Spark Lab using CDH 5.5 Quick Start

D. Howard

20 Feb 2016 Lab

Purpose

- Use CDH 5.5 QuickStart as our VM
  - CDH 5.5 Quick Start is jam packed with already installed s/w: CentOS 6.4, Java 1.7, Spark, MRV2, Eclipse , Hive and many more.
  - Less installation & configuration headaches ☺

- Run Python commands and Word Count.py using Spark's Python API.
  - Spark has APIs for: Scala, Java, Python, R

Install CDH 5.5 QuickStart Steps

- Your VMware workstation must already be installed on your Windows. Or VMFusion for Mac.

URL: http://www.cloudera.com/downloads/quickstart_vms/5-5.html

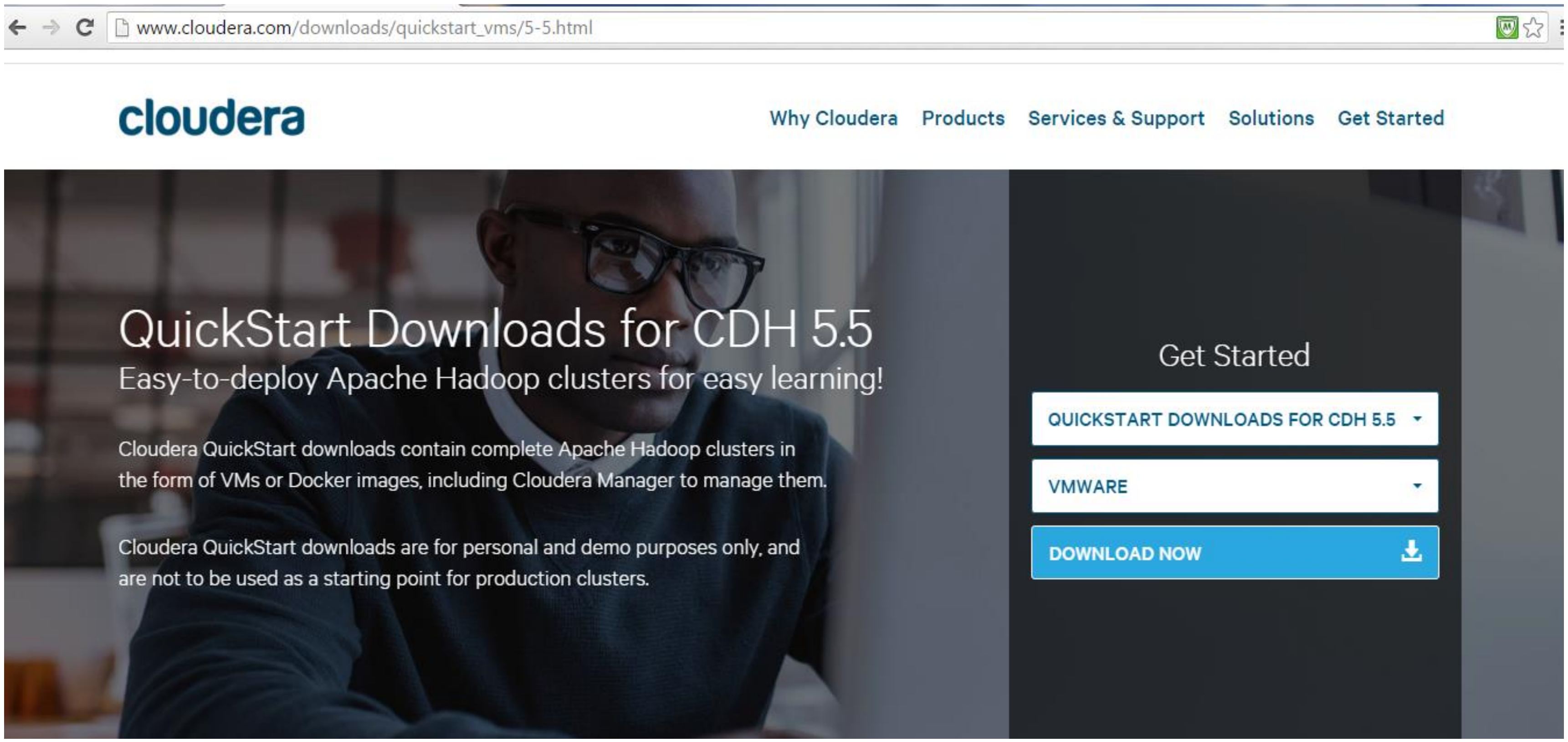
  - Select Vmware as your Platform
  - Select download Now (for CHD 5.5)

- Login as cloudera password: cloudera
- Test internet access. Open Firefox browser and go somewhere.
- Add in a Shared Folder VM as described in Professor's slide.
- MRV2, Java 1.7 and many other software packages are already installed!

Less Installation and configuration grief. Really! ☺

CDH QuickStart 5.5 Download Page

http://www.cloudera.com/downloads/quickstart_vms/5-5.html



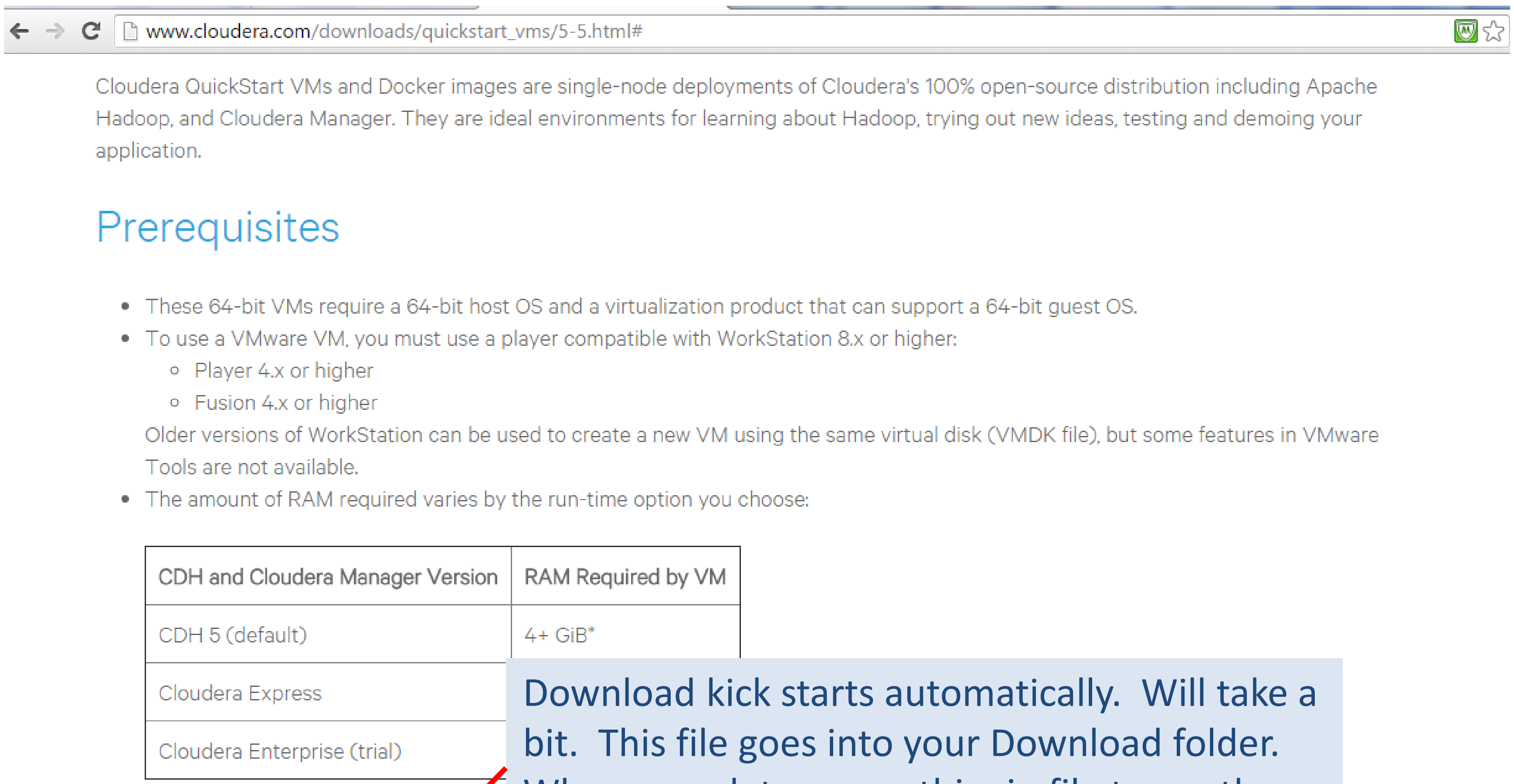
QuickStart Downloads for CDH 5.5

Easy-to-deploy Apache Hadoop clusters for easy learning!

Cloudera QuickStart downloads contain complete Apache Hadoop clusters in the form of VMs or Docker images, including Cloudera Manager to manage them.

Cloudera QuickStart downloads are for personal and demo purposes only, and are not to be used as a starting point for production clusters.

CDH QuickStart Download Page Prereqs



www.cloudera.com/downloads/quickstart_vms/5-5.html#

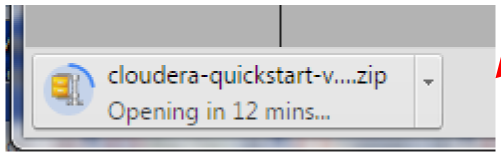
Cloudera QuickStart VMs and Docker images are single-node deployments of Cloudera's 100% open-source distribution including Apache Hadoop, and Cloudera Manager. They are ideal environments for learning about Hadoop, trying out new ideas, testing and demoing your application.

Prerequisites

- These 64-bit VMs require a 64-bit host OS and a virtualization product that can support a 64-bit guest OS.
- To use a VMware VM, you must use a player compatible with WorkStation 8.x or higher:
  - Player 4.x or higher
  - Fusion 4.x or higher

  Older versions of WorkStation can be used to create a new VM using the same virtual disk (VMDK file), but some features in VMware Tools are not available.
- The amount of RAM required varies by the run-time option you choose:

CDH and Cloudera Manager Version	RAM Required by VM
CDH 5 (default)	4+ GiB*
Cloudera Express	
Cloudera Enterprise (trial)	



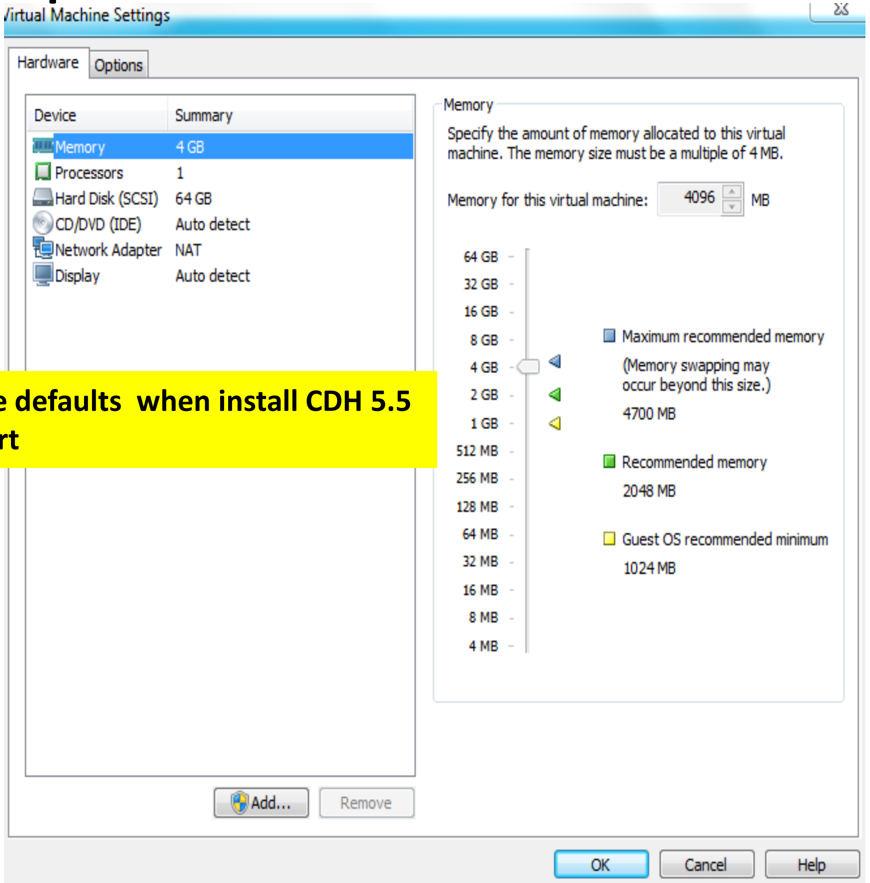
cloudera-quickstart-v....zip
Opening in 12 mins...

