Maharishi University of Management

CS522 – Big Data

Dr. Premchand Nair

Set Up a Single Node Cluster using VM or Docker

Table of Contents

Creating a Hadoop Project

Open Eclipse and Start a New Java Project.

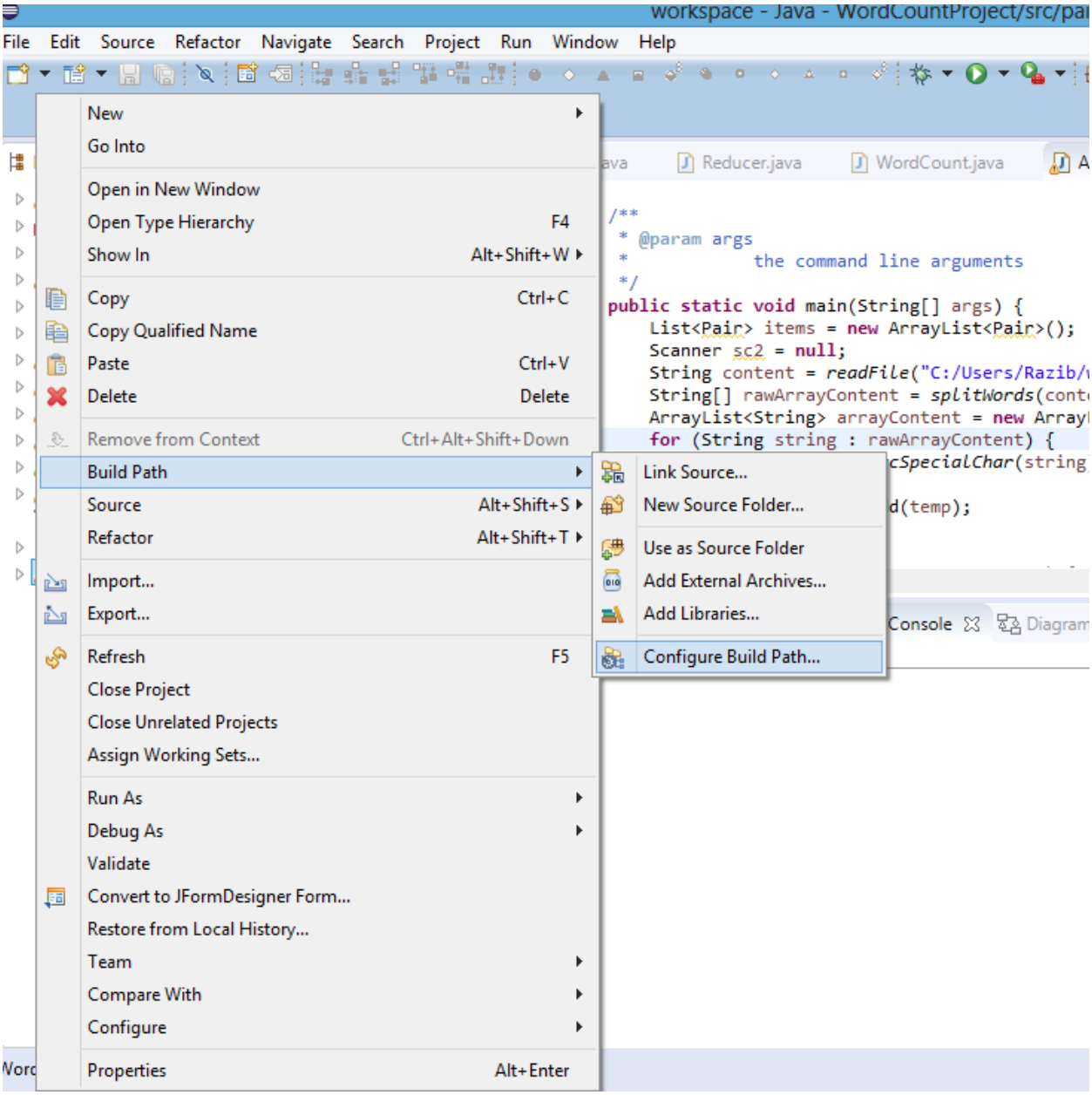
If you want to use Cloudera Quickstart VM way, follow the steps to install it first at **Error! Reference source not found.** before following this chapter.

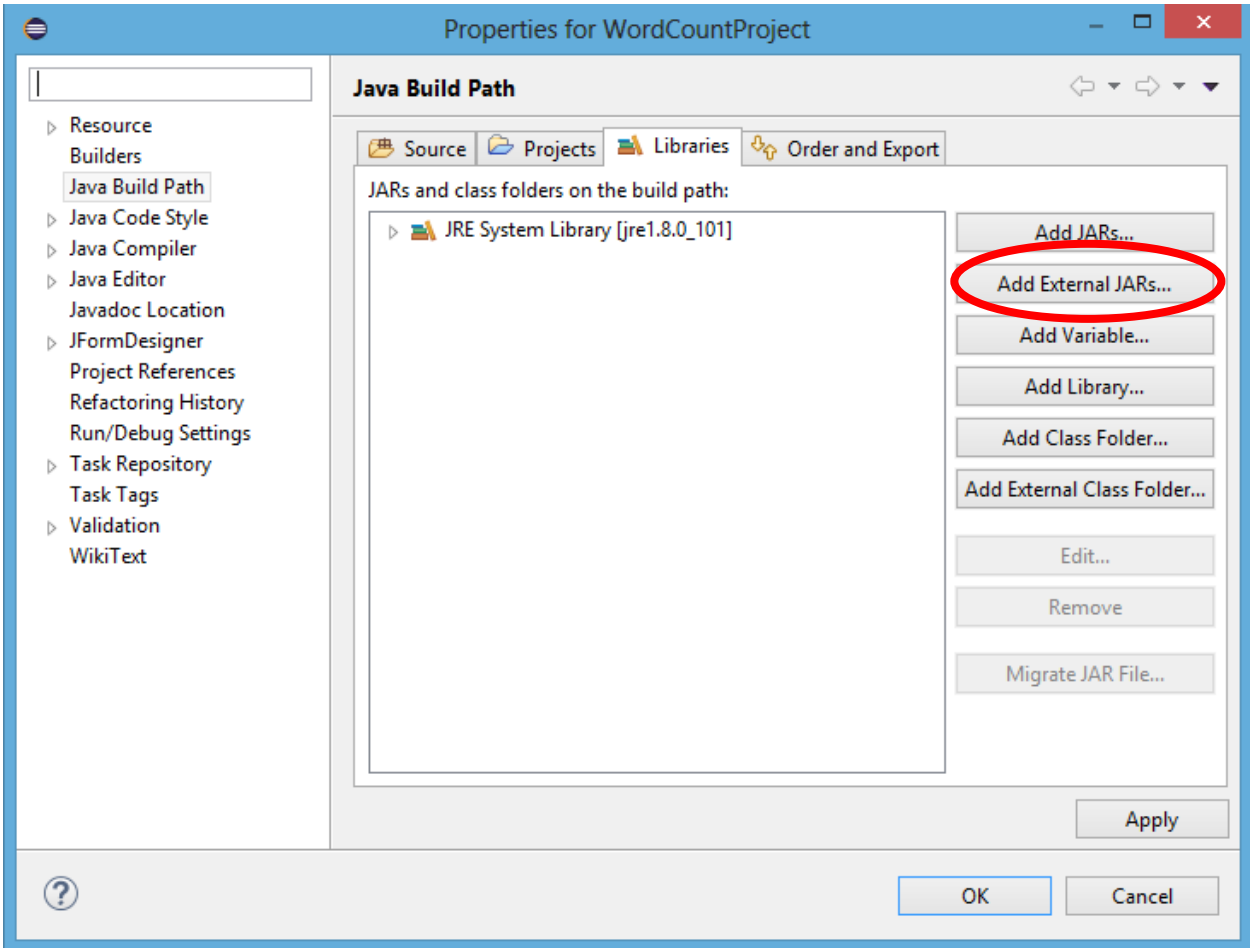
Include Hadoop Jar files to project

Download Hadoop Common and Hadoop MapReduce Client Core jar files from below links:

http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common/2.7.3
http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core/2.7.3

