

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



Lab created on: Dec, 2014

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

Estimated lab completion time: **2 hours** (spread throughout the day)

License: 

### Objective:

This lab will introduce you to using 3 Hadoop ecosystem components in Cloudera's distribution: HDFS, Spark and YARN. The lab will first walk you through the Cloudera Manager installation on a VM in Rackspace, 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, running low latency queries against HBase, etc.

**The following high level steps are in the initial part of this lab(to-edit):**

- Connect via SSH to your Rackspace 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 VM in Rackspace's Dallas data center with the following configuration: **Performance 2 w/ 15 GB of RAM and 40G of hard drive space (in a /ext3 file system) running CentOS 6.5.**

The instructor should have given you the following:

- Public IP address of your specific instance

## Resources to learn more about Spark

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

Spark homepage @ Apache:

<https://spark.apache.org>

Here are the official Apache docs for Spark 1.0.2:

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

However, docs for the latest release of Spark (which is 1.1 as of Oct 2014) can be found here:

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

Link to user + dev mailing lists:

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

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

Spark Core 1.0.0 Scala docs:

<https://spark.apache.org/docs/1.0.0/api/scala/index.html#package>

Spark Core 1.0.0 Java docs:

<https://spark.apache.org/docs/1.0.0/api/java/index.html>

Spark Core 1.0.0 Python docs:

<https://spark.apache.org/docs/1.0.0/api/python/index.html>

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

Cloudera's docs for CDH 5.1 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 5.1.3:

[http://www.cloudera.com/content/cloudera/en/documentation/cdh5/latest/CDH-Version-and-Packaging-Information/cdhvd\\_cdh\\_package\\_tarball.html?scroll=topic\\_3\\_unique\\_8](http://www.cloudera.com/content/cloudera/en/documentation/cdh5/latest/CDH-Version-and-Packaging-Information/cdhvd_cdh_package_tarball.html?scroll=topic_3_unique_8)

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 ++**

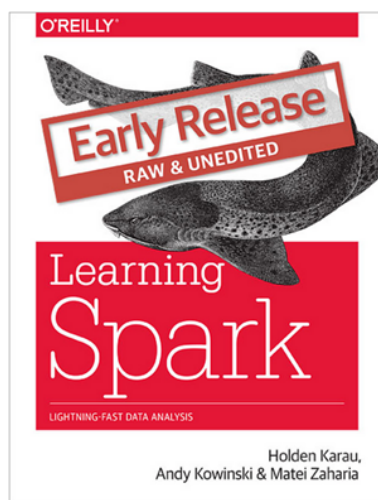
Note, when you're ready to get certified as a Spark Developer, check out the joint Spark certification program between Databricks & O'Reilly:

<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 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 (packages) 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 Rackspace Instance

Each student will get 1 virtual server in Rackspace to 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 Rackspace 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.

Your VM in Rackspace will be running CentOS 6.5.

As discussed in a [Sept 2012 Rackspace article](#), CentOS “emphasizes stability and enterprise software compatibility above cutting-edge features”:

Quote from article:

“CentOS is a distribution that emphasizes reliability. It replicates Red Hat Enterprise Linux as much as possible, omitting only the non-free components of that distribution. That means CentOS is a very stable distribution and is well-suited to production environments. It also tends to be compatible with enterprise software, though it's not always officially supported by software vendors.

The price of stability is that the software versions included with CentOS are rarely the latest and greatest. The packages included with CentOS have been tuned over time to work out as many bugs and security flaws as possible.

CentOS uses rpm for its package manager.”

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

Port: 22

Username: root

Password: *<provided by instructor, it will be unique to your VM>*

Hostname: *<public IP address instructor provided you with>*

Note, there is no .pem or .ppk key pair needed to log into this machine.

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...

SSH into the VM using this command: `ssh root@<public IP of VM>`

Type in the password when prompted and accept the rsa2 key if needed.

Please ask a fellow student or the instructor for any further help needed with logging in from an Apple laptop!

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

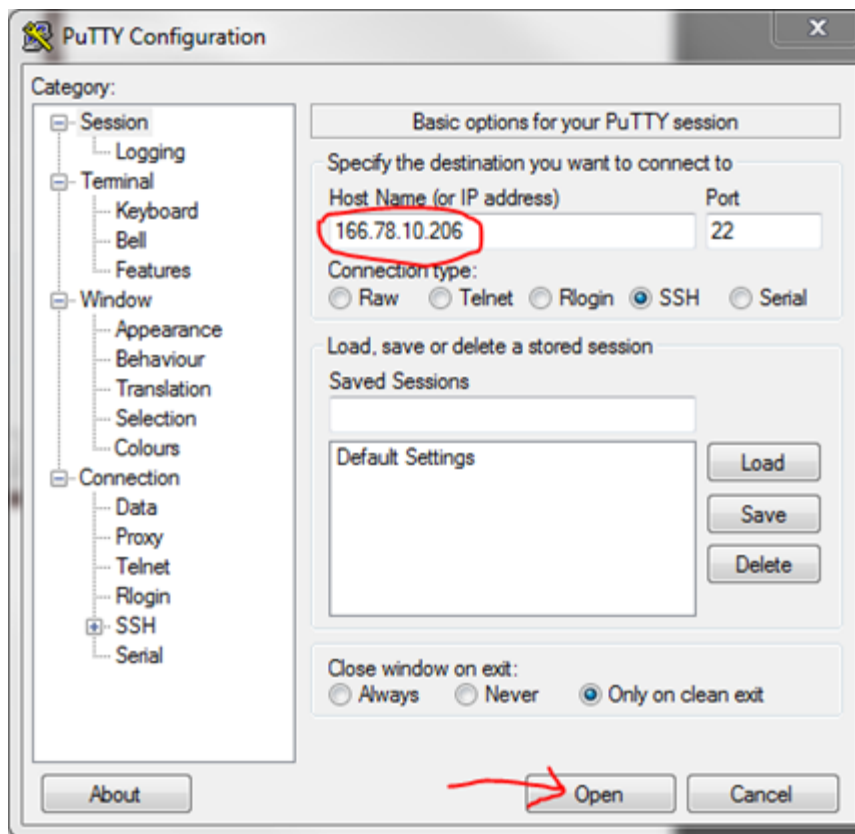
Download PuTTY.exe from:

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

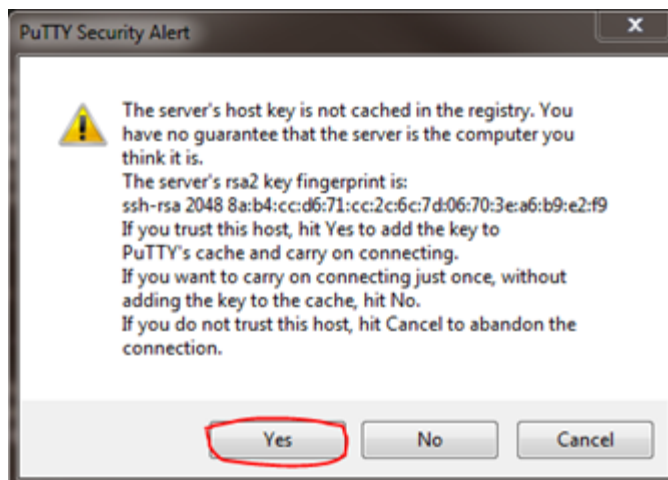
There is no installation for PuTTY. You can just run it from the downloaded .exe file.

After starting PuTTY, enter the IP address of the Rackspace VM into PuTTY. The connection type will be SSH and the port will be 22.

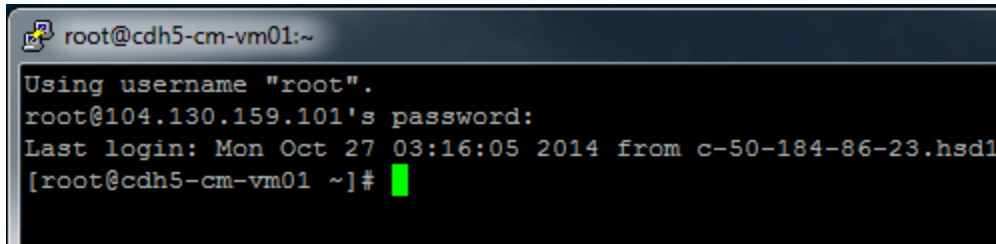
Type the IP that the instructor gave you for your individual Rackspace VM into PuTTY:



Click “Open” to connect to the VM. You will have to click “Yes” to a message about the server’s rsa2 key before a successful connection.



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



```

root@cdh5-cm-vm01:~
Using username "root".
root@104.130.159.101's password:
Last login: Mon Oct 27 03:16:05 2014 from c-50-184-86-23.hsd1
[root@cdh5-cm-vm01 ~]#
  
```

## 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:**

```

[root@cdh5-cm-vm01 ~]# free -m

```

	total	used	free	shared	buffers	cached
Mem:	15011	201	14809	0	4	46
-/+ buffers/cache:		149	14861			
Swap:	0	0	0			

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

```

[root@cdh5-cm-vm01 ~]# 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::**

```
[root@cdh4-cm-vm01 ~]$ java -version
-bash: java: command not found
```

Well, looks like Java is not available in CentOS by default. This is okay. After we install Cloudera Manager (CM), we will re-run this command and see that the CM installation installs Java for us.

**IMPORTANT: Make sure your public IP is listed above the private IP!**

Print out your `/etc/hosts` file and check if the **public IP** that the instructor gave you is listed **ABOVE** the **private IP (10.x.x.x)**. If the public IP is not above the private IP, the Cloudera Manager install may fail. Talk to a student next to you or the instructor if you need help with this step. Cloudera Manager will bind to the first IP4 address it finds on this server and we need it to bind to the public IP so that we can connect to its web server on the public IP.

Note that in half of the student's VMs, the public IP will already be correctly listed above the private IP. However, in the other half of VMs, this file will need to be fixed via a text editor like VIM, EMACS or Nano.

```
[root@cdh4-cm-vm01 ~]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4
localhost4.localhostdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localhostdomain6
2001:4800:7810:0512:e2aa:bc1f:ff04:badc cdh4-cm-vm0
166.78.10.206 cdh4-cm-vm01
10.181.7.208  cdh4-cm-vm01
```

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:**

```
[root@cdh4-cm-vm01 ~]# chkconfig --list iptables
iptables      0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

```
[root@cdh4-cm-vm01 ~]# service iptables stop
```

```
iptables: Flushing firewall rules:      [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules:            [ OK ]
```

```
[root@cdh4-cm-vm01 ~]# chkconfig --level 123456 iptables off
```

```
[root@cdh4-cm-vm01 ~]# chkconfig --list iptables
```

```
iptables          0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

```
[root@cdh4-cm-vm01 ~]# 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:**

```
[root@cdh4-cm-vm01 ~]# sestatus
```

```
SELinux status:                disabled
```

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

```
[root@cdh4-cm-vm01 ~]$ pwd
```

```
/root
```

```
[root@cdh4-cm-vm01 ~]$ 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:**

```
[root@cdh4-cm-vm01 ~]$ ls -l
```

```
total 504
```

```
-rw-r--r-- 1 root root 510569 Sep 30 21:13 cloudera-manager-installer.bin
```

```
[root@cdh4-cm-vm01 ~]$ chmod u+x cloudera-manager-installer.bin
```

```
[root@cdh4-cm-vm01 ~]$ ls -l
```

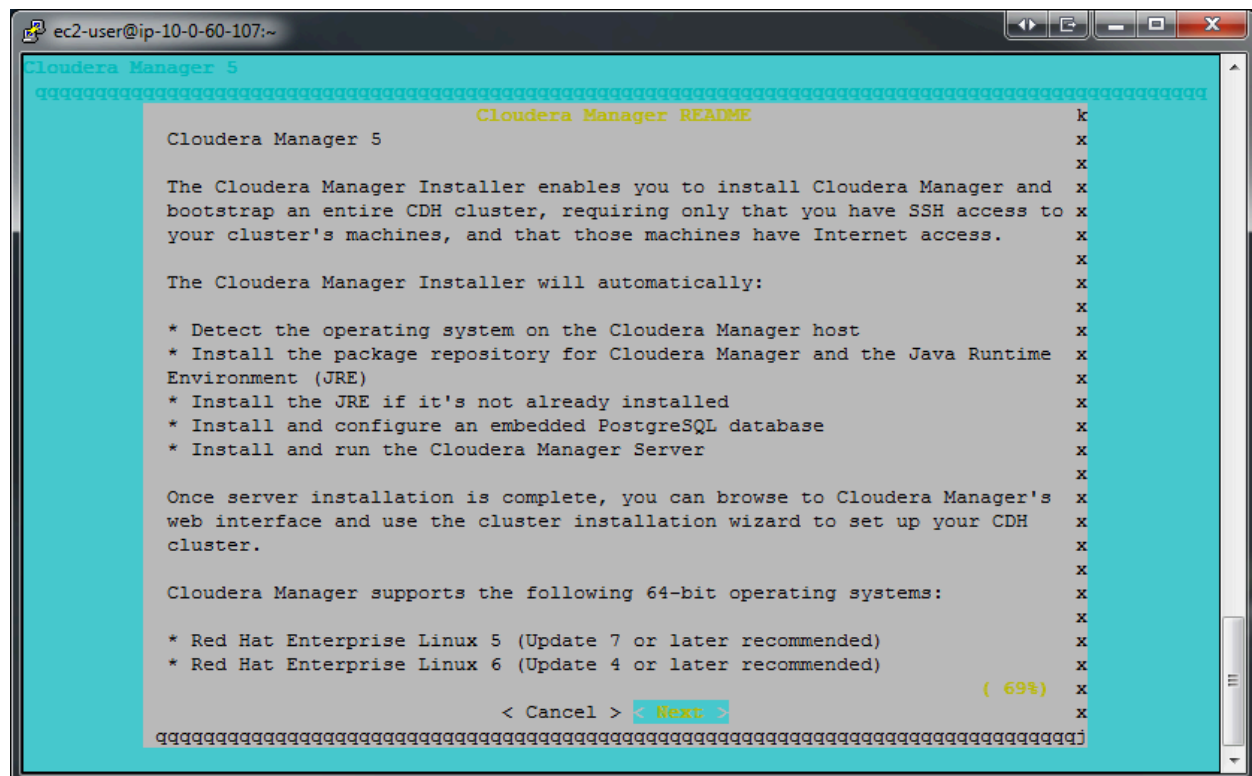
```
total 504
```

```
-rwxr--r-- 1 root root 510569 Sep 30 21:13 cloudera-manager-installer.bin
```

**Run the Cloudera Manager installer:**

```
[root@cdh4-cm-vm01 ~]$ sudo ./cloudera-manager-installer.bin
```

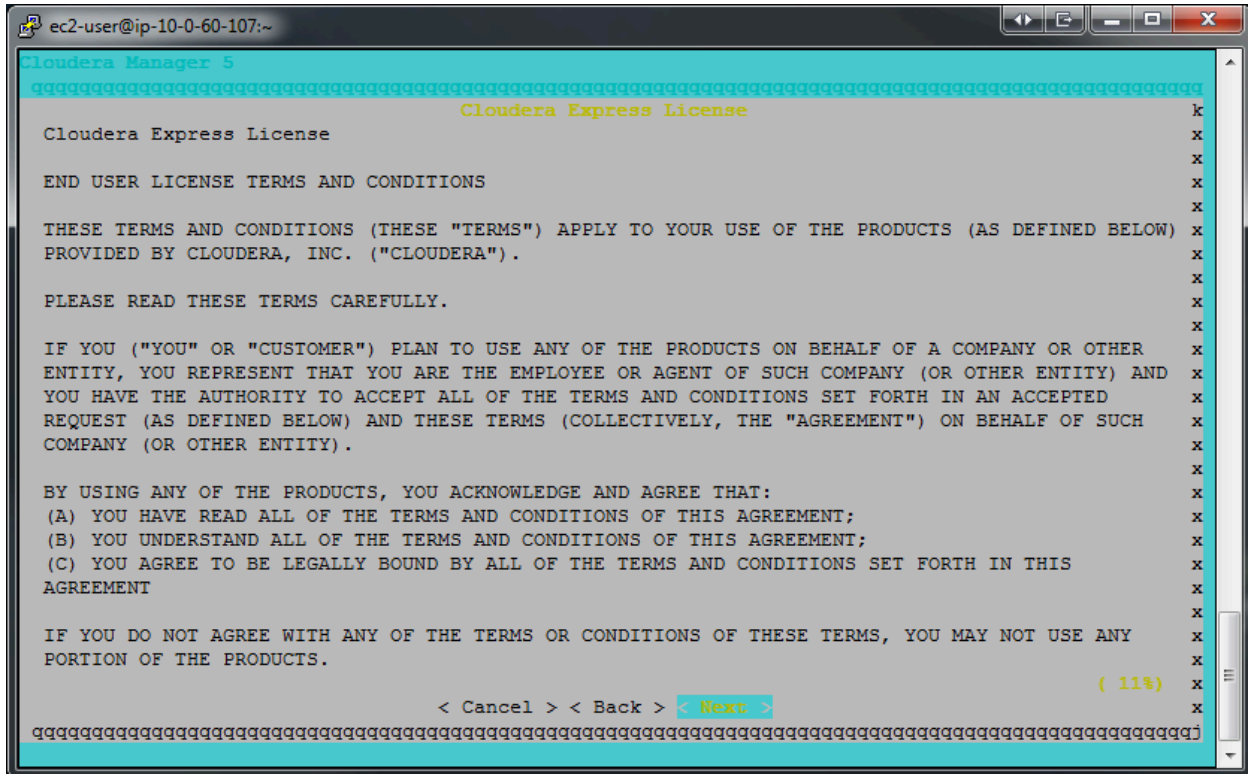
**Press **Enter** to choose Next for the Readme file:**



