**Clairvoyant**

**Intro to Apache Spark Workshop:**

**Exercises**

# General Notes

The Intro to Apache Spark workshop uses a modified verison of the CDH5.0 Quick Start VM (Cloudera’s Distribution, including Apache Hadoop 5.X) installed in Psuedo-Distributed mode. The VM comes pre-installed with Apache Spark running on Apache Hadoop as well as all the workshop code and data loaded onto it.

The original VM can be found at: <http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cloudera_quickstart_vm.html>

**Instructions to Install the VM**

1. Download VirtualBox from: <https://www.virtualbox.org/wiki/Downloads>
   1. Pick the appropriate binary for your operating system.
   2. Then follow the prompts for installing VirtualBox.
2. Once the installation is complete, extract the VM provided in the course workshop documents named:

**cloudera-quickstart-vm-5.4.2-0-.clairvoyant-spark-workshop.ova.zip**

You may need to download 7-zip from [www.7-zip.org](http://www.7-zip.org) to extract the Cloudera VM. After uncompressing, you will get a file called “cloudera-quickstart-vm-5.X.X-0-virtualbox.ovf.” Move it to location of your choice in File System.

1. Once you have extracted the VM, we will load the VM into VirtualBox.
   1. Open Virtual Box and click on **File -> Import Appliance...**
   2. From the file dialog open **cloudera-quickstart-vm-5.4.2-0-.clairvoyant-spark-workshop.ova** located in the decompressed (or unzipped) Cloudera VM download.
2. Setup "Network Adapter 2" in Network Settings in the Virtual Box as "Host-only Adapter." Another option is bridged but it has a bug in Mac when using a wireless connection.
   1. If when setting up the "Host-only Adapter," the "Name" drop down is showing only "Not selected", cancel and go back to Virtual Box preferences ("Virtual Box -> preferences ->network).

* 1. Select **Host-Only Networks** then **Add** and a new entry will be created (something like "vboxnet0").
  2. Click **OK**. Now go back to Network Settings on the VM. This time Adapter 2 should show vboxnet0 in the "Name" drop down box. Select "vboxnet0."

1. Update Virtual Box to the latest version if you are not able to add as described in step 4. The menu might be present in an older version, but may throw an exception/error message when you attempt to add.
2. Add **Port 50010** to NAT adapter.
3. Open the VM Network Settings.
   1. Go to **Adapter 1**, 'Attached to:' NAT should display.
   2. Add **Port 50010** to Port Forwarding.
4. Start the VM.
5. Once the VM starts up, you should see the Desktop within VirtualBox. This is your sandbox to play with Hadoop.
6. Test to ensure the VM was setup successfully. Open Terminal and run the following commands:
   1. $ hadoop version
      1. If the setup is successful, it will print the current version.
   2. $ spark-shell
      1. In the shell that opens run command:

scala> **sc**

1. If the setup is successful, it will print a SparkContext object

**VM Notes**

* Workshop code is available at

“/home/cloudera/spark-workshop/spark\_workshop\_codebase”

* Workshop data is available at 2 locations:
  + On the local file system of the VM at:

“/home/cloudera/spark-workshop/spark-workshop-code”

* + On HDFS of the VM at:

“/user/cloudera/spark-workshop-code”

* Credentials for the VM
  + Username: cloudera
  + Password: cloudera
* You can SSH to the VM by running the following command:
  + $ ssh -p 2222 cloudera@localhost
* You can copy files to the VM by running the following command:
  + $ scp -P 2222 {path\_to\_local\_file} cloudera@localhost:{destination}

# Exercise 1 – Running Spark Jobs

In this exercise you will practice submitting spark jobs using the methods mentioned in the slides. The job you will submit will take in a list of strings and return the strings that start with “w”.