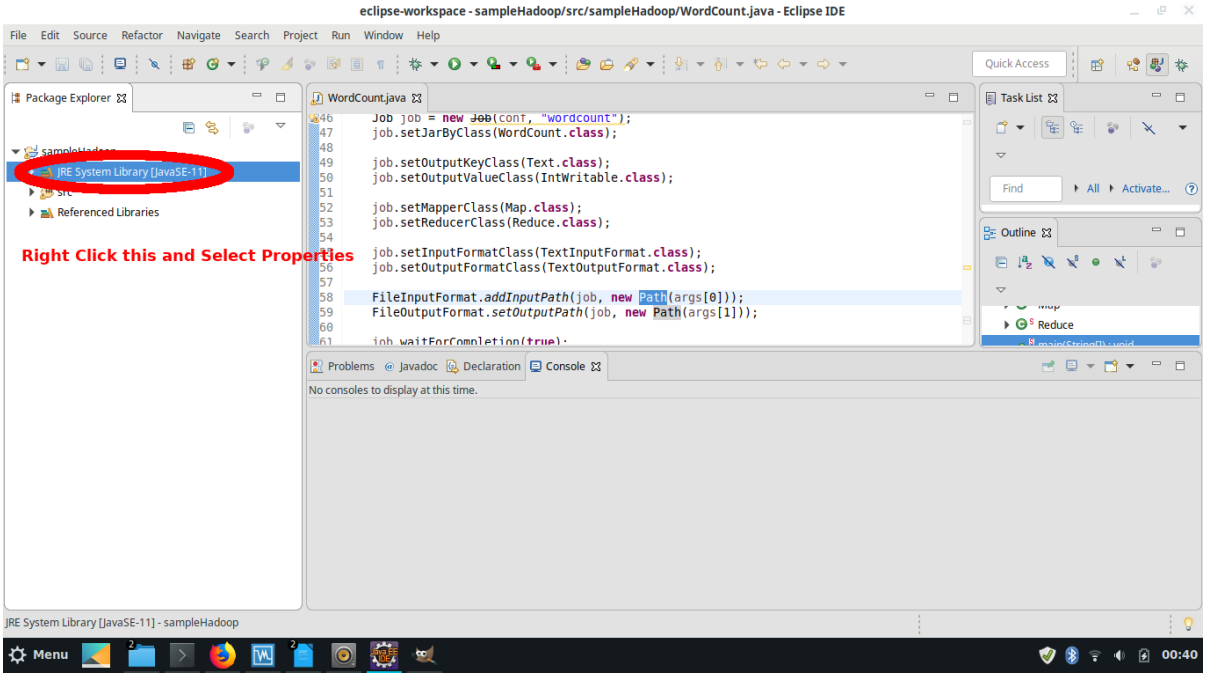
Add those JAR files to your Project

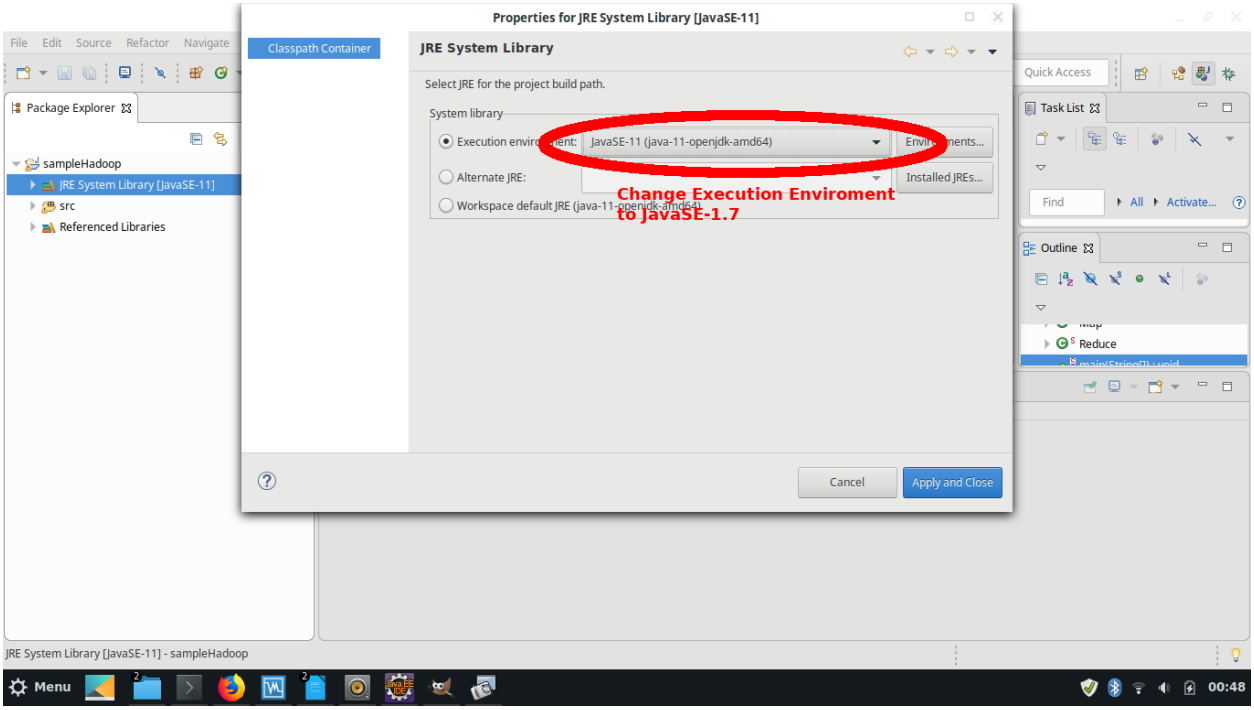




Change the Java Execution Environment

To be able to run the Jar file inside Hadoop, make sure you change your Java Environment to JDK 1.7