```

Cloudera Manager 5
=====
Cloudera Manager README
=====
x
Cloudera Manager 5
x
x
The Cloudera Manager Installer enables you to install Cloudera Manager and
bootstrap an entire CDH cluster, requiring only that you have SSH access to
your cluster's machines, and that those machines have Internet access.
x
x
The Cloudera Manager Installer will automatically:
x
x
* Detect the operating system on the Cloudera Manager host
x
* Install the package repository for Cloudera Manager and the Java Runtime
Environment (JRE)
x
* Install the JRE if it's not already installed
x
* Install and configure an embedded PostgreSQL database
x
* Install and run the Cloudera Manager Server
x
x
Once server installation is complete, you can browse to Cloudera Manager's
web interface and use the cluster installation wizard to set up your CDH
cluster.
x
x
Cloudera Manager supports the following 64-bit operating systems:
x
* Red Hat Enterprise Linux 5 (Update 7 or later recommended)
x
* Red Hat Enterprise Linux 6 (Update 4 or later recommended)
x
( 69%)
x
< Cancel > < Next >
=====
  
```

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



```

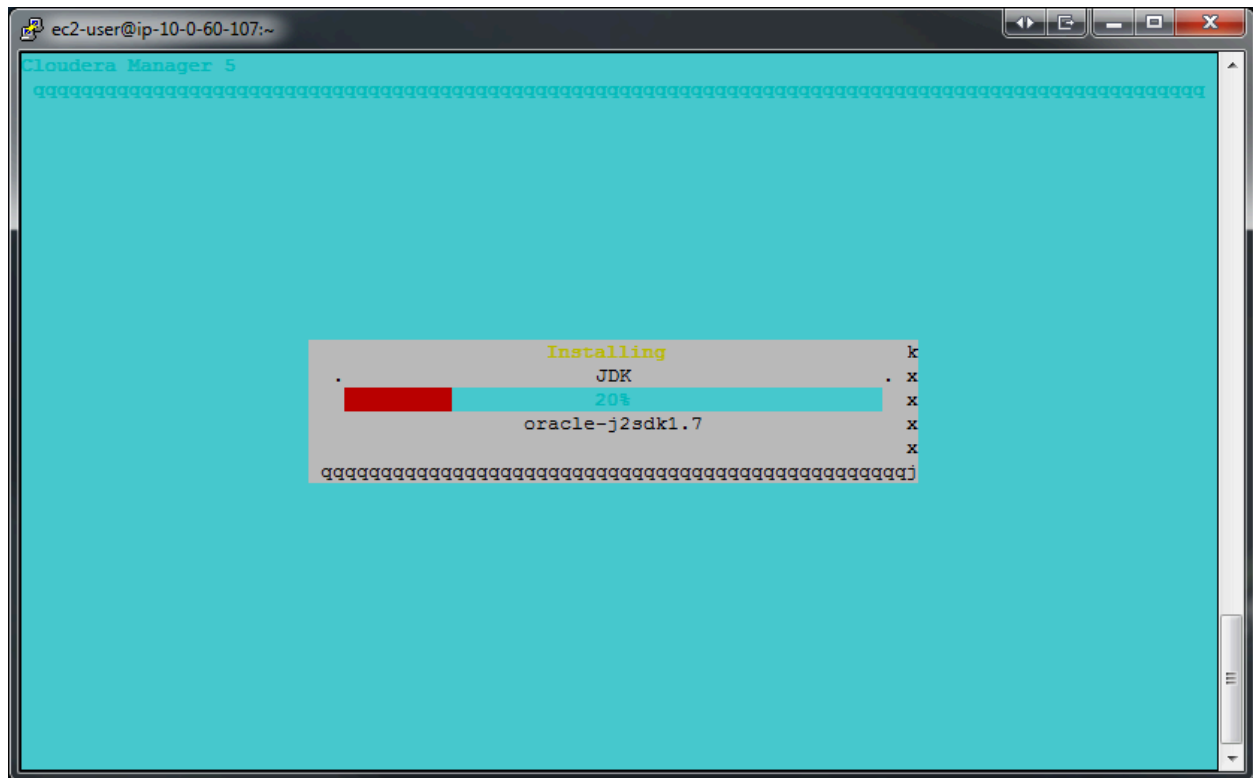
ec2-user@ip-10-0-60-107:~
Cloudera Manager 5
Cloudera Express License
END USER LICENSE TERMS AND CONDITIONS
THESE TERMS AND CONDITIONS (THESE "TERMS") APPLY TO YOUR USE OF THE PRODUCTS (AS DEFINED BELOW)
PROVIDED BY CLOUDERA, INC. ("CLOUDERA").
PLEASE READ THESE TERMS CAREFULLY.
IF YOU ("YOU" OR "CUSTOMER") PLAN TO USE ANY OF THE PRODUCTS ON BEHALF OF A COMPANY OR OTHER
ENTITY, YOU REPRESENT THAT YOU ARE THE EMPLOYEE OR AGENT OF SUCH COMPANY (OR OTHER ENTITY) AND
YOU HAVE THE AUTHORITY TO ACCEPT ALL OF THE TERMS AND CONDITIONS SET FORTH IN AN ACCEPTED
REQUEST (AS DEFINED BELOW) AND THESE TERMS (COLLECTIVELY, THE "AGREEMENT") ON BEHALF OF SUCH
COMPANY (OR OTHER ENTITY).
BY USING ANY OF THE PRODUCTS, YOU ACKNOWLEDGE AND AGREE THAT:
(A) YOU HAVE READ ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT;
(B) YOU UNDERSTAND ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT;
(C) YOU AGREE TO BE LEGALLY BOUND BY ALL OF THE TERMS AND CONDITIONS SET FORTH IN THIS
AGREEMENT
IF YOU DO NOT AGREE WITH ANY OF THE TERMS OR CONDITIONS OF THESE TERMS, YOU MAY NOT USE ANY
PORTION OF THE PRODUCTS.
( 11%)
< Cancel > < Back > < Next >

```

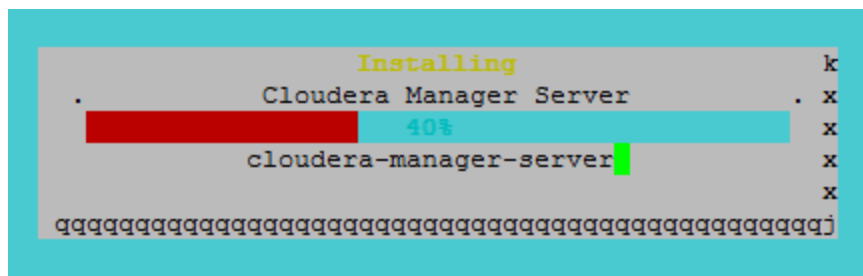
Now use the right arrow key to choose **Yes** for the license agreement.

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.

A screenshot of a terminal window titled "root@cdh5-cm-vm01:~". The background is blue. At the top, it says "Cloudera Manager 5" followed by a long string of 'q' characters. Below this, there's a light gray box containing the text "Next step" in yellow. The main text in the box reads: "Point your web browser to http://cdh5-cm-vm01:7180/. Log in to Cloudera Manager with the username and password set to 'admin' to continue installation. (Note that the hostname may be incorrect. If the url does not work, try the hostname you use when remotely connecting to this machine.) If you have trouble connecting, make sure you have disabled firewalls, like iptables." To the right of each line of text in the gray box is a column of characters: 'k', 'x', 'x', 'x', 'x', 'x', 'x', 'x'. Below the gray box is a blue button labeled "&lt; OK &gt;". At the bottom of the terminal, there's another long string of 'q' characters. On the far right edge of the terminal window, there's a vertical scrollbar and some additional characters like 'x', 'x', 'x', and a small icon at the bottom.