Download kick starts automatically. Will take a bit. This file goes into your Download folder. When complete, move this zip file to another folder e.g., VirtualMachinesforCloudera.

Cloudera QuickStart CDH 5.5

- QuickStart VM is a single-node "pseudo-distributed cluster
- Has many pre-installed packages we need (CentOS 6.4, Java 1.7, Maven, Eclipse IDE - Luna, Spark, …)

My set up:



Virtual Machine Settings

Hardware | Options

Device	Summary
Memory	4 GB
Processors	1
Hard Disk (SCSI)	64 GB
CD/DVD (IDE)	Auto detect
Network Adapter	NAT
Display	Auto detect

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

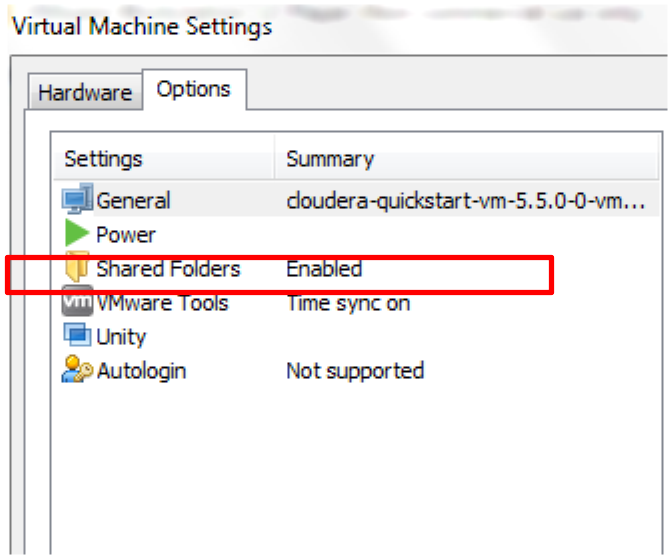
Memory for this virtual machine: 4096 MB

Maximum recommended memory (Memory swapping may occur beyond this size.) 4700 MB

Recommended memory 2048 MB

Guest OS recommended minimum 1024 MB

These are defaults when install CDH 5.5 QuickStart



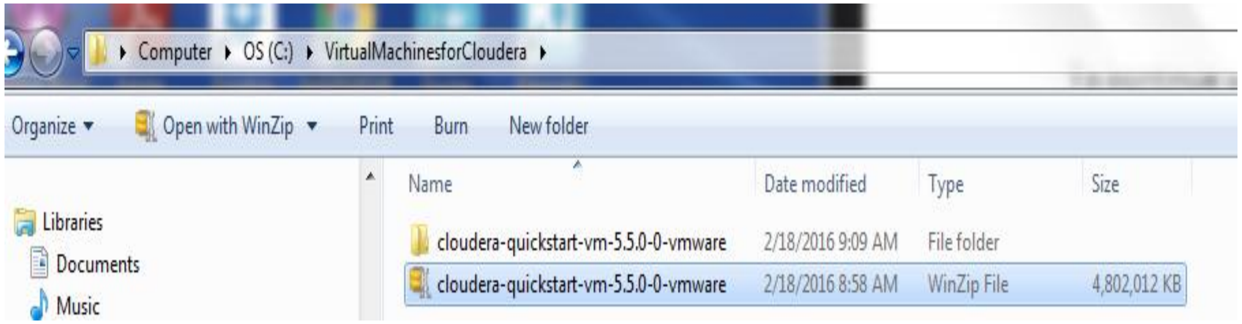
Virtual Machine Settings

Hardware | Options

Settings	Summary
General	cloudera-quickstart-vm-5.5.0-0-vm...
Power	
Shared Folders	Enabled
VMware Tools	Time sync on
Unity	
Autologin	Not supported

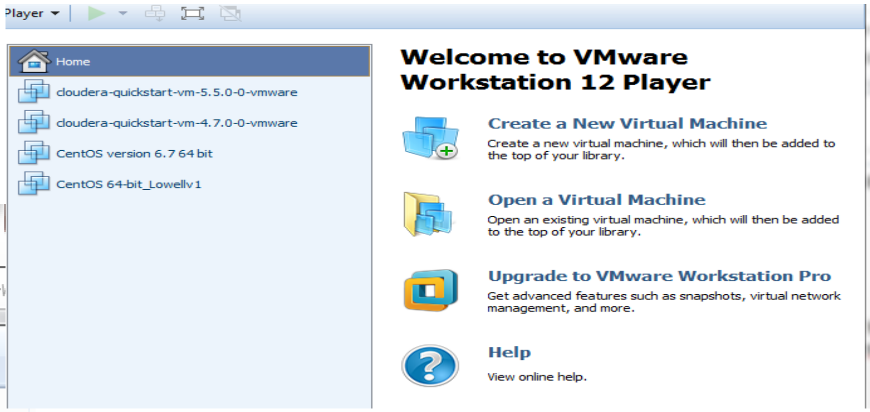
Run Cloudera QuickStart VM

- **Unzip folder:**



Name	Date modified	Type	Size
cloudera-quickstart-vm-5.5.0-0-vmware	2/18/2016 9:09 AM	File folder	
cloudera-quickstart-vm-5.5.0-0-vmware	2/18/2016 8:58 AM	WinZip File	4,802,012 KB

- **Open up your Vmware Workstation 12 Player or VMFusion. Select: Open a Virtual machine →go to your folder where QuickStart resides.**


Player ▾
Home
cloudera-quickstart-vm-5.5.0-0-vmware
cloudera-quickstart-vm-4.7.0-0-vmware
CentOS version 6.7 64 bit
CentOS 64-bit_Lowellv1

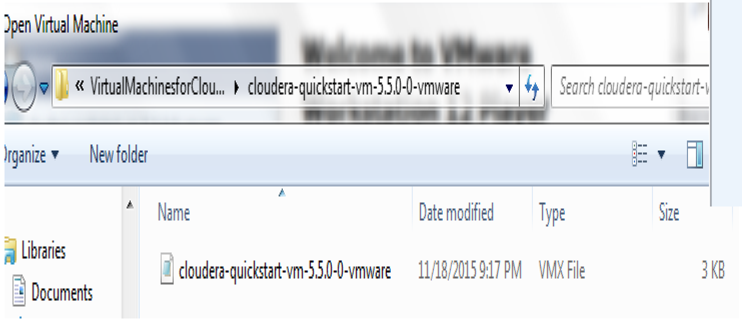
Welcome to VMware Workstation 12 Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation Pro
Get advanced features such as snapshots, virtual network management, and more.

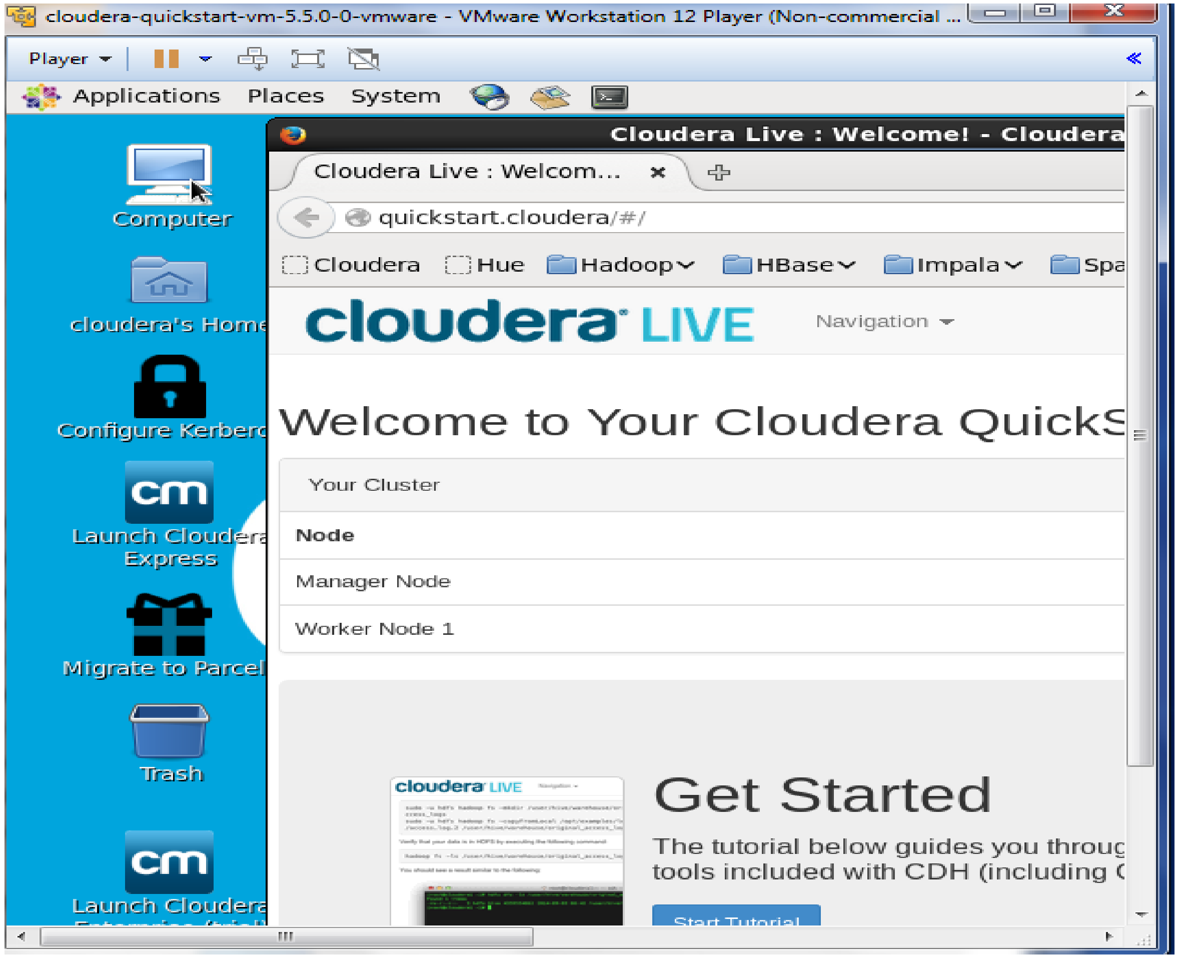
Help
View online help.


