) ( Active Only)

Command (Child commands)	Context	Status	Started at	Ended at
Stop (1)	Cluster 1	In Progress	Oct 27, 2014 5:16:51 AM UTC	

When it completes, click on Finish:

### Cluster 1 Stale Configurations

Progress

Command	Context	Status	Started at	Ended at
Restart	Cluster 1	Finished	Oct 27, 2014 5:16:51 AM UTC	Oct 27, 2014 5:23:32 AM UTC

All services successfully restarted.

Child Commands

( All) ( Failed Only) ( Active Only)

Command (Child commands)	Context	Status	Started at	Ended at
Deploy Client Configuration (5)	Cluster 1	Finished	Oct 27, 2014 5:22:07 AM UTC	Oct 27, 2014 5:23:32 AM UTC

Successfully deployed all client configurations.

Command (Child commands)	Context	Status	Started at	Ended at
Start (5)	Cluster 1	Finished	Oct 27, 2014 5:18:24 AM UTC	Oct 27, 2014 5:22:07 AM UTC

All services successfully started.

Command (Child commands)	Context	Status	Started at	Ended at
Stop (5)	Cluster 1	Finished	Oct 27, 2014 5:16:51 AM UTC	Oct 27, 2014 5:18:24 AM UTC

All services successfully stopped.

1 2 3 4 → **Finish**

The HDFS health issue will still remain as we have not changed the replication factor for the EXISTING files, only for FUTURE files. We will change the replication factor for existing files soon.

● Cluster 1 (CDH 5.2.1, Parcels)

● Hosts	
● HDFS	! 1 ✘ 2
● Spark	
● YARN (MR2 Inc...)	

## Cloudera Management Service

● C Cloudera Mana...	✗ 4
----------------------	-----

(The YARN ResourceManager service may alternate between green and red health at this time)

Switch to the cmd line and:

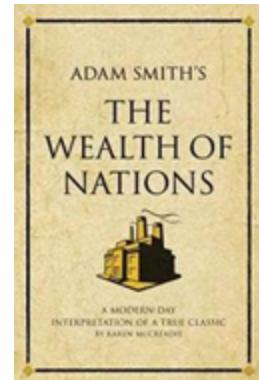
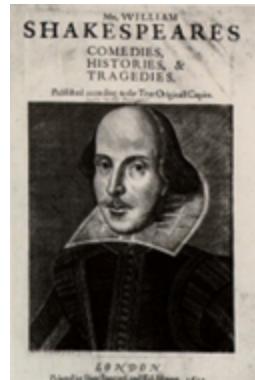
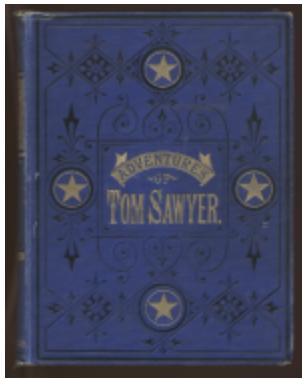
All three HDFS daemons (NameNode / DataNode/ SecondaryNameNode) start under the ‘hdfs’ user. To list all processes started under ‘hdfs’, run this command:

```
[ec2-user@ip-10-0-72-36 ~]$ ps U hdfs
 PID TTY      STAT   TIME COMMAND
32390 ?          S1      0:11 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_namenode -Xmx1000m -Dhdfs.audit.logger=INFO,
32436 ?          S1      0:08 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_datanode -Xmx1000m -Dhdfs.audit.logger=INFO,
32492 ?          S1      0:06 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_secondarynamenode -Xmx1000m -Dhdfs.audit.log
```

You will notice that these 3 daemons were started with the “java –D command” as JVMs. The –Xmx1000m option means that these JVMs were started with a max heap size of about 1 GB. However, the initial heap size is not necessarily that large.

Let's start by downloading the following three books from Project Gutenberg:

- **The Adventures of Tom Sawyer** by Mark Twain
- **The Complete Works of William Shakespeare** by William Shakespeare
- **An Inquiry into the Nature and Causes of the Wealth of Nations** by Adam Smith



Wget is free software for retrieving files using HTTP or FTP. We will use Wget to download these three files to the Rackspace VM. In the next section we will move the 3 files from the Rackspace VM's /ext3 file system to HDFS (which is coincidentally running on the same node and ultimately also persisting its data to /ext3)

You should now be in the `/home/ec2-user` folder:

```
[ec2-user@ip-10-0-72-36 ~]$ pwd  
/home/ec2-user
```

Create a new folder to store the eBooks:

```
[ec2-user@ip-10-0-72-36 ~]$ mkdir ebooks
```

```
[ec2-user@ip-10-0-72-36 ~]$ ls  
cloudera-manager-installer.bin  ebooks
```

```
[ec2-user@ip-10-0-72-36 ~]$ cd ebooks
```

**Download the three eBooks:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ wget  
http://blueplastic.com/hadoop/tom\_sawyer.txt  
--2012-08-27 06:58:43-- http://blueplastic.com/hadoop/tom_sawyer.txt  
Resolving blueplastic.com... 74.220.207.68  
Connecting to blueplastic.com|74.220.207.68|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 416148 (406K) [text/plain]  
Saving to: âtom_sawyer.txtâ  
  
100%[=====] 416,148  
1009K/s in 0.4s  
  
2012-08-27 06:58:44 (1009 KB/s) - âtom_sawyer.txtâ
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ wget  
http://blueplastic.com/hadoop/shakespeare.txt  
--2012-08-27 06:59:17-- http://blueplastic.com/hadoop/shakespeare.txt  
Resolving blueplastic.com... 74.220.207.68  
Connecting to blueplastic.com|74.220.207.68|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 5590193 (5.3M) [text/plain]  
Saving to: âshakespeare.txtâ  
  
100%[=====] 5,590,193 1.66M/s  
in 3.2s  
  
2012-08-27 06:59:20 (1.66 MB/s) - âshakespeare.txtâ
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ wget  
http://blueplastic.com/hadoop/wealth\_of\_nations.txt  
--2012-08-27 06:59:34-- http://blueplastic.com/hadoop/wealth_of_nations.txt  
Resolving blueplastic.com... 74.220.207.68  
Connecting to blueplastic.com|74.220.207.68|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2276935 (2.2M) [text/plain]  
Saving to: âwealth_of_nations.txtâ  
  
100%[=====] 2,276,935 1.55M/s  
in 1.4s  
  
2012-08-27 06:59:36 (1.55 MB/s) - âwealth_of_nations.txtâ
```

**Take a glance at the first 10 lines of one of the books:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ ls  
shakespeare.txt  tom_sawyer.txt  wealth_of_nations.txt
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ head -10 shakespeare.txt  
ii»{The Project Gutenberg EBook of The Complete Works of William Shakespeare,  
by  
William Shakespeare
```

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at [www.gutenberg.org](http://www.gutenberg.org)

```
** This is a COPYRIGHTED Project Gutenberg eBook, Details Below **  
** Please follow the copyright guidelines in this file. **
```

**That's just the license part of the file. Try running `head -200 shakespeare.txt` to see some of the actual book contents.**

These files are basically unstructured data, because it's just a blob of words.

**The `wc` command will tell you how many `lines`, `words` and `bytes` are in a file:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ wc shakespeare.txt  
124796 904087 5590193 shakespeare.txt
```



Finally, let's get deeper into HDFS.

All Hadoop commands, including HDFS and MapReduce, are run as an option under the 'hadoop' command. **To see the full list of options:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop
Usage: hadoop [--config confdir] COMMAND
      where COMMAND is one of:
        fs                  run a generic filesystem user client
        version            print the version
        jar <jar>          run a jar file
        checknative [-a|-h] check native hadoop and compression libraries
        availability
        distcp <srcurl> <desturl> copy file or directories recursively
        archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop
        archive
        classpath          prints the class path needed to get the
                           Hadoop jar and the required libraries
        daemonlog          get/set the log level for each daemon
        or
        CLASSNAME         run the class named CLASSNAME
```

Most commands print help when invoked w/o parameters.

We will specifically use the 'fs' option to run most of our HDFS commands.

**To see all of the Hadoop Filesystem commands:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs
Usage: hadoop fs [generic options]
  [-cat [-ignoreCrc] <src> ...]
  [-chgrp [-R] GROUP PATH...]
  [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
  [-chown [-R] [OWNER][:[GROUP]] PATH...]
  [-copyFromLocal <localsrc> ... <dst>]
  [-copyToLocal [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] <path> ...]
  [-cp <src> ... <dst>]
  [-df [-h] [<path> ...]]
  [-du [-s] [-h] <path> ...]
  [-expunge]
  [-get [-ignoreCrc] [-crc] <src> ... <localdst>]
  [-getmerge [-nl] <src> <localdst>]
  [-help [cmd ...]]
  [-ls [-d] [-h] [-R] [<path> ...]]
  [-mkdir [-p] <path> ...]
  [-moveFromLocal <localsrc> ... <dst>]
  [-moveToLocal <src> <localdst>]
  [-mv <src> ... <dst>]
  [-put <localsrc> ... <dst>]
    [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
  [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
  [-setrep [-R] [-w] <rep> <path/file> ...]
  [-stat [format] <path> ...]
  [-tail [-f] <file>]
  [-test -[ezd] <path>]
  [-text [-ignoreCrc] <src> ...]
  [-touchz <path> ...]
  [-usage [cmd ...]]
```

There really aren't all that many HDFS commands. If you have a linux administration background, these should mostly look familiar.

**Let's begin by doing a listing of the root HDFS folder:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls /
Found 2 items
drwxrwxrwt  - hdfs supergroup          0 2014-12-03 16:06 /tmp
drwxr-xr-x  - hdfs supergroup          0 2014-12-03 16:05 /user
```

We can see that there are two directories or folders here. Note that the above command is NOT the same as the linux 'ls' command. It doesn't matter which local directory you are in when running the above "hadoop fs -ls" command as it will always contact the NameNode to get the directory metadata for display at the command prompt. The "hadoop fs -ls" is different from the linux 'ls' command which runs on the local file system's current directory.

**Next, let's take a peek inside the /user folder:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls /user
Found 2 items
drwxrwxrwx  - mapred hadoop          0 2014-12-03 16:04 /user/history
drwxr-x--x  - spark   spark          0 2014-12-03 16:05 /user/spark
```

**If you don't specify a specific directory to list, it will default to /user/<current user> and in this case, the current linux user is 'root':**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls
ls: `.' : No such file or directory
```

This error was to be expected, since the /user/ec2-user folder does not exist under /user in HDFS as we saw in a command above.

**So, let's create a /user/root folder and retry the default -ls command:**

**(Also, note that we're running this command as the hdfs user b/c this is the superuser for HDFS, not the local 'root' user)**

```
[ec2-user@ip-10-0-72-36 ebooks]$ sudo -u hdfs hadoop fs -mkdir /user/ec2-user
[root@cdh4-cm-vm01 ebooks]$ hadoop fs -ls
```

No output from the `-ls` command means that at least the folder exists now, but there's nothing in it. From now on, if you don't provide an absolute path (starting with `/`), a relative path will be assumed under `/user/ec2-user/`.

**Check the permissions for the /user/root folder in HDFS:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls /user
Found 3 items
drwxr-xr-x  - hdfs  supergroup          0 2014-12-03 16:31 /user/ec2-user
drwxrwxrwx  - mapred hadoop           0 2014-12-03 16:04 /user/history
drwxr-x--x  - spark   spark            0 2014-12-03 16:05 /user/spark
```

The owner of the `/user/ec2-user/` folder is the user `hdfs` and the group is `supergroup`.

**Change the owner of /user/root to be root:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ sudo -u hdfs hadoop fs -chown ec2-user
/user/ec2-user
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls /user
Found 3 items
drwxr-xr-x  - ec2-user supergroup          0 2014-12-03 16:31 /user/ec2-user
drwxrwxrwx  - mapred   hadoop           0 2014-12-03 16:04 /user/history
drwxr-x--x  - spark     spark            0 2014-12-03 16:05 /user/spark
```

**Next, we will create a folder in HDFS to store these three eBooks.** This folder will be used as the input for a MapReduce job in a future lab:

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -mkdir gutenberg_input  
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls /user/ec2-user  
Found 1 items  
drwxr-xr-x - root supergroup 0 2013-03-26 23:58  
/user/root/gutenberg_input
```

Notice how in the above mkdir command since you didn't provide a full path starting with /, it automatically appended /user/ec2-user/ to the beginning of 'gutenberg\_input'.

A full list of the HDFS commands can be found here:

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

We will explore a handful of them in this lab.

**Make sure you place all the backslashes in the commands below!**

The copyFromLocal command copies files from the Linux Filesystem to HDFS (Note the case Sensitivity for the copyFromLocal command):

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -copyFromLocal tom_sawyer.txt  
/user/ec2-user/gutenberg_input/
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -copyFromLocal shakespeare.txt  
/user/ec2-user/gutenberg_input/
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -copyFromLocal  
wealth_of_nations.txt /user/ec2-user/gutenberg_input/
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls gutenberg_input  
Found 3 items  
-rw-r--r-- 1 ec2-user supergroup 5590193 2014-12-03 16:36  
gutenberg_input/shakespeare.txt  
-rw-r--r-- 1 ec2-user supergroup 416148 2014-12-03 16:35  
gutenberg_input/tom_sawyer.txt  
-rw-r--r-- 1 ec2-user supergroup 2276935 2014-12-03 16:36  
gutenberg_input/wealth_of_nations.txt
```

You will notice in the above ls command that if the absolute path is not given with /, then /user/ec2-user is assumed.