```

Finish k
Installation was successful. x
x
< OK > x
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]

```

You should now be returned to the linux cmd prompt.

## Connect to Cloudera Manager Admin Console to install CDH

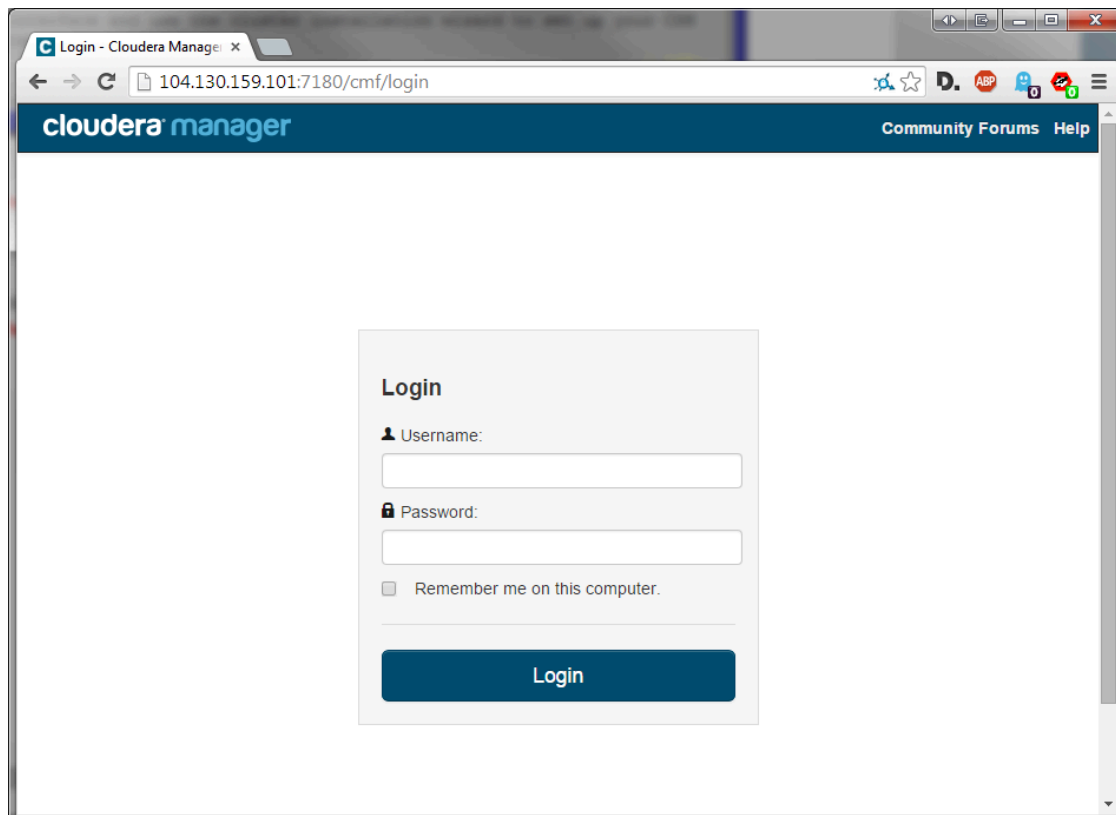
So far, only Cloudera Manager has been installed on the Amazon VM. CDH5 has not been installed yet. Time to do that now.

*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 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:

Welcome to Cloudera Manager. Which edition do you want to deploy?

Upgrading to **Cloudera Enterprise Data Hub Edition** provides important features that help you manage and monitor your Hadoop clusters in mission-critical environments.

	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 <a href="#">Upload License</a> Cloudera Enterprise is available in three editions: • Basic Edition • Flex Edition • Data Hub Edition
Node Limit	Unlimited	Unlimited	Unlimited
CDH	✓	✓	✓
Core Cloudera Manager Features	✓	✓	✓
Advanced Cloudera Manager Features		✓	✓
Cloudera Navigator		✓	✓
Cloudera Support			✓

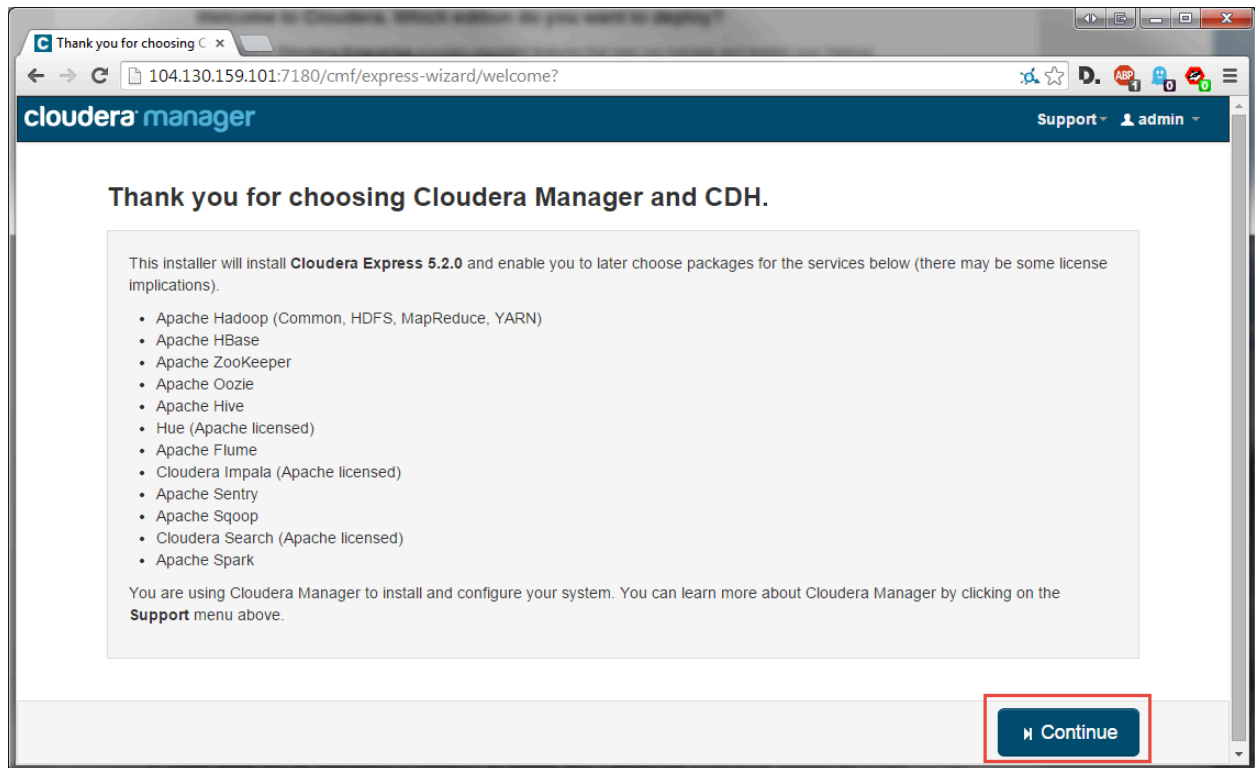
For full list of features available in **Cloudera Express** and **Cloudera Enterprise**, [click here](#).

[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:

Specify hosts for your CDH cluster installation.

Hosts should be specified using the same hostname (FQDN) that they will identify themselves with.

Cloudera recommends including Cloudera Manager Server's host. This will also enable health monitoring for that host.

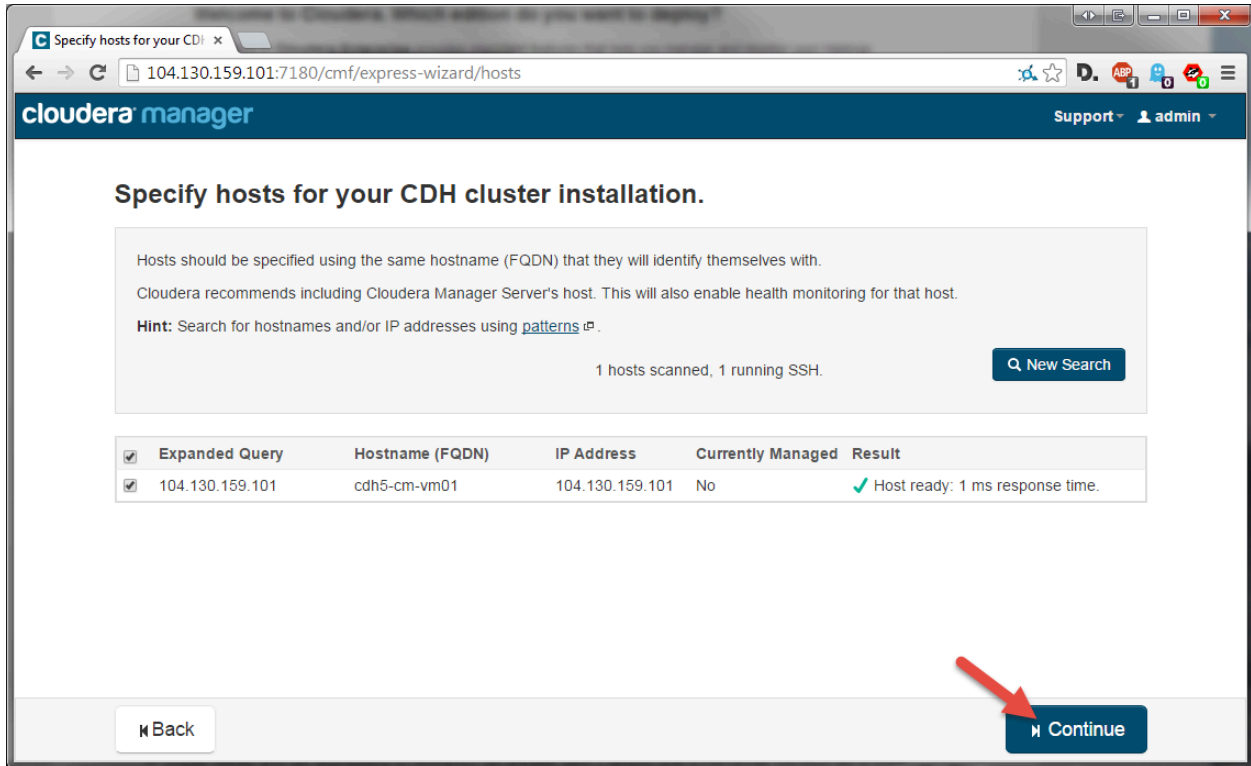
**Hint:** Search for hostnames and/or IP addresses using [patterns](#).

104.130.159.x

SSH Port: 22 **Search**

[Back](#) [Continue](#)

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



Specify hosts for your CDH cluster installation.

Hosts should be specified using the same hostname (FQDN) that they will identify themselves with.

Cloudera recommends including Cloudera Manager Server's host. This will also enable health monitoring for that host.

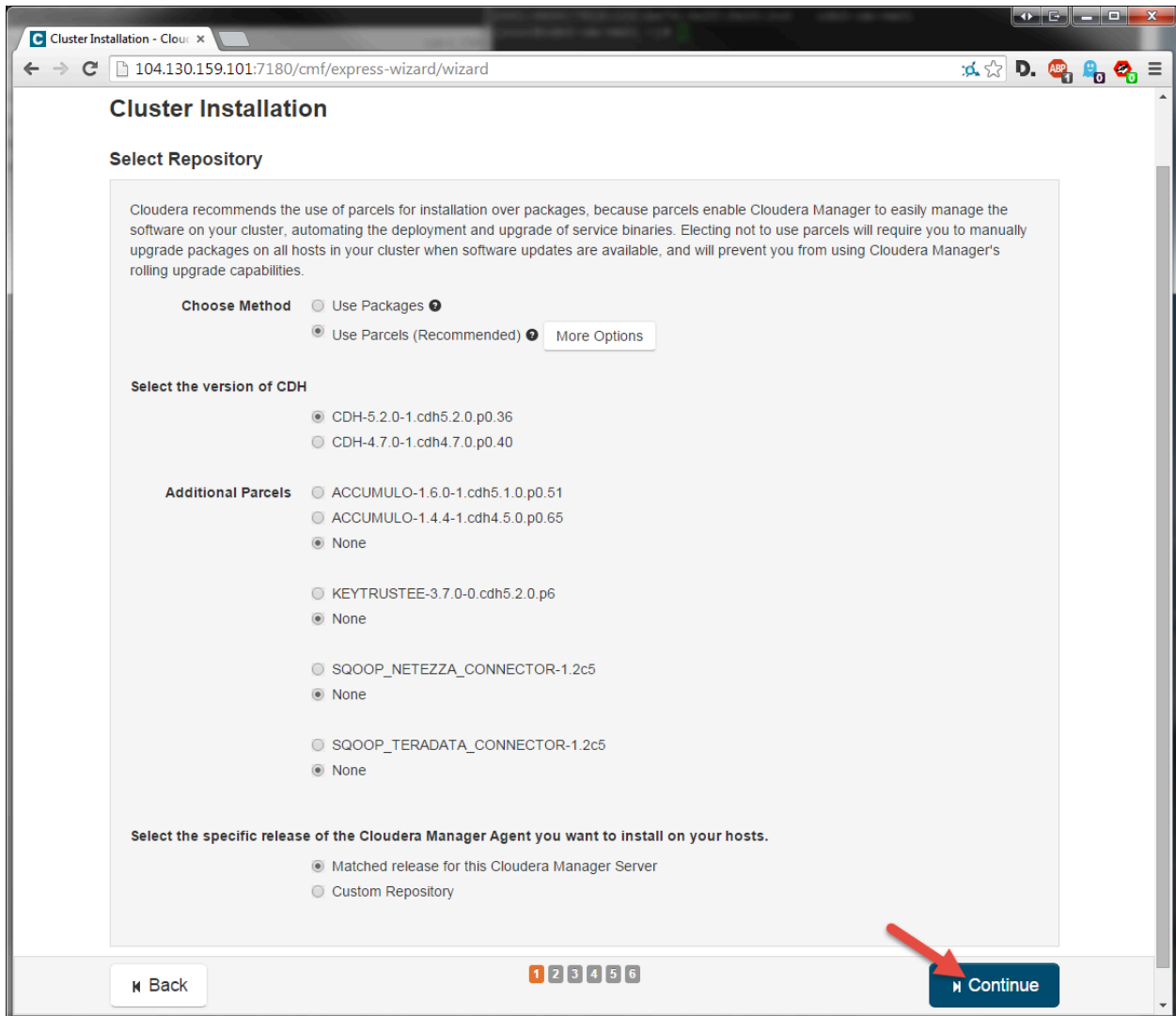
**Hint:** Search for hostnames and/or IP addresses using [patterns](#).

1 hosts scanned, 1 running SSH. [New Search](#)

<input checked="" type="checkbox"/>	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	✓ Host ready: 1 ms response time.

[Back](#) [Continue](#)

**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 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-5.2.0-1.cdh5.2.0.p0.36
- ☐ CDH-4.7.0-1.cdh4.7.0.p0.40

**Additional Parcels**

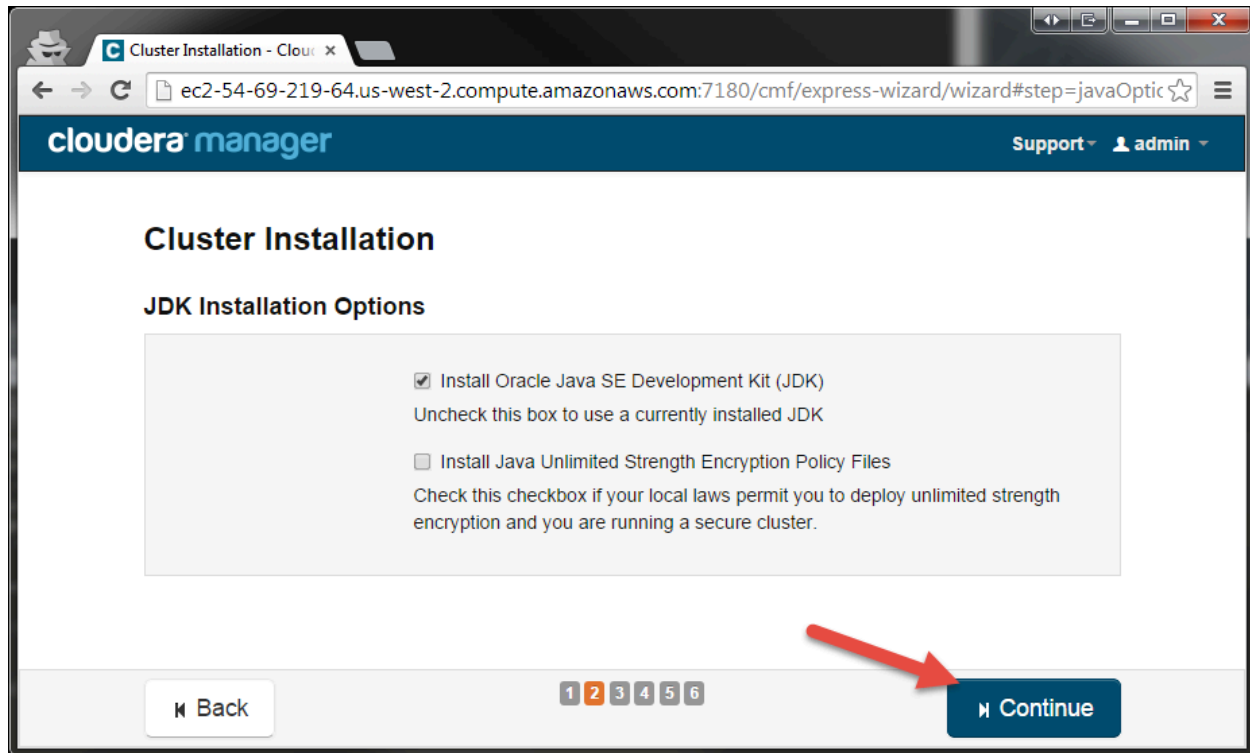
- ☐ ACCUMULO-1.6.0-1.cdh5.1.0.p0.51
- ☐ ACCUMULO-1.4.4-1.cdh4.5.0.p0.65
- ☒ None
- ☐ KEYTRUSTEE-3.7.0-0.cdh5.2.0.p6
- ☒ None
- ☐ SQOOP\_NETEZZA\_CONNECTOR-1.2c5
- ☒ None
- ☐ SQOOP\_TERADATA\_CONNECTOR-1.2c5
- ☒ None

**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

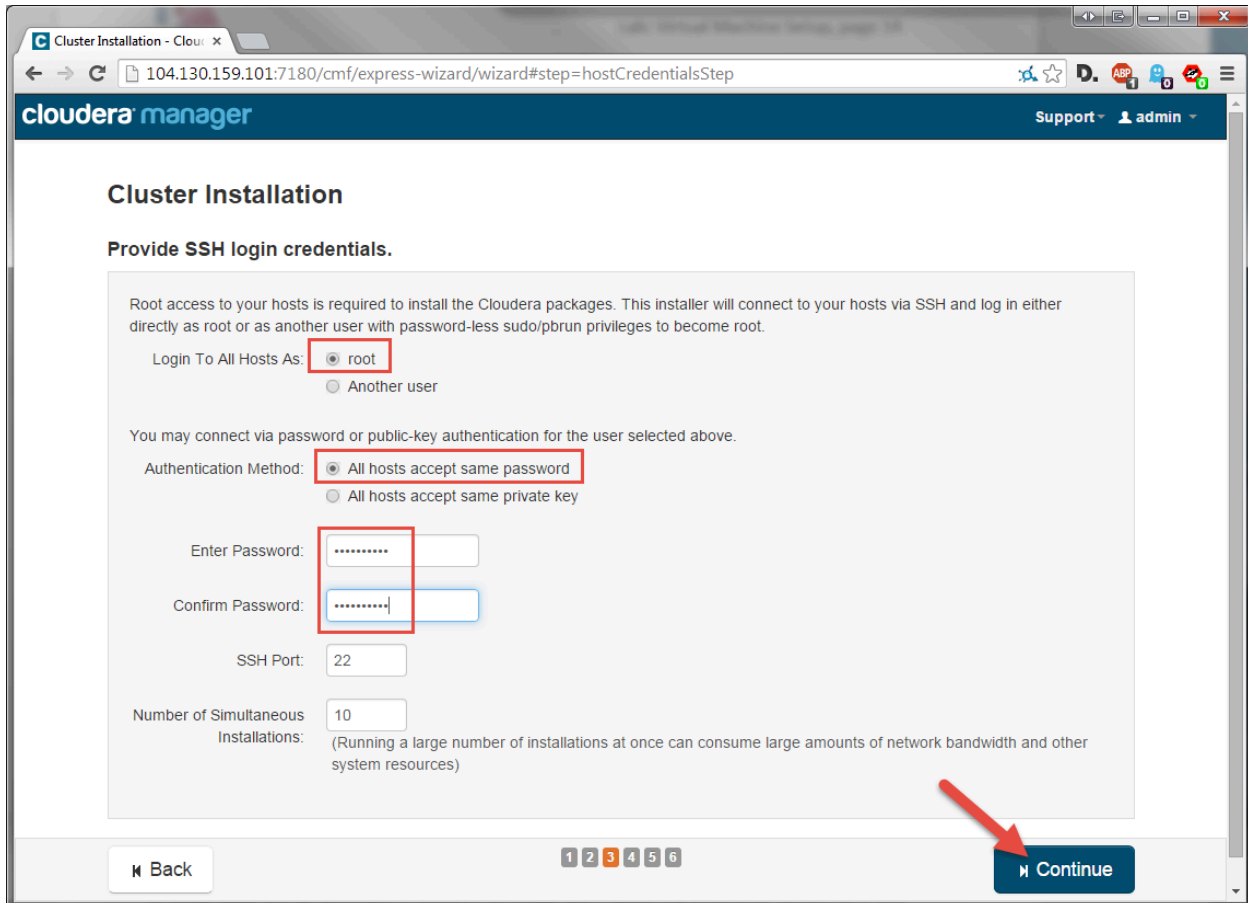
[Back](#) 1 2 3 4 5 6 [Continue](#)

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: **Root**
- Authentication Method: **All hosts accept same password**
- Enter your password twice
- Click **Continue**



Cluster Installation - Cloudera Manager

104.130.159.101:7180/cmf/express-wizard/wizard#step=hostCredentialsStep

cloudera manager Support admin

### Cluster Installation

**Provide SSH login credentials.**

Root access to your hosts is required to install the Cloudera packages. This installer will connect to your hosts via SSH and log in either directly as root or as another user with password-less sudo/pbrun privileges to become root.

Login To All Hosts As: ☒ root ☐ Another user

You may connect via password or public-key authentication for the user selected above.

Authentication Method: ☒ All hosts accept same password ☐ All hosts accept same private key

Enter Password:

Confirm Password:

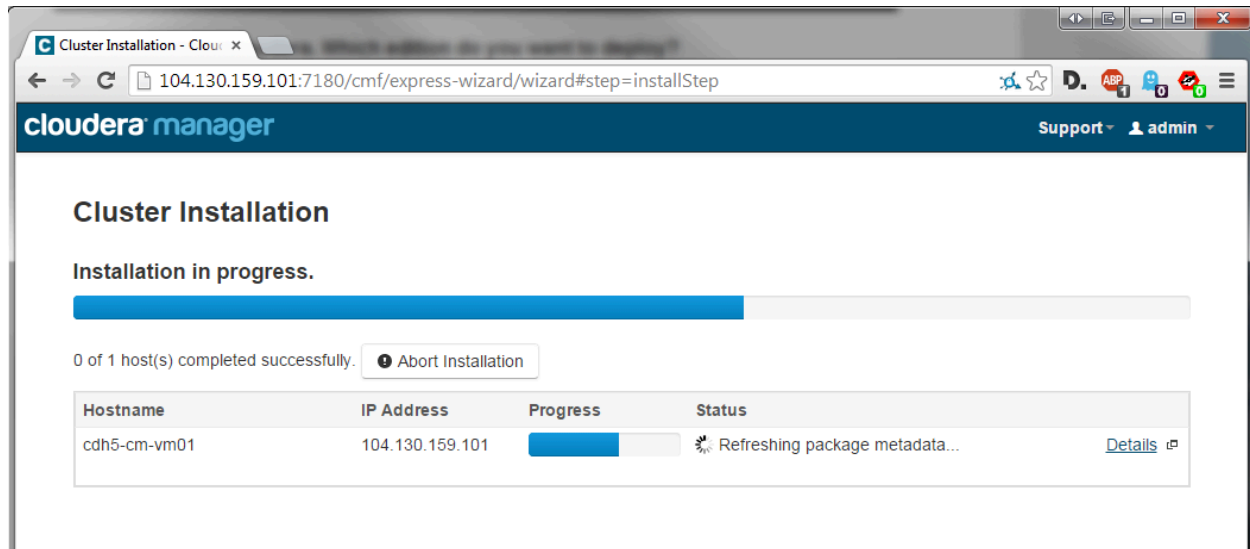
SSH Port:

Number of Simultaneous Installations:   
(Running a large number of installations at once can consume large amounts of network bandwidth and other system resources)

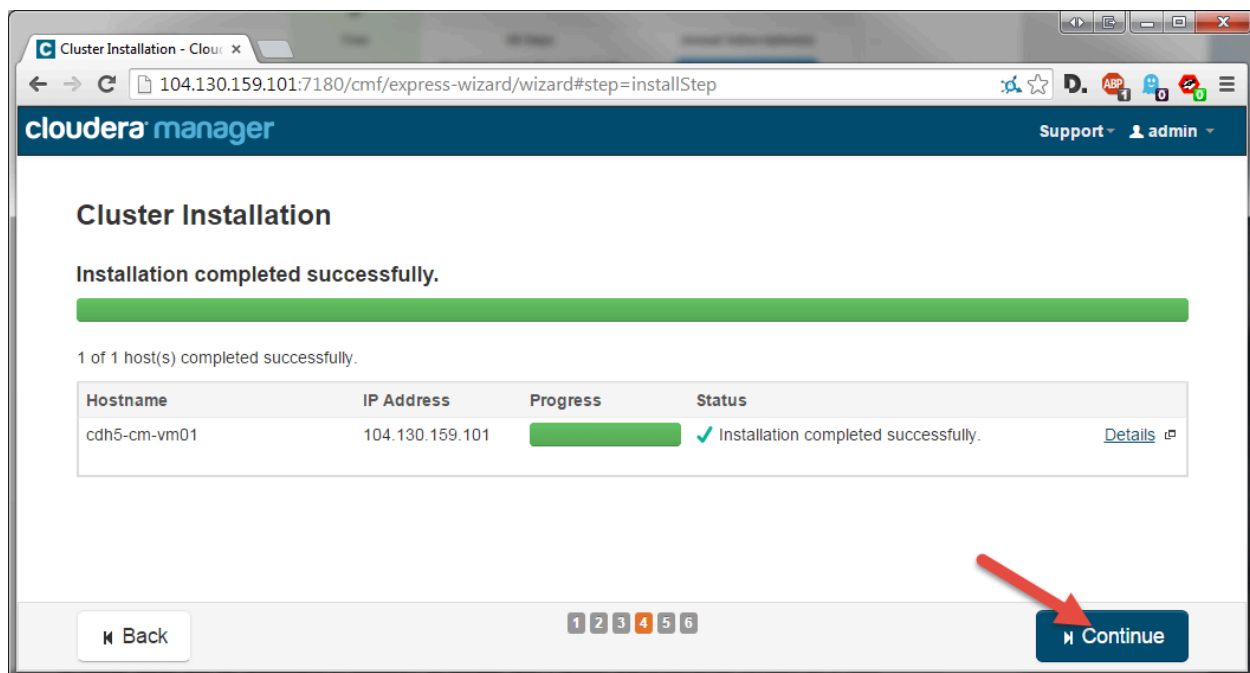
1 2 3 4 5 6

Back Continue

The cluster installation will now kick off:



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



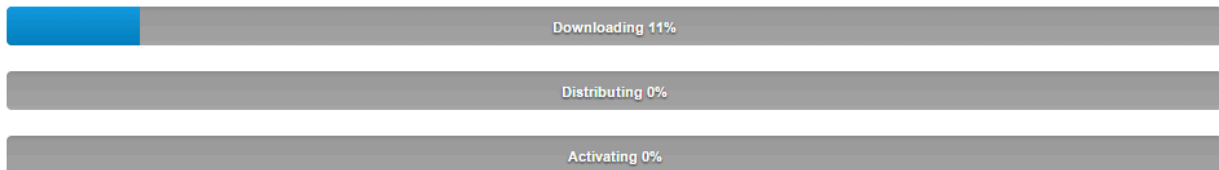
The Parcels installation will now start (note this part will take about 2 - 3 minutes):

## Cluster Installation

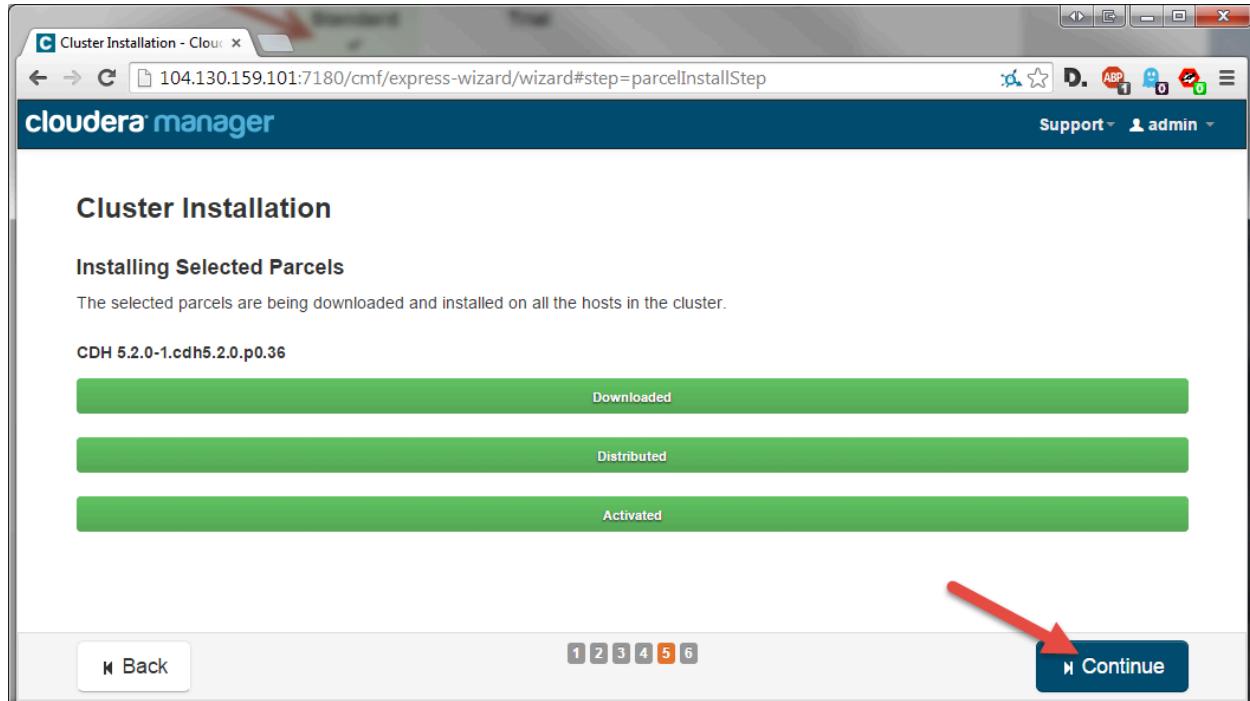
### Installing Selected Parcels

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

CDH 5.2.0-1.cdh5.2.0.p0.36



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






The host inspector will automatically run. It will have a couple of warnings about the “Inspector failed on following hosts” and “Inspector failed to run on all hosts”. You can safely ignore these warnings and continue.

## 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:

**Cluster Installation**

Inspect hosts for correctness [Run Again](#)

**Validations**

- Inspector failed on the following hosts... >
- The inspector failed to run on all hosts.
- 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**

cdh5-cm-vm01

Component	Version	Release	CDH Version
Bigtop-Tomcat (CDH 5 only)	0.7.0+cdh5.2.0+0	1.cdh5.2.0.p0.33	CDH 5
Crunch (CDH 5 only)	0.11.0+cdh5.2.0+9	1.cdh5.2.0.p0.28	CDH 5
Flume NG	1.5.0+cdh5.2.0+49	1.cdh5.2.0.p0.33	CDH 5

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


Notice that we are running Spark 1.1.0:

Solr	4.4.0+cdh5.2.0+282	1.cdh5.2.0.p0.27	CDH 5
spark	1.1.0+cdh5.2.0+56	1.cdh5.2.0.p0.35	CDH 5
Scoop	1.4.5+cdh5.2.0+47	1.cdh5.2.0.p0.26	CDH 5

Click **Finish**:

5-

build	Unavailable	Not applicable
55-		
1.cm513.p0.155.el6	Not applicable	



» 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.**

Cluster Setup - Cloudera | x

104.130.159.101:7180/cm/clusters/1/express-add-services/index

**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 checked="" 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 checked="" 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). <b>YARN (MapReduce 2 Included) is recommended instead. MapReduce is included for backward compatibility.</b>
<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 checked="" type="checkbox"/> ZooKeeper	Apache ZooKeeper is a centralized service for maintaining and synchronizing configuration data.

This wizard will also install the **Cloudera Management Service**. These are a set of components that enable monitoring, reporting, events, and alerts; these components require databases to store information, which will be configured on the next page.

Back 1 2 3 4 5 6 Continue

On the Customize Role Assignments page, there are no changes needed. Just click **Continue**:

The screenshot shows the Cloudera Manager web interface for configuring a new cluster. The browser address bar shows the URL: `104.130.159.101:7180/cm/clusters/1/express-add-services/index#step=roleAssignmentsStep`. The page title is "Cluster Setup" and the sub-header is "Customize Role Assignments".

**Customize Role Assignments**

You can customize the role assignments for your new cluster here, but if assignments are made incorrectly, such as assigning too many roles to a single host, this can impact the performance of your services. Cloudera does not recommend altering assignments unless you have specific requirements, such as having pre-selected a specific host for a specific role.

You can also view the role assignments by host. [View By Host](#)

**HBase**

- M** Master x 1 New: Same As DataNode
- HBRS** HBase REST Server: Select hosts
- HBTS** HBase Thrift Server: Select hosts
- RS** RegionServer x 1 New: Same As DataNode

**HDFS**

- NN** NameNode x 1 New: Same As DataNode
- SNN** SecondaryNameNode x 1 New: Same As DataNode
- B** Balancer x 1 New: Same As DataNode
- HFS** HttpFS: Select hosts
- NFSG** NFS Gateway: Select hosts
- DN** DataNode x 1 New: cdh5-cm-vm01

At the bottom, there is a "Back" button and a "Continue" button. A red arrow points to the "Continue" button. A progress indicator shows steps 1 through 6, with step 2 being the current step.

On the Database Setup page, leave “Use Embedded Database” selected and click on **Test Connection**:

Cluster Setup - Cloudera

104.130.159.101:7180/cm/clusters/1/express-add-services/index#step=showDbTestConnStep

cloudera manager Support admin

### Cluster Setup

#### Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the **Installing and Configuring an External Database** section of the [Installation Guide](#).

☐ Use Custom Databases  
☒ Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

**Hive**

Database Host Name:	Database Type:	Database Name :	Username:	Password:
cdh5-cm-vm01:7432	PostgreSQL	hive	hive	cf6ZYkmxqf

**Test Connection**

Back 1 2 3 4 5 6 Continue

You will quickly see a message that this step will be skipped. Now click **Continue**:

✓ Skipped. Cloudera Manager will create this database in a later step.

Database Type: PostgreSQL Database Name : hive Username: hive Password: cf6ZYkmxqf

**Test Connection**

1 2 3 4 5 6 **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**:

**Cluster Setup**

**Review Changes**

Set the following configuration values for your new role(s). Required values are marked with \*.

Parameter	Group	Value	Description
<b>Service Cloudera Management Service</b>			
Service Monitor Storage Directory* firehose.storage.base.directory	Service Monitor Default Group <a href="#">Show Members</a>	/var/lib/cloudera-service-monitor default value	The directory where Service Monitor data is stored. The Service Monitor stores metric time series and health information, as well as

Navigation: 1 2 3 4 5 6

Buttons: [Back](#) [Continue](#)

The Cluster Setup will run through 21 steps as seen here. This process will take about 10 minutes:

## Cluster Setup

### Progress

Command	Context	Status	Started at	Ended at
 <b>First Run</b>		In Progress	Oct 27, 2014 4:13:23 AM UTC	

### Command Progress

Completed 1 of 21 steps.



Initializing ZooKeeper Service  
Completed 1 steps successfully.



**Starting ZooKeeper Service**

[Details](#) 

Checking if the name directories of the NameNode are empty. Formatting HDFS only if empty.

Starting HDFS Service

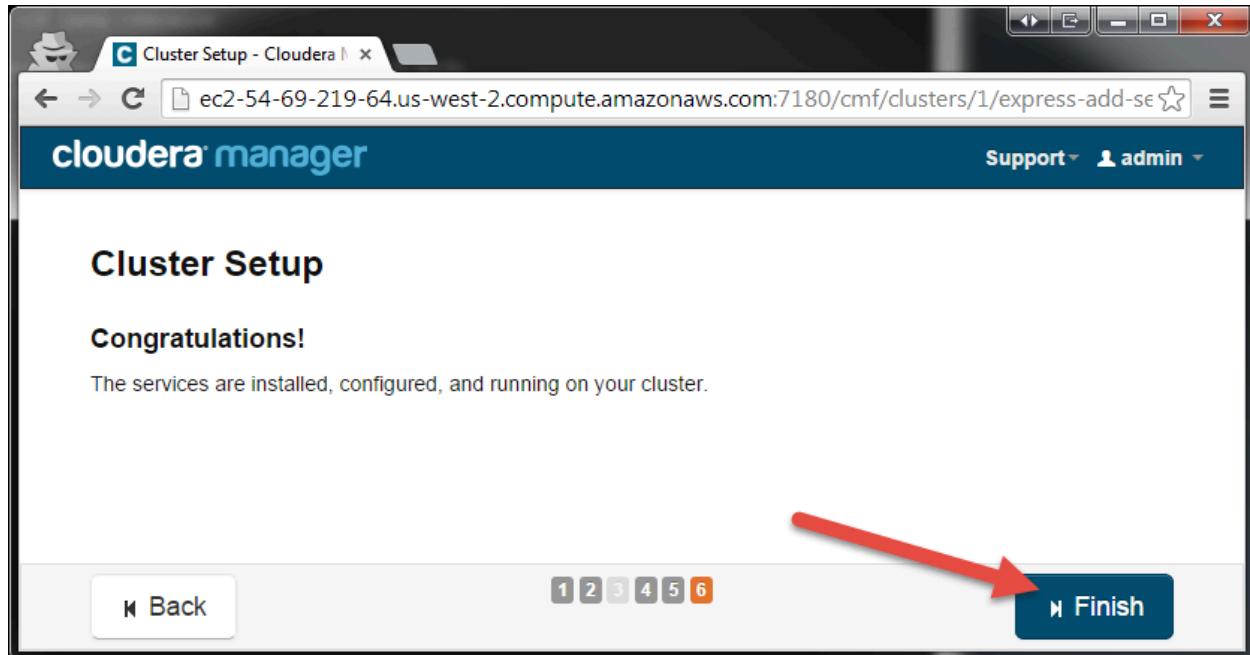
Creating HDFS /tmp directory

Click **Continue** when the 21 steps are finished:

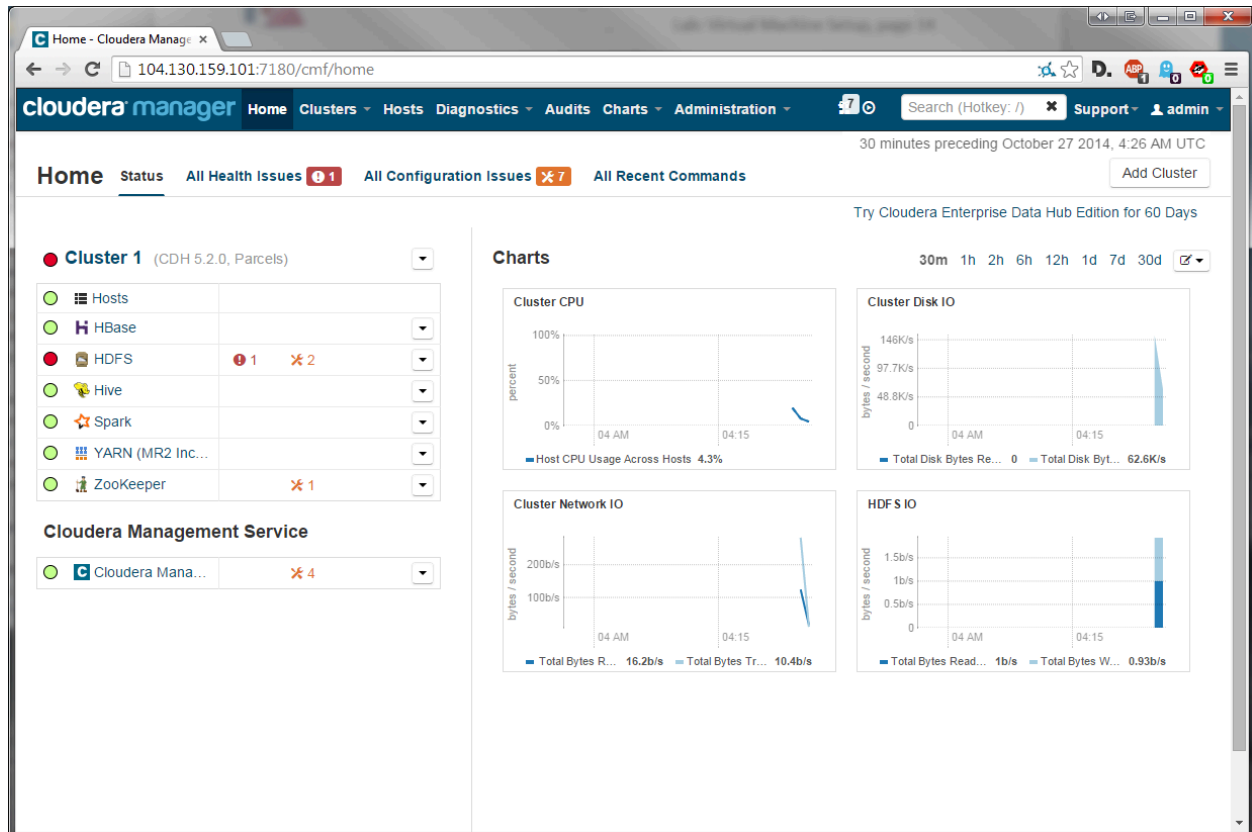




You are now finished with the CDH 5 installation and services startup! Click **Finish**:



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.



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