Copying WordCount Example from Wiki and Exporting JAR

Get WordCount example from https://wiki.apache.org/hadoop/WordCount

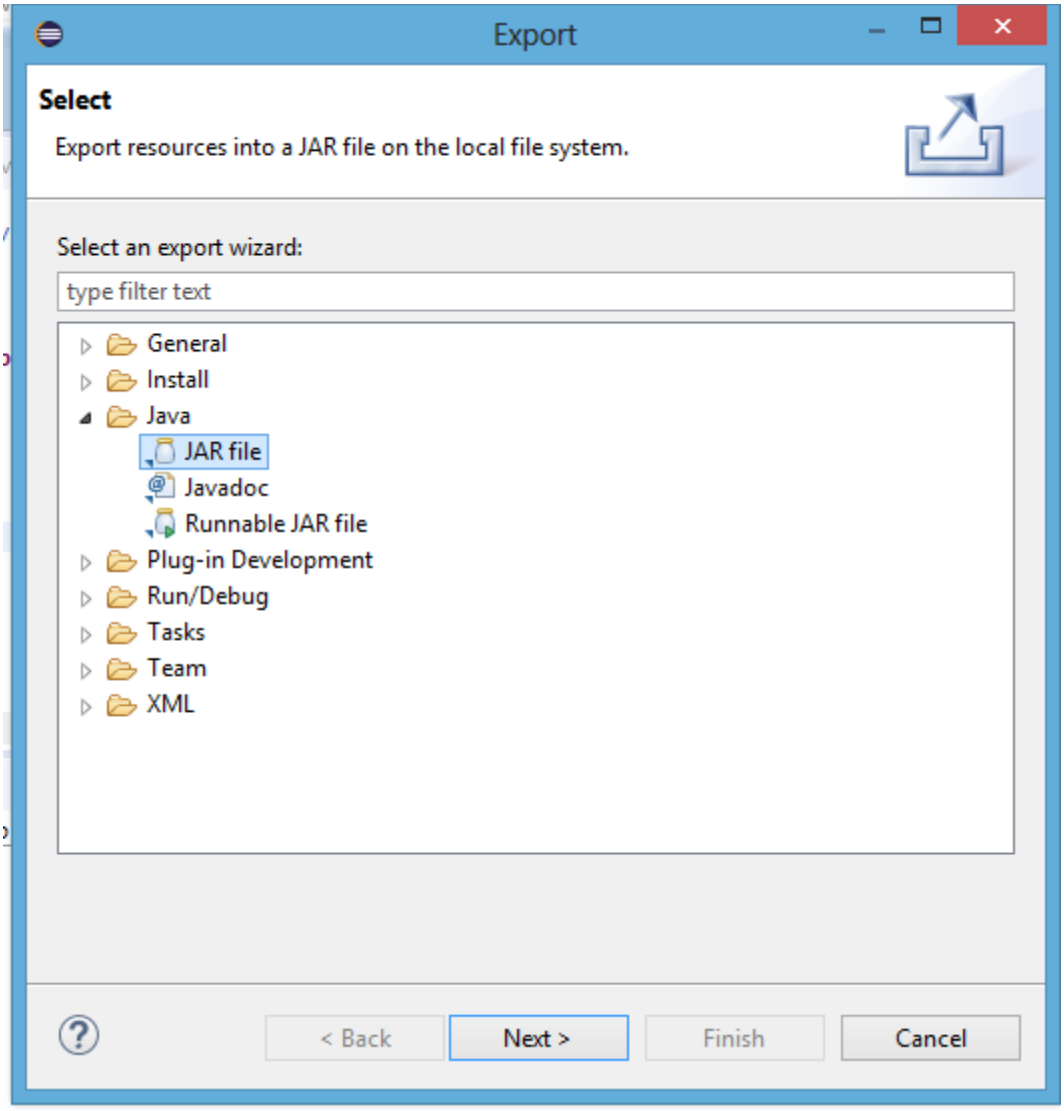
AFTER THIS LINE IN THE CODE

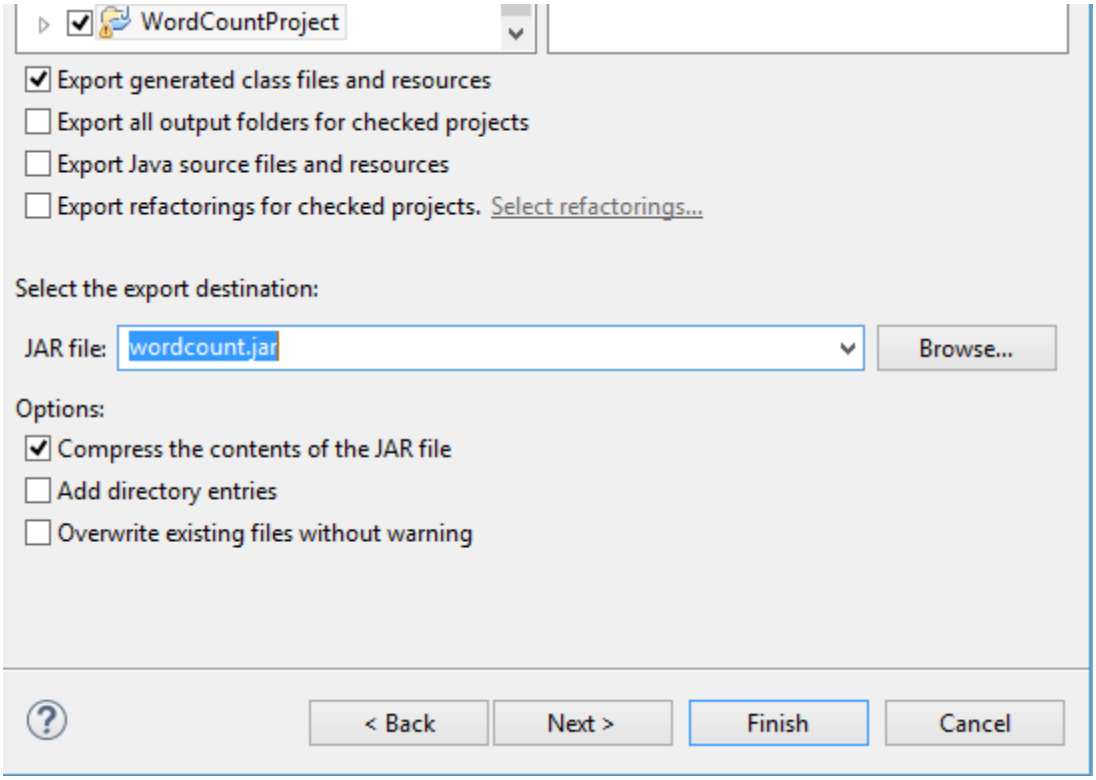
```
Job job = new Job(conf, "wordcount");
```

ADD

```
job.setJarByClass(WordCount.class);
```

Export to WordCount.jar to test the WordCount example. Right click on Project, choose “Export”. The Export window will appear, choose Java -> JAR file. Click Next.





After these steps, your JAR file is complete. Make sure you know its location as it is going to be important on the next steps.

Now you can choose either the VM approach using Cloudera Quickstart or the Hadoop Docker Container approach.