**To see the size for all three files in human readable form:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -du -h gutenberg_input
5.3m    gutenberg_input/shakespeare.txt
406.4k   gutenberg_input/tom_sawyer.txt
2.2m    gutenberg_input/wealth_of_nations.txt
```

**We can also count the # of files in the gutenberg\_input directory:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -count gutenberg_input
1           3           8283276 gutenberg_input
```

The output columns show: DIR\_COUNT, FILE\_COUNT, CONTENT\_SIZE, DIR\_NAME

**And finally to display the end of the shakespeare.txt file from HDFS:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -tail
gutenberg_input/shakespeare.txt
<output not shown>
```

Time to look into some important HDFS commands and settings.

Let's explore the deletion and trash features of the command line. Remember that if you delete a file at the command line, it will actually move to the trash folder in HDFS. Then you have to expunge trash to really delete the file.

**Let's create an empty file with touchz and then delete it with rm:**

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -touchz zerofile.txt

[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls
Found 2 items
drwxr-xr-x  - ec2-user supergroup          0 2014-12-03 16:36
gutenberg_input
-rw-r--r--  1 ec2-user supergroup          0 2014-12-03 16:37 zerofile.txt
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -rm zerofile.txt
14/12/03 16:37:51 INFO fs.TrashPolicyDefault: Namenode trash configuration:
Deletion interval = 1440 minutes, Emptier interval = 0 minutes.
```

Moved:

```
'hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/ec2-user/zerofile.txt' to trash at:  
hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/ec2-user/.Trash/Current
```

So, looks like Trash was enabled by default in this CDH5 installation. Therefore deleting anything from the CMD-line via the rm command will just move the file to trash. If you REALLY want to delete the file, you could have provided the -skipTrash option after -rm. Or you can empty/expunge the trash folder manually after the -rm command. In our lab environment, the fs.trash.interval setting for HDFS is 1 day, so if we don't expunge trash this file should be deleted at this time tomorrow.

#### Verify that the zerofile really is in the hidden .Trash folder:

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls  
Found 2 items  
drwx-----  - ec2-user supergroup          0 2014-12-03 16:37 .Trash  
drwxr-xr-x   - ec2-user supergroup          0 2014-12-03 16:36  
gutenberg_input
```

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls  
/user/ec2-user/.Trash/Current/user/ec2-user/  
Found 1 items  
-rw-r--r--   1 ec2-user supergroup          0 2014-12-03 16:37  
/user/ec2-user/.Trash/Current/user/ec2-user/zerofile.txt
```

#### Now delete everything in Trash:

```
[ec2-user@ip-10-0-72-36 ~]$ hadoop fs -expunge  
13/01/29 06:18:19 INFO fs.TrashPolicyDefault: Created trash checkpoint:  
/user/root/.Trash/1301290619
```

#### Hit the up arrow twice to verify that the file really got deleted:

```
[ec2-user@ip-10-0-72-36 ebooks]$ hadoop fs -ls  
/user/ec2-user/.Trash/Current/user/ec2-user/  
ls: `/user/ec2-user/.Trash/Current/user/ec2-user/': No such file or directory
```

Great, the .Trash directory is gone.

Let's move on to some other common HDFS commands...

**To check the health of the HDFS Filesystem at root:**

```
[root@cdh5-cm-vm01 ebooks]# sudo -u hdfs hdfs fsck /
14/10/27 05:50:40 WARN ssl.FileBasedKeyStoresFactory: The property
'ssl.client.truststore.location' has not been set, no TrustStore will be
loaded
Connecting to namenode via http://cdh5-cm-vm01:50070
FSCK started by hdfs (auth:SIMPLE) from /104.130.159.101 for path / at Mon
Oct 27 05:50:41 UTC 2014
.
/usr/spark/share/lib/spark-assembly.jar: Under replicated
BP-747346709-10.0.72.36-1417640622514:blk_1073741825_1001. Target Replicas is
3 but found 1 replica(s).
```

```
Status: HEALTHY
Total size: 103854959 B
Total dirs: 18
Total files: 5
Total symlinks: 0
Total blocks (validated): 4 (avg. block size 25963739 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 1 (25.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 2 (33.333332 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Wed Dec 03 16:40:26 EST 2014 in 8 milliseconds
```

The filesystem under path '/' is HEALTHY

You will notice that there are currently 5 files in HDFS and the block replication factor is 1 (as opposed to 3 in production clusters). There is 1 under replicated blocks and 2 missing replicas. 3 out of the 14 files are the Project Gutenberg eBooks that we uploaded.

It is possible to use the HDFS setrep command to change the replica factor for all pre-existing files from 3 down to 1. Let's try that:

```
[ec2-user@ip-10-0-72-36 ebooks]$ sudo -u hdfs hadoop fs -setrep -R -w 1 /
Replication 1 set:
/usr/ec2-user/.Trash/141203163939/user/ec2-user/zerofile.txt
Replication 1 set: /user/ec2-user/gutenberg_input/shakespeare.txt
Replication 1 set: /user/ec2-user/gutenberg_input/tom_sawyer.txt
Replication 1 set: /user/ec2-user/gutenberg_input/wealth_of_nations.txt
Replication 1 set: /user/spark/share/lib/spark-assembly.jar
Waiting for /user/ec2-user/.Trash/141203163939/user/ec2-user/zerofile.txt ...
done
Waiting for /user/ec2-user/gutenberg_input/shakespeare.txt ... done
Waiting for /user/ec2-user/gutenberg_input/tom_sawyer.txt ... done
Waiting for /user/ec2-user/gutenberg_input/wealth_of_nations.txt ... done
Waiting for /user/spark/share/lib/spark-assembly.jar ... done
```

Now, check the health status of HDFS again:

```
[ec2-user@ip-10-0-72-36 ebooks]$ sudo -u hdfs hdfs fsck /
14/10/27 05:56:31 WARN ssl.FileBasedKeyStoresFactory: The property
'ssl.client.truststore.location' has not been set, no TrustStore will be
loaded
Connecting to namenode via
http://ip-10-0-72-36.us-west-2.compute.internal:50070
FSCK started by hdfs (auth:SIMPLE) from /10.0.72.36 for path / at Wed Dec 03
16:44:40 EST 2014
.....Status: HEALTHY
Total size: 103854959 B
Total dirs: 18
Total files: 5
Total symlinks: 0
Total blocks (validated): 4 (avg. block size 25963739 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
```

Number of racks: 1  
FSCK ended at Wed Dec 03 16:44:40 EST 2014 in 5 milliseconds

The filesystem under path '/' is HEALTHY

There we go.... a healthier cluster. That red annoying HDFS Bad Health warning in the Cloudera Manager GUI will not go away yet (as a few configurations warnings may remain). HOWEVER, we are now running our cluster in a very sensitive environment! If even one of these blocks gets corrupted because of some guy walking around with a large magnet in the Amazon datacenter, then the repercussions could mean data loss for us. Cross your fingers and continue the lab...

**Home** Status All Health Issues ! 3 All Configura

---

Cluster 1 (CDH 5.2.1, Parcels) ▾

● Hosts	
● HDFS	! 1 ✖ 2
● Spark	
● YARN (MR2 Inc...	! 1

Cloudera Management Service

● C Cloudera Mana...	✖ 4
----------------------	-----

The default replication factor setting is stored in the following file:  
`/etc/hadoop/conf/hdfs-site.xml`

Let's open that file to see its contents, specifically the default replication factor and the location for where HDFS stored the 3 Project Gutenberg files in the Linux file system.

You can use either nano, vi, vim or emacs to open the XML file and all future files. If you are unfamiliar with the arcane vi/vim or emacs syntax, I recommend using nano, one of the simplest text editors to use on Linux. My preference is vi, so you will see me opening all files with vi for the rest of the labs, but feel free to replace the word 'vi' with 'nano' or 'emacs' on your end.

If you want a 3 min crash course in vim, go to this link and graduate levels 1 and 2 and then come back: <http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/>

Going back to our original goal of reviewing the XML file, let's do that now. Choose of the following two commands (or use emacs):

If using VI:

```
[ec2-user@ip-10-0-72-36 ebooks]$ vi /etc/hadoop/conf/hdfs-site.xml
```

If using nano:

```
[ec2-user@ip-10-0-72-36 ebooks]$ nano /etc/hadoop/conf/hdfs-site.xml
```

(Note, when you want to exit nano, hit CTRL + X and choose y or n if prompted to save the file)

On line 27 of the file, or about 6 settings down, you can see that the default replication factor is set to 1:

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
```

The setting is not marked as 'final' so a developer writing via the HDFS API can choose to overwrite this setting to something higher for a specific file. It is the Hadoop administrator's job to decide which settings should be final and untamperable, and which should be allowed to be overwritten by a developer.

In this file, you will also notice the dfs.blocksize set to 134217728 bytes (128 MB) on line 31. This is the default block size in Hadoop, and it can also be overwritten individually for files.

Cloudera Manager makes it easy to update these settings via the CM web UI and then just Deploy Client Configuration to all the nodes and restart the affected service. If you were using Hadoop straight from apache.org, you would have to use something like rsync to update all of the XML files if you wanted to change the replication factor and then maybe a bash script to restart the HDFS daemons across the cluster nodes.

Go ahead and **exit out of vim or nano** and don't save any changes. (In VI, you can do this by hitting **<ESC>**, then type **:q!**)

Another interesting directory to look at is `/dfs/dn`. Note that in the Cloudera Manager GUI, `/dfs/dn` is the location of the DataNode Data Directory:

The screenshot shows the Cloudera Manager HDFS Configuration page. The 'DataNode Default Group' is selected in the left sidebar. In the main table, the 'Property' column shows 'DataNode Data Directory' and the 'Value' column shows '/dfs/dn'. A green circle highlights the value input field. The 'Description' column provides details about the property.

Category	Property	Value	Description
► Service-Wide	dfs.data.dir, dfs.datanode.data.dir	/dfs/dn	Comma-delimited list of directories on the local file system where the DataNode stores HDFS block data. Typical values are /data/N/dfs/dn for N = 1, 2, 3.. These directories should be mounted using the noatime
► Balancer Default Group			
▼ DataNode Default Group			
Advanced			
Logs			
Monitoring			

**There is the directory in ext3 where HDFS stores its blocks. The following commands track down the blocks:**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs ls /dfs/dn/current/
BP-747346709-10.0.72.36-1417640622514 VERSION
```

**WARNING: In the command below, you will have to replace the # with whatever unique # appears in your machine!**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs ls
/dfs/dn/current/BP-747346709-10.0.72.36-1417640622514/current/finalized/subdir0/subdir0
blk_1073741825          blk_1073741855          blk_1073741856
blk_1073741857
blk_1073741825_1001.meta blk_1073741855_1031.meta blk_1073741856_1032.meta
blk_1073741857_1033.meta
```

You will notice that each block file has a corresponding meta file with the checksum and generation stamp for the block. There can also be many sub directories named 'subdir#', depending on the amount of files in HDFS.

At this point though, it is hard to tell which block # corresponds to which file in HDFS.

How would we find the block that corresponds to the Tom Sawyer file? Continue on to the next section to find out.

Next, let's try to figure out what the block ID is for the ebook tom\_sawyer.txt:

The Tom Sawyer ebook is just half a MB in size, so it will be able to fit on 1 block. It will not take up the entire 128 MB max block size and instead the block will be < 1 MB in size and the underlying file in ext3 will also just be a < 1 MB in size.

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs hdfs fsck  
/user/ec2-user/gutenberg_input/tom_sawyer.txt -files -blocks -racks  
14/12/03 20:03:03 WARN ssl.FileBasedKeyStoresFactory: The property  
'ssl.client.truststore.location' has not been set, no TrustStore will be  
loaded  
Connecting to namenode via  
http://ip-10-0-72-36.us-west-2.compute.internal:50070  
FSCK started by hdfs (auth:SIMPLE) from /10.0.72.36 for path  
/user/ec2-user/gutenberg_input/tom_sawyer.txt at Wed Dec 03 20:03:04 EST 2014  
/user/ec2-user/gutenberg_input/tom_sawyer.txt 416148 bytes, 1 block(s): OK  
0. BP-747346709-10.0.72.36-1417640622514:blk_1073741855_1031 len=416148  
rep=1 [/default/10.0.72.36:50010]
```

```
Status: HEALTHY  
Total size: 416148 B  
Total dirs: 0  
Total files: 1  
Total symlinks: 0  
Total blocks (validated): 1 (avg. block size 416148 B)  
Minimally replicated blocks: 1 (100.0 %)  
Over-replicated blocks: 0 (0.0 %)  
Under-replicated blocks: 0 (0.0 %)  
Mis-replicated blocks: 0 (0.0 %)  
Default replication factor: 1  
Average block replication: 1.0  
Corrupt blocks: 0  
Missing replicas: 0 (0.0 %)
```

```
Number of data-nodes:          1
Number of racks:              1
FSCK ended at Wed Dec  3 20:03:04 EST 2014 in 1 milliseconds
```

The filesystem under path '/user/ec2-user/gutenberg\_input/tom\_sawyer.txt' is  
HEALTHY

In the above output, we can see this file is made of one block with block ID:  
blk\_1073741855\_1031

Well, the actual block id is just the # preceding the underscore, so: blk\_1073741855

Note, the block ID will be different for your instance of the block.