```
[root@cdh4-cm-vm01 ~]$ sudo /usr/java/jdk1.7.0_55-cloudera/bin/jps
15601 JobHistoryServer
3604 Main
13238 QuorumPeerMain
21781 Main
21742 EventCatcherService
13485 NameNode
19991 RunJar
15888 ResourceManager
14273 HRegionServer
14854 Canary
19977 RunJar
13525 SecondaryNameNode
13578 DataNode
23550 Jps
21806 AlertPublisher
21451 HistoryServer
14309 HMaster
15563 NodeManager
21758 Main
```

We will study these later on in the class.

Note, if you see any messages like the following, you can safely ignore them:

**You have mail in /var/spool/mail/root**

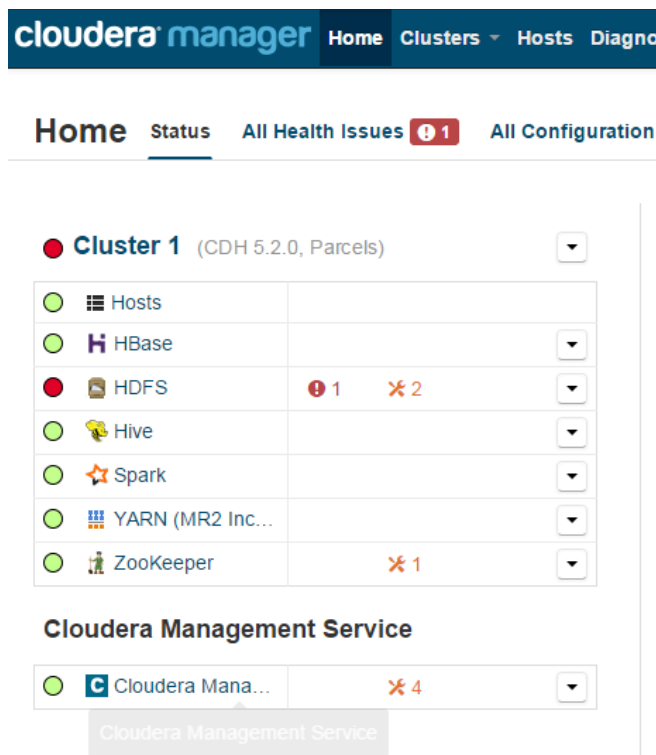
**These are just alerts from Cloudera Manager since there are some warnings and health alerts being generated (which we'll address soon). For now you can print out the alerts to verify this like so:**

