These steps will cover two different ways of starting Hadoop for this class. Section 1 goes through the steps of using Horton Works. Section 2 walks through all the steps for using Hadoop on a local machine without a virtual machine. Section 3 contains the steps needed to import libraries and compile Hadoop code in Java.

1 Hortonworks

This section describes all the steps you need to complete in order to install HortonWorks and get Hadoop started.

1.1 What to download

1. HortonWorks
2. Virtual Box

1.2 Installation

First you need to install the virtual machine you downloaded. I suggest the virtual box, as the least amount of students have had a problem with that. Once your virtual machine is installed, go ahead and download HortonWorks from the link provided. Once downloaded double click the downloaded HortonWorks and it should instantiate it in your virtual machine software. Go ahead and start the virtual machine.

1.2.1 Known Bugs

On occasion you may receive an error about BIOS. If this occurs go into your BIOS setting make sure that virtual machines are enabled. If that doesn’t fix the problem then try to update your BIOS. If it still doesn’t work, then I would suggest moving to Section 2.
1.3 Virtual Machine Commands

Once in HortonWorks, you will need to login as root. The default user and password are shown to you and you will be required to change your password. Once you are logged in as root, you can view the HortonWorks GUI from any browser. Below is a list of useful commands that you will need to know.

**URL To HortonWorks:** 127.0.0.1:8000

- **Copy jar:** `cp /media/<yoursharedfolder>/<your_jar> /usr/lib/hue`
- **Change permissions as root:** `chmod 777 /usr/lib/hue/<yourjar>`
- **Run your jar:** `hadoop jar <yourjar> <commandlinearguments>`

Before you run your program, you will need to create a directory for input on the HDFS in `/user/hue/`. Call it whatever you like. Then, using the GUI in the browser, upload all the input data you want into that directory.

1.3.1 Known Bugs

Remember to not have an output directory already existing when you try to run your program. Hadoop will create one for you and you will get an error if one already exists.

2 Hadoop Without Virtual Machine

This section will describe the method you need to use in order to run Hadoop without a virtual machine. First download Hadoop from Apache. I will be using version 2.7.1 for the following examples. Alternatively, you an use HomeBrew to install hadoop. If you do this, it will take care of adding hadoop to the PATH variables. I would highly suggest this if you can. Otherwise, in your hadoop folder add the `/bin/hadoop` to your path, as that is the main Hadoop program.

2.1 localhost

First you will need to be able to SSH into your localhost. To do this, make sure you have ssh installed. If you are windows distribution, I would suggest using Putty instead of the default cmd that comes with Windows. If you need to login with a password, then you will have to you create an authorized Key for your SSH command. This will be different for all distributions of operating systems, so I will let you look up how to do that. You will know you did it right if you can `ssh localhost` without the system asking for a password. You are now sshed into your localhost machine.
2.2 Adding Java and Hadoop to PATH

Now you need to add your Java Home and the Hadoop distribution to the settings folder in the hadoop folder you downloaded etc/hadoop/hadoop-env.sh.

Copy the following lines into that folder

```sh
export JAVA_HOME={your java home directory}
export HADOOP_PREFIX={your hadoop distribution directory}
```

2.3 Hadoop Configuration Files

You will now need to configure some of the files in the hadoop directory. These configurations are to modify what the default settings are to fit your local machine.

Listing 1: etc/hadoop/core-site.xml

```xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Listing 2: etc/hadoop/hdfs-site.xml

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

Listing 3: etc/hadoop/mapred-site.xml

```xml
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Listing 4: etc/hadoop/yarn-site.xml

```xml
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```
2.4 Formatting and Starting HDFS and Yarn

Now you need to format the HDFS portion. This only needs to be done once. To do this, enter the following command. Remember, the bin here is in reference to the Hadoop folder you downloaded. If you want to use the commands as is here, then you will need to change directory to the Hadoop download.

```
bin/hdfs namenode -format
```

Then start the name node
```
sbin/start-dfs.sh
```

If everything worked right, you can view your DFS from the browser now at NameNode. Now you need to create the user directory with a user. The username needs to be the same as the user of the machine you are currently on. I will use the following examples with my username as an example.

```
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/benmccamish
```

Now you can start up Yarn. You can start Yarn by `sbin/start-yarn.sh` Again you can figure out if it worked by going to This Address in your browser.

```
Start DFS: sbin/start-dfs.sh
Start Yarn: sbin/start-yarn.sh
Stop DFS: sbin/stop-dfs.sh
Stop Yarn: sbin/stop-yarn.sh
```

2.5 Running Hadoop Files

You don’t need to copy your jar anywhere in this scenario, since you are not on a virtual machine. You still need to copy the input directory though with the input files in it. To do this, you will need to execute the following commands.

```
Create input directory: hdfs dfs -mkdir <any directory name>
Upload a input file: hdfs dfs -put <file on local machine>
To Copy Output: hdfs dfs -get <output directory on hdfs>
To view it: hdfs dfs -cat <output directory on hdfs>/*
```
3 Java Build

First you will need to create a project. Create a java project like normal. Then you will want to go to Right Click Project > Built Path... > Add External Libraries. Then add the following libraries. The path starts in the hadoop main folder that you downloaded.

share/hadoop/common/hadoop-common-2.7.1.jar
share/hadoop/common/hadoop-nfs-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-app-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-hs-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.1.jar
share/hadoop/mapreduce/hadoop-mapreduce-client-shuffle-2.7.1.jar

3.1 Importing Libraries

When you are creating java source files for your hadoop project, be sure to include the proper libraries that you need at the top of your class. Eclipse should prompt you for any that you may be missing. Below I have included a list of libraries that I included in my examples.

Listing 5: Main

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

Listing 6: Mapper

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
```
3.2 Exporting Java

First make sure you build your program. Then you can export it as a Runnable Jar. Be sure when doing this you selected the proper main and included external jars in your export. Once you are done, the jar file will be where you defined and all you need to do is run it from Hadoop.

Listing 7: Reducer

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```