While at the linux command line, under the finalized folder, run this command to search for the Tom Sawyer block:

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u root find / -name "blk_1073741855"
/dfs/dn/current/BP-747346709-10.0.72.36-1417640622514/current/finalized/subdir0/subdir0/blk_1073741855
```

So, this is file in ext3 representing the Tom Sawyer block. You can verify this by opening the block/file like this:

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs vi
/dfs/dn/current/BP-747346709-10.0.72.36-1417640622514/current/finalized/subdir0/subdir0/blk_1073741855
```

*[output of file not shown]*

**The NameNode stores its file system metadata files under /dfs/nn. I know this because that is how it is configured in the Cloudera Manager GUI**

Cloudera Manager Home Clusters ▾ Hosts Diagnostics ▾ Audits Charts ▾ Administration ▾ Search (Hotkey: /) ×

Cluster 1 >

HDFS Status Instances Configuration Commands Audits Charts ▾

Configuration

Search Role Groups Notes

⚠ 1 validation warning below.

Category	Property	Value	Description
▶ Service-Wide	NameNode Data Directories dfs.name.dir, dfs.namenode.name.dir	/dfs/nn	Deterr local fi Name name redund comm direct name direct are /d N=1..3
▶ Balancer Default Group		<a href="#">Reset to empty default value</a>	
▶ DataNode Default Group			
▶ Failover Controller Default Group			
▶ Gateway Default Group			
▶ HttpFS Default Group			
▶ JournalNode Default Group			
▶ NFS Gateway Default Group			
▶ <b>NameNode Default Group</b>	NameNode Edits Directories dfs.namenode.edits.dir	Default value is empty. Click to edit.	Direct system Name
▶ SecondaryNameNode Default Group			

Let's explore that now.

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs ls /dfs/nn/current
 edits_00000000000000000001-00000000000000000106
 edits_0000000000000000000107-00000000000000000200
 edits_0000000000000000000201-00000000000000000588
 edits_000000000000000000589-00000000000000000944
 edits_000000000000000000945-00000000000000001300
 edits_00000000000000001301-00000000000000001656
 edits_00000000000000001657-00000000000000002018
 edits_00000000000000002019-00000000000000002374
 edits_00000000000000002375-00000000000000002724
 edits_00000000000000002725-00000000000000003080
 edits_00000000000000003081-00000000000000003436
 edits_00000000000000003437-00000000000000003792
 edits_00000000000000003793-00000000000000004148
 edits_00000000000000004149-00000000000000004504
 edits_00000000000000004505-00000000000000004860
 edits_00000000000000004861-00000000000000005216
 edits_00000000000000005217-00000000000000005572
 edits_00000000000000005573-00000000000000005928
 edits_inprogress_00000000000000005929
 fsimage_00000000000000005572
 fsimage_00000000000000005572.md5
```

```
fsimage_00000000000000005928
fsimage_00000000000000005928.md5
seen_txid
VERSION
```

These Journal (edits) and Checkpoint (fsimage) files are the most critical files to backup in the cluster. Typically multiple directories are defined in Cloudera Manager for the NN Data Directory so that Hadoop automatically writes them to multiple locations (or disks). One of the locations should ideally be a remote NAS.

**Hadoop dfsadmin is a useful command to see a report of basic statistics for the file system:**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs hdfs dfsadmin -report
Configured Capacity: 38048189645 (35.44 GB)
Present Capacity: 30903599104 (28.78 GB)
DFS Remaining: 30798868480 (28.68 GB)
DFS Used: 104730624 (99.88 MB)
DFS Used%: 0.34%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
```

-----  
Live datanodes (1):

```
Name: 10.0.72.36:50010 (ip-10-0-72-36.us-west-2.compute.internal)
Hostname: ip-10-0-72-36.us-west-2.compute.internal
Rack: /default
Decommission Status : Normal
Configured Capacity: 38048189645 (35.44 GB)
DFS Used: 104730624 (99.88 MB)
Non DFS Used: 7144590541 (6.65 GB)
DFS Remaining: 30798868480 (28.68 GB)
DFS Used%: 0.28%
DFS Remaining%: 80.95%
Configured Cache Capacity: 1505755136 (1.40 GB)
Cache Used: 0 (0 B)
Cache Remaining: 1505755136 (1.40 GB)
Cache Used%: 0.00%
Cache Remaining%: 100.00%
```

Xceivers: 2

Last contact: Thu Dec 04 08:47:27 EST 2014

**Apache Hadoop comes with several web GUIs to monitor the cluster.**

To access the HDFS web GUI, visit the following URL: <ip address>:50070

The screenshot shows a web browser window titled 'Namenode information' with the URL '104.130.159.101:50070/dfshealth.html#tab-overview'. The page has a dark blue header with tabs for 'Hadoop', 'Overview', 'Datanodes', 'Snapshot', 'Startup Progress', and 'Utilities'. The 'Overview' tab is currently active. Below the header is a large section titled 'Overview 'cdh5-cm-vm01:8020' (active)'. Underneath this title is a table with five rows of cluster information:

<b>Started:</b>	Mon Oct 27 05:18:43 UTC 2014
<b>Version:</b>	2.5.0-cdh5.2.0, re1f20a08bde76a33b79df026d00a0c91b2298387
<b>Compiled:</b>	2014-10-11T21:00Z by jenkins from Unknown
<b>Cluster ID:</b>	cluster14
<b>Block Pool ID:</b>	BP-1545055139-104.130.159.101-1414383262813

## Overview 'cdh5-cm-vm01:8020' (active)

<b>Started:</b>	Mon Oct 27 05:18:43 UTC 2014
<b>Version:</b>	2.5.0-cdh5.2.0, re1f20a08bde76a33b79df026d00a0c91b2298387
<b>Compiled:</b>	2014-10-11T21:00Z by jenkins from Unknown
<b>Cluster ID:</b>	cluster14
<b>Block Pool ID:</b>	BP-1545055139-104.130.159.101-1414383262813

## Summary

Security is off.

Safemode is off.

190 files and directories, 15 blocks = 205 total filesystem object(s).

Heap Memory used 28.39 MB of 278.44 MB Heap Memory. Max Heap Memory is 278.44 MB.

Non Heap Memory used 41.5 MB of 41.94 MB Committed Non Heap Memory. Max Non Heap Memory is 130 MB.

<b>Configured Capacity:</b>	35.43 GB
<b>DFS Used:</b>	100.07 MB
<b>Non DFS Used:</b>	5.73 GB

Take a few minutes to browse around the page and explore the various links.

A screenshot of the Databricks web interface. At the top, there's a header bar with a search field containing ".html#tab-overview" and several icons. Below the header is a navigation bar with tabs for "Snapshot", "Startup Progress", and "Utilities". A red arrow points from the text below to the "Utilities" tab. A dropdown menu is open under "Utilities", showing options like "Browse the file system" and "Logs".

vm01:8020' (active)

## Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxrwxrwt	hdfs	supergroup	0 B	0	0 B	tmp
drwxr-xr-x	hdfs	supergroup	0 B	0	0 B	user

Try navigating to the Project Gutenberg files on your own:

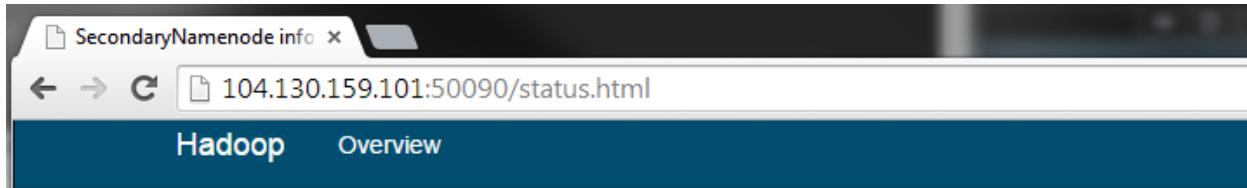
## Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	ec2-user	supergroup	5.33 MB	1	128 MB	shakespeare.txt
-rw-r--r--	ec2-user	supergroup	406.39 KB	1	128 MB	tom_sawyer.txt
-rw-r--r--	ec2-user	supergroup	2.17 MB	1	128 MB	wealth_of_nations.txt

You can see the Secondary NameNode status and last checkpoint taken time with the following URL:

<http://<ip address>:50090>



## Overview

<b>Version</b>	2.5.0-cdh5.2.0
<b>Compiled</b>	2014-10-11T21:00Z by jenkins from Unknown
<b>NameNode Address</b>	cdh5-cm-vm01:8022
<b>Started</b>	10/26/2014, 10:18:41 PM
<b>Last Checkpoint</b>	12/31/1969, 6:15:35 PM
<b>Checkpoint Period</b>	3600 seconds
<b>Checkpoint Transactions</b>	1000000

### Checkpoint Image URI

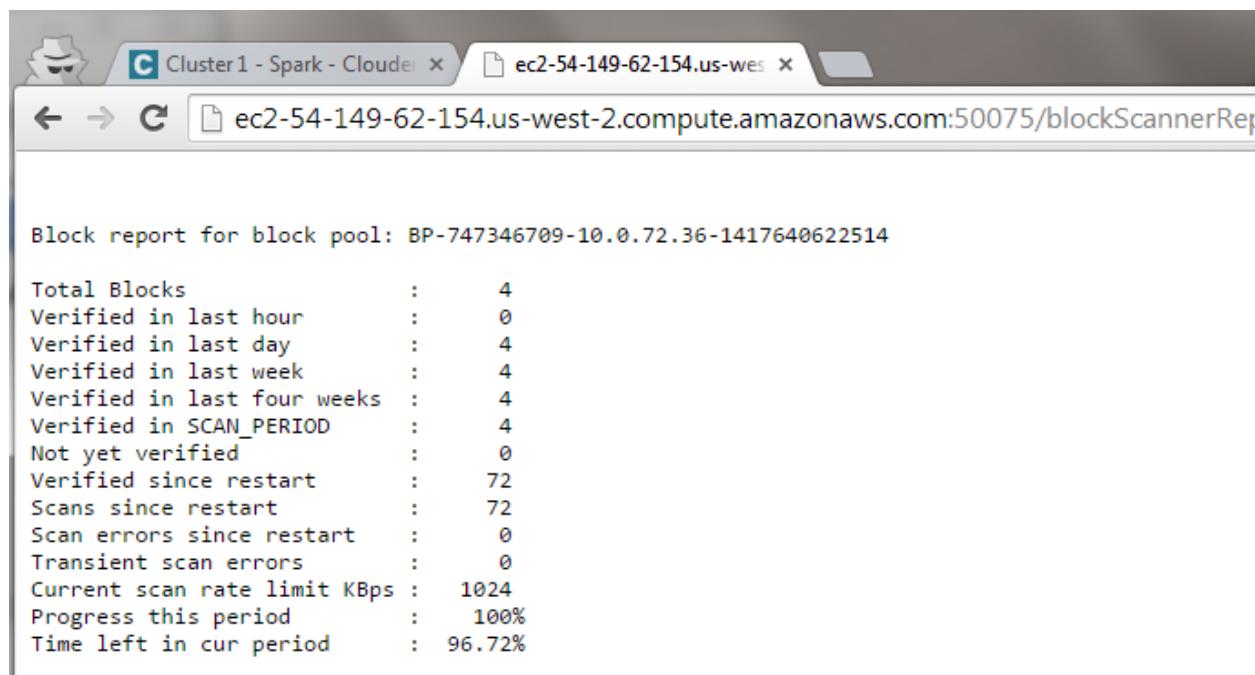
- file:///dfs/snn

### Checkpoint Editlog URI

- file:///dfs/snn

You can see the Block Scanner Report with the following URL:

<http://<ip address>:50075/blockScannerReport>



The screenshot shows a web browser window with the following details:

- Tab bar: Cluster1 - Spark - Cloud, ec2-54-149-62-154.us-west-2.compute.amazonaws.com
- Address bar: ec2-54-149-62-154.us-west-2.compute.amazonaws.com:50075/blockScannerReport
- Content area:

```
Block report for block pool: BP-747346709-10.0.72.36-1417640622514

Total Blocks : 4
Verified in last hour : 0
Verified in last day : 4
Verified in last week : 4
Verified in last four weeks : 4
Verified in SCAN_PERIOD : 4
Not yet verified : 0
Verified since restart : 72
Scans since restart : 72
Scan errors since restart : 0
Transient scan errors : 0
Current scan rate limit KBps : 1024
Progress this period : 100%
Time left in cur period : 96.72%
```

Finally you can see all of the blocks in the HDFS file system with:

**<http://<ip address>:50075/blockScannerReport?listblocks>**

```
Block report for block pool: BP-747346709-10.0.72.36-1417640622514
blk_1073741825_1001      : status : ok    type : local  scan time : 2002871      1969-12-31 19:33:22,871
blk_1073741855_1031      : status : ok    type : local  scan time : 3708062      1969-12-31 20:01:48,062
blk_1073741856_1032      : status : ok    type : local  scan time : 3717104      1969-12-31 20:01:57,104
blk_1073741857_1033      : status : ok    type : local  scan time : 3724119      1969-12-31 20:02:04,119

Total Blocks          :     4
Verified in last hour :     0
Verified in last day   :     4
Verified in last week  :     4
Verified in last four weeks :     4
Verified in SCAN_PERIOD :     4
Not yet verified       :     0
Verified since restart :    72
Scans since restart   :    72
Scan errors since restart :     0
Transient scan errors :     0
Current scan rate limit KBps : 1024
Progress this period   : 100%
Time left in cur period : 96.72%
```

The columns in the first four lines are: Block ID / status / scan type / scan time.

Great! You have successfully completed the HDFS cmd-line lab.

To continue learning about HDFS on your own time, I recommend the following links:

Read Chapter 8 of “The Architecture of Open Source Applications” for a great high-level overview of how HDFS works:

<http://www.aosabook.org/en/hdfs.html>

Brad Hedlund of Dell has an excellent blog post with diagrams explaining HDFS:

<http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/>

I have a 30 min YouTube video with a high-level overview of HDFS:

<https://www.youtube.com/watch?v=ziqx2hJY8Hg>

Here is a page with all of the HDFS settings in the respective XML file:

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

And here is the official Apache Hadoop docs website:

<https://hadoop.apache.org/docs/current/>



## Spark and YARN

Let's explore YARN at a high level first.

Take a look at the YARN resource manager UI:

<http://<ip address>:8088>

A screenshot of a web browser displaying the YARN Resource Manager UI. The title bar shows "Cluster 1 - YARN (MR2 Inc.)" and "All Applications". The main content area is titled "All Applications" with a "hadoop" logo. On the left, there is a sidebar with a tree view under "Cluster" and a "Scheduler" section. The "Scheduler" section lists application states: NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED. Below the sidebar is a "Tools" section. The main area contains two tables: "Cluster Metrics" and "User Metrics for dr.who". Both tables have several columns including Apps Submitted, Apps Pending, Apps Running, Apps Completed, Containers Running, Memory Used, Memory Total, Memory Reserved, VCores Used, VCores Total, VCores Reserved, Active Nodes, and Dead Nodes. The "User Metrics" table also includes columns for Containers Pending and Containers Reserved. At the bottom, there is a table header for "Applications" with columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, and Final Status. A message "No data available in table" is displayed. A footer message "Showing 0 to 0 of 0 entries" is also present.

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status
No data available in table								

There should be no applications showing up as we have not submitted or started the Spark application within YARN yet.

Take a look at the YARN history manager UI:

<http://<ip address>:19888>

The screenshot shows a web browser window titled "JobHistory". The left sidebar has a "hadoop" logo and links for "Application", "About", "Jobs", and "Tools". The main area is titled "Retired Jobs" and contains a table header with columns: Submit Time, Start Time, Finish Time, Job ID, Name, User, Queue, State, Maps Total, Maps Completed, Reduces Total, and Reduces Completed. Below the header, a message says "No data available in table". At the bottom, there are search and filter fields for "Submit Ti", "Start Tim", "Finish Tin", "Job ID", "Name", "User", "Queue", "State", "Maps Total", "Maps Comp", "Reduces", and "Reduces Co". A footer shows "Showing 0 to 0 of 0 entries" and navigation links "First", "Previous", "Next", and "Last".

Okay, nothing there either.

Don't confuse the YARN history server with the Spark history server, which has its own UI on a different port:

<http://<ip address>:18088>

The screenshot shows a web browser window titled "History Server". The left sidebar has a "Spark" logo and links for "All Applications" and "JobHistory". The main area displays the text "Event Log Location: hdfs://cdh5-cm-vm01:8020/user/spark/applicationHistory" and "No Completed Applications Found".

Next, let's take a look at some Spark configuration files:

```
[ec2-user@ip-10-0-72-36 ~]$ cat  
/etc/spark/conf.cloudera.spark_on_yarn/spark-defaults.conf  
spark.eventLog.dir=hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/  
spark/applicationHistory  
spark.eventLog.enabled=true  
spark.yarn.historyServer.address=http://ip-10-0-72-36.us-west-2.compute.inter  
nal:18088
```

**The above file is where you can add additional settings such as (however, do not add this line at this time):**

```
spark.executor.memory      2g
```

**Here are the rest of the Spark configuration files:**

```
[ec2-user@ip-10-0-72-36 ~]$ ls /etc/spark/conf.cloudera.spark_on_yarn/  
log4j.properties  spark-defaults.conf  spark-env.sh
```

```
[ec2-user@ip-10-0-72-36 ~]$ cat  
/etc/spark/conf.cloudera.spark_on_yarn/spark-env.sh  
#!/usr/bin/env bash  
##  
# Generated by Cloudera Manager and should not be modified directly  
##  
  
export SPARK_HOME=/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/lib/spark  
export  
DEFAULT_HADOOP_HOME=/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/lib/hado  
op  
  
### Path of Spark assembly jar in HDFS  
export  
SPARK_JAR_HDFS_PATH=${SPARK_JAR_HDFS_PATH:-/user/spark/share/lib/spark-assemb  
ly.jar}  
  
### Let's run everything with JVM runtime, instead of Scala  
export SPARK_LAUNCH_WITH_SCALA=0  
export SPARK_LIBRARY_PATH=${SPARK_HOME}/lib  
export SCALA_LIBRARY_PATH=${SPARK_HOME}/lib  
  
export HADOOP_HOME=${HADOOP_HOME:-$DEFAULT_HADOOP_HOME}  
  
if [ -n "$HADOOP_HOME" ]; then
```

```
export SPARK_LIBRARY_PATH=$SPARK_LIBRARY_PATH:${HADOOP_HOME}/lib/native  
fi  
  
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-/etc/hadoop/conf}
```

**We should now make a change in the log4j file to reduce the level of logging for Spark, or else we will be overwhelmed by too many INFO messages. You may use VI or NANO to do this.**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo vi  
/etc/spark/conf.cloudera.spark_on_yarn/log4j.properties
```

**Then change line #2 from INFO to ERROR:**

```
# Set everything to be logged to the console  
log4j.rootCategory=ERROR, console  
log4j.appender.console=org.apache.log4j.ConsoleAppender  
log4j.appender.console.target=System.err  
log4j.appender.console.layout=org.apache.log4j.PatternLayout  
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p  
%c{1}: %m%n  
  
# Settings to quiet third party logs that are too verbose  
log4j.logger.org.eclipse.jetty=WARN  
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO  
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
```

**Save & Quit out of the file.**

**This is the Spark user's directory in HDFS:**

```
[ec2-user@ip-10-0-72-36 ~]$ sudo -u hdfs hadoop fs -ls /user/spark  
Found 2 items  
drwxrwxrwt - spark spark 0 2014-10-27 04:21  
/user/spark/applicationHistory  
drwxr-xr-x - spark spark 0 2014-10-27 04:21 /user/spark/share
```

**Local machine logs for Spark are here:**

```
[ec2-user@ip-10-0-72-36 ~]$ cd /var/log/spark/
```

```
[ec2-user@ip-10-0-72-36 spark]# ls
```

```
spark-history-server-ip-10-0-72-36.us-west-2.compute.internal.log
```

```
[ec2-user@ip-10-0-72-36 spark]# tail -3  
spark-history-server-ip-10-0-72-36.us-west-2.compute.internal.log  
2014-10-27 05:21:50,298 INFO org.eclipse.jetty.server.AbstractConnector:  
Started SelectChannelConnector@0.0.0.0:18088  
2014-10-27 05:21:50,303 INFO org.apache.spark.util.Utils: Successfully  
started service on port 18088.  
2014-10-27 05:21:50,313 INFO org.apache.spark.deploy.history.HistoryServer:  
Started HistoryServer at http://cdh5-cm-vm01:18088
```

**Finally, let's actually do something interesting with Spark. How about a WordCount job?**

**Go to the linux tmp dir and download animals.txt:**

```
[ec2-user@ip-10-0-72-36 spark]# cd /tmp  
  
[ec2-user@ip-10-0-72-36 tmp]# wget http://blueplastic.com/hadoop/animals.txt  
--2014-10-27 06:37:46-- http://blueplastic.com/hadoop/animals.txt  
Resolving blueplastic.com... 74.220.207.68  
Connecting to blueplastic.com|74.220.207.68|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 44 [text/plain]  
Saving to: "animals.txt"  
  
100%[=====] 44  
---.K/s in 0s  
  
2014-10-27 06:37:46 (3.87 MB/s) - "animals.txt" saved [44/44]
```

```
[ec2-user@ip-10-0-72-36 tmp]$ cat animals.txt  
cat cat  
dog dog  
fish fish fish fish  
fly
```

**Copy the file to the Spark users' folder in HDFS:**

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u spark hadoop fs -copyFromLocal  
animals.txt /user/spark/
```

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u hdfs hadoop fs -ls /user/spark
Found 3 items
-rw-r--r-- 1 spark spark          44 2014-10-27 06:41
/user/spark/animals.txt
drwxrwxrwt - spark spark          0 2014-10-27 04:21
/user/spark/applicationHistory
drwxr-xr-x - spark spark          0 2014-10-27 04:21 /user/spark/share
```

**Start the Spark scala shell:**

```
[ec2-user@ip-10-0-72-36 tmp]$ spark-shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
[jar:file:/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/jars/spark-assembl
y-1.1.0-cdh5.2.0-hadoop2.5.0-cdh5.2.0.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: Found binding in
[jar:file:/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/jars/slf4j-log4j12
-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an
explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to
```

```
   _/ _/_ _ _ _ _ _ / _/_/
  _\ \_ \_ \_ \_ ` / _/_ ' _/
 /__/_ . __/\_, _/_ / / \_ \_ version 1.1.0
 /_/_
```

```
Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc.
```

```
scala>
```

**Ignore the binding messages at the beginning.**

**At port 4040, you will now see the Spark Stages UI for this shell application that is currently running in the cmd prompt:**

<http://<ip address>:4040>

The screenshot shows a web browser window with four tabs: 'Cluster 1 - Spark - Cloud', 'All Applications', 'JobHistory', and 'Spark shell - Spark Stages'. The 'Spark shell - Spark Stages' tab is active. The URL in the address bar is '104.130.159.101:4040/stages/'. The page title is 'Spark Stages'. It displays the following statistics:  
Total Duration: 1.7 min  
Scheduling Mode: FIFO  
Active Stages: 0  
Completed Stages: 0  
Failed Stages: 0

Below these stats, there are three sections: 'Active Stages (0)', 'Completed Stages (0)', and 'Failed Stages (0)', each with a table header row.

**Notice that no stages have been submitted yet.**

**Let's run through some Scala commands. Most of the commands we're running should be self-explanatory.**

```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@287a3e19
```

At a high level, every Spark application consists of a driver program that launches various parallel operations on a cluster. The driver program contains your application's main function and defines distributed datasets on the cluster, then applies operations to them. In the example above, the driver program was the Spark shell itself, and you could just type in the operations you wanted to run.

Driver programs access Spark through a `SparkContext` object, which represents a connection to a computing cluster. In the shell, a `SparkContext` is automatically created for you, as the variable called `sc`.

**The help command shows common commands:**