```
[root@cdh5-cm-vm01 ~]# cat /var/spool/mail/root
From noreply@localhost.localdomain Mon Oct 27 04:23:58 2014
Return-Path: <noreply@localhost.localdomain>
X-Original-To: root@localhost
Delivered-To: root@localhost.localdomain
Received: from cdh5-cm-vm01 (localhost [127.0.0.1])
    by cdh5-cm-vm01.localdomain (Postfix) with ESMTP id 54967B431B
    for <root@localhost>; Mon, 27 Oct 2014 04:23:58 +0000 (UTC)
Date: Mon, 27 Oct 2014 04:23:58 +0000 (UTC)
From: noreply@localhost.localdomain
To: root@localhost.localdomain
```

Message-ID: <1073529197.0.1414383838322.JavaMail.cloudera-scm@localhost>  
 Subject: [Cloudera Alert] 5 Alerts since 4:23:08 AM  
 MIME-Version: 1.0  
 Content-Type: text/html; charset=UTF-8  
 Content-Transfer-Encoding: 7bit

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>[Cloudera Alert] 5 Alerts since 4:23:08 AM</title>
  </head>
```

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



The screenshot shows the Cloudera Manager web interface. At the top, there's a navigation bar with 'Home', 'Clusters', 'Hosts', and 'Diagnose'. Below this, a sub-navigation bar shows 'Home', 'Status', 'All Health Issues' (with a red icon and '1'), and 'All Configuration'. The main content area displays 'Cluster 1' (CDH 5.2.0, Parcels). A table lists the services and their status:

Service	Status	Alerts
Hosts	Running (Green)	0
HBase	Running (Green)	0
HDFS	Warning (Yellow)	1
Hive	Running (Green)	0
Spark	Running (Green)	0
YARN (MR2 Inc...)	Running (Green)	0
ZooKeeper	Running (Green)	0

Below the table, the 'Cloudera Management Service' is shown with a status of 'Warning (Yellow)' and 4 alerts.

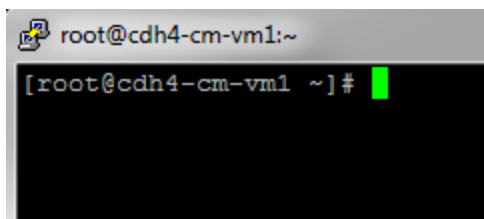
Note above that all services have actually started for us, including HDFS. HDFS is in a warning state, though, so we'll address that in the HDFS lab.

## HDFS Command Line

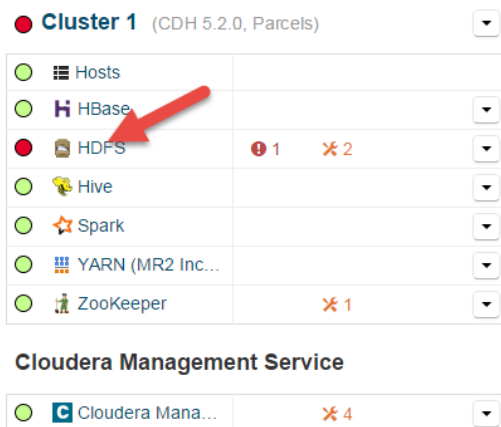
This 30 min lab introduces you to the HDFS command line. We will:

- 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 Rackspace VM command prompt as user 'root':



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:



## Health Tests [Expand All](#)

● 11 under replicated blocks in the cluster. 13 total blocks in the cluster. Percentage under replicated blocks: 84.62%. Critical threshold: 40.00%.

➤ ● 6 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 a 1-node cluster. The top navigation bar includes links for Home, Clusters, Hosts, Diagnostics, Audits, Charts, Administration, and a search bar. Below this, the 'HDFS' service page is selected, with sub-tabs for Status, Instances, Configuration, Commands, Audits, and Charts. A red arrow points to the 'Configuration' tab. The main content area is divided into two sections: 'HDFS Summary' and 'Charts'.

**HDFS Summary**

Configured Capacity	5.8 GiB/35.4 GiB
Quick Links	<a href="#">NameNode Web UI (Active)</a>
Event Search	<a href="#">Alerts</a> , <a href="#">Critical</a> , <a href="#">All</a>

**Status Summary**

SecondaryNameNode	● 1 Good Health
NameNode	● 1 Good Health (Active)
Balancer	○ None

**Charts**

30m 1h 2h 6h 12h 1d 7d 30d

**HDFS Capacity**

The chart shows HDFS Capacity over time. The y-axis represents bytes, with a marker at 18.6G. The x-axis shows time intervals from 04:15 to 04:30. The legend indicates: Configured Cap... 35.4G (blue line), HDFS Used 92.1M (blue bar), and Non-HDFS Used 5.7G (green bar).

**Total Bytes Read Across DataNodes**

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

Click on the validation warning:

[Cluster 1](#) »

**HDFS** Status Instances Configuration Commands Audits Charts ▾

### Configuration

1 validation warning below.

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 288 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	<input type="text" value="288"/> <input type="text" value="MiB"/> <input type="button" value="Reset to the default value: 1 GiB ↶"/>	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			

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:**

```
[root@cdh5-cm-vm01 ~]# free -g
```

	total	used	free	shared	buffers	cached
Mem:	14	11	3	0	0	6
-/+ buffers/cache:		4	10			
Swap:	0	0	0			

```
[root@cdh5-cm-vm01 ~]# df -Th
```

Filesystem	Type	Size	Used	Avail	Use%	Mounted on
/dev/xvda1	ext3	40G	7.8G	30G	21%	/
tmpfs	tmpfs	7.4G	0	7.4G	0%	/dev/shm
cm_processes	tmpfs	7.4G	6.0M	7.4G	1%	
/var/run/cloudera-scm-agent/process						

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

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

**HDFS** Status Instances Configuration Commands

### Configuration

Q filterWARNING X

⚠ 1 validation warning below. (Remove Filter)

Category	Property	Value
NameNode Default Group /	Java Heap Size of	2



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

**HDFS** Status Instances Configuration Commands Audits Charts ▾

Configuration

🔍 Search ✕ Role Groups 🗨

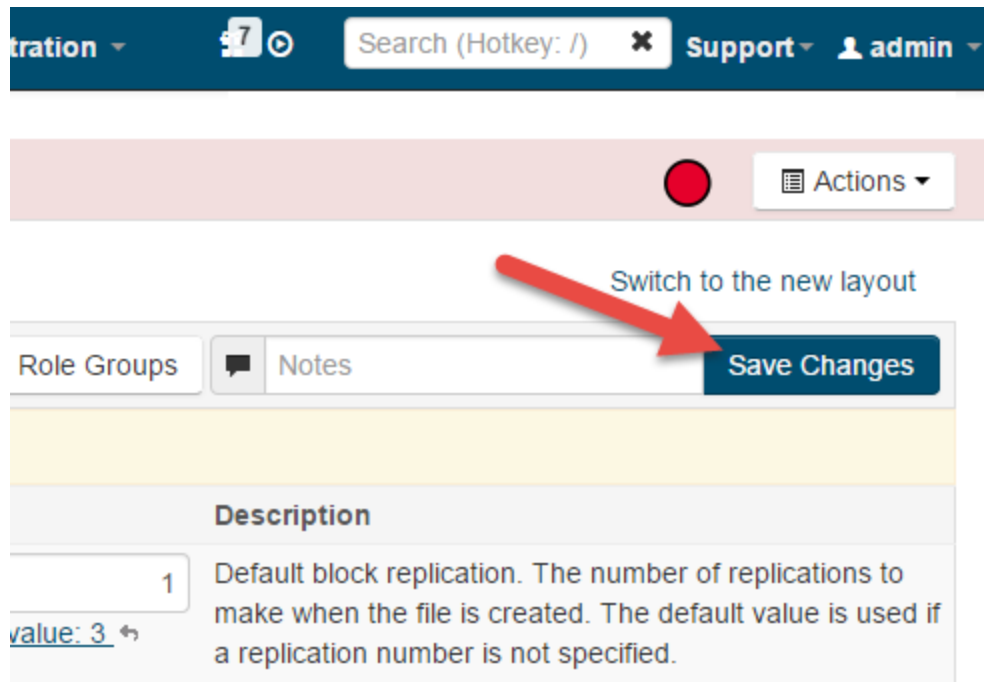
⚠ 1 validation warning below.

Category	Property	Value	Description
Service-Wide	Replication Factor dfs.replication	3 default value	Default block replication. The number of blocks replicated for each file is created. The specified.
Advanced	Minimal Block Replication dfs.replication.min, dfs.namenode.replication.min	1 default value	The minimal block replication.
Cloudera Navigator	Maximal Block Replication dfs.replication.max	512 default value	The maximal block replication.
High Availability			
Logs			
Monitoring			
Performance			
Ports and Addresses			
Proxy			
Replication			
Security			

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.

Property	Value	Description
Replication Factor dfs.replication	3 <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:

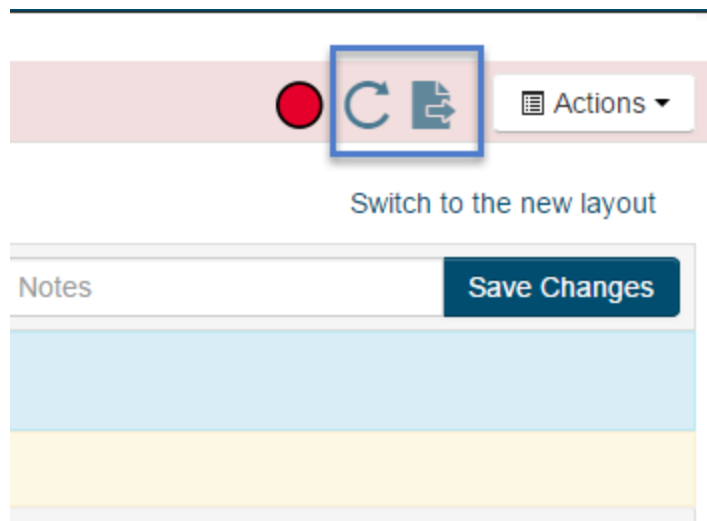


Saving the changes only applies it to Cloudera Manager's settings (stored in PostgreSQL) but not to the underlying Hadoop XML configuration files. We have to Deploy Client Configuration for the entire cluster and restart the HDFS service... then the new setting will take effect.

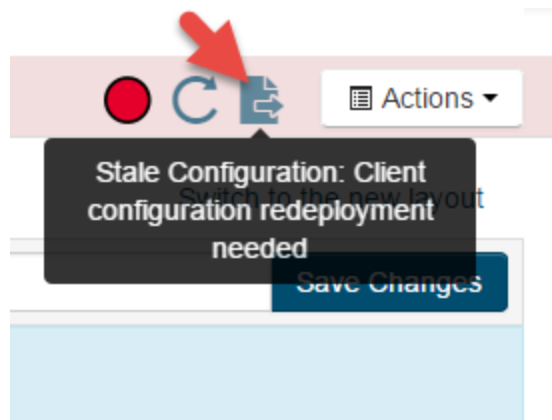
According to Cloudera: "Although Cloudera Manager will deploy client configuration files automatically in many cases, if you have modified the configurations for a service, you may need to redeploy those configuration files."

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

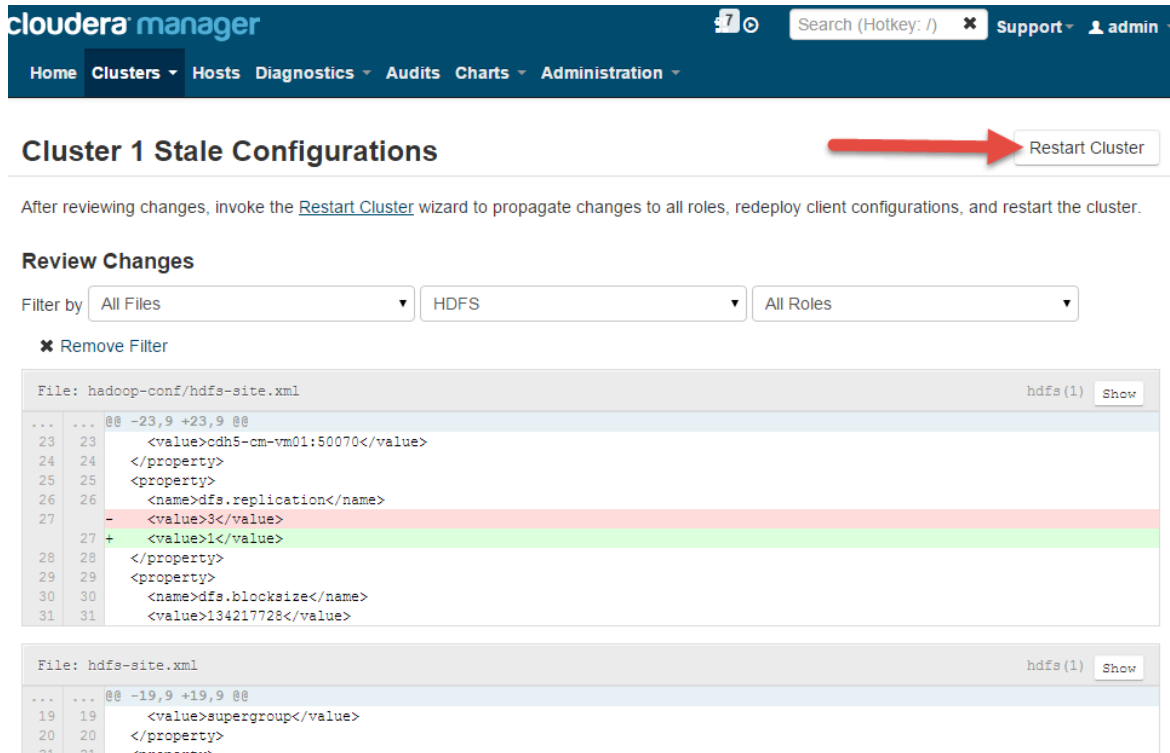
Notice that a couple of new icons have appeared to deploy client configuration and restart the HDFS service:



**Click the Deploy Client Configuration button:**



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



**cloudera manager** Search (Hotkey: /) Support admin

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 hdfs (1) Show

```

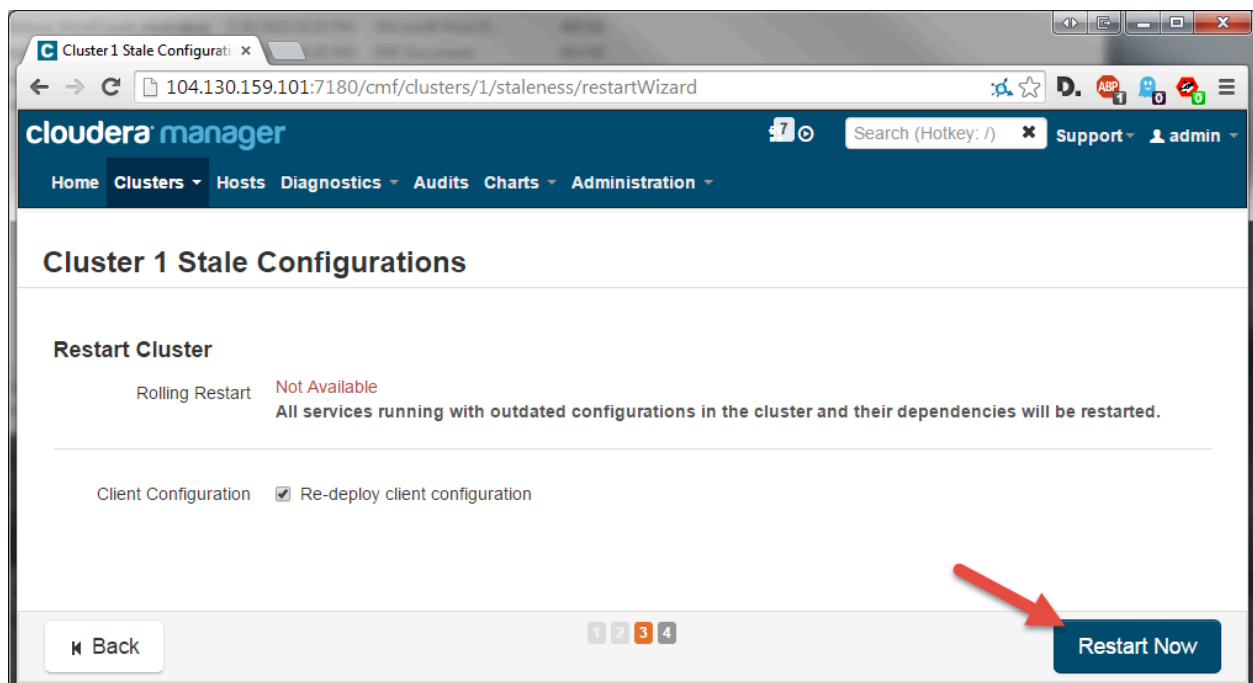
... @@ -23,9 +23,9 @@
23 23     <value>cdh5-cm-vm01:50070</value>
24 24   </property>
25 25   <property>
26 26     <name>dfs.replication</name>
27 26     <value>3</value>
27 +   <value>1</value>
28 28   </property>
29 29   <property>
30 30     <name>dfs.blocksize</name>
31 31     <value>134217728</value>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

File: hdfs-site.xml hdfs (1) Show

```

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

Click **Restart Now**:



**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 5 - 7 mins:

## Cluster 1 Stale Configurations

### Progress

Command	Context	Status	Started at	Ended at
Restart	<a href="#">Cluster 1</a>	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)	<a href="#">Cluster 1</a>	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	<a href="#">Cluster 1</a>	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
✓ <a href="#">Deploy Client Configuration</a> (5)	<a href="#">Cluster 1</a>	Finished	Oct 27, 2014 5:22:07 AM UTC	Oct 27, 2014 5:23:32 AM UTC
Successfully deployed all client configurations.				
✓ <a href="#">Start</a> (5)	<a href="#">Cluster 1</a>	Finished	Oct 27, 2014 5:18:24 AM UTC	Oct 27, 2014 5:22:07 AM UTC
All services successfully started.				
✓ <a href="#">Stop</a> (5)	<a href="#">Cluster 1</a>	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.

[Home](#)
[Status](#)
[All Health Issues 1](#)
[All Configurati...](#)

---

● **Cluster 1** (CDH 5.2.0, Parcels)

<span>●</span> Hosts		
<span>●</span> HBase		
<span>●</span> HDFS	<span>!</span> 1 <span>✖</span> 2	
<span>●</span> Hive		
<span>●</span> Spark		
<span>●</span> YARN (MR2 Inc...		
<span>●</span> ZooKeeper	<span>✖</span> 1	

**Cloudera Management Service**

<span>●</span> Cloudera Mana...	<span>✖</span> 4	
---------------------------------	------------------	--

**Switch to the cmd line and:**

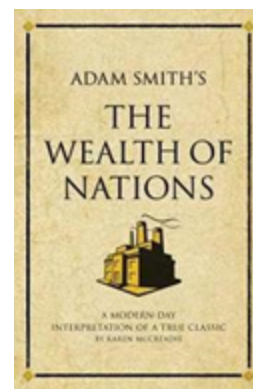
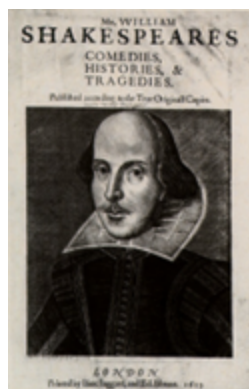
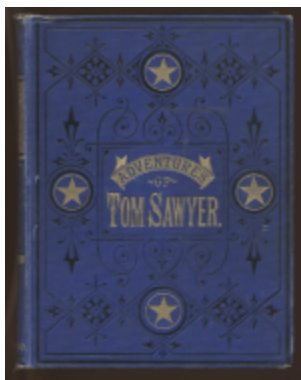
All three HDFS daemons (NameNode / DataNode/ SecondaryNameNode) start under the 'hdfs' user. To list all processes started under 'hdfs', run this command:

```
[root@cdh5-cm-vm01 ~]# ps U hdfs
  PID TTY          STAT       TIME COMMAND
 4395 ?            S1          0:09 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_secondarynamenode -Xmx1000m -
 4439 ?            S1          0:08 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_datanode -Xmx1000m -Dhdfs.aud
 4495 ?            S1          0:15 /usr/java/jdk1.7.0_67-cloudera/bin/java
-Dproc_namenode -Xmx1000m -Dhdfs.aud
```

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 /root folder:**

```
[root@cdh4-cm-vm01 ~]# pwd
/root
```

**Create a new folder to store the eBooks:**

```
[root@cdh4-cm-vm01 ~]# mkdir ebooks
```

```
[root@cdh4-cm-vm01 ~]# ls
cloudera-manager-installer.bin  ebooks
```

```
[root@cdh4-cm-vm01 ~]# cd ebooks
```

**Download the three eBooks:**

```
[root@cdh4-cm-vm01 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
```

```
[root@cdh4-cm-vm01 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
```

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 ebooks]# ls
shakespeare.txt  tom_sawyer.txt  wealth_of_nations.txt

[root@cdh4-cm-vm01 ebooks]# head -10 shakespeare.txt
i»¿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:**

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls /
Found 3 items
drwxr-xr-x   - hbase hbase                0 2013-03-26 07:14 /hbase
drwxrwxrwt   - hdfs supergroup            0 2013-03-26 07:14 /tmp
drwxr-xr-x   - hdfs supergroup            0 2013-03-26 07:17 /user
```

We can see that there are three 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:**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls /user
Found 4 items
drwxrwxrwx   - mapred hadoop              0 2014-10-27 04:17 /user/history
drwxrwxr-t   - hive  hive                  0 2014-10-27 04:19 /user/hive
drwxrwxr-x   - hue   hue                   0 2014-10-27 04:23 /user/hue
drwxr-x--x   - spark spark                0 2014-10-27 04:21 /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':**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls
ls: `.': No such file or directory
```

This error was to be expected, since the /user/root 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)

```
[root@cdh4-cm-vm01 ebooks]# sudo -u hdfs hadoop fs -mkdir /user/root
[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/root/`.

### Check the permissions for the `/user/root` folder in HDFS:

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls /user
Found 5 items
drwxrwxrwx   - mapred  hadoop           0 2014-10-27 04:17 /user/history
drwxrwxr-t   - hive    hive           0 2014-10-27 04:19 /user/hive
drwxrwxr-x   - hue     hue           0 2014-10-27 04:23 /user/hue
drwxr-xr-x   - hdfs    supergroup    0 2014-10-27 05:39 /user/root
drwxr-x--x   - spark   spark         0 2014-10-27 04:21 /user/spark
```

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

### Change the owner of `/user/root` to be root:

```
[root@cdh4-cm-vm01 ebooks]# sudo -u hdfs hadoop fs -chown root /user/root
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls /user
Found 5 items
drwxrwxrwx   - mapred  hadoop           0 2014-10-27 04:17 /user/history
drwxrwxr-t   - hive    hive           0 2014-10-27 04:19 /user/hive
drwxrwxr-x   - hue     hue           0 2014-10-27 04:23 /user/hue
drwxr-xr-x   - root    supergroup      0 2014-10-27 05:39 /user/root
drwxr-x--x   - spark   spark         0 2014-10-27 04:21 /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:

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -mkdir gutenberg_input
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls /user/root
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/root/ 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):**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -copyFromLocal tom_sawyer.txt
/user/root/gutenberg_input/
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -copyFromLocal shakespeare.txt
/user/root/gutenberg_input/
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -copyFromLocal wealth_of_nations.txt
/user/root/gutenberg_input/
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls gutenberg_input
Found 3 items
-rw-r--r--   1 root supergroup    5590193 2013-03-27 00:02
gutenberg_input/shakespeare.txt
-rw-r--r--   1 root supergroup    416148 2013-03-27 00:02
gutenberg_input/tom_sawyer.txt
-rw-r--r--   1 root supergroup    2276935 2013-03-27 00:02
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/root is assumed.

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

```
[root@cdh4-cm-vm01 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 gutenber**g\_input directory:

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 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:**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -touchz zerofile.txt
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls
```

Found 2 items

```
drwxr-xr-x  - root supergroup          0 2013-07-16 05:37 gutenberg_input
-rw-r--r--  1 root supergroup          0 2013-07-16 05:40 zerofile.txt
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -rm zerofile.txt
```

```
Moved: 'hdfs://cdh4-cm-vm32:8020/user/root/zerofile.txt' to trash at:
hdfs://cdh4-cm-vm1:8020/user/root/.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:**

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls
Found 2 items
drwx----- - root supergroup          0 2013-07-16 05:42 .Trash
drwxr-xr-x - root supergroup          0 2013-07-16 05:37 gutenber_input
```

```
[root@cdh4-cm-vm01 ebooks]# hadoop fs -ls
/user/root/.Trash/Current/user/root/
Found 1 items
-rw-r--r--  1 root supergroup          0 2013-07-16 05:40
.Trash/Current/user/root/zerofile.txt
```

**Now delete everything in Trash:**

```
[root@cdh4-cm-vm01 ~]# 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:**

```
[root@cdh4-cm-vm01 ~]# hadoop fs -ls /user/root/.Trash/Current/user/root
ls: `/user/root/.Trash/Current/user/root': 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
.
```



```
/hbase/data/hbase/meta/.tabledesc/.tableinfo.0000000001: Under replicated  
BP-1545055139-104.130.159.101-1414383262813:blk_1073741829_1005. Target  
Replicas is 3 but found 1 replica(s).
```

*<output from the rest of the under replicated blocks is hidden>*

```
Status: HEALTHY  
Total size: 103854164 B (Total open files size: 166 B)  
Total dirs: 150  
Total files: 14  
Total symlinks: 0 (Files currently being written: 2)  
Total blocks (validated): 13 (avg. block size 7988781 B) (Total open  
file blocks (not validated): 2)  
Minimally replicated blocks: 13 (100.0 %)  
Over-replicated blocks: 0 (0.0 %)  
Under-replicated blocks: 10 (76.92308 %)  
Mis-replicated blocks: 0 (0.0 %)  
Default replication factor: 1  
Average block replication: 1.0  
Corrupt blocks: 0  
Missing replicas: 20 (60.60606 %)  
Number of data-nodes: 1  
Number of racks: 1  
FSCK ended at Mon Oct 27 05:50:41 UTC 2014 in 25 milliseconds
```

The filesystem under path '/' is HEALTHY

You will notice that there are currently 14 files in HDFS and the block replication factor is 1 (as opposed to 3 in production clusters). There are 10 under replicated blocks and 20 missing replicas. 3 out of the 14 files are the Project Gutenberg eBooks that we uploaded.

This makes sense if you think about it. The 10 under replicated blocks were written to HDFS by the installer. Since 2 replicas of these 10 blocks are missing,  $10 \text{ blocks} \times 2 = 20$ .

**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:**

```
[root@cdh4-cm-vm01 ebooks]# sudo -u hdfs hadoop fs -setrep -R -w 1 /
Replication 1 set: /hbase/-ROOT-/.tableinfo.0000000001
Replication 1 set: /hbase/-ROOT-/70236052/.oldlogs/hlog.1373949222869
Replication 1 set: /hbase/-ROOT-/70236052/.regioninfo
Replication 1 set:
/hbase/-ROOT-/70236052/info/4cd35eddb16d4d00a46e9d7dfacadb26
```

*<output for the rest of the blocks is hidden>*

**Now, check the health status of HDFS again:**

```
[root@cdh4-cm-vm01 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://cdh5-cm-vm01:50070
FSCK started by hdfs (auth:SIMPLE) from /104.130.159.101 for path / at Mon
Oct 27 05:56:32 UTC 2014
.....Status: HEALTHY
Total size:      103854164 B (Total open files size: 166 B)
Total dirs:      156
Total files:      14
Total symlinks:      0 (Files currently being written: 2)
Total blocks (validated):      13 (avg. block size 7988781 B) (Total open
file blocks (not validated): 2)
Minimally replicated blocks:      13 (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 Mon Oct 27 05:56:32 UTC 2014 in 25 milliseconds
```

The filesystem under path '/' is HEALTHY

There we go.... a healthy cluster. That red annoying HDFS Bad Health warning in the Cloudera Manager GUI should go away (but 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 Rackspace Dallas datacenter, then the repercussions could mean data loss for us. Cross your fingers and continue the lab...

**Cluster 1** (CDH 5.2.0, Parcels) ▼

Hosts		
HBase		▼
HDFS	✖ 2	▼
Hive		▼
Spark		▼
YARN (MR2 Inc...)		▼
ZooKeeper	✖ 1	▼

**Cloudera Management Service**

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

Cloudera Management Service

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:

```
[root@cdh4-cm-vm01 ebooks]# vi /etc/hadoop/conf/hdfs-site.xml
```

If using nano:

```
[root@cdh4-cm-vm01 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 interface for configuring HDFS. Under the 'DataNode Default Group', the 'DataNode Data Directory' property is set to `/dfs/dn`. A green circle highlights the value field. The description explains that this is a comma-delimited list of local file system directories where the DataNode stores HDFS block data, with typical values like `/data/N/dfs/dn` for `N = 1, 2, 3..` and that these should be mounted using the `noatime` option.

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

```
[root@cdh4-cm-vm01 ebooks]# cd /dfs/dn
```

```
[root@cdh4-cm-vm01 dn]# ls
current  in_use.lock
```

```
[root@cdh4-cm-vm01 dn]# cd current
```

```
[root@cdh4-cm-vm01 current]# ls
BP-930791293-108.166.81.199-1346079203773  VERSION
```

**WARNING:** In the command below, you will have to replace the `#` with whatever unique `#` appears in your machine! Try just typing `cd BP-` and then hitting `TAB` on your keyboard to autocomplete.

```
[root@cdh4-cm-vm01 current]# cd BP-930791293-108.166.81.199-1346079203773
```

```
[root@cdh4-cm-vm01 BP-930791293-108.166.81.199-1346079203773]# ls
current                                dncp_block_verification.log.prev
dncp_block_verification.log.curr      tmp
```

```
[root@cdh4-cm-vm01 BP-930791293-108.166.81.199-1346079203773]# cd current/
```

```
[root@cdh4-cm-vm01 current]# ls
finalized rbw VERSION
```

```
[root@cdh4-cm-vm01 current]# cd finalized/
```

```
[root@cdh4-cm-vm01 finalized]# pwd
/dfs/dn/current/BP-930791293-108.166.81.199-1346079203773/current/finalized
```

```
[root@cdh4-cm-vm01 finalized]# ls
subdir0
```

```
[root@cdh5-cm-vm01 finalized]# cd subdir0/
```

```
[root@cdh5-cm-vm01 subdir0]# ls
subdir0
```

```
[root@cdh5-cm-vm01 subdir0]# cd subdir0/
```

```
[root@cdh5-cm-vm01 subdir0]# ls
blk_1073741825          blk_1073741828_1004.meta  blk_1073741837
blk_1073741923_1099.meta
blk_1073741825_1001.meta  blk_1073741829          blk_1073741837_1013.meta
blk_1073741924
blk_1073741826          blk_1073741829_1005.meta  blk_1073741893
blk_1073741924_1100.meta
blk_1073741826_1002.meta  blk_1073741832          blk_1073741893_1069.meta
blk_1073741925
blk_1073741827          blk_1073741832_1008.meta  blk_1073741894
blk_1073741925_1101.meta
blk_1073741827_1003.meta  blk_1073741833          blk_1073741894_1070.meta
blk_1073741828          blk_1073741833_1009.meta  blk_1073741923
```

```
[root@cdh5-cm-vm01 subdir0]# pwd
/dfs/dn/current/BP-1545055139-104.130.159.101-1414383262813/current/finalized
/subdir0/subdir0
```

You will notice that each block file has a corresponding meta file with the checksum and generation stamp for the block. There are also many sub directories named 'subdir#'.

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.