Open Virtual Machine
« VirtualMachinesforClou... ▸ cloudera-quickstart-vm-5.5.0-0-vmware
Search cloudera-quickstart-v
Organize ▾
New folder
Libraries
Documents

Name	Date modified	Type	Size
cloudera-quickstart-vm-5.5.0-0-vmware	11/18/2015 9:17 PM	VMX File	3 KB

- **Select Play virtual machine and your Cloudera QuickStart VM will boot up (Note your operating system is already loaded).**
- **Once up and running login as cloudera/cloudera.**
- **Open up Mozilla Firefox and go to a site to check Internet connectivity.**

Cloudera QuickStart Welcome Page



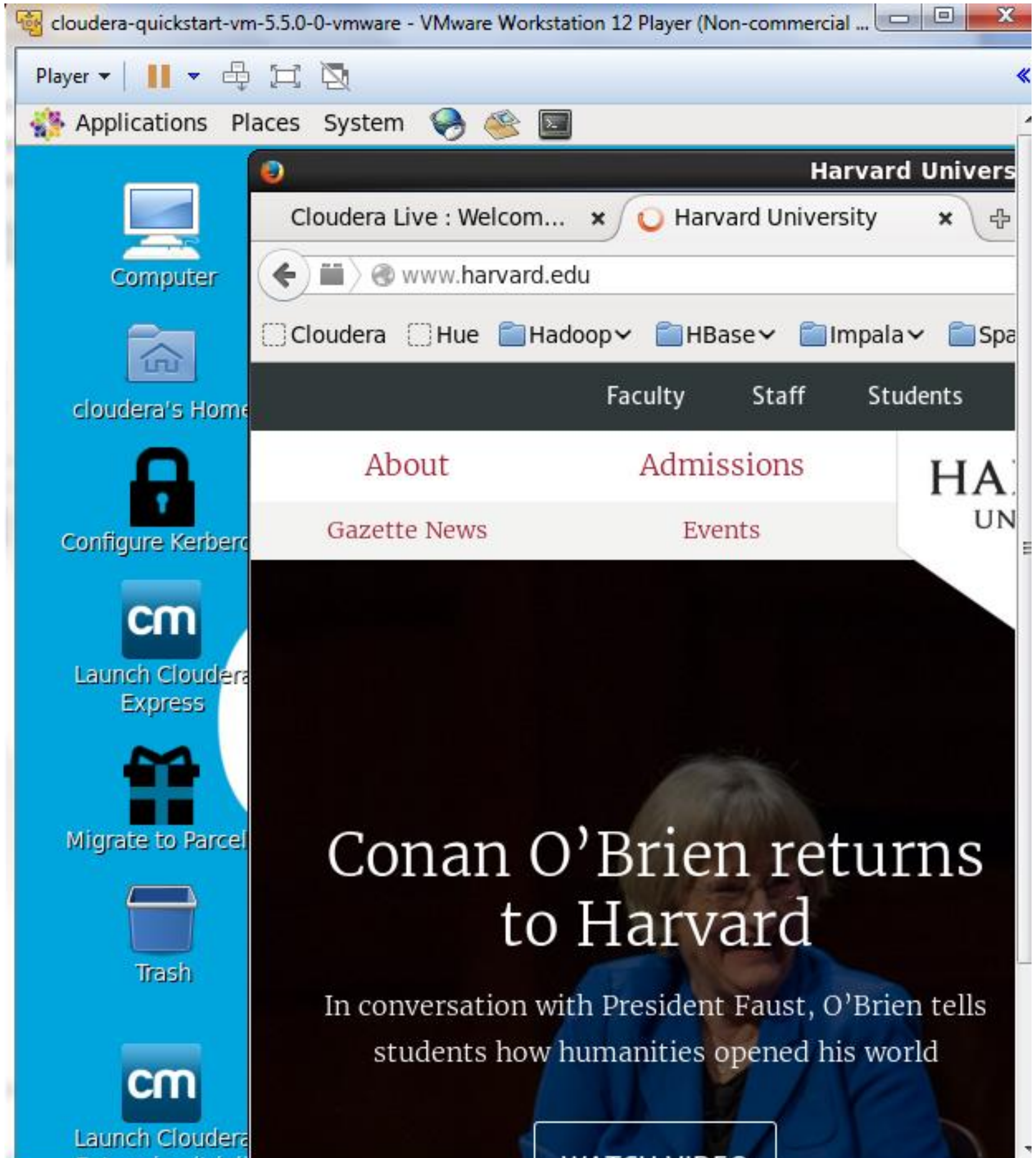
Welcome to Your Cloudera QuickS

Your Cluster
Node
Manager Node
Worker Node 1

Get Started

The tutorial below guides you throug
tools included with CDH (including

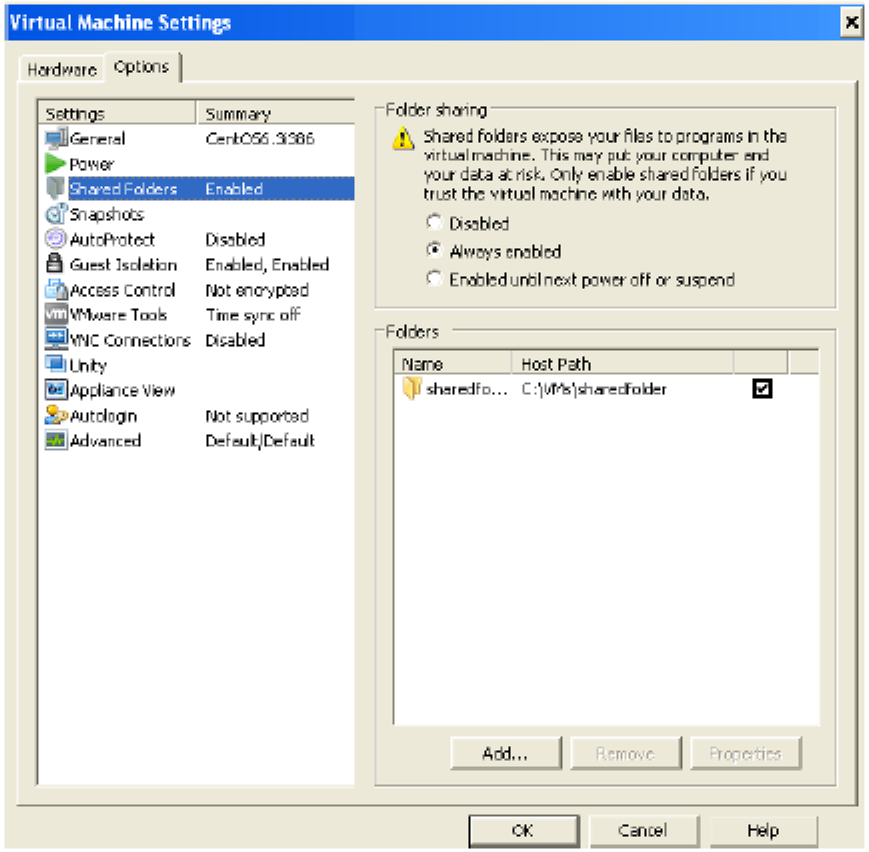
Check Internet Connectivity



Add a Shared Folder
Reference: Lab02_CreatingVM_CentOS.pdf, slide 21.

Enable Shared Folders

- Another way to share files with the host OS is to enable `Shared Folders`.
- Power down VM. Right click on the VM, select `Edit virtual machine settings > Options`
- Select `Shared Folders > Add`
- Add folder `c:\VMs\sharedfolder`
- Check `Always enable > Finish > OK`
- Power up VM
- Login as `cloudera`.
- Shared folder will shows as `/mnt/hgfs/sharefolder`


Virtual Machine Settings

Hardware | Options

Settings	Summary
General	CentOS6.3386
Power	
Shared Folders	Enabled
Snapshots	
AutoProtect	Disabled
Guest Isolation	Enabled, Enabled
Access Control	Not encrypted
VMware Tools	Time sync off
VNC Connections	Disabled
Unity	
Appliance View	
Autologin	Not supported
Advanced	Default/Default

Folder sharing

Shared folders expose your files to programs in the virtual machine. This may put your computer and your data at risk. Only enable shared folders if you trust the virtual machine with your data.

( ) Disabled
(●) Always enabled
( ) Enabled until next power off or suspend

Folders

Name	Host Path	
sharedfo...	C:\VMs\sharedfolder	✓

Spark™ *Lightning-fast cluster computing*

Packages you need to have for Spark (and JAVA API)

In terminal window in your **CDH QuickStart VM**

- **Java**

$ which java

```
[cloudera@quickstart SF]$ which java
/usr/java/jdk1.7.0_67-cloudera/bin/java
```

- **Spark**

[cloudera@quickstart ~]$ /usr/bin/pyspark

```
Python 2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-3)] on linux2
```

OR you can use…

[cloudera@quickstart ~]$ pyspark

```
[cloudera@quickstart SF]$ cd /
[cloudera@quickstart /]$ ./usr/bin/pyspark
Python 2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-3)] on linux2
```

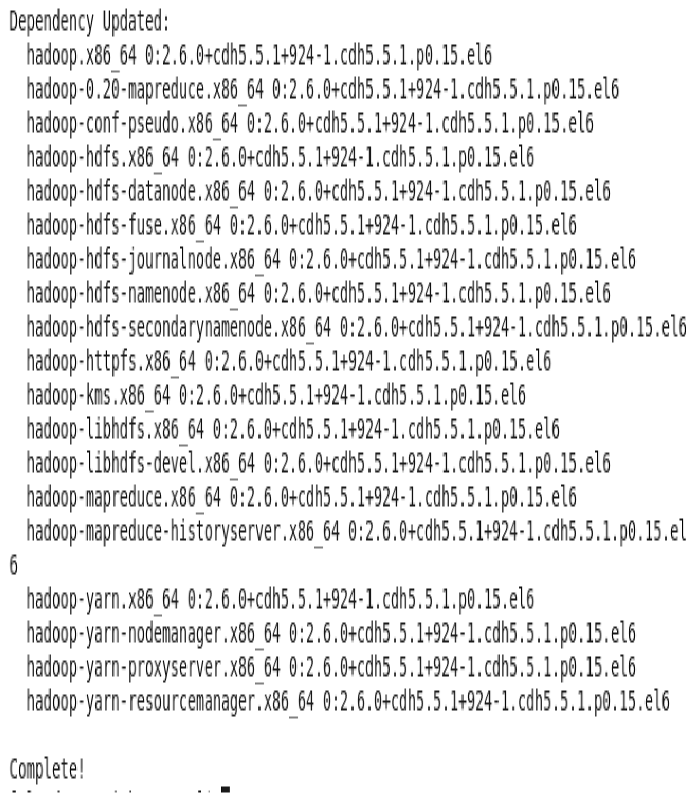
- **Maven**

[cloudera@quickstart ~]$ mvn --version

```
Apache Maven 3.0.4 (r1232337; 2012-01-17 00:44:56-0800)
Maven home: /usr/local/apache-maven/apache-maven-3.0.4
Java version: 1.7.0_67, vendor: Oracle Corporation
Java home: /usr/java/jdk1.7.0_67-cloudera/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "2.6.32-358.el6.x86_64", arch: "amd64",
family: "unix"
```

- **Hadoop-Client**

$*sudo yum install hadoop-client*



```
Dependency Updated:
  hadoop.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-0.20-mapreduce.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-conf-pseudo.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs-datanode.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs-fuse.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs-journalnode.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs-namenode.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-hdfs-secondarynamenode.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-httpfs.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-kms.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-libhdfs.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-libhdfs-devel.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-mapreduce.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-mapreduce-historyserver.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el
6
  hadoop-yarn.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-yarn-nodemanager.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-yarn-proxyserver.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6
  hadoop-yarn-resourcemanager.x86_64 0:2.6.0+cdh5.5.1+924-1.cdh5.5.1.p0.15.el6

Complete!
```

If you are not using CDH QuickStart…

In your CentOS 6.7 MRv2 VM:

Log onto your VM as user cloudera and issue the command:

$ sudo yum install hadoop-client

Download the tarball for ***apache-maven-3.3.1 or later*** and expand it using command:

$ tar -zxvf apache-maven-3.3.1-bin.tar.gz

Save this expanded folder to the '*/usr/lib/maven*' folder by issuing command:

$ sudo mv apache-maven-3.3.1 /usr/lib/maven

Install **spark-core, spark-master, spark-worker, spark-historyserver** and **spark-python** by issuing command:

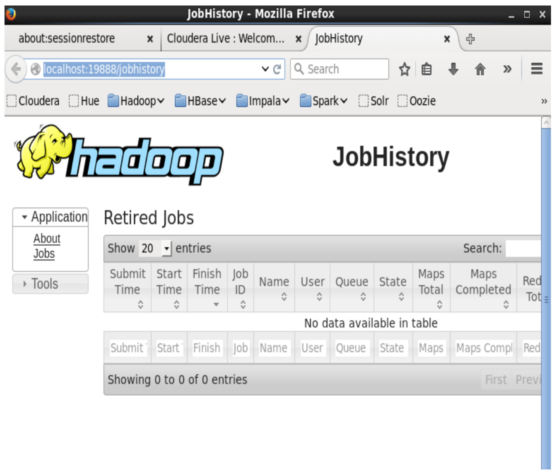
$ sudo yum install spark-core spark-master spark-worker spark-historyserver spark-python

Check Hadoop daemons, job status, health

[cloudera@quickstart ~]$ hdfs dfsadmin -report

```
Configured Capacity: 58665738240 (54.64 GB)
Present Capacity: 47990501376 (44.69 GB)
DFS Remaining: 47240380416 (44.00 GB)
DFS Used: 750120960 (715.37 MB)
DFS Used%: 1.56%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

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

Name: 127.0.0.1:50010 (quickstart.cloudera)
Hostname: quickstart.cloudera
Decommission Status : Normal
Configured Capacity: 58665738240 (54.64 GB)
DFS Used: 750120960 (715.37 MB)
Non DFS Used: 10675236864 (9.94 GB)
DFS Remaining: 47240380416 (44.00 GB)
DFS Used%: 1.28%
DFS Remaining%: 80.52%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 6
Last contact: Thu Feb 18 16:31:51 PST 2016
```

Check job status: http://localhost: 19888



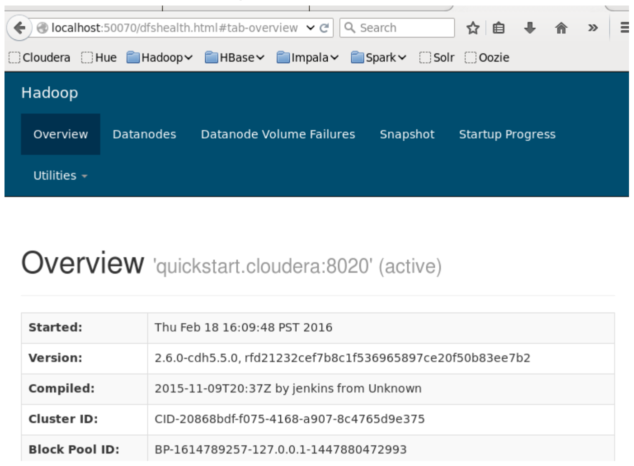
Retired Jobs

Show 20 entries Search:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Red Tot
No data available in table										
Submit	Start	Finish	Job	Name	User	Queue	State	Maps	Maps Compl	Red

Showing 0 to 0 of 0 entries First Previ

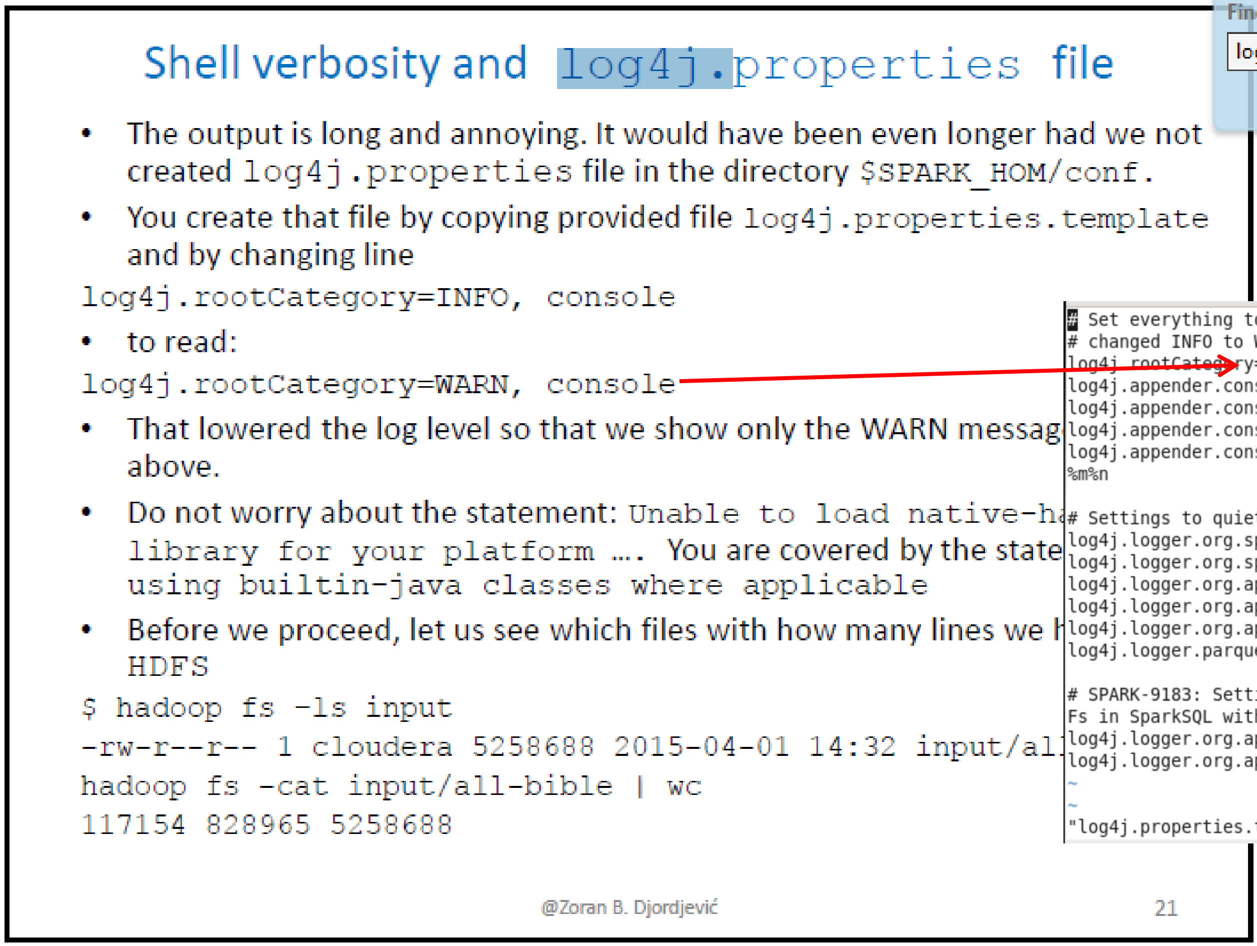
Check health: http://localhost:50070



Overview 'quickstart.cloudera:8020' (active)

Started:	Thu Feb 18 16:09:48 PST 2016
Version:	2.6.0-cdh5.5.0, rfd21232cef7b8c1f536965897ce20f50b83ee7b2
Compiled:	2015-11-09T20:37Z by jenkins from Unknown
Cluster ID:	CID-20868bdf-f075-4168-a907-8c4765d9e375
Block Pool ID:	BP-1614789257-127.0.0.1-1447880472993

Remove 3rd party verbose messages in Spark



Shell verbosity and `log4j.properties` file

- The output is long and annoying. It would have been even longer had we not created `log4j.properties` file in the directory `$SPARK_HOM/conf`.
- You create that file by copying provided file `log4j.properties.template` and by changing line

```
log4j.rootCategory=INFO, console
```

- to read:

```
log4j.rootCategory=WARN, console
```

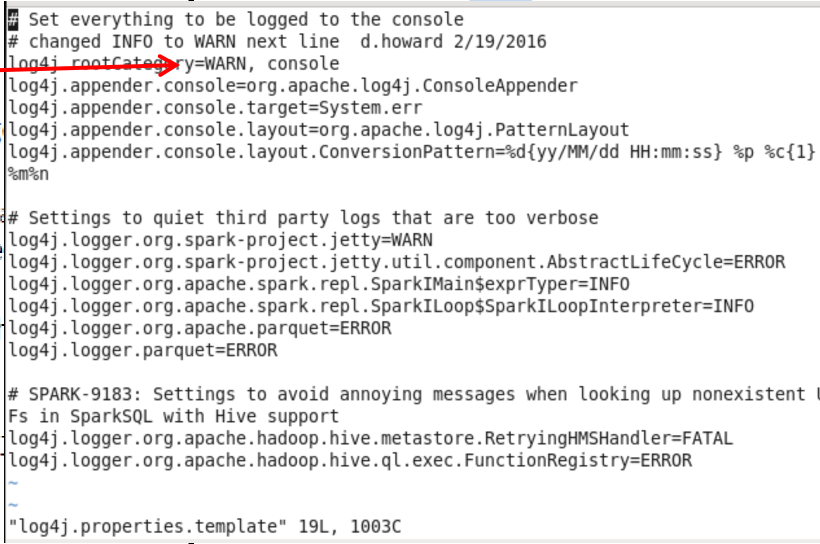
- That lowered the log level so that we show only the WARN messag above.
- Do not worry about the statement: `Unable to load native-ha library for your platform ….` You are covered by the state `using builtin-java classes where applicable`
- Before we proceed, let us see which files with how many lines we h HDFS

```
$ hadoop fs -ls input
-rw-r--r-- 1 cloudera 5258688 2015-04-01 14:32 input/al
hadoop fs -cat input/all-bible | wc
117154 828965 5258688
```

@Zoran B. Djordjević 21

Find
log4

I changed this default.
Note: I did this as user root.



```
# Set everything to be logged to the console
# changed INFO to WARN next line  d.howard 2/19/2016
log4j.rootCategory=WARN, 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.spark-project.jetty=WARN
log4j.logger.org.spark-project.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
log4j.logger.org.apache.parquet=ERROR
log4j.logger.parquet=ERROR

# SPARK-9183: Settings to avoid annoying messages when looking up nonexistent U
Fs in SparkSQL with Hive support
log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR
~
~
"log4j.properties.template" 19L, 1003C
```