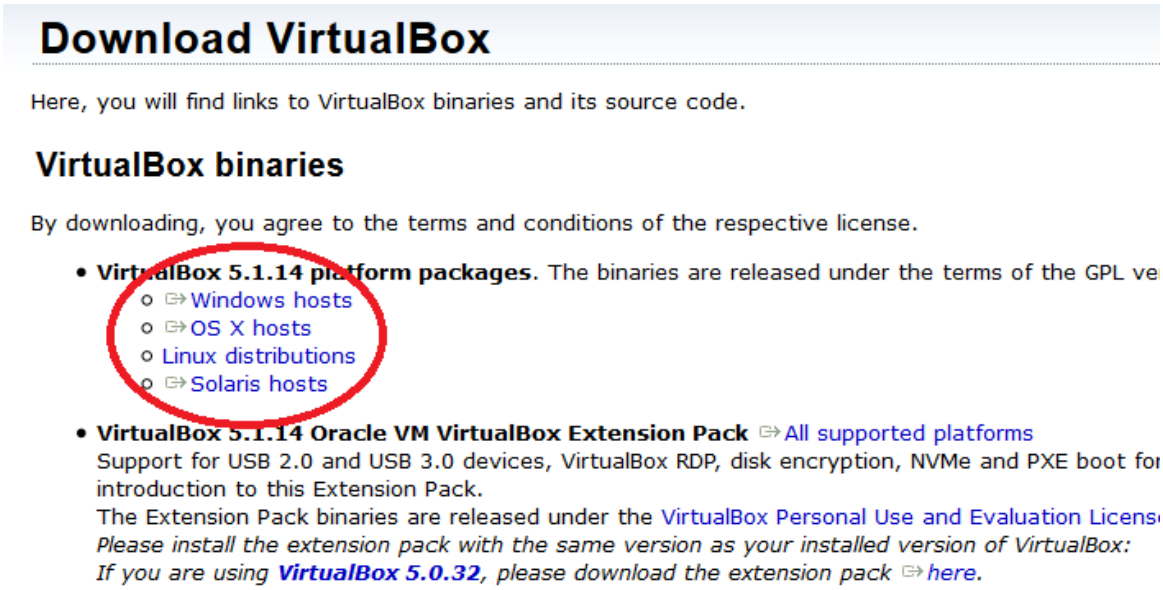
Using Cloudera Quickstart VM

Download Cloudera

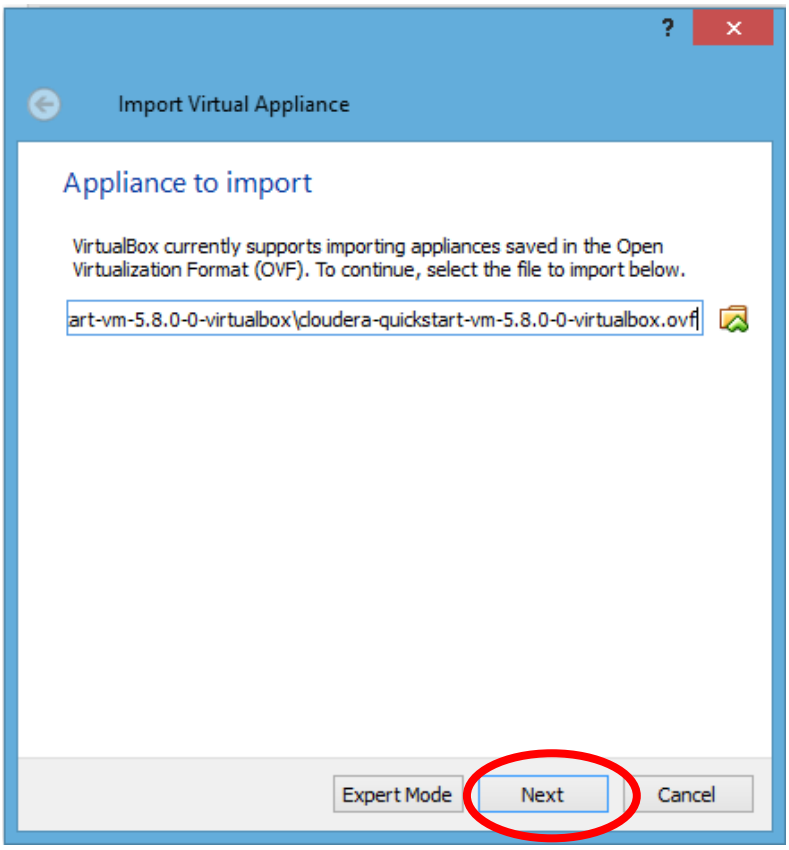
download the latest free version of Cloudera QuickStart from here
https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip

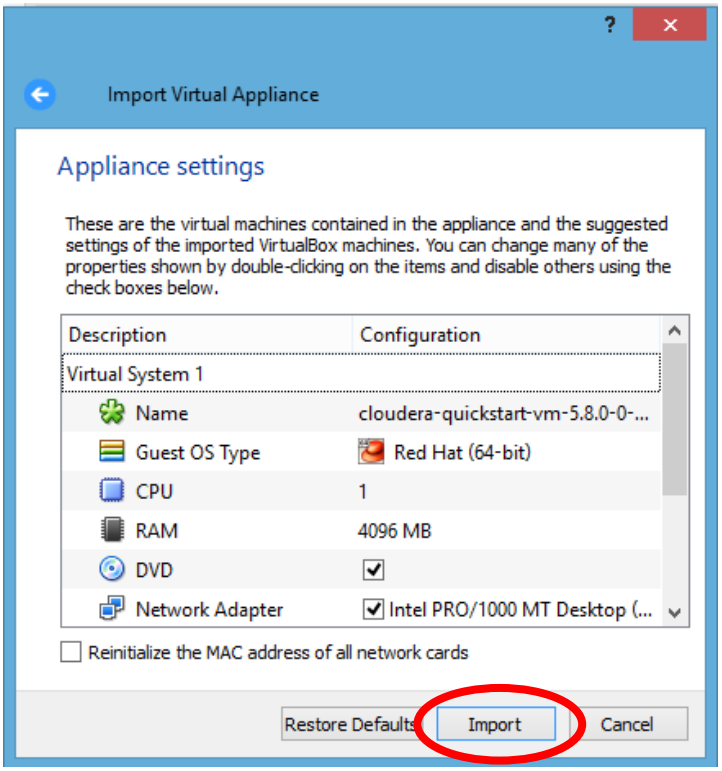
Download Virtual Box

from https://www.virtualbox.org/wiki/Downloads

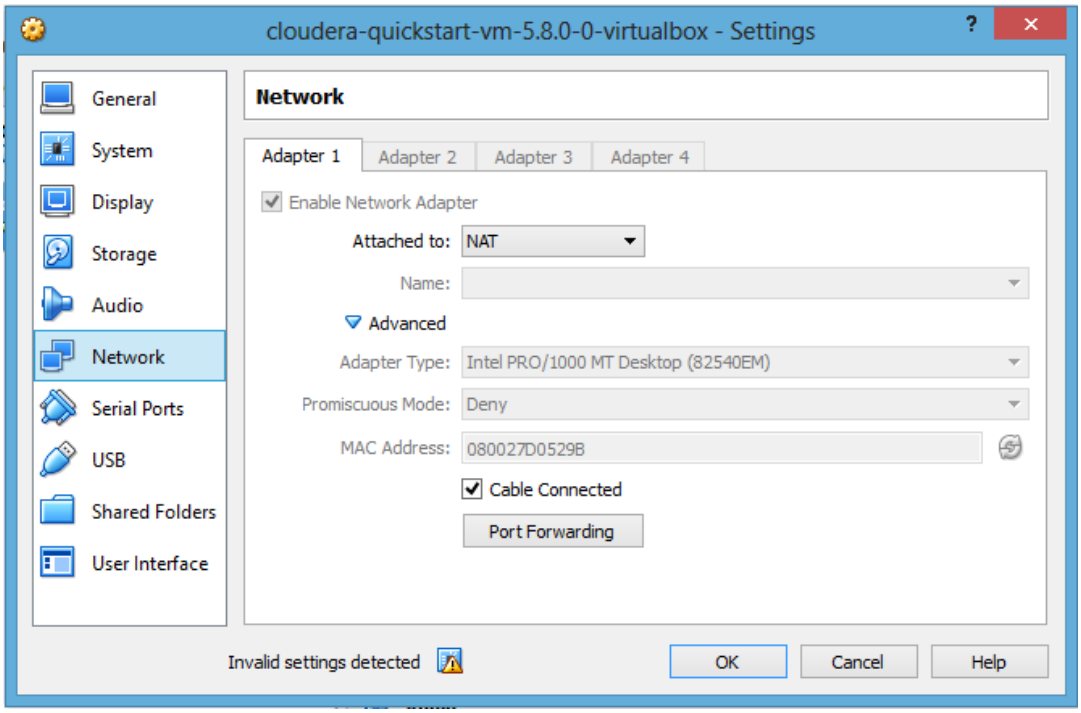


Click on File > Import Appliance

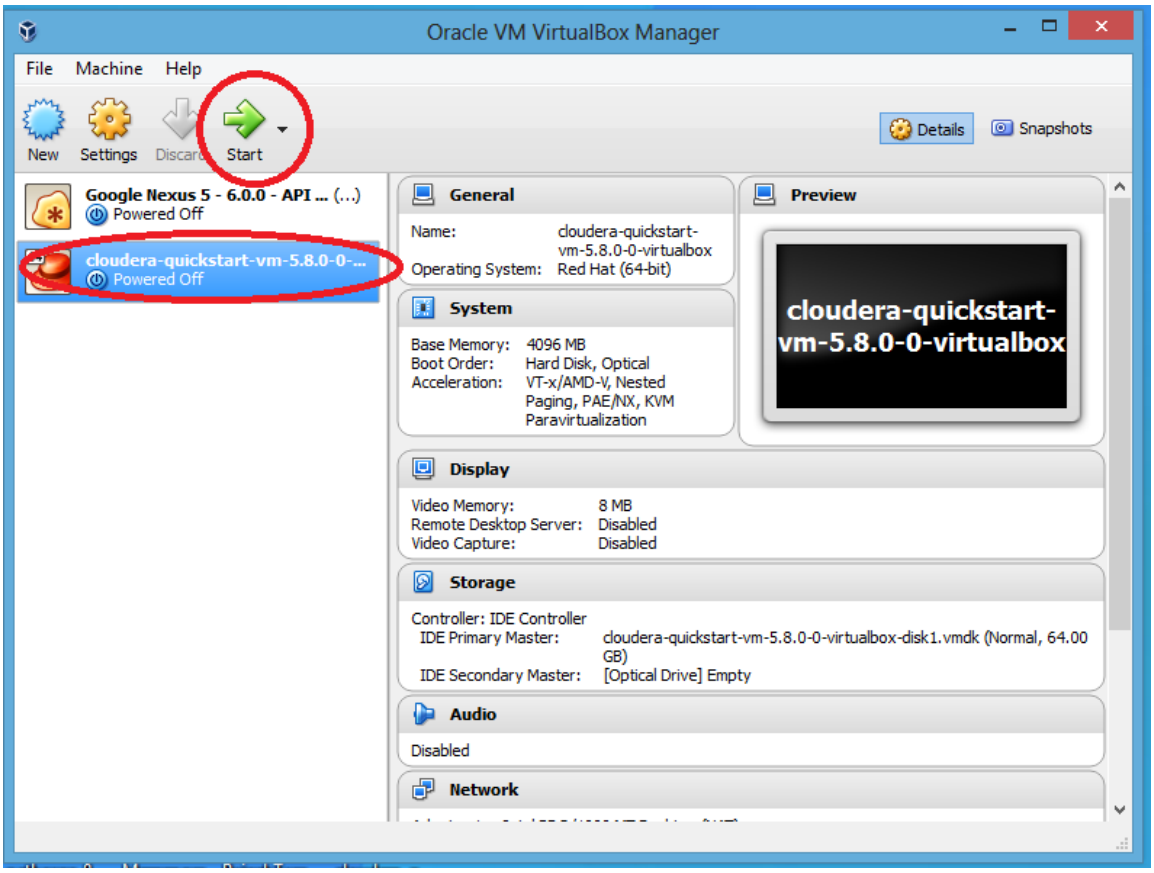

Import Virtual Appliance
Appliance to import
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.
art-vm-5.8.0-0-virtualbox\cloudera-quickstart-vm-5.8.0-0-virtualbox.ovf
Expert Mode
Next
Cancel

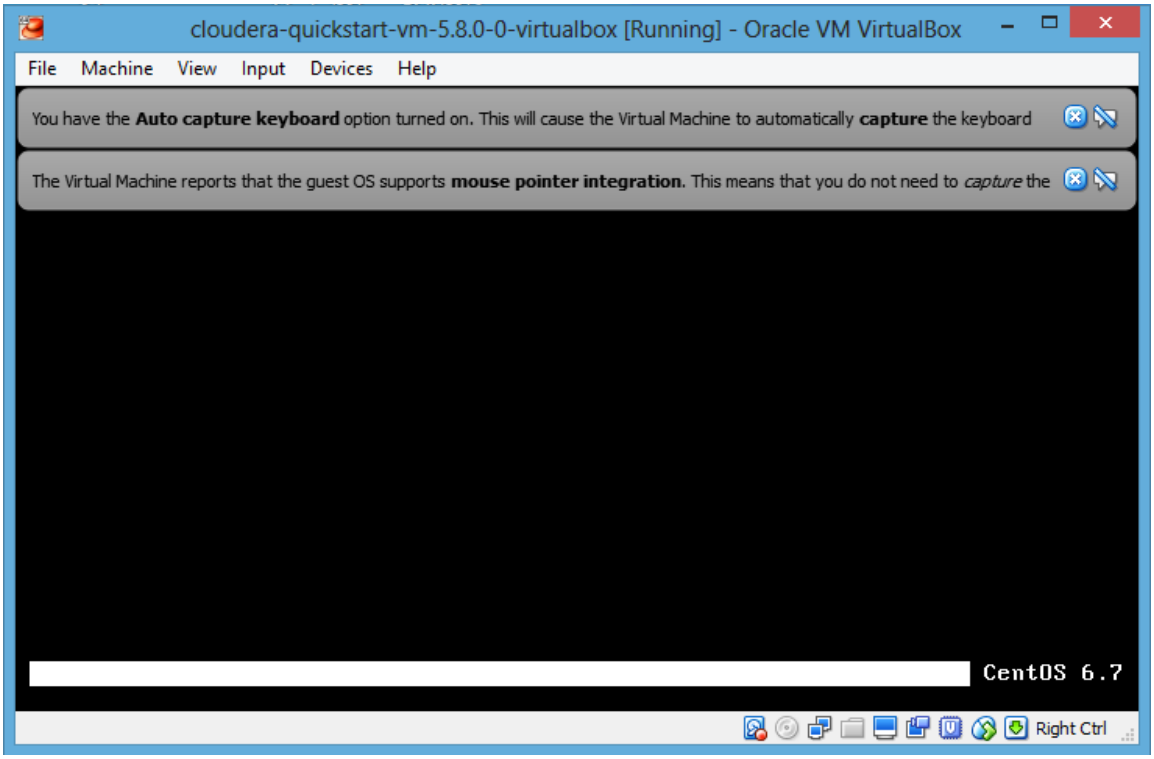

Import Virtual Appliance
Appliance settings
These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.
Description
Configuration
Virtual System 1
Name
cloudera-quickstart-vm-5.8.0-0-...
Guest OS Type
Red Hat (64-bit)
CPU
1
RAM
4096 MB
DVD
Network Adapter
Intel PRO/1000 MT Desktop (...
Reinitialize the MAC address of all network cards
Restore Defaults
Import
Cancel

To enable network access go to Machine > Settings > Network and choose NAT in Attached to option as below:

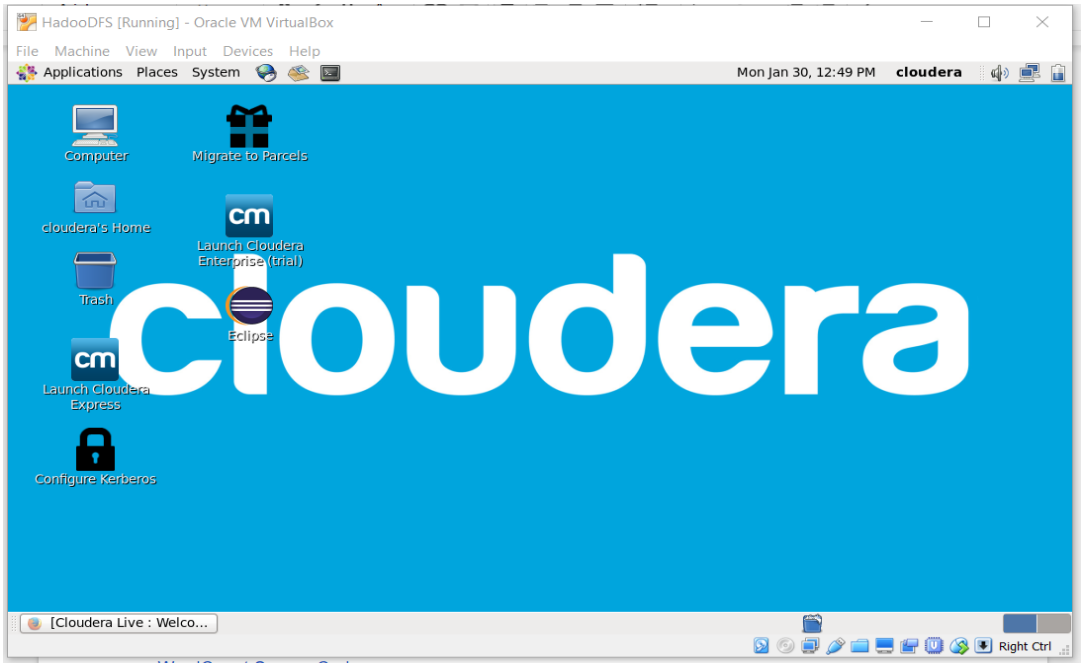


Now start Cloudera VM by clicking start button:

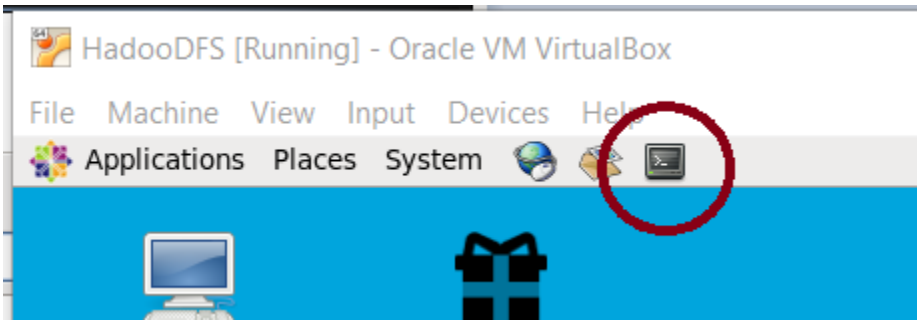

Oracle VM VirtualBox Manager
File
Machine
Help
New
Settings
Discard
Start
Details
Snapshots
Google Nexus 5 - 6.0.0 - API ... (...)
Powered Off
cloudera-quickstart-vm-5.8.0-0-...
Powered Off
General
Name: cloudera-quickstart-vm-5.8.0-0-virtualbox
Operating System: Red Hat (64-bit)
System
Base Memory: 4096 MB
Boot Order: Hard Disk, Optical
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization
Preview
cloudera-quickstart-vm-5.8.0-0-virtualbox
Display
Video Memory: 8 MB
Remote Desktop Server: Disabled
Video Capture: Disabled
Storage
Controller: IDE Controller
IDE Primary Master: cloudera-quickstart-vm-5.8.0-0-virtualbox-disk1.vmdk (Normal, 64.00 GB)
IDE Secondary Master: [Optical Drive] Empty
Audio
Disabled
Network


cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically capture the keyboard
The Virtual Machine reports that the guest OS supports mouse pointer integration. This means that you do not need to capture the
CentOS 6.7
Right Ctrl

Desktop will look like below:



Open terminal program:



After these steps, your Cloudera Environment is ready and you can skip to

Using Docker

Install Docker

First install Docker on your machine using the instructions from official website: https://docs.docker.com/install/

After installing it, check if Docker is installed correctly opening your terminal and typing the following command:

```
$ docker -v
```

It should return something like this (it can be another version, make sure yours is equal or greater than 18):

```
Docker version 18.09.7, build 2d0083d
```

If this is OK proceed to the next step, if it isn't, try to reinstall Docker.

Install Hadoop Image on Docker

Use the following command to pull the image to your docker machine:

```
# docker pull sequenceiq/hadoop-docker
```

Attention! Each time you see a # before a command, it means you have to run that command as a superuser (root user). That user is very similar to an Administrator account in Windows. If you receive permission denied error, try to run using sudo before the command. For example, if you want to run docker pull sequenceiq/hadoop-docker and it results in permission denied, try to run sudo docker pull sequenceiq/hadoop-docker instead. If this doesn't work on your Linux Distribution try to search on the internet how to run a command as administrator in that distribution.

After pulling the image, try to start the container using the following command:

```
# docker run -it sequenceiq/hadoop-docker:2.7.0 /etc/bootstrap.sh -
bash
```

The above command output should be something similar to this:

```
edgar@edgar-Aspire-A515-41G:~/eclipse/jee-2019-06/eclipse$ sudo docker
run -it sequenceiq/hadoop-docker /etc/bootstrap.sh -bash
/
Starting sshd:                                             [  OK  ]
Starting namenodes on [efea29053abf]
efea29053abf: starting namenode, logging to
/usr/local/hadoop/logs/hadoop-root-namenode-efea29053abf.out
localhost: starting datanode, logging to
/usr/local/hadoop/logs/hadoop-root-datanode-efea29053abf.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to
/usr/local/hadoop/logs/hadoop-root-secondarynamenode-efea29053abf.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--
resourcemanager-efea29053abf.out
localhost: starting nodemanager, logging to
/usr/local/hadoop/logs/yarn-root-nodemanager-efea29053abf.out
bash-4.1#
```

Leave that Terminal Window alone for now, that is the terminal for the Hadoop Docker Container.

To check if everything is OK, open another terminal window and type:

```
# docker ps
```

The expected output should be:

```
edgar@edgar-Aspire-A515-41G:~/eclipse/jee-2019-06/eclipse$ sudo docker
ps
CONTAINER ID        IMAGE                      COMMAND
CREATED             STATUS              PORTS
NAMES
efea29053abf        sequenceiq/hadoop-docker   "/etc/bootstrap.sh -…"
7 minutes ago       Up 7 minutes        2122/tcp, 8030-8033/tcp,
8040/tcp, 8042/tcp, 8088/tcp, 19888/tcp, 49707/tcp, 50010/tcp,
50020/tcp, 50070/tcp, 50075/tcp, 50090/tcp   silly_beaver
```

Sharing files (JAR and Input Files) between your machine and the Docker container

For us the be able to run Hadoop commands, like hadoop jar, we need to copy some files from our local machine to that Docker Container containing Hadoop. To do this, you have to know its Container ID as it is randomly generated each time you run it. To see it, run docker ps as was shown above (in the example above the Container ID was efea29053abf).

When you know your docker id you can upload your jar file using the following command (REPLACE efea29053abf with your Container ID and JarFileLocation with your real Jar File Location in your machine!):

```
# docker cp JarFileLocation efea29053abf:/usr/local/hadoop
```

The above command copies the file JarFileLocation from Local Machine to folder */usr/local/hadoop* inside the Docker Container with ID efea29053abf.

With this, you can also copy any file from your machine to the container, not just the jar.

Executing Hadoop commands using Docker

Once you are done with the above commands, you can run any Hadoop command **inside the container terminal.**

First we change our current directory to the Hadoop directory

```
$ cd /usr/local/hadoop
```

Then we can execute hadoop commands using bin/hadoop, for example:

```
$ bin/hadoop fs -mkdir /user/cloudera
```

PS: Don't forget that the command in this Docker Container is bin/hadoop, not just hadoop!!

In this setup, each time you terminate the Docker Container all your modifications to the HDFS are lost and you have to repeat all those steps!

Creating the Directory Structure inside HDFS

Open Terminal and first create input locations in HDFS. Using the following commands:

```
hadoop fs -mkdir /user/cloudera /user/cloudera/wordcount
/user/cloudera/wordcount/input
```

We also have to send files to the input folder. We can do using the following commands.

```
echo "Hadoop is an elephant" > file0
echo "Hadoop is as yellow as can be" > file1
echo "Oh what a yellow fellow is Hadoop" > file2
hadoop fs -put file* /user/cloudera/wordcount/input
```

With these commands we are creating three files named file0, file1 and file2 and sending them to the HDFS folder /user/cloudera/wordcount/input using hadoop fs -put.

Attention: The output folder is going to be created by the JAR Application by the time you run it with Hadoop, don't create it manually or else it is going to fail.