```
scala> :help
```

All commands can be abbreviated, e.g. :he instead of :help.

Those marked with a \* have more detailed help, e.g. :help imports.

:cp <path>	add a jar or directory to the classpath
:help [command]	print this summary or command-specific help
:history [num]	show the history (optional num is commands to show)
:h? <string>	search the history
:imports [name name ...]	show import history, identifying sources of names
:implicits [-v]	show the implicits in scope
:javap <path class>	disassemble a file or class name
:load <path>	load and interpret a Scala file
:paste	enter paste mode: all input up to ctrl-D compiled together
:quit	exit the repl
:replay	reset execution and replay all previous commands
:reset	reset the repl to its initial state, forgetting all session entries
:sh <command line>	run a shell command (result is implicitly => List[String])
:silent	disable/enable automatic printing of results
:fallback	disable/enable advanced repl changes, these fix some issues but may introduce others.
This mode will be removed once these fixes stabilize	
:type [-v] <expr>	display the type of an expression without evaluating it
:warnings	show the suppressed warnings from the most recent line which had any

Try to create a base RDD from an incorrect animals file and count the # of lines in it:

```
scala> val fileBaseRDD = sc.textFile("/user/spark/animals-fake.txt")
```

```
fileBaseRDD: org.apache.spark.rdd.RDD[String] = /user/spark/animals.txt
```

```
MappedRDD[1] at textFile at <console>:12
```

```
scala> fileBaseRDD.count()
```

```
org.apache.hadoop.mapred.InvalidInputException: Input path does not exist:  
hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/spark/animals-fake.  
txt  
        at  
org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFo  
rmat.java:285)  
        at  
org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:228)  
        at  
org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:313)  
        at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:180)
```

**Woah, we got an error! Notice that this is basically a file not found error. More specifically, HDFS was not found at localhost. Notice that we did not get a file missing error until we called an ACTION on the base RDD.**

Let's fix this by giving the full path for HDFS:

```
scala> val fileBaseRDD = sc.textFile("hdfs://<YOUR PUBLIC  
hostname>:8020/user/spark/animals.txt")  
fileBaseRDD: org.apache.spark.rdd.RDD[String] =  
hdfs://ec2-54-149-62-154.us-west-2.compute.amazonaws.com:8020/user/spark/anim  
als.txt MappedRDD[5] at textFile at <console>:12  
  
scala> fileBaseRDD.count()  
res2: Long = 4  
  
scala> fileBaseRDD.collect()  
res4: Array[String] = Array(cat cat, dog dog, fish fish fish fish, fly)  
  
scala> fileBaseRDD.partitions.length  
res5: Int = 2  
  
scala> val counts = fileBaseRDD.flatMap(line => line.split(" ")).map(word =>  
(word,1)).reduceByKey(_ + _)  
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[6] at  
reduceByKey at <console>:14  
  
scala> counts.collect()  
res6: Array[(String, Int)] = Array((fish,5), (dog,2), (cat,2), (fly,1))
```

```
scala> counts.saveAsTextFile("hdfs://<YOUR PUBLIC  
hostname>:8020/user/spark/wc-results.txt")  
org.apache.hadoop.security.AccessControlException: Permission denied:  
user=root, access=WRITE, inode="/user/spark":spark:spark:drwxr-x--x  
at  
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.checkFsPe  
rmission(DefaultAuthorizationProvider.java:255)  
at  
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.check(Def  
aultAuthorizationProvider.java:236)
```

Hmm, another error. It seems we have started the Spark shell as the `ec2-user`, but are trying to write to HDFS into the Spark users' dir, which gives us an error.

**Try writing the results to the `ec2-user`'s dir:**

```
scala> counts.saveAsTextFile("hdfs://<YOUR PUBLIC  
hostname>:8020/user/ec2-user/wc-results.txt")
```

**Switch to the browser and reload or refresh the Spark stages UI at port 4040:**

**Spark Stages**

Total Duration: 1.6 min  
Scheduling Mode: FIFO  
Active Stages: 0  
Completed Stages: 5  
Failed Stages: 0

**Active Stages (0)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
----------	-------------	-----------	----------	------------------------	-------	--------------	---------------

**Completed Stages (5)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
4	saveAsTextFile at <console>:17	+details	2014/10/27 06:57:56	2/2			
2	collect at <console>:17	+details	2014/10/27 06:57:15	2/2			
3	map at <console>:14	+details	2014/10/27 06:57:15	2/2	44.0 B		369.0 B
1	collect at <console>:15	+details	2014/10/27 06:56:59	2/2	44.0 B		
0	count at <console>:15	+details	2014/10/27 06:56:55	2/2	44.0 B		

**Failed Stages (0)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write	Failure Reason
----------	-------------	-----------	----------	------------------------	-------	--------------	---------------	----------------

**Notice all of the completed stages!**

If you click on the Storage tab, you will not see anything since we have not persisted anything to memory yet:

**Storage**

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk
----------	---------------	-------------------	-----------------	----------------	-----------------	--------------

**Check out the Environment tab:**

The screenshot shows the Databricks web interface at the URL [104.130.159.101:4040/environment/](http://104.130.159.101:4040/environment/). The top navigation bar includes links for Spark, Stages, Storage, Environment (which is highlighted with a red arrow), and Executors. To the right of the navigation are icons for Databricks, App, and Help. The main content area is titled "Environment" and contains two sections: "Runtime Information" and "Spark Properties".

**Runtime Information**

Name	Value
Java Home	/usr/java/jdk1.7.0_67-cloudera/jre
Java Version	1.7.0_67 (Oracle Corporation)
Scala Version	version 2.10.4

**Spark Properties**

Name	Value
spark.app.name	Spark shell
spark.driver.host	cdh5-cm-vm01
spark.driver.port	53596
spark.eventLog.dir	hdfs://cdh5-cm-vm01:8020/user/spark/applicationHistory
spark.eventLog.enabled	true
spark.filesServer.uri	<a href="http://104.130.159.101:40943">http://104.130.159.101:40943</a>
spark.jars	
spark.master	local[*]

And the Executors' tab:

A screenshot of the Spark UI Executors page. At the top, there is a navigation bar with tabs: Spark (selected), Stages, Storage, Environment, Executors (highlighted with a red arrow), and Spark shell app. Below the navigation bar, the title "Executors (1)" is displayed. A summary section shows "Memory: 0.0 B Used (265.4 MB Total)" and "Disk: 0.0 B Used". The main content is a table with 14 columns: Executor ID, Address, RDD Blocks, Memory Used, Disk Used, Active Tasks, Failed Tasks, Complete Tasks, Total Tasks, Task Time, Input, Shuffle Read, and Shuffle Write. There is one row in the table, corresponding to the driver node.

Executor ID	Address	RDD Blocks	Memory Used	Disk Used	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time	Input	Shuffle Read	Shuffle Write
<driver>	cdh5-cm-vm01:46368	0	0.0 B / 265.4 MB	0 B	0	0	10	10	717 ms	132.0 B	0.0 B	369.0 B

Next, go ahead and quit out of the Scala spark shell:

```
scala> :q  
Stopping spark context.
```

The Stages UI will not stop loading as we have terminated the Spark application (which was the scala shell in this case):

A screenshot of the Spark UI Stages page. At the top, there is a navigation bar with tabs: Spark (selected), Stages (highlighted with a red arrow), Storage, Environment, Executors, and Spark shell app. Below the navigation bar, the title "Stages" is displayed. The main content area shows a large error message: "This webpage is not available". There are two buttons at the bottom: "Reload" and "Details".

If you are interested, you can explore more of the Scala transformations and actions by reading this page:

<https://spark.apache.org/docs/1.1.0/programming-guide.html#transformations>

**Let's take a look at the results in HDFS:**

```
[ec2-user@ip-10-0-72-36 tmp]$ hadoop fs -ls /user/ec2-user/
Found 3 items
drwx-----  ec2-user supergroup          0 2014-12-03 16:39
/user/ec2-user/.Trash
drwxr-xr-x  ec2-user supergroup          0 2014-12-03 16:36
/user/ec2-user/gutenberg_input
drwxr-xr-x  ec2-user supergroup          0 2014-12-04 09:19
/user/ec2-user/wc-results.txt
```

**Notice that wc-results.txt is actually a directory! So in the future, keep in mind that when you save some RDD results to HDFS, you are actually giving a dir location, not a final output file name!**

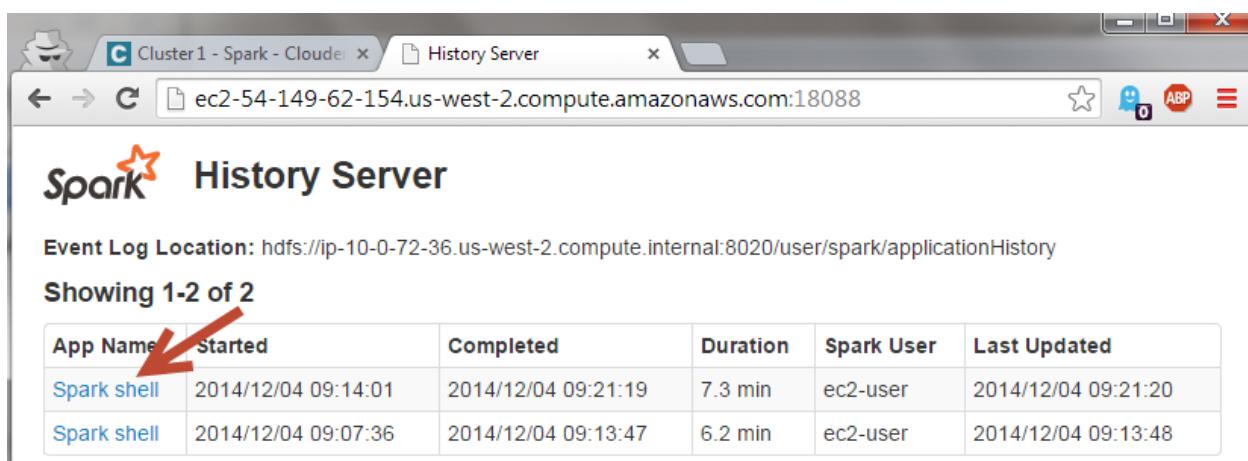
```
[ec2-user@ip-10-0-72-36 tmp]$ hadoop fs -ls /user/root/wc-results.txt/
Found 3 items
-rw-r--r--  1 ec2-user supergroup          0 2014-12-04 09:19
/user/ec2-user/wc-results.txt/_SUCCESS
-rw-r--r--  1 ec2-user supergroup         25 2014-12-04 09:19
/user/ec2-user/wc-results.txt/part-00000
-rw-r--r--  1 ec2-user supergroup          8 2014-12-04 09:19
/user/ec2-user/wc-results.txt/part-00001
```

**Since we had to partitions in our RDD, we will have 2 results files:**

```
[ec2-user@ip-10-0-72-36 tmp]$ hadoop fs -cat
/user/ec2-user/wc-results.txt/part-00000
(fish,5)
(dog,2)
(cat,2)
```

```
[ec2-user@ip-10-0-72-36 tmp]$ hadoop fs -cat
/user/ec2-user/wc-results.txt/part-00001
(fly,1)
```

Try refreshing the Spark history server UI on **port 18088** to see the now terminated Scala Spark shell application (*note you will only see one old app, instead of two in the screenshot below!*)



The screenshot shows a web browser window with the title "Cluster1 - Spark - CloudBees Jenkins" and the tab "History Server". The URL in the address bar is "ec2-54-149-62-154.us-west-2.compute.amazonaws.com:18088". The main content is the "Spark History Server" interface. It displays the following information:

Event Log Location: hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/spark/applicationHistory

Showing 1-2 of 2

App Name	Started	Completed	Duration	Spark User	Last Updated
Spark shell	2014/12/04 09:14:01	2014/12/04 09:21:19	7.3 min	ec2-user	2014/12/04 09:21:20
Spark shell	2014/12/04 09:07:36	2014/12/04 09:13:47	6.2 min	ec2-user	2014/12/04 09:13:48

Try clicking on one of the “Spark shell” links in the screenshot above to see the stages of execution of that specific application.

**Spark Stages**

Scheduling Mode: FIFO  
 Active Stages: 0  
 Completed Stages: 5  
 Failed Stages: 0

Active Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Sh

Completed Stages (5)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuf Read
4	saveAsTextFile at <console>:17 +details	2014/12/04 09:19:52	0.1 s	2/2		
2	collect at <console>:17 +details	2014/12/04 09:18:01	28 ms	2/2		
3	map at <console>:14 +details	2014/12/04 09:18:01	60 ms	2/2	44.0 B	
1	collect at <console>:15 +details	2014/12/04	18 ms	2/2	44.0	

**Let's try to do something a bit more complicated. How about using the Python Spark shell and submitting Spark SQL commands from python code?**

First, let's download some data to play with. We will download the MovieLens database that the University of Minnesota has made available for free. MovieLens is a structured data set which is available in three sizes: 100,000 reviews, 1 million reviews and 10 million review. For this lab, we'll use the 1 million ratings and first move that data set to HDFS. You can browse the MovieLens webpage here:

<http://grouplens.org/datasets/movielens/>

**You should be in the /tmp directory:**