How to start Spark

```
[cloudera@quickstart ~]$ pyspark
```

You will see many messages:

```
16/02/17 20:02:26 INFO spark.SparkEnv: Registering OutputCommitCoordinator
16/02/17 20:02:27 INFO server.Server: jetty-8.y.z-SNAPSHOT
16/02/17 20:02:27 INFO server.AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040
16/02/17 20:02:27 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.
16/02/17 20:02:27 INFO ui.SparkUI: Started SparkUI at http://192.168.133.131:4040
16/02/17 20:02:28 WARN metrics.MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.
16/02/17 20:02:28 INFO executor.Executor: Starting executor ID driver on host localhost
16/02/17 20:02:28 INFO util.Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 54280.
16/02/17 20:02:28 INFO netty.NettyBlockTransferService: Server created on 54280
16/02/17 20:02:28 INFO storage.BlockManagerMaster: Trying to register BlockManager
16/02/17 20:02:28 INFO storage.BlockManagerMasterEndpoint: Registering block manager localhost:54280 with 534.5 MB RAM, BlockManagerId(driver,
localhost, 54280)
16/02/17 20:02:28 INFO storage.BlockManagerMaster: Registered BlockManager
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.5.0-cdh5.5.0
      /_/

Using Python version 2.6.6 (r266:84292, Feb 22 2013 00:00:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Enter Python commands here

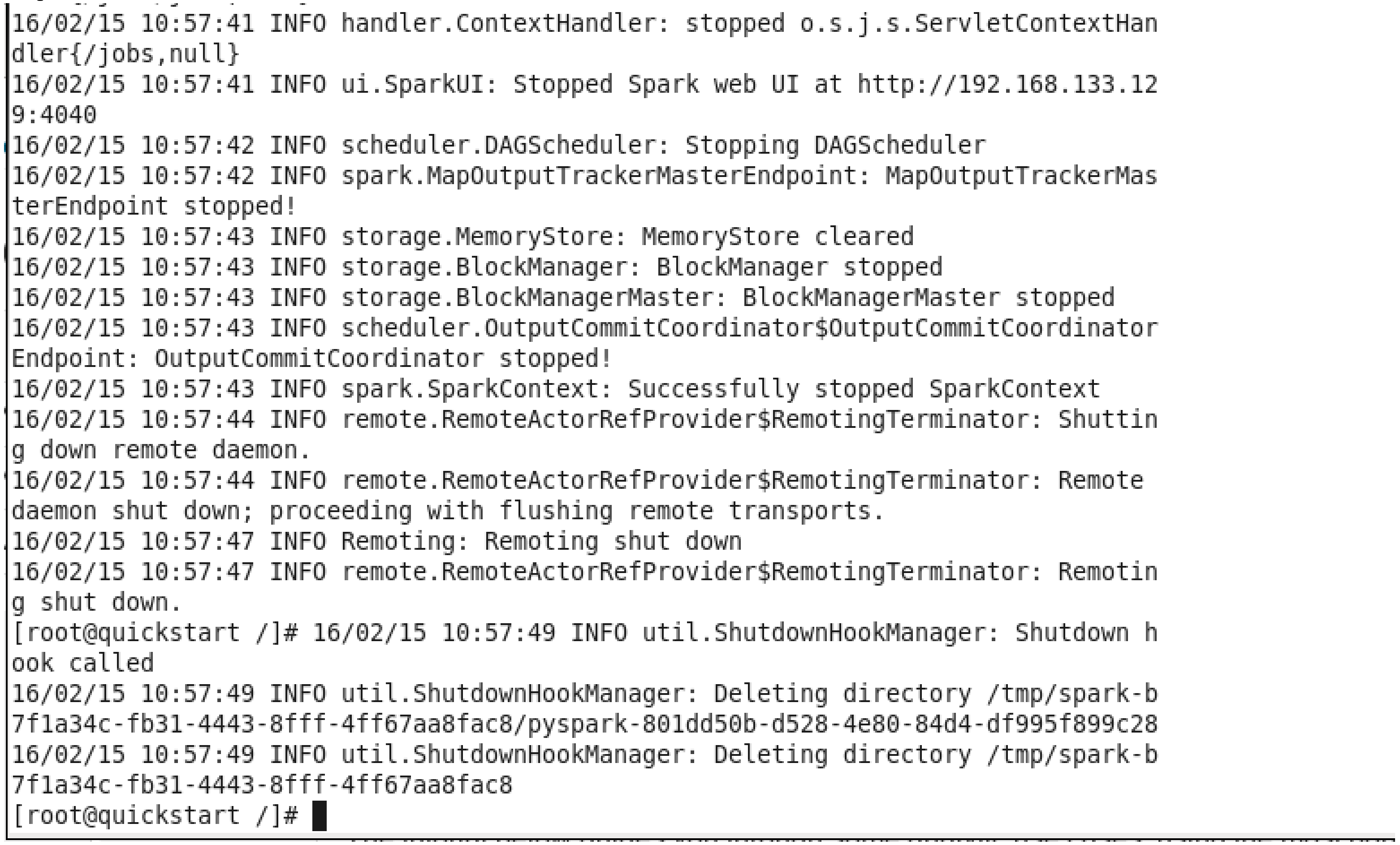
SparkContext & HiveContext
is pre-loaded so you don't need:

```
>>>from pyspark import SparkConf, SparkContext
>>>conf = SparkConf().setMaster("local").setAppName("YourApp")
>>>sc = SparkContext(conf = conf)
```

(unless you get message SparkContext is shutdown).

How to exit SPARK

- Enter: CTRL + D
- Output from shutdown:



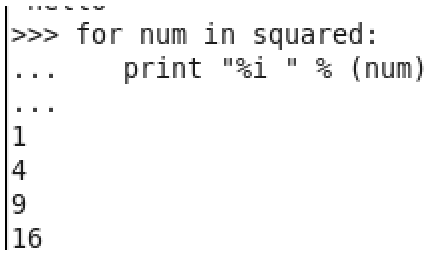
```
16/02/15 10:57:41 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/jobs,null}
16/02/15 10:57:41 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.133.12
9:4040
16/02/15 10:57:42 INFO scheduler.DAGScheduler: Stopping DAGScheduler
16/02/15 10:57:42 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMas
terEndpoint stopped!
16/02/15 10:57:43 INFO storage.MemoryStore: MemoryStore cleared
16/02/15 10:57:43 INFO storage.BlockManager: BlockManager stopped
16/02/15 10:57:43 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
16/02/15 10:57:43 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinator
Endpoint: OutputCommitCoordinator stopped!
16/02/15 10:57:43 INFO spark.SparkContext: Successfully stopped SparkContext
16/02/15 10:57:44 INFO remote.RemoteActorRefProvider$RemotingTerminator: Shuttin
g down remote daemon.
16/02/15 10:57:44 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remote
daemon shut down; proceeding with flushing remote transports.
16/02/15 10:57:47 INFO Remoting: Remoting shut down
16/02/15 10:57:47 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remotin
g shut down.
[root@quickstart /]# 16/02/15 10:57:49 INFO util.ShutdownHookManager: Shutdown h
ook called
16/02/15 10:57:49 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-b
7f1a34c-fb31-4443-8fff-4ff67aa8fac8/pyspark-801dd50b-d528-4e80-84d4-df995f899c28
16/02/15 10:57:49 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-b
7f1a34c-fb31-4443-8fff-4ff67aa8fac8
[root@quickstart /]# █
```

Examples to try in Spark

- ## Python example:

Calculates SQUARES:

```
>>>nums = sc.parallelize([1, 2, 3, 4])
>>>squared = nums.map(lambda x: x * x).collect()  (note: returns output – Running task)
>>>for num in squared:
... (add 3 spaces)  print "%i " % (num)
... (press return)
---------------------
1
4
9
16
```

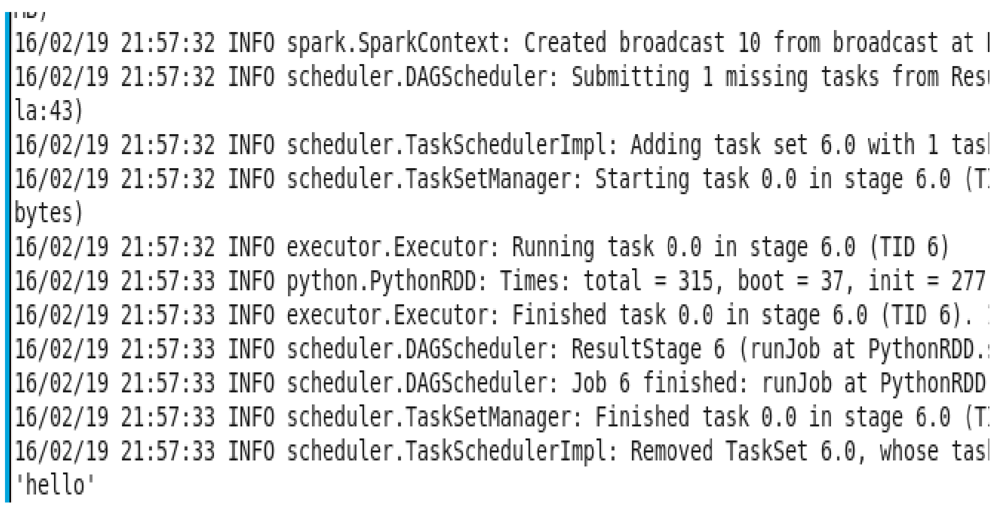


```
>>> for num in squared:
...     print "%i " % (num)
...
1
4
9
16
```

- ## PYSPARK example:

Splits lines into words:

```
lines = sc.parallelize(["hello world", "hi"])
words = lines.flatMap(lambda line: line.split(" "))
words.first()
-------------------------------
'hello'
```



```
16/02/19 21:57:32 INFO spark.SparkContext: Created broadcast 10 from broadcast at 
16/02/19 21:57:32 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Res
la:43)
16/02/19 21:57:32 INFO scheduler.TaskSchedulerImpl: Adding task set 6.0 with 1 tas
16/02/19 21:57:32 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 6.0 (T
bytes)
16/02/19 21:57:32 INFO executor.Executor: Running task 0.0 in stage 6.0 (TID 6)
16/02/19 21:57:33 INFO python.PythonRDD: Times: total = 315, boot = 37, init = 277
16/02/19 21:57:33 INFO executor.Executor: Finished task 0.0 in stage 6.0 (TID 6).
16/02/19 21:57:33 INFO scheduler.DAGScheduler: ResultStage 6 (runJob at PythonRDD.
16/02/19 21:57:33 INFO scheduler.DAGScheduler: Job 6 finished: runJob at PythonRDD
16/02/19 21:57:33 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 6.0 (T
16/02/19 21:57:33 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 6.0, whose tas
'hello'
```

Adding all-bible.txt to your CDH VM Linux file system

Note: You must have your shared folder set up. Otherwise, next steps won't work.

1. On your laptop put the all-bible.txt in your shared folder directory.

2. On your VM go to: /mnt/hgfs/SF

```
[cloudera@quickstart ~]$ cd /mnt/hgfs
[cloudera@quickstart hgfs]$ ls
SF    (this is my shared folder directory called "SF")
[cloudera@quickstart hgfs]$ cd SF
[cloudera@quickstart SF]$ ls -l
total 28543
-rwxrwxrwx 1 root root  5258688 Feb 15 08:49 all-bible.txt
```

Move all-bible.txt from Linux File System to HDFS

- Create an input HDFS directory called ‘***input***’

```
[cloudera@quickstart SF]$ hadoop fs -mkdir input
[cloudera@quickstart SF]$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - cloudera cloudera          0 2016-02-15 21:54 input
```

- Copy the all-bible.txt file from the shared folder directory to the HDFS

$\text{[cloudera@quickstart SF]\$ hadoop fs -put }\underbrace{\text{/mnt/hgfs/SF/}}_{\textit{my shared folder path}}\text{all-bible.txt input}$

- Check the hdfs for directory input

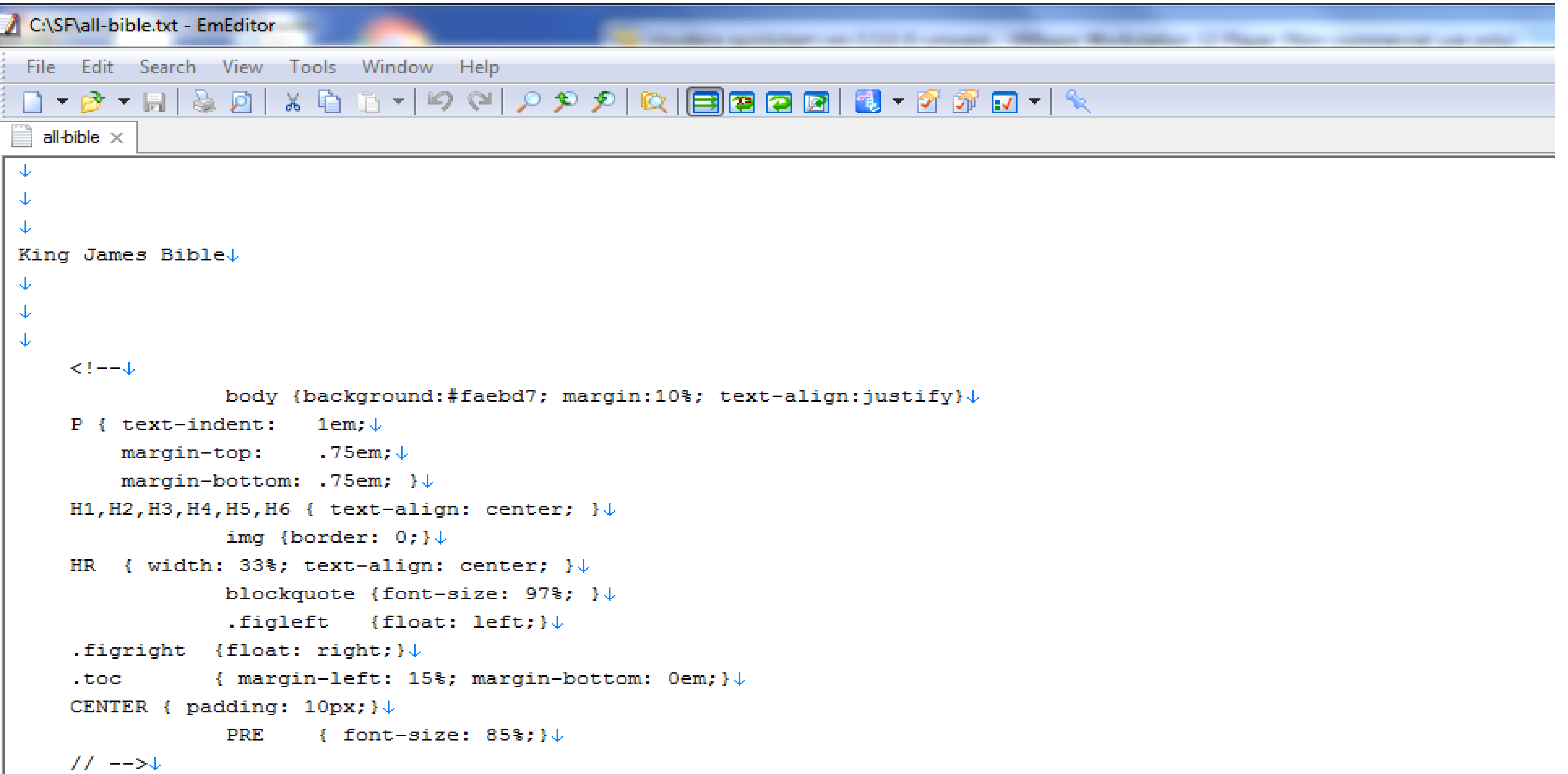
```
[cloudera@quickstart SF]$ ls
Found 1 items
-rw-r--r--   1 cloudera cloudera    5258688 2016-02-15 21:56 input/all-bible.txt
```

- Remove directory in hdfs