```
[root@cdh5-cm-vm01 subdir0]# sudo -u hdfs hdfs fsck
/user/root/gutenberg_input/tom_sawyer.txt -files -blocks -racks
14/10/27 06:05:19 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
/user/root/gutenberg_input/tom_sawyer.txt at Mon Oct 27 06:05:20 UTC 2014
/user/root/gutenberg_input/tom_sawyer.txt 416148 bytes, 1 block(s): OK
0. BP-1545055139-104.130.159.101-1414383262813:blk_1073741923_1099 len=416148
repl=1 [/default/104.130.159.101:50010]
```

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 Mon Oct 27 06:05:20 UTC 2014 in 1 milliseconds

The filesystem under path '/user/root/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\_1073741923\_1099

Well, the actual block id is just the # preceding the underscore, so: blk\_1073741923

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:

```
[root@cdh5-cm-vm01 subdir0]# find . -name "blk_1073741923"
./blk_1073741923
```

So, the file in ext3 representing the Tom Sawyer block is in the same linux directory that you're in.

**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**

The screenshot shows the Cloudera Manager interface. The top navigation bar includes links for Home, Clusters, Hosts, Diagnostics, Audits, Charts, and Administration. Below this, the 'Cluster 1' section is active, showing tabs for HDFS, Status, Instances, Configuration, Commands, Audits, and Charts. The 'Configuration' tab is selected, displaying a table of properties. A search bar and 'Role Groups' and 'Notes' buttons are at the top of the configuration area. A yellow warning banner indicates '1 validation warning below.' The table has four columns: Category, Property, Value, and Description. The 'NameNode Default Group' is selected in the left sidebar. The 'NameNode Data Directories' property is highlighted, showing its value as '/dfs/nn', which is circled in green. A '+', 'Reset to empty default value', and a '-' button are next to the value field. The 'NameNode Edits Directories' property is also visible below it.

Category	Property	Value	Description
Service-Wide	NameNode Data Directories	/dfs/nn	Determines the local file system directory where NameNode metadata files are stored. Default value is /dfs/nn.
Balancer Default Group	dfs.name.dir, dfs.namenode.name.dir		
DataNode Default Group			
Failover Controller Default Group			
Gateway Default Group			
HttpFS Default Group			
JournalNode Default Group			
NFS Gateway Default Group			
NameNode Default Group	NameNode Edits Directories	Default value is empty. Click to edit.	Directory where NameNode metadata files are stored.
SecondaryNameNode Default Group	dfs.namenode.edits.dir		

Let's explore that now.

```
[root@cdh4-cm-vm01 finalized]# cd /dfs/nn
```



```
[root@cdh4-cm-vm01 nn]# ls
current  in_use.lock
```

```
[root@cdh4-cm-vm01 nn]# cd current
```

```
[root@cdh4-cm-vm01 current]# ls
edits_00000000000000000001-00000000000000000004
fsimage_0000000000000000000612.md5
edits_00000000000000000005-000000000000000000612  fsimage_0000000000000000000695
edits_0000000000000000000613-000000000000000000674
fsimage_0000000000000000000695.md5
edits_0000000000000000000675-000000000000000000695  seen_txid
edits_inprogress_0000000000000000000696              VERSION
fsimage_0000000000000000000612
```

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:

```
[root@cdh4-cm-vm01 current]# cd ~
```

```
[root@cdh4-cm-vm01 ~]# sudo -u hdfs hdfs dfsadmin -report
Configured Capacity: 38047245927 (35.43 GB)
Present Capacity: 31897604096 (29.71 GB)
DFS Remaining: 31792676864 (29.61 GB)
DFS Used: 104927232 (100.07 MB)
DFS Used%: 0.33%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
```

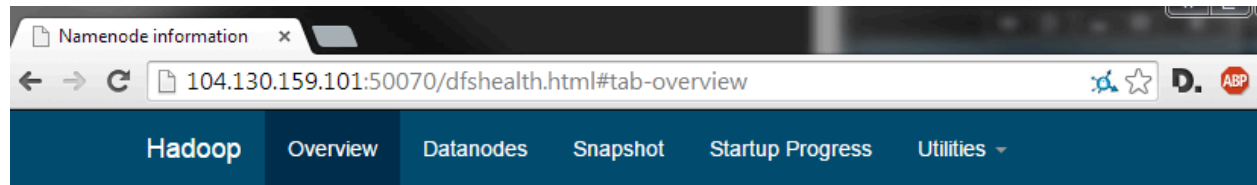
```
-----
Live datanodes (1):
```

```
Name: 104.130.159.101:50010 (cdh5-cm-vm01)
Hostname: cdh5-cm-vm01
```

Rack: /default  
Decommission Status : Normal  
Configured Capacity: 38047245927 (35.43 GB)  
DFS Used: 104927232 (100.07 MB)  
Non DFS Used: 6149641831 (5.73 GB)  
DFS Remaining: 31792676864 (29.61 GB)  
DFS Used%: 0.28%  
DFS Remaining%: 83.56%  
Configured Cache Capacity: 1051721728 (1003 MB)  
Cache Used: 0 (0 B)  
Cache Remaining: 1051721728 (1003 MB)  
Cache Used%: 0.00%  
Cache Remaining%: 100.00%  
Xceivers: 6  
Last contact: Mon Oct 27 06:10:25 UTC 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**



## 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.

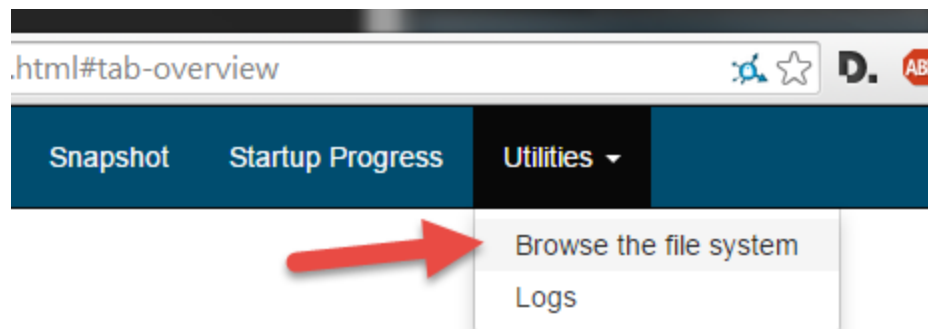
190 files and directories, 15 blocks = 205 total filesystem object(s).

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.



vm01:8020' (active)

## Browse Directory

<input type="text" value="/"/>						Go!
Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hbase	hbase	0 B	0	0 B	<a href="#">hbase</a>
drwxrwxrwt	hdfs	supergroup	0 B	0	0 B	<a href="#">tmp</a>
drwxr-xr-x	hdfs	supergroup	0 B	0	0 B	<a href="#">user</a>

Try navigating to the Project Gutenberg files on your own:

# Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	root	supergroup	5.33 MB	1	128 MB	<a href="#">shakespeare.txt</a>
-rw-r--r--	root	supergroup	406.39 KB	1	128 MB	<a href="#">tom_sawyer.txt</a>
-rw-r--r--	root	supergroup	2.17 MB	1	128 MB	<a href="#">wealth_of_nations.txt</a>

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

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

SecondaryNameNode info x
104.130.159.101:50090/status.html

Hadoop Overview

## 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

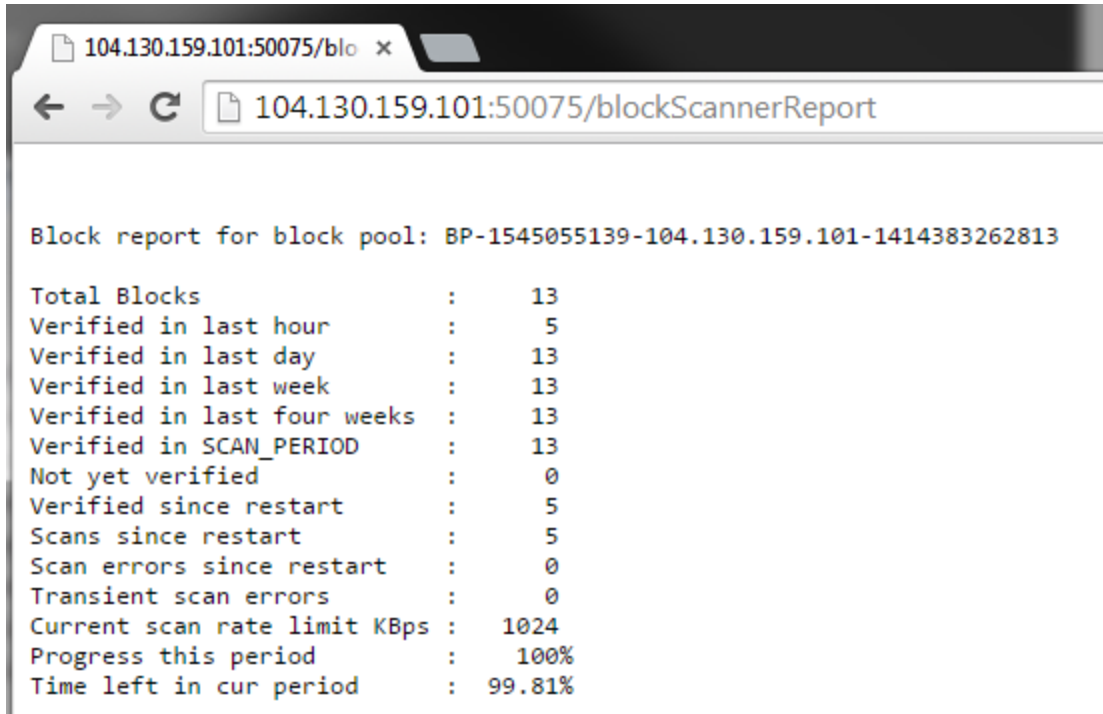
- file:///dfs/snn

### 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 a single tab titled "104.130.159.101:50075/blo x". The address bar contains the URL "104.130.159.101:50075/blockScannerReport". The main content area displays a block report for a specific block pool. The report is formatted as a text-based table with two columns: a description of the metric and its corresponding value.

```
Block report for block pool: BP-1545055139-104.130.159.101-1414383262813

Total Blocks                :      13
Verified in last hour       :        5
Verified in last day        :      13
Verified in last week       :      13
Verified in last four weeks :      13
Verified in SCAN_PERIOD     :      13
Not yet verified            :        0
Verified since restart       :        5
Scans since restart          :        5
Scan errors since restart    :        0
Transient scan errors        :        0
Current scan rate limit KBps :    1024
Progress this period         :    100%
Time left in cur period      :   99.81%
```

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

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

```

104.130.159.101:50075/blk
104.130.159.101:50075/blockScannerReport?listblocks

Block report for block pool: BP-1545055139-104.130.159.101-1414383262813
blk_1073741828_1004 : status : ok type : local scan time : 4337173 1970-01-01 01:12:17,173
blk_1073741825_1001 : status : ok type : local scan time : 4337175 1970-01-01 01:12:17,175
blk_1073741829_1005 : status : ok type : local scan time : 4337175 1970-01-01 01:12:17,175
blk_1073741826_1002 : status : ok type : local scan time : 4337176 1970-01-01 01:12:17,176
blk_1073741827_1003 : status : ok type : local scan time : 4337176 1970-01-01 01:12:17,176
blk_1073741832_1008 : status : ok type : local scan time : 4342199 1970-01-01 01:12:22,199
blk_1073741833_1009 : status : ok type : local scan time : 4342200 1970-01-01 01:12:22,200
blk_1073741837_1013 : status : ok type : local scan time : 4767470 1970-01-01 01:19:27,470
blk_1073741893_1069 : status : ok type : local scan time : 7957738 1970-01-01 02:12:37,738
blk_1073741894_1070 : status : ok type : local scan time : 7962743 1970-01-01 02:12:42,743
blk_1073741923_1099 : status : ok type : local scan time : 9628048 1970-01-01 02:40:28,048
blk_1073741924_1100 : status : ok type : local scan time : 9637088 1970-01-01 02:40:37,088
blk_1073741925_1101 : status : ok type : local scan time : 9644103 1970-01-01 02:40:44,103

Total Blocks : 13
Verified in last hour : 5
Verified in last day : 13
Verified in last week : 13
Verified in last four weeks : 13
Verified in SCAN_PERIOD : 13
Not yet verified : 0
Verified since restart : 5
Scans since restart : 5
Scan errors since restart : 0
Transient scan errors : 0
Current scan rate limit KBps : 1024
Progress this period : 100%
Time left in cur period : 99.81%

```

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.

Please email me at [sameer@blueplastic.com](mailto:sameer@blueplastic.com) with any feedback about this lab and corrections you've identified.

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=ziqx2hJY8Hq>

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

<http://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/>

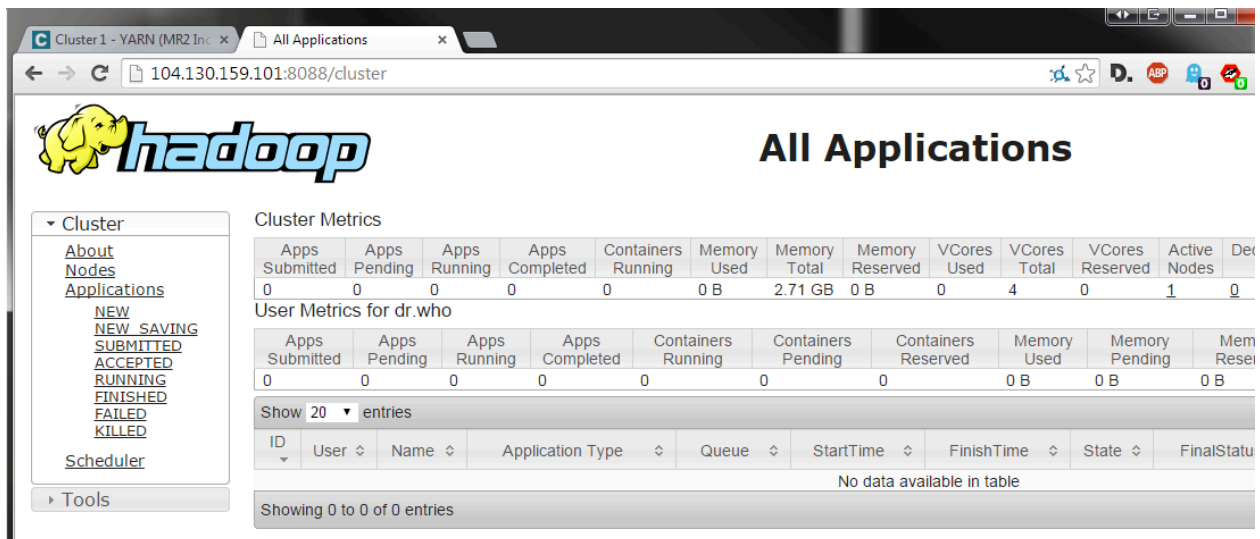


## Spark and YARN

Let's explore YARN at a high level first.

Take a look at the YARN resource manager UI:

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



The screenshot shows the Hadoop YARN Resource Manager UI. The browser address bar displays `104.130.159.101:8088/cluster`. The page title is "All Applications". On the left, a sidebar menu includes "Cluster", "About", "Nodes", "Applications" (selected), "NEW", "NEW SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED", and "Scheduler". The main content area displays "Cluster Metrics" and "User Metrics for dr.who". Both tables show zero values for all metrics. Below the tables, a message states "No data available in table" and "Showing 0 to 0 of 0 entries".

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Deallocated Nodes
0	0	0	0	0	0 B	2.71 GB	0 B	0	4	0	1	0

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B

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>

Okay, nothing there either.

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

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

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

```
[root@cdh5-cm-vm01 ~]# cat
/etc/spark/conf.cloudera.spark_on_yarn/spark-defaults.conf
spark.eventLog.dir=hdfs://cdh5-cm-vm01:8020/user/spark/applicationHistory
spark.eventLog.enabled=true
spark.yarn.historyServer.address=http://cdh5-cm-vm01:18088
```

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

```
[root@cdh5-cm-vm01 ~]# ls /etc/spark/conf.cloudera.spark_on_yarn/
log4j.properties  spark-defaults.conf  spark-env.sh
```

```
[root@cdh5-cm-vm01 ~]# 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-5.2.0-1.cdh5.2.0.p0.36/lib/spark
export
DEFAULT_HADOOP_HOME=/opt/cloudera/parcels/CDH-5.2.0-1.cdh5.2.0.p0.36/lib/hado
op

### Path of Spark assembly jar in HDFS
export
SPARK_JAR_HDFS_PATH=${SPARK_JAR_HDFS_PATH:-/user/spark/share/lib/spark-assembly.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.**

```
[root@cdh5-cm-vm01 conf.cloudera.spark_on_yarn]# 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:**

```
[root@cdh5-cm-vm01 ~]# 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:**

```
[root@cdh5-cm-vm01 ~]# cd /var/log/spark/
```

```
[root@cdh5-cm-vm01 spark]# ls
spark-history-server-cdh5-cm-vm01.log
```

```
[root@cdh5-cm-vm01 spark]# tail -3 spark-history-server-cdh5-cm-vm01.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:**