**Spark-Shell**

1. Open Spark Shell

spark-shell --master yarn-client

1. Wait for the shell to come up with the following prompt

scala>

1. Type in the following Scala code

val list = List("who", "what", "when" ,"where", "how")

val data = sc.parallelize(list)

val wData = data.filter(\_.startsWith("w"))

wData.collect()

1. After running the above code you should get the following result

res5: Array[String] = Array(who, what, when, where)

1. Congratulations you just ran a Spark job using Scala!

**Pyspark**

1. Open Pyspark

pyspark --master yarn-client

1. Wait for the shell to come up with the following prompt

>>>

1. Type in the following Python code

list = ["who", "what", "when" ,"where", "how"]

data = sc.parallelize(list)

wData = data.filter(lambda x: x.startswith("w"))

wData.collect()

1. After running the above code you should get the result:

['who', 'what', 'when', 'where']

1. Congratulations you just ran a Spark job using Python!

**Spark Submit**

1. Go to the spark\_workshop code base provided (on VM at /home/cloudera/spark-workshop/spark\_workshop\_codebase) and go to the exercise1 module. Run maven install to build the needed jar file:
   1. Note 1: The maven build has not been configured to set the main class. So when you submit the job you will need to define the main class to run as a command line argument.

$ cd /home/cloudera/spark-workshop/spark\_workshop\_codebase/exercise1

$ mvn clean install

1. Verify the required jar was built

$ cd /home/cloudera/spark-workshop/spark\_workshop\_codebase/exercise1/target

# The jar “com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar” should have been built

1. Submit Java Code

$ cd /home/cloudera/spark-workshop/spark\_workshop\_codebase/exercise1/target

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise1.java.Exercise1JavaSparkApp com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar who what when were why how

#Output: [who, what, when, were, why]

1. Submit Scala Code

$ cd /home/cloudera/spark-workshop/spark\_workshop\_codebase/exercise1/target

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise1.scala.Exercise1ScalaSparkApp com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar who what when were why how

#Output: who what when where

1. Submit Python Code

$ cd /home/cloudera/spark-workshop/

spark\_workshop\_codebase/exercise1/src/main/python

$ spark-submit Exercise1PythonSparkApp.py who what when were why how

#Output: ['who', 'what', 'when', 'where', 'why']

1. Congratulations you just ran a Spark job as a pre-packaged/built file!

# Exercise 2 – Access Logs

In this exercise you will analyze the access.log file using spark by calculate the following:

* Count how many times the “/health” URL was hit
* Get all events that occurred on May 19th 2014 and save them to HDFS

Access log file can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “logs” subdirectory
* In HDFS (on the VM provided) at

/user/cloudera/spark-workshop-data/logs/access.log

# Exercise 3 – Joining Datasets

In this exercise you will be finding out how many times the work “Spark” shows up in the README.md and CHANGES.txt by following the bellow steps:

1. Create RDD’s to filter each file for the keyword “Spark”
2. Perform a WordCount on each of the resulting datasets so the results are (K, V) pairs of (word, count)
3. Join the two RDDs

Files can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “spark” subdirectory
* In HDFS (on the VM provided) at

/user/cloudera/spark-workshop-data/spark/

# Exercise 4 – Shared Variables

In this exercise you will take a file with mock bank transaction data and process it using Shared Variables.

File can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “transactions” subdirectory
* In HDFS (on the VM provided) at

/user/cloudera/spark-workshop-data/transactions/

File is a tab-separated value file without a header. The file had the scheme:

UserID, Name, TransactionID, TransactionCode, Reason, BankID

Steps

1. Create a map with the following key value pairs (where the key is the TransactionCode and the value is a translated TransactionCode) and **Broadcast** it to the nodes:

C -> CASH\_ADVANCE

S -> BALANCE\_INQUIRY

B -> BALANCE\_TRANSFER

A -> OTHER

V -> OTHER

O -> OTHER

P -> PREAUTHORIZED

R -> AUTHORIZED

1. Use an **Accumulator** to count how many transactions from Bank “A” were of type “OTHER”.