```
[cloudera@quickstart ~]$ hadoop fs -rm -r output
```

A Peek at all-bible.txt

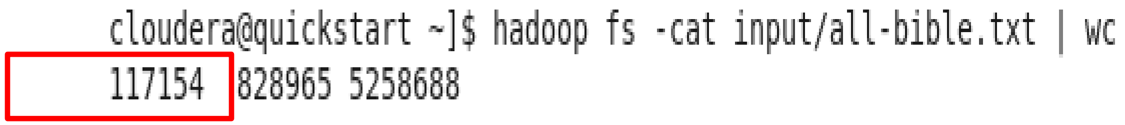
- I used EmEditor – great editor for very large text files.


C:\SF\all-bible.txt - EmEditor
File Edit Search View Tools Window Help
all-bible

```
↓
↓
↓
King James Bible↓
↓
↓
↓
    <!--↓
                body {background:#faebd7; margin:10%; text-align:justify}↓
    P { text-indent:   1em;↓
        margin-top:    .75em;↓
        margin-bottom: .75em; }↓
    H1,H2,H3,H4,H5,H6 { text-align: center; }↓
                img {border: 0;}↓
    HR  { width: 33%; text-align: center; }↓
                blockquote {font-size: 97%; }↓
                .figleft   {float: left;}↓
    .figright  {float: right;}↓
    .toc       { margin-left: 15%; margin-bottom: 0em;}↓
    CENTER { padding: 10px;}↓
                PRE     { font-size: 85%;}↓
    // -->↓
```

Count # words in all-bible.txt

1. **hadoop fs -cat input/all-bible.txt | wc**



```
cloudera@quickstart ~]$ hadoop fs -cat input/all-bible.txt | wc
 117154 828965 5258688
```

2. **From Spark using Python:**

cloudera@quickstart ~]$ pyspark

```
Python 2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

>>RDDvar = sc.textFile("input/all-bible.txt") → ***Loads data into RDD variable called RDDvar***

```
6/02/16 18:18:25 INFO storage.MemoryStore: ensureFreeSpace(187856) called with curMem=262459, maxMem=560497950
16/02/16 18:18:25 INFO storage.MemoryStore: Block broadcast_3 stored as values in memory (estimated size 183.5 KB, free 534.1 MB)
```

>>RDDvar.count() → ***count the # of words***

```
16/02/16 18:18:43 INFO python.PythonRDD: Times: total = 2276, boot = -47307, init = 47350, finish = 2233
16/02/16 18:18:43 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 2127 bytes result sent to driver
16/02/16 18:18:43 INFO scheduler.DAGScheduler: ResultStage 1 (count at <stdin>:1) finished in 2.355 s
16/02/16 18:18:43 INFO scheduler.DAGScheduler: Job 1 finished: count at <stdin>:1, took 2.398193 s
16/02/16 18:18:43 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 2351 ms on localhost (1/1)
16/02/16 18:18:43 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
117154
```

Lambda function to filter words

```
>>> RDDvar = sc.textFile("input/all-bible.txt")
```
(note: we don't have to rerun this line since we already stored our data in the RDD)

```
>>>Jacob_filter = RDDvar.filter(lambda line: "Jacob" in line)
>>> Jacob_filter.count()
16/02/16 18:58:29 INFO spark.SparkContext: Starting job: count at <stdin>:1
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Got job 6 (count at <stdin>:1) with 1 output partitions
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Final stage: ResultStage 6(count at <stdin>:1)
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Parents of final stage: List()
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Missing parents: List()
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Submitting ResultStage 6 (PythonRDD[8] at count at <stdin>:1), which has no missing parents
16/02/16 18:58:29 INFO storage.MemoryStore: ensureFreeSpace(6208) called with curMem=169254, maxMem=560497950
16/02/16 18:58:29 INFO storage.MemoryStore: Block broadcast_7 stored as values in memory (estimated size 6.1 KB, free 534.4 MB)
16/02/16 18:58:29 INFO storage.MemoryStore: ensureFreeSpace(3757) called with curMem=175462, maxMem=560497950
16/02/16 18:58:29 INFO storage.MemoryStore: Block broadcast_7_piece0 stored as bytes in memory (estimated size 3.7 KB, free 534.4 MB)
16/02/16 18:58:29 INFO storage.BlockManagerInfo: Added broadcast_7_piece0 in memory on localhost:35685 (size: 3.7 KB, free: 534.5 MB)
16/02/16 18:58:29 INFO spark.SparkContext: Created broadcast 7 from broadcast at DAGScheduler.scala:861
16/02/16 18:58:29 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 6 (PythonRDD[8] at count at <stdin>:1)
16/02/16 18:58:29 INFO scheduler.TaskSchedulerImpl: Adding task set 6.0 with 1 tasks
16/02/16 18:58:29 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 6.0 (TID 6, localhost, partition 0,ANY, 2174 bytes)
16/02/16 18:58:29 INFO executor.Executor: Running task 0.0 in stage 6.0 (TID 6)
16/02/16 18:58:29 INFO rdd.HadoopRDD: Input split: hdfs://quickstart.cloudera:8020/user/cloudera/input/all-bible.txt:0+5258688
16/02/16 18:58:31 INFO python.PythonRDD: Times: total = 1870, boot = 8, init = 52, finish = 1810
16/02/16 18:58:31 INFO executor.Executor: Finished task 0.0 in stage 6.0 (TID 6). 2125 bytes result sent to driver
16/02/16 18:58:31 INFO scheduler.DAGScheduler: ResultStage 6 (count at <stdin>:1) finished in 2.002 s
16/02/16 18:58:31 INFO scheduler.DAGScheduler: Job 6 finished: count at <stdin>:1, took 2.061797 s
16/02/16 18:58:31 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 6.0 (TID 6) in 2001 ms on localhost (1/1)
16/02/16 18:58:31 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 6.0, whose tasks have all completed, from pool
368
```

→ 368 is the # of times Jacob is mentioned in all-bible.txt

Python Word Count for Spark

```
>>>from pyspark import SparkConf, SparkContext
>>>conf = SparkConf().setMaster("local").setAppName("YourApp")
>>>sc = SparkContext(conf = conf)
>>> RDDvar = sc.textFile("input/all-bible.txt")
16/02/17 20:18:11 INFO storage.MemoryStore: ensureFreeSpace(124088) called with curMem=0, maxMem=560497950
16/02/17 20:18:11 INFO storage.MemoryStore: Block broadcast_0 stored as values in memory (estimated size 121.2 KB, free 534.4 MB)
…
>>>words = RDDvar.flatMap(lambda x: x.split(" "))
>>>result = words.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)
16/02/17 20:19:46 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
16/02/17 20:19:47 INFO mapred.FileInputFormat: Total input paths to process : 1

>>>result.saveAsTextFile("hdfs://localhost:8020/user/cloudera/output2")
…
16/02/17 13:17:13 INFO scheduler.DAGScheduler: ResultStage 4 (saveAsTextFile at NativeMethodAccessorImpl.java:-2) finished in 5.953 s
16/02/17 13:17:13 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 4.0 (TID 4) in 5953 ms on localhost (1/1)
16/02/17 13:17:13 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
16/02/17 13:17:13 INFO scheduler.DAGScheduler: Job 3 finished: saveAsTextFile at NativeMethodAccessorImpl.java:-2, took 26.181604 s
```

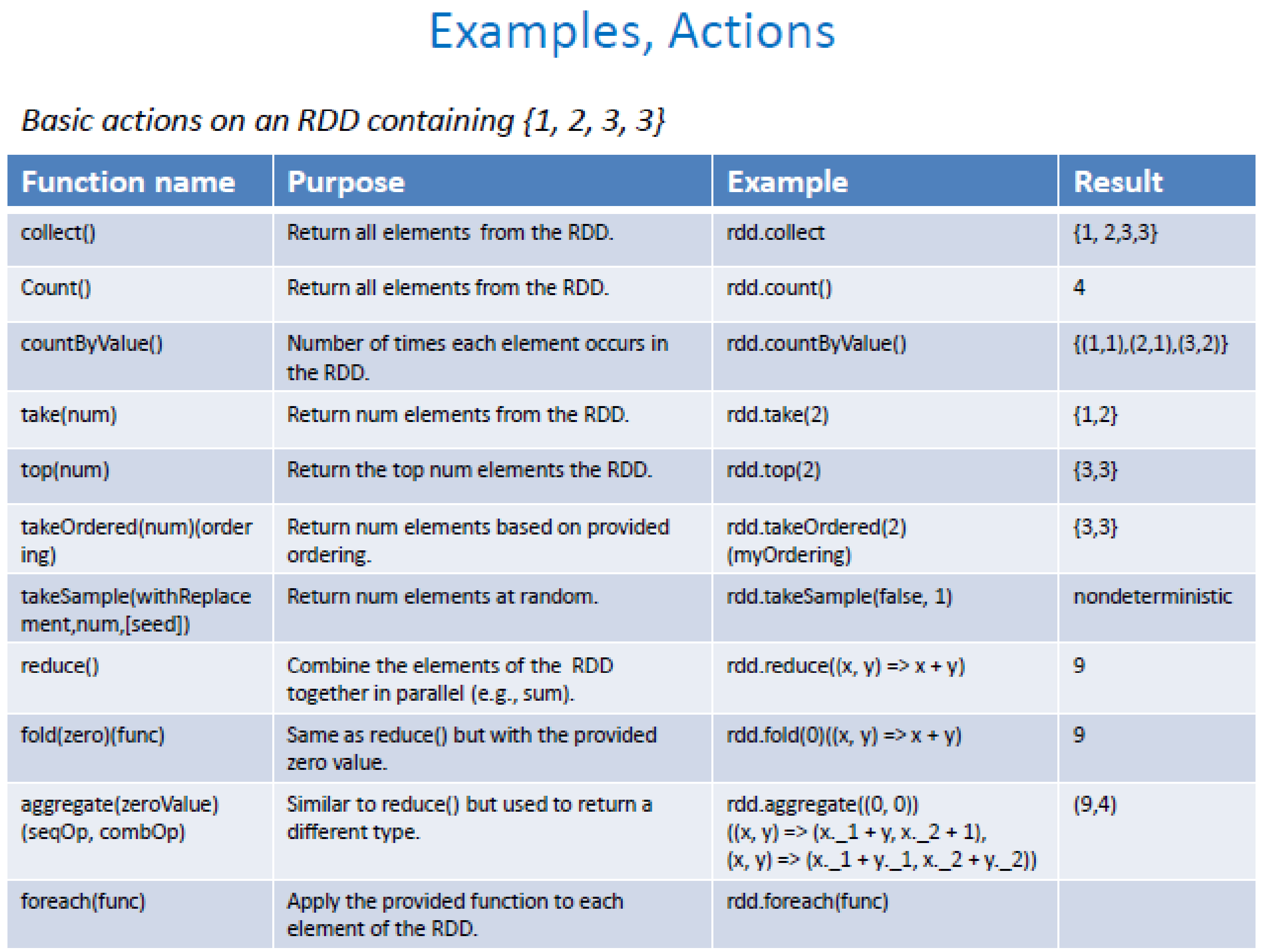
You don’t need these lines if SparkContext is running. See slide 16 but you do need them in scripts.