```
[ec2-user@ip-10-0-72-36 tmp]$ pwd
/tmp
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wget http://blueplastic.com/hadoop/movies.dat
--2014-10-27 07:17:14--  http://blueplastic.com/hadoop/movies.dat
Resolving blueplastic.com... 74.220.207.68
Connecting to blueplastic.com|74.220.207.68|:80... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: 163512 (160K) [text/plain]
Saving to: "movies.dat"
```

```
100%[=====>] 163,512
385K/s  in 0.4s
```

```
2014-10-27 07:17:15 (385 KB/s) - "movies.dat" saved [163512/163512]
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wget http://blueplastic.com/hadoop/ratings.dat
--2014-10-27 07:17:20--  http://blueplastic.com/hadoop/ratings.dat
Resolving blueplastic.com... 74.220.207.68
Connecting to blueplastic.com|74.220.207.68|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21593504 (21M) [text/plain]
Saving to: "ratings.dat"
```

```
83% [=====>] 
18,005,776 4.62M/s eta 2s
w100%[=====>]
21,593,504 5.40M/s in 5.1s
```

```
2014-10-27 07:17:26 (4.01 MB/s) - "ratings.dat" saved [21593504/21593504]
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wget http://blueplastic.com/hadoop/users.dat
--2014-10-27 07:17:27--  http://blueplastic.com/hadoop/users.dat
Resolving blueplastic.com... 74.220.207.68
Connecting to blueplastic.com|74.220.207.68|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 110208 (108K) [text/plain]
Saving to: "users.dat"
```

```
100%[=====>] 110,208
341K/s  in 0.3s
```

```
2014-10-27 07:17:28 (341 KB/s) - "users.dat" saved [110208/110208]
```

**Take a look at the contents of each of the 3 .dat files and do a line count on each:**

```
[ec2-user@ip-10-0-72-36 tmp]$ head -5 movies.dat
1#Toy Story (1995)#Animation|Children's|Comedy
2#Jumanji (1995)#Adventure|Children's|Fantasy
3#Grumpier Old Men (1995)#Comedy|Romance
```

```
4#Waiting to Exhale (1995)#Comedy|Drama  
5#Father of the Bride Part II (1995)#Comedy
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wc -l movies.dat  
3883 movies.dat
```

```
[ec2-user@ip-10-0-72-36 tmp]$ head -5 ratings.dat  
1#1193#5#978300760  
1#661#3#978302109  
1#914#3#978301968  
1#3408#4#978300275  
1#2355#5#978824291
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wc -l ratings.dat  
1000209 ratings.dat
```

```
[ec2-user@ip-10-0-72-36 tmp]$ head -5 users.dat  
1#F#1#10#48067  
2#M#56#16#70072  
3#M#25#15#55117  
4#M#45#7#02460  
5#M#25#20#55455
```

```
[ec2-user@ip-10-0-72-36 tmp]$ wc -l users.dat  
6040 users.dat
```

**The structure of each file is as follows:**

**movies.dat:** <movieID>#<Title>#<Genres>

**ratings.dat:** <UserID>#<MovieID>#<Rating>#<Timestamp>

**users.dat:** <UserID>#<Gender>#<Age>#<Occupation>#<ZipCode>

The movie rating is on a 5-star scale. Timestamp refers to when the rating was recorded.

**Copy the 3 files into the Spark user's dir in HDFS:**

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u spark hadoop fs -copyFromLocal  
movies.dat /user/spark/
```

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u spark hadoop fs -copyFromLocal  
users.dat /user/spark/
```

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u spark hadoop fs -copyFromLocal  
ratings.dat /user/spark/
```

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u hdfs hadoop fs -ls /user/spark  
Found 6 items  
-rw-r--r-- 1 spark spark 44 2014-10-27 06:41  
/user/spark/animals.txt  
drwxrwxrwt - spark spark 0 2014-10-27 07:13  
/user/spark/applicationHistory  
-rw-r--r-- 1 spark spark 163512 2014-10-27 07:22 /user/spark/movies.dat  
-rw-r--r-- 1 spark spark 21593504 2014-10-27 07:22  
/user/spark/ratings.dat  
drwxr-xr-x - spark spark 0 2014-10-27 04:21 /user/spark/share  
-rw-r--r-- 1 spark spark 110208 2014-10-27 07:22 /user/spark/users.dat
```

### Start the Spark Python shell as the spark user:

```
[ec2-user@ip-10-0-72-36 tmp]$ sudo -u spark pyspark  
Python 2.6.6 (r266:84292, Jan 22 2014, 09:42:36)  
[GCC 4.4.7 20120313 (Red Hat 4.4.7-4)] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in  
[jar:file:/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/jars/spark-assembl  
y-1.1.0-cdh5.2.0-hadoop2.5.0-cdh5.2.0.jar!/org/slf4j/impl/StaticLoggerBinder.  
class]  
SLF4J: Found binding in  
[jar:file:/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/jars/slf4j-log4j12  
-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an  
explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
Welcome to
```

```
 _ _ _ / _ _ _ _ _ / _ /  
 _ \ \ \ _ \ \ _ \ / _ / ' /  
 /_ / . _ / \ _ , _ / / / _ / \ _ \ version 1.1.0  
 /_ /
```

```
Using Python version 2.6.6 (r266:84292, Jan 22 2014 09:42:36)
SparkContext available as sc.
>>>
```

**Let's get right to work and play around with some Python and Spark SQL commands. See if you can figure out what each command is doing... it's pretty simple!**

```
>>> moviesBaseRDD = sc.textFile("hdfs://<your public
IP>:8020/user/spark/movies.dat")

>>> moviesBaseRDD.count()
3883

>>> moviesBaseRDD.take(5)
[u"1#Toy Story (1995)#Animation|Children's|Comedy", u"2#Jumanji
(1995)#Adventure|Children's|Fantasy", u'3#Grumpier Old Men
(1995)#Comedy|Romance', u'4#Waiting to Exhale (1995)#Comedy|Drama',
u'5#Father of the Bride Part II (1995)#Comedy']

>>> type(moviesBaseRDD.take(5))
<type 'list'>

>>> moviesPartsRDD = moviesBaseRDD.map(lambda l: l.split("#"))
```

**Now we have a list of lists (with each nested list having 3 items):**

```
>>> moviesPartsRDD.take(5)
[[u'1', u'Toy Story (1995)', u"Animation|Children's|Comedy"], [u'2',
u'Jumanji (1995)', u"Adventure|Children's|Fantasy"], [u'3', u'Grumpier Old
Men (1995)', u'Comedy|Romance'], [u'4', u'Waiting to Exhale (1995)',
u'Comedy|Drama'], [u'5', u'Father of the Bride Part II (1995)', u'Comedy']]
```

**Let's see if we can clean up the 2nd element in the list, which has both the title and the year of the movie. It would be nice to be able to separate them.**

**Now, take the 2nd element in each sub-list and split on " (" and then remove the trailing ):**

**This takes out just the 2nd item:**

```
>>> tRDD = moviesPartsRDD.map(lambda l: l[1])
>>> tRDD.take(2)
[u'Toy Story (1995)', u'Jumanji (1995)']
```

```
>>> import string
```

**Define a custom function in Python:**

```
>>> def custFunc(item):
...     item[2:2] = [item[2]]
...     item[2] = (item[1].rstrip(')'))[-4:]
...     item[1] = string.split(item[1], ' (')[0]
...     return item
...
>>> moviesPartsRDD2 = moviesPartsRDD.map(lambda l: custFunc(l))

>>> moviesPartsRDD2.take(4)
[[u'1', u'Toy Story', u'1995', u"Animation|Children's|Comedy"], [u'2',
u'Jumanji', u'1995', u"Adventure|Children's|Fantasy"], [u'3', u'Grumpier Old
Men', u'1995', u'Comedy|Romance'], [u'4', u'Waiting to Exhale', u'1995',
u'Comedy|Drama']]
```

**Interesting. Did you see how we can define a custom function in Python to separate the title from the year of the movie and apply it to the data set?**

**Let's do some Spark SQL integration now...**

```
>>> moviesRDD = moviesPartsRDD2.map(lambda m: {"movieid": m[0], "title": m[1], "year": int(m[2]), "genres": m[3]})

>>> from pyspark.sql import SQLContext, Row

>>> sqlContext = SQLContext(sc)

>>> moviesTable = sqlContext.inferSchema(moviesRDD)
/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/lib/spark/python/pyspark/sql
.py:1039: UserWarning: Using RDD of dict to inferSchema is deprecated, please
use pyspark.Row instead
    warnings.warn("Using RDD of dict to inferSchema is deprecated,"
```

**Ignore the warning and continue...**

```
>>> moviesTable.registerTempTable("movies")
```

```
>>> newFilmsRDD = sqlContext.sql("SELECT title, year FROM movies WHERE year > 1995")  
  
>>> newFilmsRDD.take(2)  
[Row(title=u'Eye for an Eye', year=1996), Row(title=u"Don't Be a Menace to South Central While Drinkin Your Juice in the Hood", year=1996)]  
  
>>> newFilmsRDD.count()  
1436L
```

Now set up the ratings table...

```
>>> ratingsBaseRDD = sc.textFile("hdfs://<Your Public IP>:8020/user/spark/ratings.dat")  
  
>>> ratingsBaseRDD.take(5)  
[u'1#1193#5#978300760', u'1#661#3#978302109', u'1#914#3#978301968',  
u'1#3408#4#978300275', u'1#2355#5#978824291']  
  
>>> ratingsPartsRDD = ratingsBaseRDD.map(lambda l: l.split("#"))  
  
>>> ratingsPartsRDD.take(5)  
[[u'1', u'1193', u'5', u'978300760'], [u'1', u'661', u'3', u'978302109'],  
[u'1', u'914', u'3', u'978301968'], [u'1', u'3408', u'4', u'978300275'],  
[u'1', u'2355', u'5', u'978824291']]  
  
>>> ratingsRDD = ratingsPartsRDD.map(lambda m: {"userid": int(m[0]),  
"movieid": int(m[1]), "rating": int(m[2]), "tstamp": m[3]})  
  
>>> ratingsTable = sqlContext.inferSchema(ratingsRDD)  
  
>>> ratingsTable.registerTempTable("ratings")
```

```
>>> myRDD = sqlContext.sql("SELECT movieid, rating FROM ratings WHERE rating > 2")  
  
>>> myRDD.take(3)  
[Row(movieid=1193, rating=5), Row(movieid=661, rating=3), Row(movieid=914, rating=3)]
```

Can you figure out what the next few commands do? Hint, we are focusing on just the Toy Story movie, which is ID = 1.

```
>>> complexRDD = sqlContext.sql("SELECT ratings.rating, COUNT(ratings.rating) FROM ratings WHERE movieid=1 GROUP BY ratings.rating")  
  
>>> complexRDD.count()  
5L  
  
>>> complexRDD.collect()  
[Row(rating=1, c1=16), Row(rating=2, c1=61), Row(rating=3, c1=345), Row(rating=4, c1=835), Row(rating=5, c1=820)]
```

To get a list of movie ratings with movie titles, we will need to join the ratings and movies tables:

```
>>> joinRDD = sqlContext.sql("SELECT ratings.userid, ratings.rating, movies.title FROM ratings JOIN movies ON (ratings.movieid = movies.movieid) LIMIT 5")  
  
>>> joinRDD.count()  
5L  
  
>>> joinRDD.collect()  
[Row(userid=2, rating=3, title=u"Soldier's Story, A"), Row(userid=2, rating=5, title=u'Amadeus'), Row(userid=2, rating=2, title=u'Thin Red Line, The'), Row(userid=2, rating=1, title=u'Get Shorty'), Row(userid=3, rating=5, title=u'Caddyshack')] 
```

Next, let's see how we can cache an RDD to memory, so we can operate on it at memory speed in the future:

```
>>> joinRDD.cache()
MapPartitionsRDD[92] at mapPartitions at SchemaRDD.scala:413
```

Actually, the RDD is not cached to memory until we call an action on it. For example, visit the Spark Storage UI and you won't see any RDD's listed:

<http://<Your public hostname>:4040/storage/>

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk

Call an action on the joinRDD:

```
>>> joinRDD.collect()
[Row(userid=2, rating=3, title=u"Soldier's Story, A"), Row(userid=2,
rating=5, title=u'Amadeus'), Row(userid=2, rating=2, title=u'Thin Red Line,
The'), Row(userid=2, rating=1, title=u'Get Shorty'), Row(userid=3, rating=5,
title=u'Caddyshack')]
```

Now the RDD is cached:

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk
74	Memory Deserialized 1x Replicated	1	100%	720.0 B	0.0 B	0.0 B

Try clicking on the RDD name in the left-most column to see more details about the RDD:

The screenshot shows the PySparkShell application UI. At the top, there are navigation icons (back, forward, search) and a URL bar displaying 'ec2-54-149-62-154.us-west-2.compute.amazonaws.com:4040/storage/rdd/?id=74'. Below the URL bar is a header with tabs: 'Spark' (selected), 'Stages', 'Storage', 'Environment', 'Executors', and 'PySparkShell application UI'. The main content area is titled 'RDD Storage Info for 74'. It displays the following statistics:

- Storage Level:** Memory Deserialized 1x Replicated
- Cached Partitions:** 1
- Total Partitions:** 1
- Memory Size:** 720.0 B
- Disk Size:** 0.0 B

A section titled 'Data Distribution on 1 Executors' follows, showing a table with one row:

Host	Memory Usage	Disk Usage
ip-10-0-72-36.us-west-2.compute.internal:55977	720.0 B (265.4 MB Remaining)	0.0 B

Below this is a section titled '1 Partitions' with a table:

Block Name	Storage Level	Size in Memory	Size on Disk	Executors
rdd_74_0	Memory Deserialized 1x Replicated	720.0 B	0.0 B	ip-10-0-72-36.us-west-2.compute.internal:55977

**Notice that the RDD is made of 1 partition and the size it is taking up in the Executor's JVM memory on-heap is 720 Bytes.**

**Move back to the Spark python shell and exit it:**

```
>>> quit()  
[ec2-user@ip-10-0-72-36 tmp]$
```

**Currently Spark is running in YARN mode in CDH 5.2.1.**

Let's take a look at the Spark configuration settings. Go to the Cloudera Manager UI and click on Spark:

The screenshot shows the Cloudera Manager Home page for Cluster 1 (CDH 5.2.1, Parcels). The left sidebar lists services: Hosts (red), HDFS (red), Spark (green with a red arrow pointing to it), and YARN (red). The right side features a 'Charts' section with a 'Cluster CPU' graph showing host CPU usage across hosts over a 1% scale from 0% to 100%. The graph shows a small peak around 09:45. Below the charts is a 'Cluster Disk IO' section.

From the Spark service page, click on Configuration:

**cloudera manager**

Home Clusters ▾ Hosts Diagnostics ▾ Audits Charts ▾ Administration ▾

Cluster 1 » 30 minutes preceding December 4

Spark Status Instances Configuration Commands Audits Charts Library

**Quick Links**

Quick Links History Server Web UI ▾ Event Search Alerts ▾ , Critical ▾ , All ▾

**Status Summary**

History Server Bad Health  
Gateway None

**Health Tests** [Collapse All](#)

**Charts** 30m 1h

Informational Events

events  
0 0.5 1  
09:45 10 AM  
Informational Events 0

Spark Status Instances Configuration Commands Audits Charts Library Actions ▾

**Configuration** Switch to the new layout

Category	Property	Value	Description
▶ Service-Wide	Spark Jar Location (HDFS)	/user/spark/share/lib/spark-assembly.jar	The location of the Spark jar in HDFS
▶ Gateway Default Group	spark_jar_hdfs_path	default value	
▶ History Server Default Group	Spark History Location (HDFS)	/user/spark/applicationHistory	The location of Spark application history logs in HDFS. Changing this value will not move existing logs to the new location.
	spark.eventLog.dir	default value	
	YARN (MR2 Included) Service	<input checked="" type="radio"/> YARN (MR2 Included) <a href="#">Reset to empty default value</a>	Name of the YARN (MR2 Included) service that this Spark service instance depends on

You can expand “Service-Wide” on the left and click on “Advanced” to see what user the Spark service runs as:

Category	Property	Value	Description
▼ Service-Wide	System User	spark default value	The user that this service's processes should run as.
Advanced	System Group	spark default value	The group that this service's processes should run as.
Monitoring	Spark Service Environment Advanced Configuration Snippet (Safety Valve)	Default value is empty. Click to edit.	For advanced use only, key-value pairs (one on each line) to be inserted into a role's environment. Applies to configurations of all roles in this service except client configuration.
► Gateway Default Group			
► History Server Default Group			

To understand what the properties here mean, check out the CDH 5.2 Spark Standalone properties page:

[http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm\\_props\\_cdh520\\_spark\\_standalone.html](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_props_cdh520_spark_standalone.html)

**Spark (Standalone) properties in CDH 5.2.0**

This is the documentation for Cloudera 5.2.x. Documentation for other versions is available at [Cloudera Documentation](#).

**Role groups:**

- Gateway Default Group
- History Server Default Group
- Master Default Group
- Service-Wide
- Worker Default Group

**Gateway Default Group**

**Advanced**

Display Name	Description	Related Name	Default Value	API Name
Deploy Directory	The directory where the client configs will be deployed	/etc/spark		client_config_root_dir
Spark (Standalone) Client Advanced Configuration Snippet (Safety Valve) for spark-conf/log4j.properties	For advanced use only, a string to be inserted into the client configuration for <code>spark-conf/log4j.properties</code>			spark-conf/log4j.properties_client_conf
Spark (Standalone) Client Advanced Configuration	For advanced use only, a string to be inserted into the client			spark-conf/spark-defaults.conf_client_config_safet

## Submitting Spark apps in YARN client and cluster mode

Next, we will use the information from the following page to submit a SparkPi application in YARN Client Mode and then YARN Cluster Mode:

[http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh\\_ig\\_running\\_spark\\_apps.html](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_ig_running_spark_apps.html)

The code for SparkPi is available here:

<https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/SparkPi.scala>

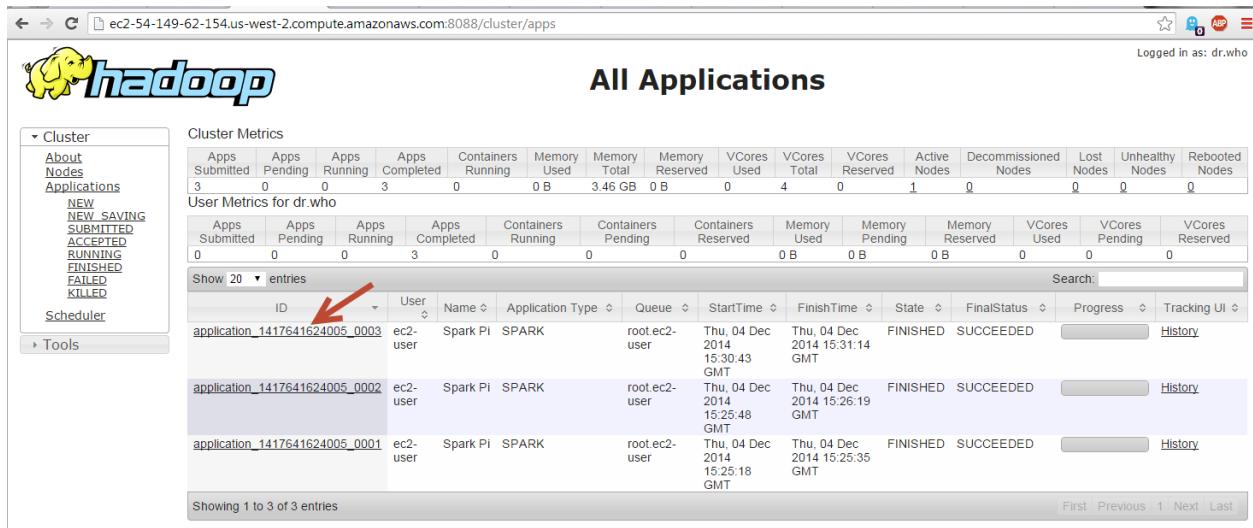
```
[ec2-user@ip-10-0-72-36 /]$ cd ~  
  
[ec2-user@ip-10-0-72-36 ~]$ spark-submit --class  
org.apache.spark.examples.SparkPi --deploy-mode client --master yarn  
/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/spark-examples-1.1.0-cd  
h5.2.1-hadoop2.5.0-cdh5.2.1.jar 10  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in  
[jar:file:/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/spark-assembl  
y-1.1.0-cdh5.2.1-hadoop2.5.0-cdh5.2.1.jar!/org/slf4j/impl/StaticLoggerBinder.  
class]  
SLF4J: Found binding in  
[jar:file:/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/slf4j-log4j12  
-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an  
explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
--args is deprecated. Use --arg instead.  
Pi is roughly 3.144032
```

Note that in client mode, the output from the Spark app is sent directly back to the cmd-line.

At this point, you can go to the YARN applications page to see the SparkPi application launched. (Note, the screenshot below shows 3 apps, but you should only see one)

Visit the URL:

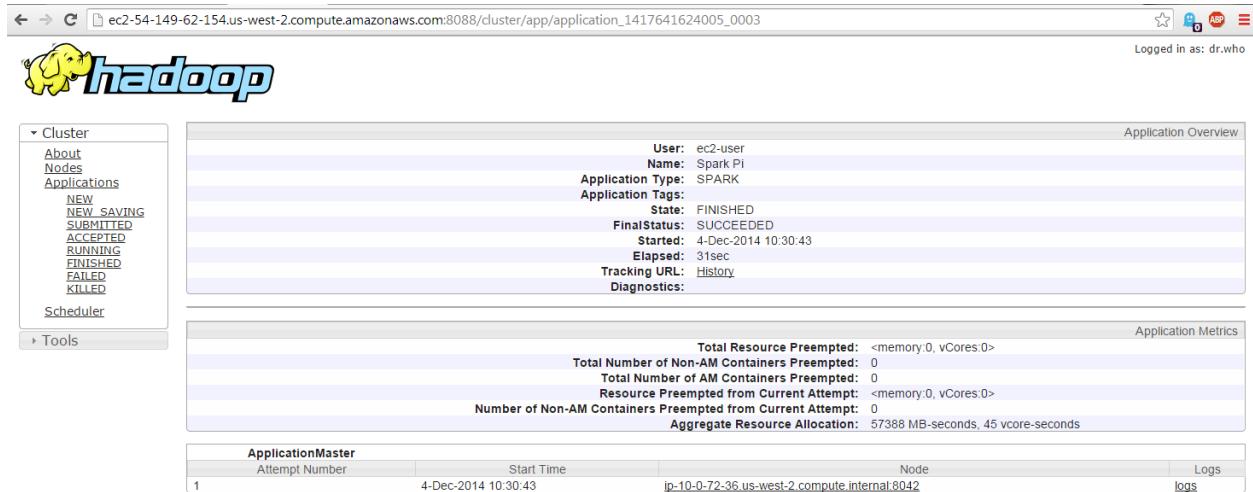
<http://<your public hostname>:8088/cluster/apps>



The screenshot shows the Hadoop YARN Applications page at <http://ec2-54-149-62-154.us-west-2.compute.amazonaws.com:8088/cluster/apps>. The page title is "All Applications". On the left, there's a sidebar with a "hadoop" logo and sections for Cluster Metrics, User Metrics for dr.who, and a list of applications. The application list shows three entries, each with details like ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking URL. A red arrow points to the first application entry: "application\_1417641624005\_0003" by user "ec2-user".

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Progress	Tracking URL
application_1417641624005_0003	ec2-user	Spark Pi	SPARK	root.ec2-user	Thu, 04 Dec 2014 15:30:43 GMT	Thu, 04 Dec 2014 15:31:14 GMT	FINISHED	SUCCEEDED		History
application_1417641624005_0002	ec2-user	Spark Pi	SPARK	root.ec2-user	Thu, 04 Dec 2014 15:25:48 GMT	Thu, 04 Dec 2014 15:26:19 GMT	FINISHED	SUCCEEDED		History
application_1417641624005_0001	ec2-user	Spark Pi	SPARK	root.ec2-user	Thu, 04 Dec 2014 15:25:18 GMT	Thu, 04 Dec 2014 15:25:35 GMT	FINISHED	SUCCEEDED		History

Click on a specific application from the web page above and you will see details of the app:



The screenshot shows the Hadoop YARN Application Overview page for the application with ID "application\_1417641624005\_0003". The page title is "Application Overview". It displays various metrics and logs for the application. A red arrow points to the "Logs" link in the bottom right corner of the application details section.

Application Overview	
User:	ec2-user
Name:	Spark Pi
Application Type:	SPARK
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	4-Dec-2014 10:30:43
Elapsed:	31sec
Tracking URL:	<a href="#">History</a>
Diagnostics:	

Application Metrics	
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	57388 MB-seconds, 45 vcore-seconds

ApplicationMaster			
Attempt Number	Start Time	Node	Logs
1	4-Dec-2014 10:30:43	ip-10-0-72-36.us-west-2.compute.internal:8042	<a href="#">logs</a>

Run the following command to run the same SparkPi app in cluster mode:

```
[ec2-user@ip-10-0-72-36 ~]$ spark-submit --class
org.apache.spark.examples.SparkPi --deploy-mode cluster --master yarn
/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/spark-examples-1.1.0-cd
h5.2.1-hadoop2.5.0-cdh5.2.1.jar 10
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
[jar:file:/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/spark-assembl
y-1.1.0-cdh5.2.1-hadoop2.5.0-cdh5.2.1.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: Found binding in
[jar:file:/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/slf4j-log4j12
-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

Notice that this time, the output is not shown on the cmd line. The output will be printed to the stdout log file for the application. We'll find that now...