Running the Hadoop Application using the JAR file

Run the WordCount application from the JAR file we created on Eclipse, giving the paths to the input and output directories in HDFS.

```
hadoop jar wordcount.jar org.myorg.WordCount
/user/cloudera/wordcount/input /user/cloudera/wordcount/output
```

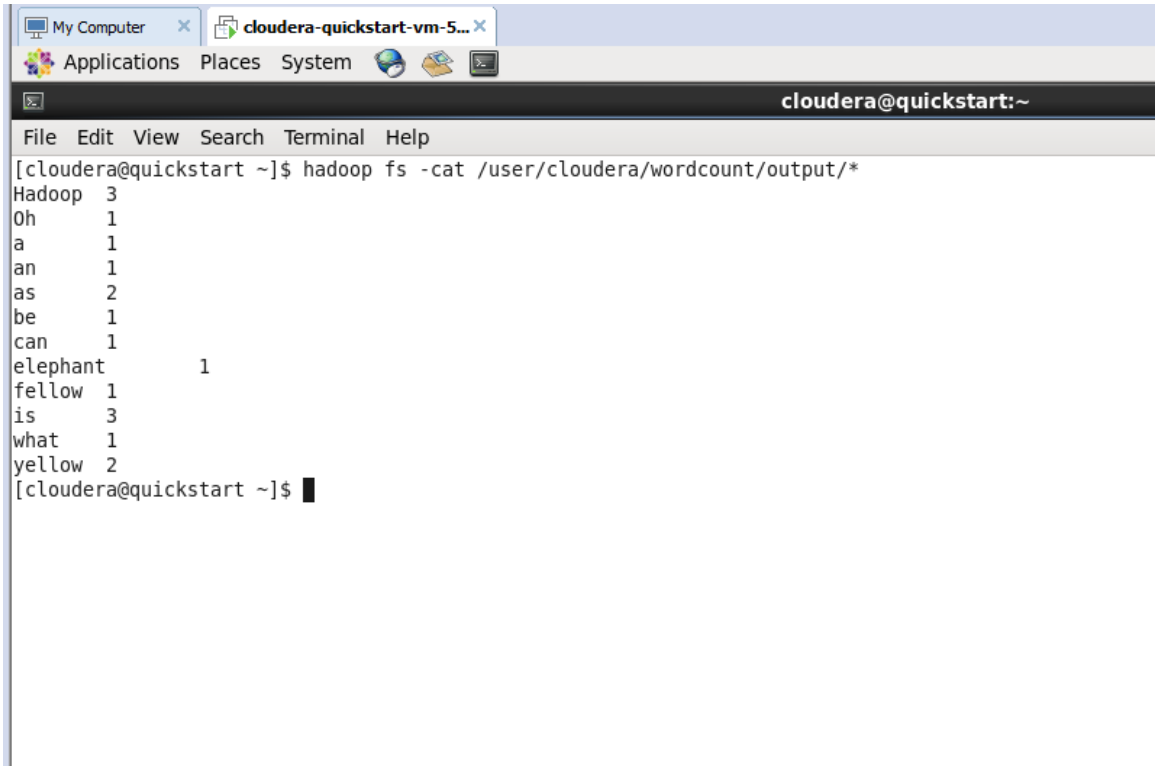
(Make sure that the package name org.myorg is the same as you used on your project! Change it if it is different.)

The output should be as following:

```
17/02/01 12:38:44 INFO mapreduce.Job:  map 0% reduce 0%
17/02/01 12:38:58 INFO mapreduce.Job:  map 100% reduce 0%
17/02/01 12:39:08 INFO mapreduce.Job:  map 100% reduce 100%
17/02/01 12:39:09 INFO mapreduce.Job: Job job_1485979610631_0001 completed successfully
17/02/01 12:39:09 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=226
                FILE: Number of bytes written=233771
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=204
                HDFS: Number of bytes written=36
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
```

In order to see the output use the following command which will be as follows or you can see from the browser (http://ipAddressOfHadoopMachine:50070/explorer.html#/) by downloading the output file.

```
hadoop fs -cat /user/cloudera/wordcount/output/*
```



If you want to run the sample again, you first need to remove the output directory. Use the following command.

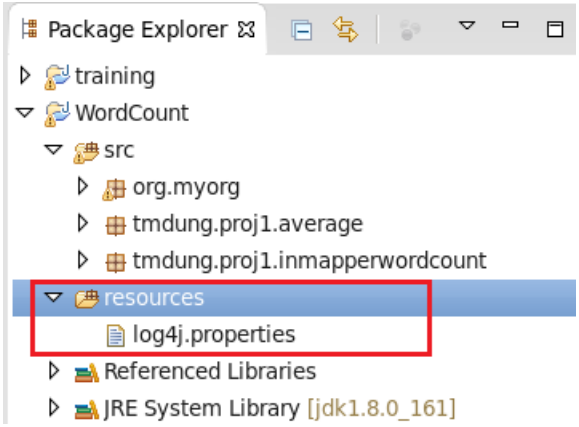
```
hadoop fs -rm -r /user/cloudera/wordcount/output
```

Hadoop setup is complete. Continue only if you want to debug Hadoop.

Add logging (Debugging) Hadoop

1. Hadoop uses log4j by default. Firsts step is to configure its settings:

    a. Add **resources** folder to your project. Then, add **log4j.properties** file to that folder



    b. Open log4j.properties file and add the following settings:

log4j.rootLogger=DEBUG, CA

log4j.appender.CA=org.apache.log4j.ConsoleAppender

log4j.appender.CA.layout=org.apache.log4j.PatternLayout

log4j.appender.CA.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n

2. Add code to log the info we need:
    a. Create Logger instance:
        i. private Logger logger = Logger.getLogger(MyMapper.class);
    b. Use respective log methods for your purpose:
        i. logger.info("Log information");
        ii. logger.error("Log some error");
        iii. logger.debug("Debug information");
3. Sample:

MyMapper.java | AverageMapper.java | InMapperWordCount.java | log4j.properties

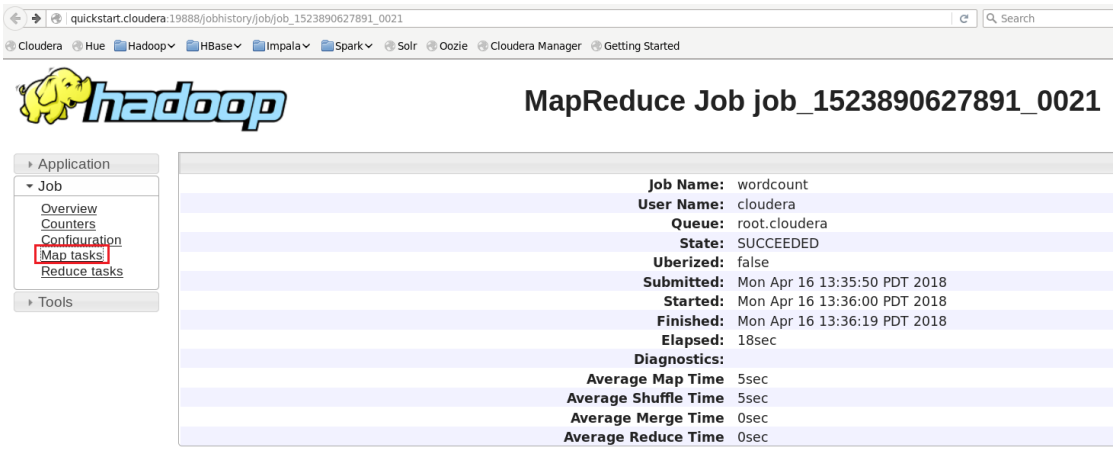
```
import org.apache.hadoop.mapreduce.Mapper;

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private HashMap<String, Integer> associativeArray;
    private Logger logger = Logger.getLogger(MyMapper.class);


    @Override protected void setup(Mapper<LongWritable,Text,Text,IntWritable>.Context context) throws IOException ,InterruptedException {
        associativeArray = new HashMap<String, Integer>();
        logger.info("Initializing NoSQL Connection.");
    };

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String line = value.toString().toLowerCase();
        line = line.replace("\"", "").replace("'", "");
        for(String s : line.split("\\s+|-")) {
            if (s.matches("[A-Za-z]+[\\.|\\,]?")) {
                s = s.replace(".", "").replace(",", "");
                int count = associativeArray.containsKey(s) ?  associativeArray.get(s) + 1 : 1;
                associativeArray.put(s, count);
            }
        }
    }

    @Override
    protected void cleanup(Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {

        logger.info("---MAPPER OUTPUT---");

        for(Entry<String, Integer> e : associativeArray.entrySet()) {
            Text key = new Text(e.getKey());
            IntWritable value = new IntWritable(e.getValue());

            logger.info("(" + e.getKey() + ", " + e.getValue() + ")");
            context.write( key, value);
        }
        super.cleanup(context);
    }
}
```