Check for directory output2 in HDFS:

```
[cloudera@quickstart ~]$ hadoop fs -ls output2
Found 2 items
-rw-r--r--   1 cloudera cloudera        0 2016-02-17 13:17 output2/_SUCCESS
-rw-r--r--   1 cloudera cloudera  1081681 2016-02-17 13:17 output2/part-00000
```

RDD Methods (actions supported by Spark)

http://spark.apache.org/docs/latest/programming-guide.html



Examples, Actions

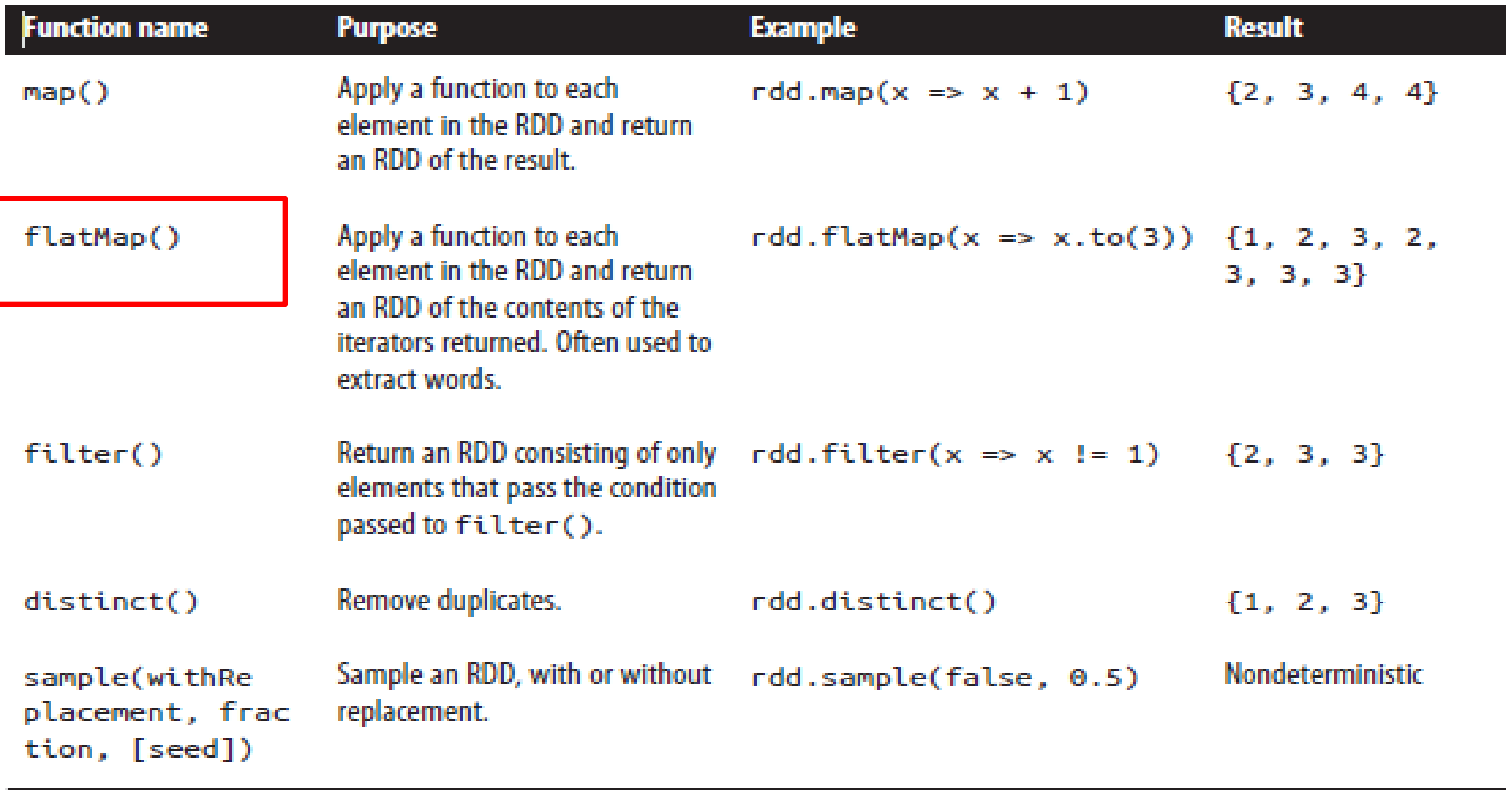
Basic actions on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
collect()	Return all elements from the RDD.	rdd.collect	{1, 2,3,3}
Count()	Return all elements from the RDD.	rdd.count()	4
countByValue()	Number of times each element occurs in the RDD.	rdd.countByValue()	{(1,1),(2,1),(3,2)}
take(num)	Return num elements from the RDD.	rdd.take(2)	{1,2}
top(num)	Return the top num elements the RDD.	rdd.top(2)	{3,3}
takeOrdered(num)(ordering)	Return num elements based on provided ordering.	rdd.takeOrdered(2)(myOrdering)	{3,3}
takeSample(withReplacement,num,[seed])	Return num elements at random.	rdd.takeSample(false, 1)	nondeterministic
reduce()	Combine the elements of the RDD together in parallel (e.g., sum).	rdd.reduce((x, y) => x + y)	9
fold(zero)(func)	Same as reduce() but with the provided zero value.	rdd.fold(0)((x, y) => x + y)	9
aggregate(zeroValue)(seqOp, combOp)	Similar to reduce() but used to return a different type.	rdd.aggregate((0, 0))((x, y) => (x._1 + y, x._2 + 1), (x, y) => (x._1 + y._1, x._2 + y._2))	(9,4)
foreach(func)	Apply the provided function to each element of the RDD.	rdd.foreach(func)	

Common Transformations on RDDs supported by Spark

http://spark.apache.org/docs/latest/programming-guide.html

Table 3-2. Basic RDD transformations on an RDD containing {1, 2, 3, 3}



Function name	Purpose	Example	Result
`map()`	Apply a function to each element in the RDD and return an RDD of the result.	`rdd.map(x => x + 1)`	`{2, 3, 4, 4}`
`flatMap()`	Apply a function to each element in the RDD and return an RDD of the contents of the iterators returned. Often used to extract words.	`rdd.flatMap(x => x.to(3))`	`{1, 2, 3, 2, 3, 3, 3}`
`filter()`	Return an RDD consisting of only elements that pass the condition passed to `filter()`.	`rdd.filter(x => x != 1)`	`{2, 3, 3}`
`distinct()`	Remove duplicates.	`rdd.distinct()`	`{1, 2, 3}`
`sample(withReplacement, fraction, [seed])`	Sample an RDD, with or without replacement.	`rdd.sample(false, 0.5)`	Nondeterministic

Breaking down Word Count example

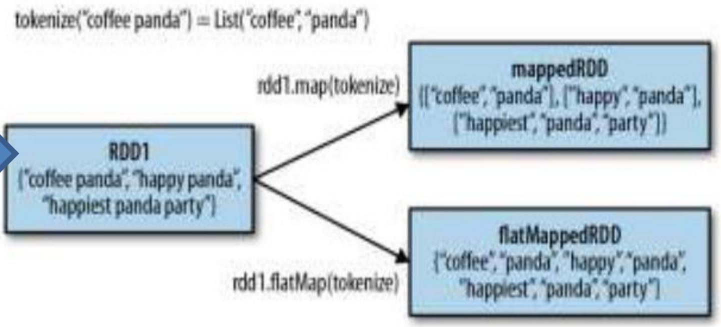
Read input

```
Myvar = sc.textFile("input/all-bible.txt")
```

Load file all-bible.txt from hdfs direcotry
Input into shell variable Myvar
(this is your RDD).

Mapper

```
words = Myvar.flatMap(lambda x: x.split(" "))
```

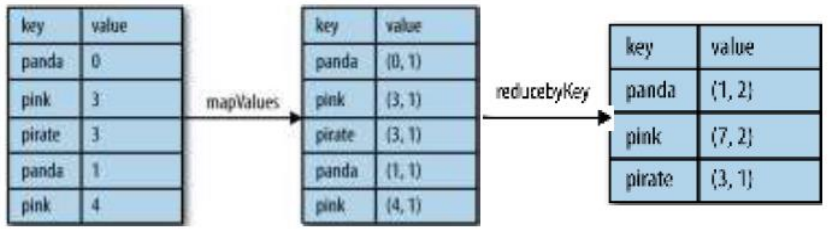

tokenize("coffee panda") = List("coffee", "panda")
RDD1
("coffee panda", "happy panda", "happiest panda party")
rdd1.map(tokenize)
mappedRDD
{["coffee", "panda"], ["happy", "panda"], ["happiest", "panda", "party"]}
rdd1.flatMap(tokenize)
flatMappedRDD
{"coffee", "panda", "happy", "panda", "happiest", "panda", "party"}

Reducer

create pairs of words with 1

```
result = words.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)
```

sums all words together



key	value
panda	0
pink	3
pirate	3
panda	1
pink	4

key	value
panda	(0, 1)
pink	(3, 1)
pirate	(3, 1)
panda	(1, 1)
pink	(4, 1)

key	value
panda	(1, 2)
pink	(7, 2)
pirate	(3, 1)

Save output

```
result.saveAsTextFile("hdfs://localhost:8020/user/cloudera/output2")
```

Save contents of RDD in output directory called output2 on HDFS located in /user/cloudera.