```
[root@cdh5-cm-vm01 conf.cloudera.spark_on_yarn]# cd tmp

[root@cdh5-cm-vm01 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]
```

```
[root@cdh5-cm-vm01 tmp]# cat animals.txt
cat cat
dog dog
fish fish fish fish fish
```

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

```
[root@cdh5-cm-vm01 tmp]# sudo -u spark hadoop fs -copyFromLocal animals.txt
/user/spark/

[root@cdh5-cm-vm01 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:**

```
[root@cdh5-cm-vm01 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-assembly-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
/_ \
```

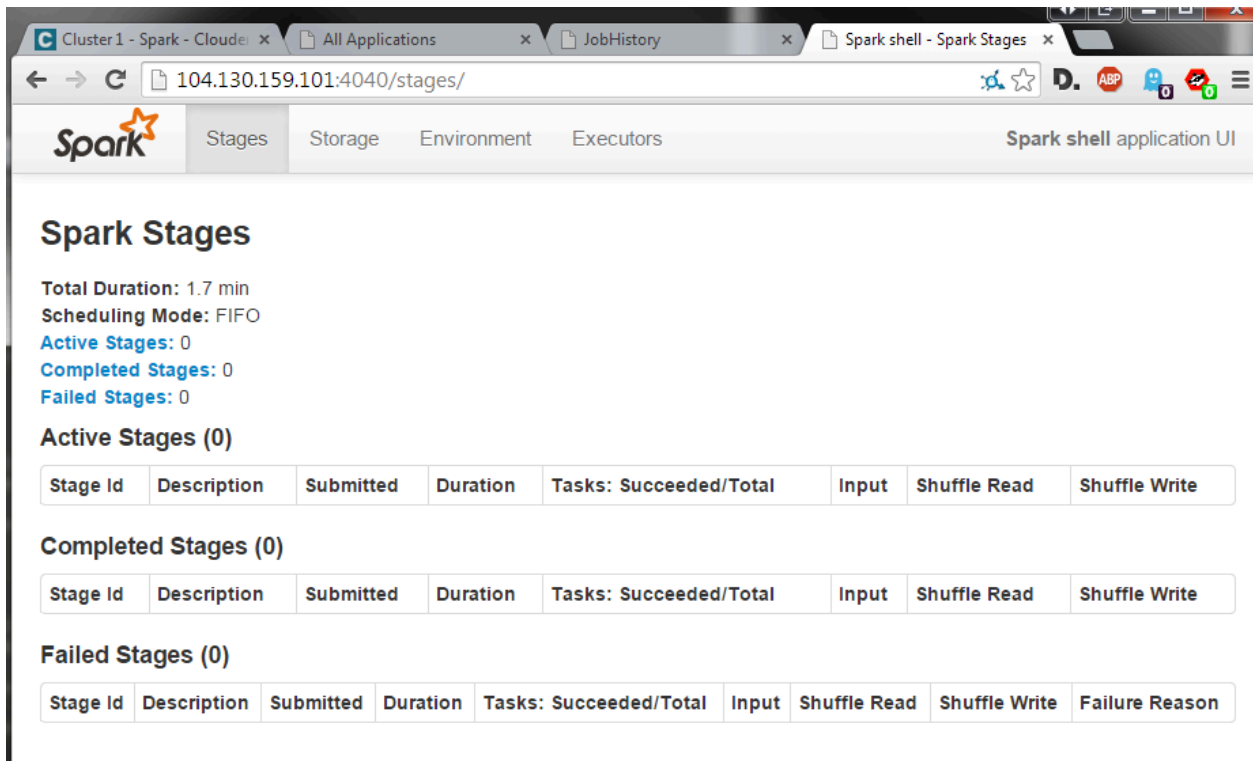
```
Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc.
```

```
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`**



**Spark Stages**

Total Duration: 1.7 min  
 Scheduling Mode: FIFO  
 Active Stages: 0  
 Completed Stages: 0  
 Failed Stages: 0

**Active Stages (0)**

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

**Completed Stages (0)**

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

**Failed Stages (0)**

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

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
```

```
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.
```

<code>:cp &lt;path&gt;</code>	add a jar or directory to the classpath
<code>:help [command]</code>	print this summary or command-specific help
<code>:history [num]</code>	show the history (optional num is commands to show)
<code>:h? &lt;string&gt;</code>	search the history
<code>:imports [name name ...]</code>	show import history, identifying sources of names
<code>:implicit &lt;-v&gt;</code>	show the implicit in scope
<code>:javap &lt;path class&gt;</code>	disassemble a file or class name



<code>:load &lt;path&gt;</code>	load and interpret a Scala file
<code>:paste</code>	enter paste mode: all input up to ctrl-D compiled together
<code>:quit</code>	exit the repl
<code>:replay</code>	reset execution and replay all previous commands
<code>:reset</code>	reset the repl to its initial state, forgetting all session entries
<code>:sh &lt;command line&gt;</code>	run a shell command (result is implicitly => List[String])
<code>:silent</code>	disable/enable automatic printing of results
<code>:fallback</code>	disable/enable advanced repl changes, these fix some issues but may introduce others.
This mode will be removed once these fixes stabilize	
<code>:type [-v] &lt;expr&gt;</code>	display the type of an expression without evaluating it
<code>:warnings</code>	show the suppressed warnings from the most recent line which had any

```
scala> fileBaseRDD.count()
java.net.ConnectException: Call From cdh5-cm-vm01/104.130.159.101 to
localhost:8020 failed on connection exception: java.net.ConnectException:
Connection refused; For more details see:
http://wiki.apache.org/hadoop/ConnectionRefused
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native
Method)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccess
orImpl.java:57)
    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstruct
orAccessorImpl.java:45)
    at java.lang.reflect.Constructor.newInstance(Constructor.java:526)
    at org.apache.hadoop.net.NetUtils.wrapWithMessage(NetUtils.java:783)
    at org.apache.hadoop.net.NetUtils.wrapException(NetUtils.java:730)
```

**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  
IP>:8020/user/spark/animals.txt")  
fileBaseRDD: org.apache.spark.rdd.RDD[String] =  
hdfs://104.130.159.101:8020/user/spark/animals.txt MappedRDD[3] 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 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  
IP>: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 root user, but are trying to write to HDFS into the Spark users' dir, which gives us an error.

**Try writing the results to the root users' dir:**

```
scala>  
counts.saveAsTextFile("hdfs://104.130.159.101:8020/user/root/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	<a href="#">saveAsTextFile at &lt;console&gt;:17</a> <a href="#">+details</a>	2014/10/27 06:57:56	0.1 s	<a href="#">2/2</a>			
2	<a href="#">collect at &lt;console&gt;:17</a> <a href="#">+details</a>	2014/10/27 06:57:15	34 ms	<a href="#">2/2</a>			
3	<a href="#">map at &lt;console&gt;:14</a> <a href="#">+details</a>	2014/10/27 06:57:15	68 ms	<a href="#">2/2</a>	44.0 B		369.0 B
1	<a href="#">collect at &lt;console&gt;:15</a> <a href="#">+details</a>	2014/10/27 06:56:59	19 ms	<a href="#">2/2</a>	44.0 B		
0	<a href="#">count at &lt;console&gt;:15</a> <a href="#">+details</a>	2014/10/27 06:56:55	0.1 s	<a href="#">2/2</a>	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:

104.130.159.101:4040/storage/

Spark Stages **Storage** Environment Executors Spark shell application UI

### Storage

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

Check out the Environment tab:

104.130.159.101:4040/environment/

Spark Stages Storage **Environment** Executors Spark shell application

### Environment

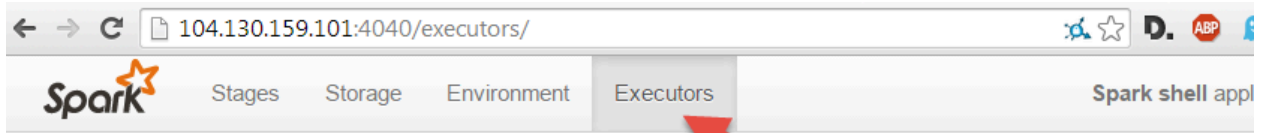
#### Runtime Information

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.fileserver.uri	http://104.130.159.101:40943
spark.jars	
spark.master	local[*]

And the Executors' tab:



## Executors (1)

**Memory:** 0.0 B Used (265.4 MB Total)

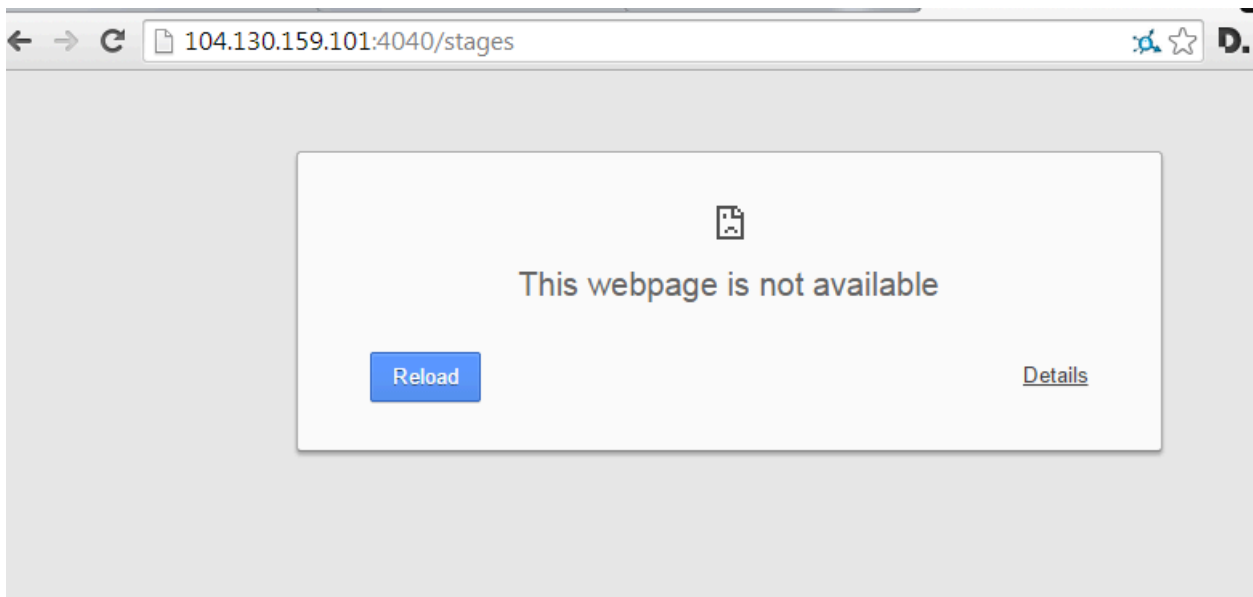
**Disk:** 0.0 B Used

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	0.0 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):



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:**

```
[root@cdh5-cm-vm01 tmp]# hadoop fs -ls /user/root/
Found 3 items
drwx----- - root supergroup          0 2014-10-27 07:04 /user/root/.Trash
drwxr-xr-x - root supergroup          0 2014-10-27 05:44
/user/root/gutenberg_input
drwxr-xr-x - root supergroup          0 2014-10-27 06:55
/user/root/wc-results.txt
```

**Notice that wc-results.txt is actually a directory!**

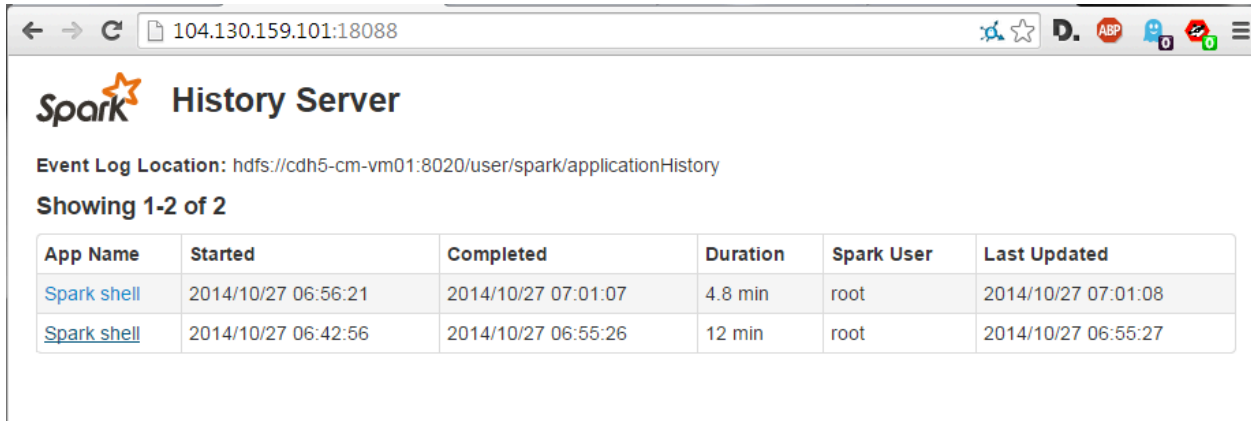
```
[root@cdh5-cm-vm01 tmp]# hadoop fs -ls /user/root/wc-results.txt/
Found 3 items
-rw-r--r--  1 root supergroup          0 2014-10-27 06:55
/user/root/wc-results.txt/_SUCCESS
-rw-r--r--  1 root supergroup        25 2014-10-27 06:55
/user/root/wc-results.txt/part-00000
-rw-r--r--  1 root supergroup          8 2014-10-27 06:55
/user/root/wc-results.txt/part-00001
```

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

```
[root@cdh5-cm-vm01 tmp]# hadoop fs -cat /user/root/wc-results.txt/part-00000
(fish,5)
(dog,2)
(cat,2)

[root@cdh5-cm-vm01 tmp]# hadoop fs -cat /user/root/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!)**



App Name	Started	Completed	Duration	Spark User	Last Updated
<a href="#">Spark shell</a>	2014/10/27 06:56:21	2014/10/27 07:01:07	4.8 min	root	2014/10/27 07:01:08
<a href="#">Spark shell</a>	2014/10/27 06:42:56	2014/10/27 06:55:26	12 min	root	2014/10/27 06:55:27

**As the final section in the Spark lab, 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:**

```
[root@cdh5-cm-vm01 tmp]# pwd
/tmp
```

```
[root@cdh5-cm-vm01 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]
```

```
[root@cdh5-cm-vm01 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]

```
[root@cdh5-cm-vm01 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:**

```
[root@cdh5-cm-vm01 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
```

```
[root@cdh5-cm-vm01 tmp]# wc -l movies.dat
3883 movies.dat
```

```
[root@cdh5-cm-vm01 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
```

```
[root@cdh5-cm-vm01 tmp]# wc -l ratings.dat
1000209 ratings.dat
```

```
[root@cdh5-cm-vm01 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
```

```
[root@cdh5-cm-vm01 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:**

```
[root@cdh5-cm-vm01 tmp]# sudo -u spark hadoop fs -copyFromLocal movies.dat
/user/spark/
```

```
[root@cdh5-cm-vm01 tmp]# sudo -u spark hadoop fs -copyFromLocal users.dat /user/spark/
```

```
[root@cdh5-cm-vm01 tmp]# sudo -u spark hadoop fs -copyFromLocal ratings.dat /user/spark/
```

```
[root@cdh5-cm-vm01 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:

```
[root@cdh5-cm-vm01 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.
```

```

/ _/ _ _ _ _/ _/ _
\_ \_ \_ \_ \_ \_ \_ \_
/_ / _ . _ \_ , _/ / / _ \_ \_
/_ /

```

version 1.1.0

**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!**

**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:

```
>>> trRDD = moviesPartsRDD.map(lambda l: l[1])
>>> trRDD.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.registerAsTable("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")
```

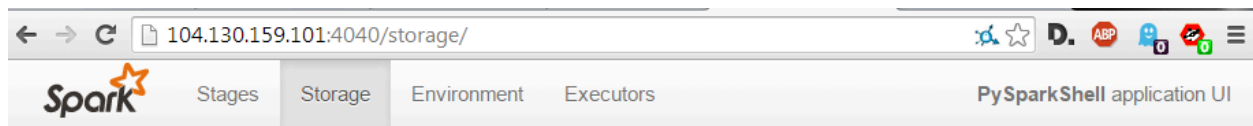
```
>>> joinRDD.count()
5L
```

```
>>> 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')]
```

Finally, 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:



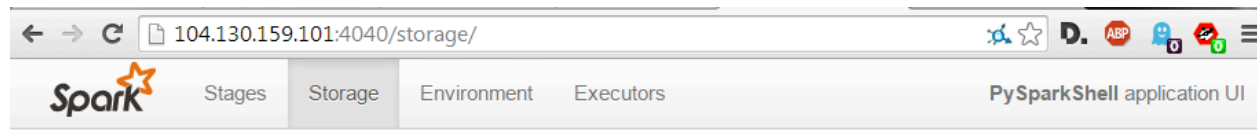
## Storage

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:



## Storage

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

This concludes the Spark lab. There is a lot more to Spark, we just brushed the surface!