4. View logs:

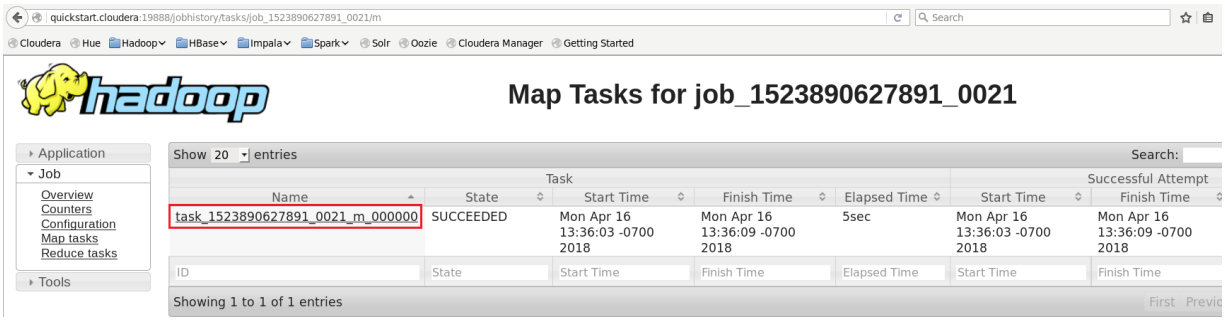
    a. The logger will by default write log to Hadoop tracker system's log file.

    b. One way to see it is open http://quickstart.cloudera:8088/proxy/<application_ID>/ in browser if you use Cloudera Quickstart VM or http://IP_OF_DOCKER_CONTAINER:8088/proxy/<application_ID>/ if you use Docker.

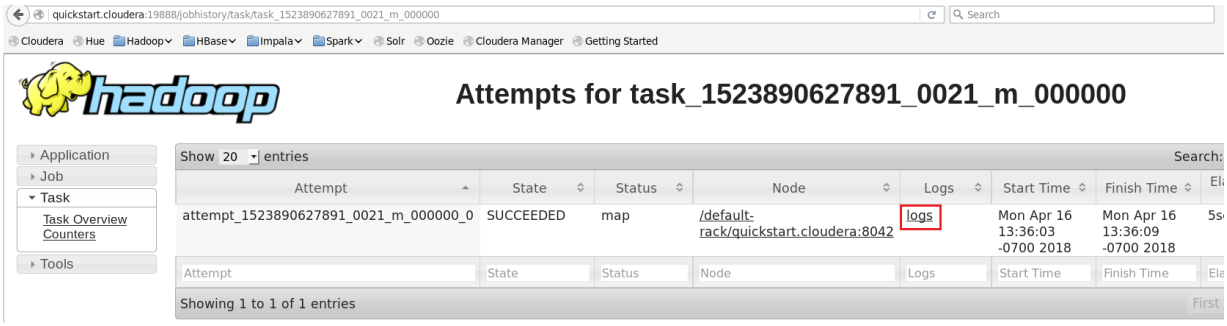
    c. Select Map Tasks on left menu



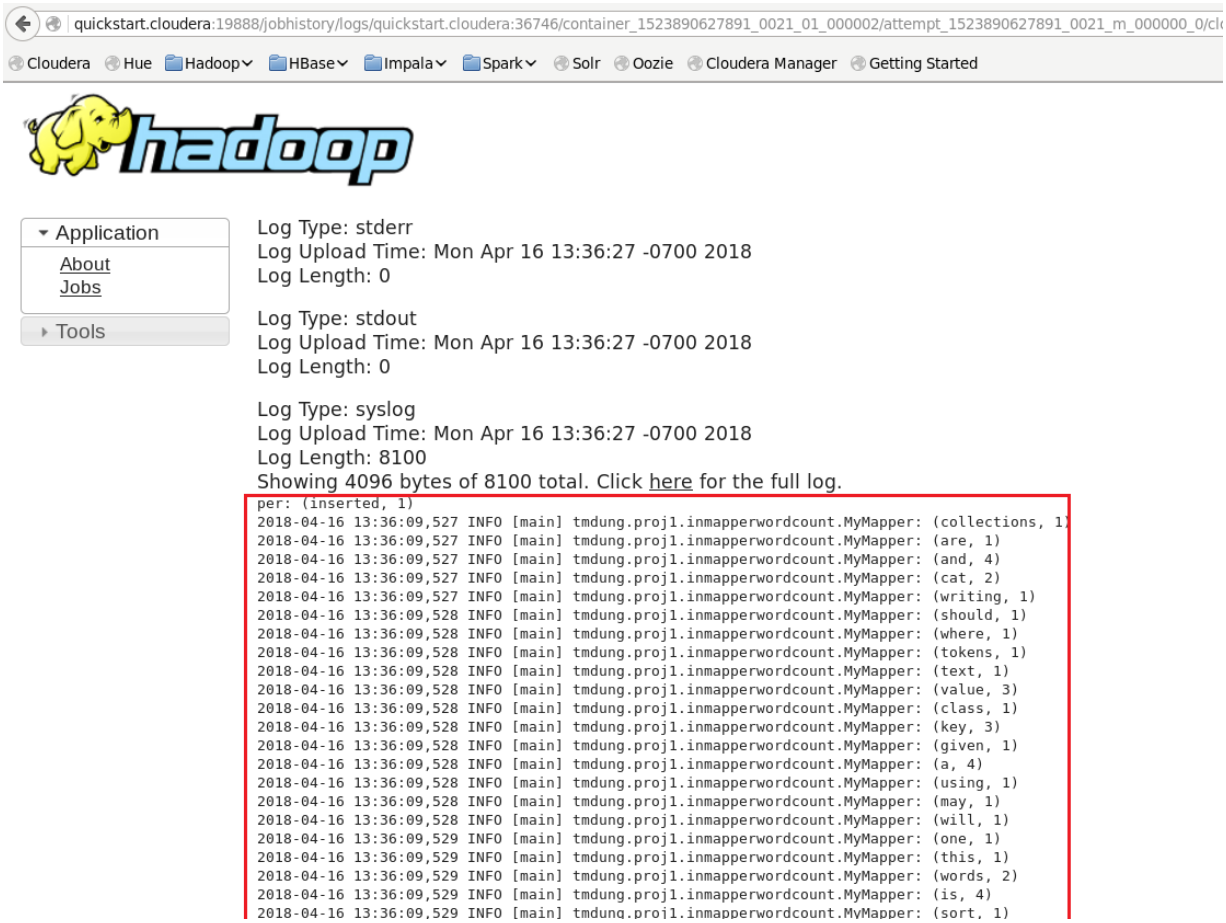
    d. Select on the task link:



e. Select logs:



f. You can see the content of logging in syslog section:



End

Running multiple MapReduce tasks

Option 1:

https://coe4bd.github.io/HadoopHowTo/multipleJobsSingle/multipleJobsSingle.html

Option 2:

Use Oozie

http://oozie.apache.org/