Check output from your job various ways

[cloudera@quickstart ~]$ hadoop fs -cat output2/part-00000 | head -20

```
(u'', 605968)
(u'sending;', 1)
(u'16:010:019', 1)
(u'16:010:018', 1)
(u'16:010:017', 1)
(u'16:010:016', 1)
(u'16:010:015', 1)
(u'16:010:014', 1)
(u'16:010:013', 1)
(u'16:010:012', 1)
(u'16:010:011', 1)
(u'16:010:010', 1)
(u'42:017:011', 1)
(u'hanging', 17)
(u'30:009:013', 1)
(u'whither,', 1)
(u'whither.', 2)
(u'11:013:016', 1)
(u'Tappuah,', 3)
(u'40:022:007', 1)
cat: Unable to write to output stream.
```

[cloudera@quickstart ~]$ hadoop fs -cat output2/part-00000 | tail -10

```
(u'circumcision,', 8)
(u'kingdom:', 6)
(u'roar,', 6)
(u'45:015:007', 1)
(u'Lust', 1)
(u'Caphtorims,', 1)
(u'helmets,', 1)
(u'roar;', 2)
(u'Zohar,', 3)
(u'jawbone', 3)
```

Creating Python Script

SCRIPT example (findJacob.py)

$spark-submit findJacob.py

from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("MyApp")

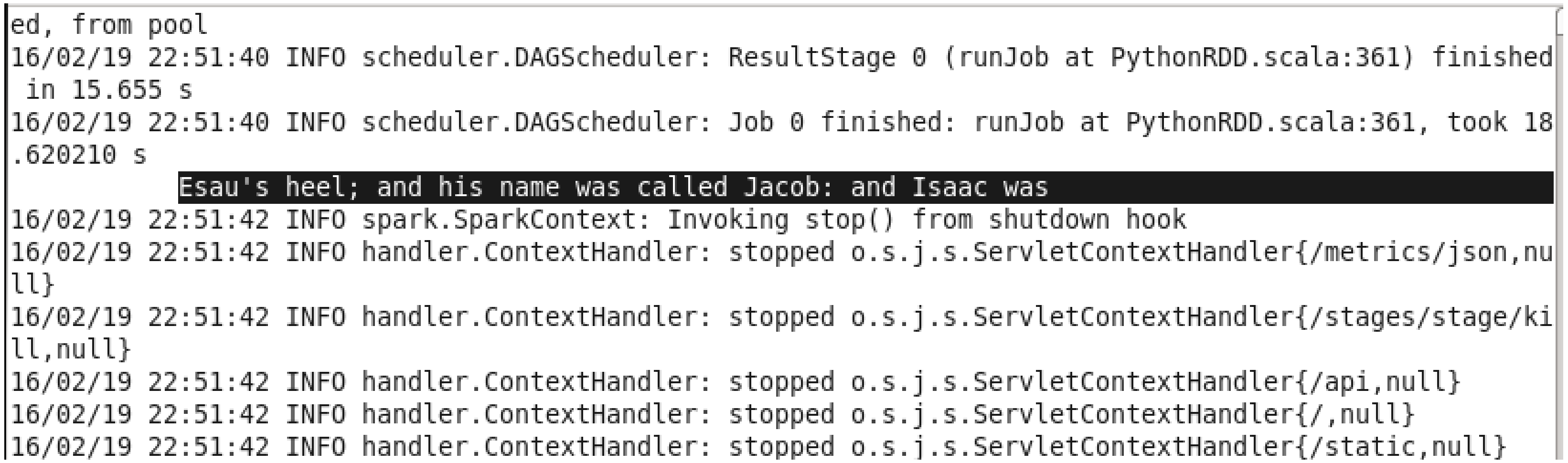
sc = SparkContext(conf = conf)

RDDvar = sc.textFile("hdfs://localhost:8020/user/cloudera/input/all-bible.txt")

lifeLines = RDDvar.filter(lambda line: "Jacob" in line)

print lifeLines.first()

Output:



```
ed, from pool
16/02/19 22:51:40 INFO scheduler.DAGScheduler: ResultStage 0 (runJob at PythonRDD.scala:361) finished
 in 15.655 s
16/02/19 22:51:40 INFO scheduler.DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:361, took 18
.620210 s
            Esau's heel; and his name was called Jacob: and Isaac was
16/02/19 22:51:42 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/02/19 22:51:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,nu
ll}
16/02/19 22:51:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/ki
ll,null}
16/02/19 22:51:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
16/02/19 22:51:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
16/02/19 22:51:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
```

Creating Word Count Script

- Run Word Count on all-bible.txt and convert to all lowercase and remove punctuation
- To run: $**SPARK_HOME/bin/spark-submit mywordcount.py**

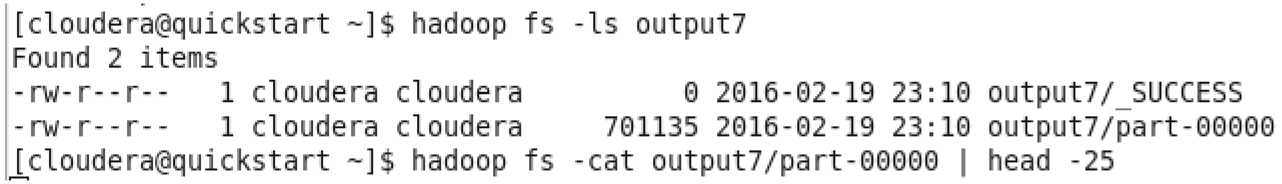
  OR

  $spark-submit mywordcount.py

```
# mywordcount.py       2/19/2016 d. howard
# counts words and converts to lowercase, removes punctuation in file all-bible.txt
# import classes (SparkConf, SparkContext, string)
from pyspark import SparkConf, SparkContext
import string

# Set the Spark configuration so WordCount is run locally
conf = SparkConf().setMaster('local').setAppName('WordCount')
sc = SparkContext(conf = conf)

# Read all-bible.txt into RDD (Variable called RDDvar)
RDDvar = sc.textFile("hdfs://localhost:8020/user/cloudera/input/all-bible.txt")

# Tokenize each line
words = RDDvar.flatMap(lambda line: line.split())

# Converts tokens to lower-case & removes punctuation before creating a tuple (token, 1)
result = words.map(lambda word: (str(word.lower())\
       .translate(None,string.punctuation), 1))
aggreg1    = result.reduceByKey(lambda a, b: a+b)

# Saves text file to hdfs directory: output
aggreg1.saveAsTextFile("hdfs://localhost:8020/user/cloudera/output7")
```

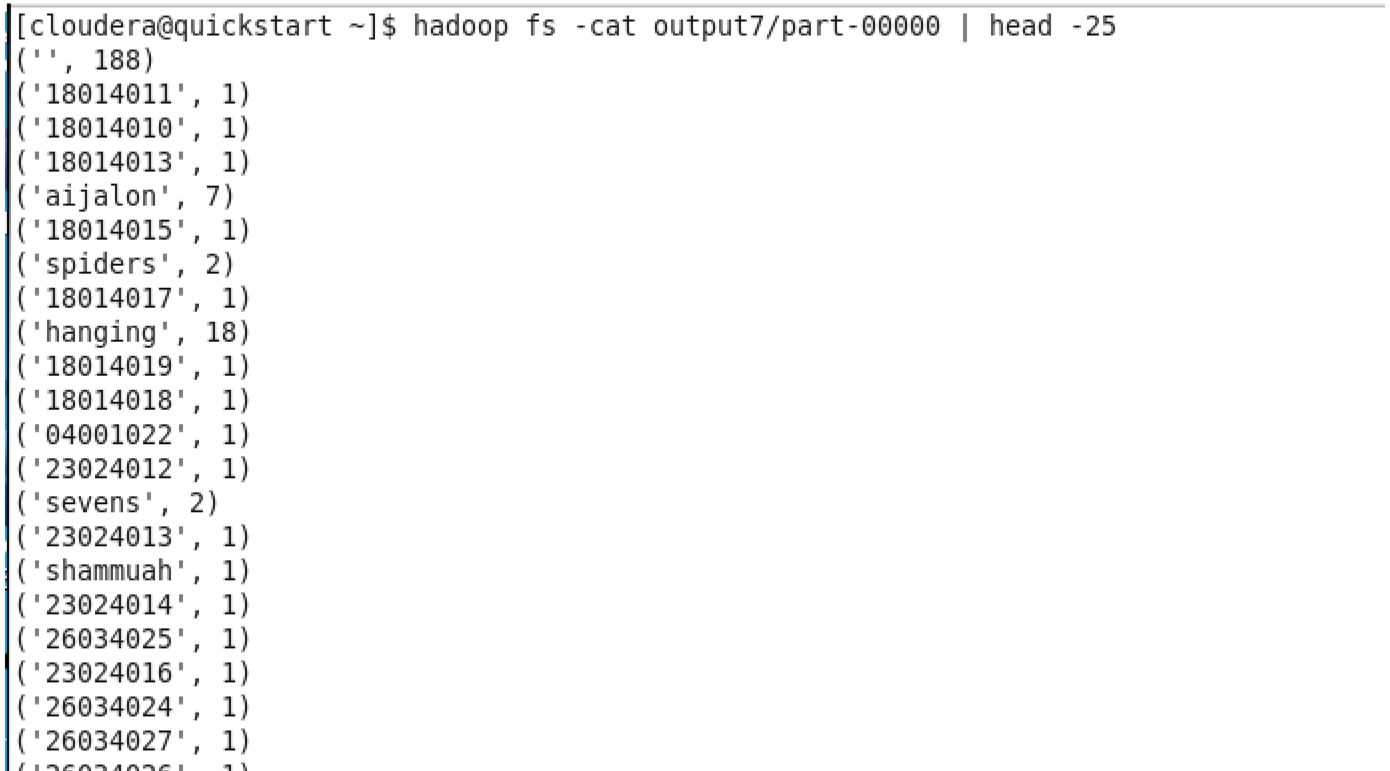
Check output folder

```
$hadoop fs -ls output7
```



```
[cloudera@quickstart ~]$ hadoop fs -ls output7
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2016-02-19 23:10 output7/_SUCCESS
-rw-r--r--   1 cloudera cloudera     701135 2016-02-19 23:10 output7/part-00000
[cloudera@quickstart ~]$ hadoop fs -cat output7/part-00000 | head -25
```

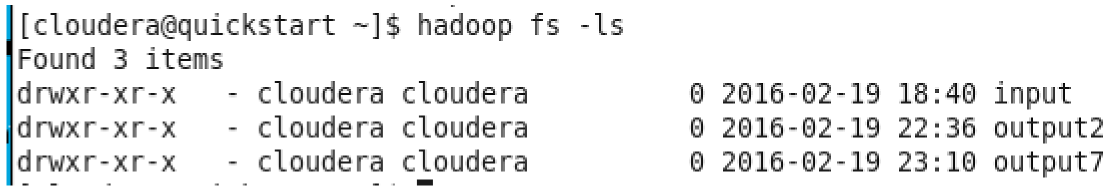
```
$hadoop fs -cat output7/part-00000 | head -25
```



```
[cloudera@quickstart ~]$ hadoop fs -cat output7/part-00000 | head -25
('', 188)
('18014011', 1)
('18014010', 1)
('18014013', 1)
('aijalon', 7)
('18014015', 1)
('spiders', 2)
('18014017', 1)
('hanging', 18)
('18014019', 1)
('18014018', 1)
('04001022', 1)
('23024012', 1)
('sevens', 2)
('23024013', 1)
('shammuah', 1)
('23024014', 1)
('26034025', 1)
('23024016', 1)
('26034024', 1)
('26034027', 1)
```

Remove your output directory!

- If you run your same script you will get an error because your output directory already exists.
- Remove your output directory after every MR run!



```
[cloudera@quickstart ~]$ hadoop fs -ls
Found 3 items
drwxr-xr-x   - cloudera cloudera          0 2016-02-19 18:40 input
drwxr-xr-x   - cloudera cloudera          0 2016-02-19 22:36 output2
drwxr-xr-x   - cloudera cloudera          0 2016-02-19 23:10 output7
```

hadoop fs -rm r output7 (note:output → your output directory)

Useful URLs and documentation

- https://www.python.org/
- https://docs.python.org/3/tutorial/index.html
- http://spark.apache.org/documentation.html

"*Learning Spark*" by Holden Karau, Andy Konwinski, Patrick Wendell & Mathei Zaharia, O'Reilly 2015

Summary

- Use Cloudera 5.5 QuickStart out of the box without having to do any install/config changes
  - login: cloudera/cloudera
- Run Spark : spark
  - Always run some simple test first
- Run Python scripts: spark-submit mywordcount.py
- Try Eclipse/Java
- Run MapReduce jobs from previous homeworks
- Always delete your hdfs output directories before you rerun same job
- Have patience Spark appears to be slow but very busy performing in-memory operations (very powerful).
  - Do not have too many other resources (windows open) or you will see slow performance.

Have fun!

GOOD LUCK!