Visit the URL:

<http://<your public hostname>:8088/cluster/apps>

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1417641624005_0004	ec2-user	org.apache.spark.examples.SparkPi	SPARK	root.ec2-user	Thu, 04 Dec 2014 15:37:10 GMT	Thu, 04 Dec 2014 15:37:54 GMT	FINISHED	SUCCESS
application_1417641624005_0003	ec2-user	Spark PI	SPARK	root.ec2-user	Thu, 04 Dec 2014 15:39:42	Thu, 04 Dec 2014 15:39:44	FINISHED	SUCCESS

Notice above that the name of the SparkPi app now appears with the full class path. Click on the latest Spark Application ID as seen in the screenshot above.

The next page will show an ApplicationMaster's logs at the bottom right corner. Click on 'logs':

Logged in as: dr.who

**Application Overview**

User:	ec2-user
Name:	org.apache.spark.examples.SparkPi
Application Type:	SPARK
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	4-Dec-2014 10:37:10
Elapsed:	43sec
Tracking URL:	<a href="#">History</a>
Diagnostics:	

**Application Metrics**

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	83705 MB-seconds, 66 vcore-seconds

**ApplicationMaster**

Attempt Number	Start Time	Node	Logs
1	4-Dec-2014 10:37:10	ip-10-0-72-36.us-west-2.compute.internal:8042	<a href="#">logs</a>

At the point, you'll see a 404 page. Simply replace the internal hostname of the EC2 instance with the public hostname:

This webpage is not available

[Reload](#) [Details](#)

So, replace this:

[http://ip-10-0-72-36.us-west-2.compute.internal:8042/node/containerlogs/container\\_1417641624005\\_0004\\_01\\_000001/ec2-user](http://ip-10-0-72-36.us-west-2.compute.internal:8042/node/containerlogs/container_1417641624005_0004_01_000001/ec2-user)

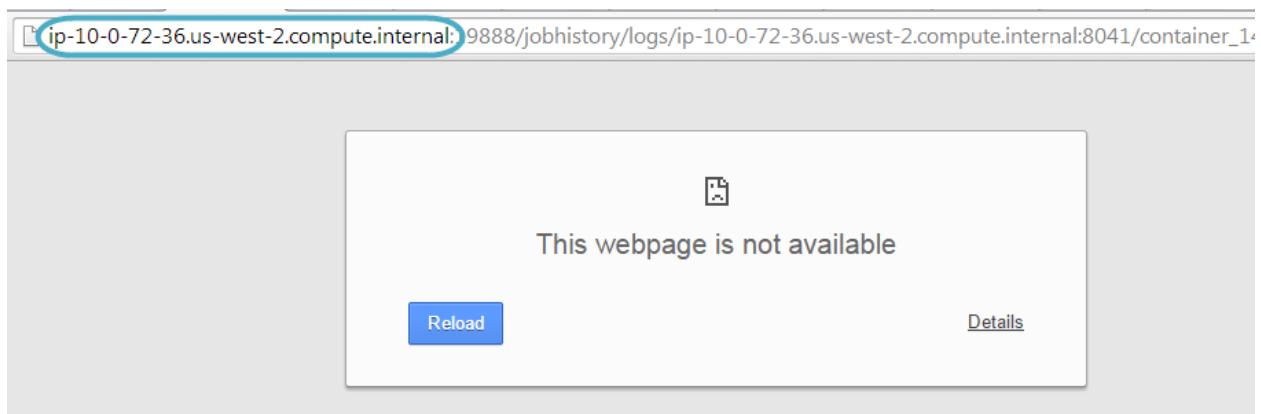
with something like this:

[http://<your public hostname>:8042/node/containerlogs/container\\_1417641624005\\_0004\\_01\\_000001/ec2-user](http://<your public hostname>:8042/node/containerlogs/container_1417641624005_0004_01_000001/ec2-user)

When you hit enter on the fixed URL, you will see this page for about 1-2 seconds:

The screenshot shows the Hadoop ResourceManager web interface. At the top left is the Hadoop logo. On the right, it says "Logged in as: dr.who". In the center, the text "Redirecting to log server for container\_1417641624005\_0004\_01\_000001" is displayed. Below this, an error message states: "java.lang.Exception: Unknown container. Container either has not started or has already completed or doesn't belong to this node at all." On the left, there is a sidebar with links: "ResourceManager" (selected), "RM Home", "NodeManager", and "Tools".

But then you'll get another 404 page:



Once again, replace the internal hostname in the beginning of the URL with the public hostname and finally, you'll see:

```

Log Type: stderr
Log Length: 22704
Showing 4096 bytes of 22704 total. Click here for the full log.
./sparkStaging/application_1417641624005_0004/spark-assembly-1.1.0-cdh5.2.1-hadoop2.5.0-cdh5.2.1.jar" } size: 95571683 timestamp
14/12/04 10:37:52 INFO yarn.YarnAllocationHandler: Completed container container_1417641624005_0004_01_000002 (state: COMPLETE,
14/12/04 10:37:52 INFO yarn.ExecutorRunnable: Setting up executor with environment: Map(CLASSPATH -> $PWD:$PWD/_spark_.jar:$H
14/12/04 10:37:52 INFO yarn.ExecutorRunnable: Setting up executor with commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError=-0
14/12/04 10:37:52 INFO impl.ContainerManagementProtocolProxy: Opening proxy : ip-10-0-72-36.us-west-2.compute.internal:8041
14/12/04 10:37:52 INFO yarn.ApplicationMaster: Allocating 1 containers to make up for (potentially) lost containers
14/12/04 10:37:52 INFO yarn.YarnAllocationHandler: Will Allocate 1 executor containers, each with 1408 memory
14/12/04 10:37:52 INFO spark.MapOutputTrackerMasterActor: MapOutputTrackerActor stopped!
14/12/04 10:37:53 INFO network.ConnectionManager: Selector thread was interrupted!
14/12/04 10:37:53 INFO network.ConnectionManager: ConnectionManager stopped
14/12/04 10:37:53 INFO storage.MemoryStore: MemoryStore cleared
14/12/04 10:37:53 INFO storage.BlockManager: BlockManager stopped
14/12/04 10:37:53 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
14/12/04 10:37:53 INFO remote.RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.
14/12/04 10:37:53 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remote daemon shut down; proceeding with flushing remo
14/12/04 10:37:53 INFO Remoting: Remoting shut down
14/12/04 10:37:53 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remoting shut down.
14/12/04 10:37:54 INFO spark.SparkContext: Successfully stopped SparkContext
14/12/04 10:37:54 INFO yarn.ApplicationMaster: Unregistering ApplicationMaster with SUCCEEDED
14/12/04 10:37:54 INFO impl.AMRMClientImpl: Waiting for application to be successfully unregistered.
14/12/04 10:37:54 INFO yarn.ApplicationMaster: All executors have launched.
14/12/04 10:37:54 INFO yarn.ApplicationMaster: Started progress reporter thread - heartbeat interval : 5000
14/12/04 10:37:54 INFO yarn.ApplicationMaster: AppMaster received a signal.
14/12/04 10:37:54 INFO yarn.ApplicationMaster: Deleting staging directory .sparkStaging/application_1417641624005_0004
14/12/04 10:37:54 INFO yarn.ApplicationMaster$$anon$1: Invoking sc stop from shutdown hook
14/12/04 10:37:54 INFO ui.SparkUI: Stopped Spark web UI at http://ip-10-0-72-36.us-west-2.compute.internal:41025
14/12/04 10:37:54 INFO spark.SparkContext: SparkContext already stopped

Log Type: stdout
Log Length: 23
Pi is roughly 3.142392

```

Also, in the screenshot above, the 3rd log line in stderr will show the **JVM size of the Executor memory**:

```
14/12/04 10:37:52 INFO yarn.ExecutorRunnable: Setting up executor with
commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError='kill
%p', -Xms1024m -Xmx1024m <rest of line truncated>
```

And the second to last line shows that there was a Spark web UI available while the Spark app was running (but it will no longer be available):

```
14/12/04 10:37:54 INFO ui.SparkUI: Stopped Spark web UI at
http://ip-10-0-72-36.us-west-2.compute.internal:41025
```

Now that we've run several Spark Apps (via the Scala shell, Python Shell and YARN client/cluster modes), we can visit the Spark History server to see the details of each of the jobs that was run:

<http://<your public hostname>:18088/>

The screenshot shows a browser window with the URL <http://ec2-54-149-62-154.us-west-2.compute.amazonaws.com:18088>. The page title is "Spark History Server". Below it, a message says "Event Log Location: hdfs://ip-10-0-72-36.us-west-2.compute.internal:8020/user/spark/applicationHistory". A heading "Showing 1-5 of 5" is followed by a table with the following data:

App Name	Started	Completed	Duration	Spark User	Last Updated
Spark Pi	2014/12/04 10:37:21	2014/12/04 10:37:52	31 s	ec2-user	2014/12/04 10:37:54
Spark Pi	2014/12/04 10:30:41	2014/12/04 10:31:13	32 s	ec2-user	2014/12/04 10:31:15
Spark Pi	2014/12/04 10:25:46	2014/12/04 10:26:18	32 s	ec2-user	2014/12/04 10:26:20
Spark shell	2014/12/04 09:14:01	2014/12/04 09:21:19	7.3 min	ec2-user	2014/12/04 09:21:20
Spark shell	2014/12/04 09:07:36	2014/12/04 09:13:47	6.2 min	ec2-user	2014/12/04 09:13:48

Try clicking on one of the App Names above to see the details of that app:

The screenshot shows a browser window with the URL <http://ec2-54-149-62-154.us-west-2.compute.amazonaws.com:18088/history/spark-pi-1417707441277/stages/>. The page title is "spark-pi-1417707441277 application UI". Below it, a navigation bar has "Stages" selected. The main content is titled "Spark Stages" and includes the following statistics:

- Scheduling Mode: FIFO
- Active Stages: 0
- Completed Stages: 1
- Failed Stages: 0

Under "Active Stages (0)", there is a table with columns: Stage Id, Description, Submitted, Duration, Tasks: Succeeded/Total, Input, Shuffle Read, and Shuffle Write.

Under "Completed Stages (1)", there is a table with columns: Stage Id, Description, Submitted, Duration, Tasks: Succeeded/Total, Input, Shuffle Read, and Shuffle Write. One row is highlighted with a red arrow pointing to the "Description" column, which contains the text "reduce at SparkPi.scala:35".

Under "Failed Stages (0)", there is a table with columns: Stage Id, Description, Submitted, Duration, Tasks: Succeeded/Total, Input, Shuffle Read, Shuffle Write, and Failure Reason.

Here you will see many details of the Spark application:

	Stages	Storage	Environment	Executors
---	--------	---------	-------------	-----------

## Details for Stage 0

Total task time across all tasks: 0.2 s

### Summary Metrics for 10 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile
Result serialization time	0 ms	0 ms	0 ms	0 ms
Duration	8 ms	11 ms	12 ms	13 ms
Time spent fetching task results	0 ms	0 ms	0 ms	0 ms
Scheduler delay	9 ms	11 ms	11 ms	16 ms

### Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	SI W
1	ip-10-0-72-36.us-west-2.compute.internal:54423	0.6 s	10	0	10	0.0 B	0.0 B	0.

At the end of the page, you will see 10 tasks which corresponds to the fact that you passed in the #10 when running the SparkPi application:

### Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:51	0.1 s			
1	1	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	13 ms			
2	2	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	11 ms			
3	3	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	11 ms			
4	4	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	13 ms			
5	5	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	11 ms			
6	6	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	14 ms			
7	7	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	8 ms			
8	8	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	12 ms			
9	9	0	SUCCESS	PROCESS_LOCAL	ip-10-0-72-36.us-west-2.compute.internal	2014/12/04 10:37:52	12 ms			

Before moving on, note that you can also pass in the # of executors and the amount of memory to assign to each executor like this. This command is just FYI - do not run it!

```
[ec2-user@ip-10-0-72-36 ~]$ spark-submit --class  
org.apache.spark.examples.SparkPi --deploy-mode client --master yarn --name  
"My Pi App" --num-executors 3 --executor-memory 2g  
/opt/cloudera/parcels/CDH-5.2.1-1.cdh5.2.1.p0.12/jars/spark-examples-1.1.0-cd  
h5.2.1-hadoop2.5.0-cdh5.2.1.jar 10
```

You can learn more details about running Spark on YARN here:

<http://spark.apache.org/docs/latest/running-on-yarn.html>