



---

# Hadoop Setup

---

Erick Skorupa Parolin

Erik Jonsson School of Engineering and Computer Science

---

# Learning Outcomes

*In this tutorial, we will cover the following steps:*

- Install Virtual Machine on your computers
- Java installation on VM
- Hadoop installation on VM
- Install & Configure IDE for running Hadoop job
- Run our first Hadoop code (WordCount)



THE UNIVERSITY OF TEXAS AT DALLAS

---

# Virtual Machine

---



Installation Guide

# Learning Outcomes

*Upon completion of this lesson, students will be able to:*

- Install Virtual Machine on their computers

Note: This step is required only for those who use Windows or other non-POSIX OS

# OS INFO

- Linux is well supported by almost all big data frameworks.
- Mac (Unix-like) is similar to Linux and should be fine for most frameworks.
- Windows is not well supported and may require extra effort to make it work.
- However, building your virtual machine with Linux as your OS is highly recommended for all of you.
- **Never try to run codes on windows in this course. It is not fully compatible and hard to debug!**

# Installation of VIRTUAL Machine

- Please always test your code on your machine before try to submit and run it on a server.
  - The Server might shut down due to unexpected accidents. Running code on your own machine alleviates the effort of debugging.
  - Make sure your machine meet following requirements:
    - >=8G RAM (GUI-supported Linux only)
    - >=4G RAM (for command-line Linux only)
    - >=16G RAM is highly recommended for running big data framework without significant delay

# Linux info

- In this course, we choose Ubuntu as desired Linux distribution.
- If you are familiar with Linux, CentOS is also a good choice.
- Linux info:
  - Ubuntu 20.04.2 LTS (my setup: 18.04.3 LTS)
  - <https://www.ubuntu.com/download/desktop>
  - Click above url and click on download to download Ubuntu disk image
- VirtualBox
  - <https://www.virtualbox.org/wiki/Downloads>
  - Click above url to download VirtualBox

# VirtualBox

- Click on url corresponding to your operating system
- We take mac os as example
- Double click on the downloaded file (e.g. [VirtualBox-6.1.18-142142-Win.exe](#)) to install



The screenshot shows the official VirtualBox download page. At the top right, there is a search bar, a 'Login' button, and a 'Preferences' link. The main title 'VirtualBox' is prominently displayed in large blue letters, with 'Download VirtualBox' below it. To the left of the main content area is a sidebar with links: 'About', 'Screenshots', 'Downloads', 'Documentation', 'End-user docs', 'Technical docs', 'Contribute', and 'Community'. The main content area contains text about the availability of binaries and source code, links to platform packages for various hosts (Windows, OS X, Linux, Solaris), and information about the GPL license and changelog. It also includes a note about SHA256 checksums and a note about upgrading guest additions.

VirtualBox  
Download VirtualBox

search...  
Login Preferences

About  
Screenshots  
Downloads  
Documentation  
End-user docs  
Technical docs  
Contribute  
Community

Here you will find links to VirtualBox binaries and its source code.

**VirtualBox binaries**

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 6.0 packages, see [VirtualBox 6.0 builds](#). Please also use version 6.0 if you need to run VMs with software virtualization, as this has been discontinued in 6.1. Version 6.0 will remain supported until July 2020.

If you're looking for the latest VirtualBox 5.2 packages, see [VirtualBox 5.2 builds](#). Please also use version 5.2 if you still need support for 32-bit hosts, as this has been discontinued in 6.0. Version 5.2 will remain supported until July 2020.

**VirtualBox 6.1.18 platform packages**

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)
- [Solaris 11 IPS hosts](#)

The binaries are released under the terms of the GPL version 2.

See the [changelog](#) for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- [SHA256 checksums, MD5 checksums](#)

**Note:** After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

**VirtualBox 6.1.18 Oracle VM VirtualBox Extension Pack**

- [All supported platforms](#)

# Installation

- Open VirtualBox



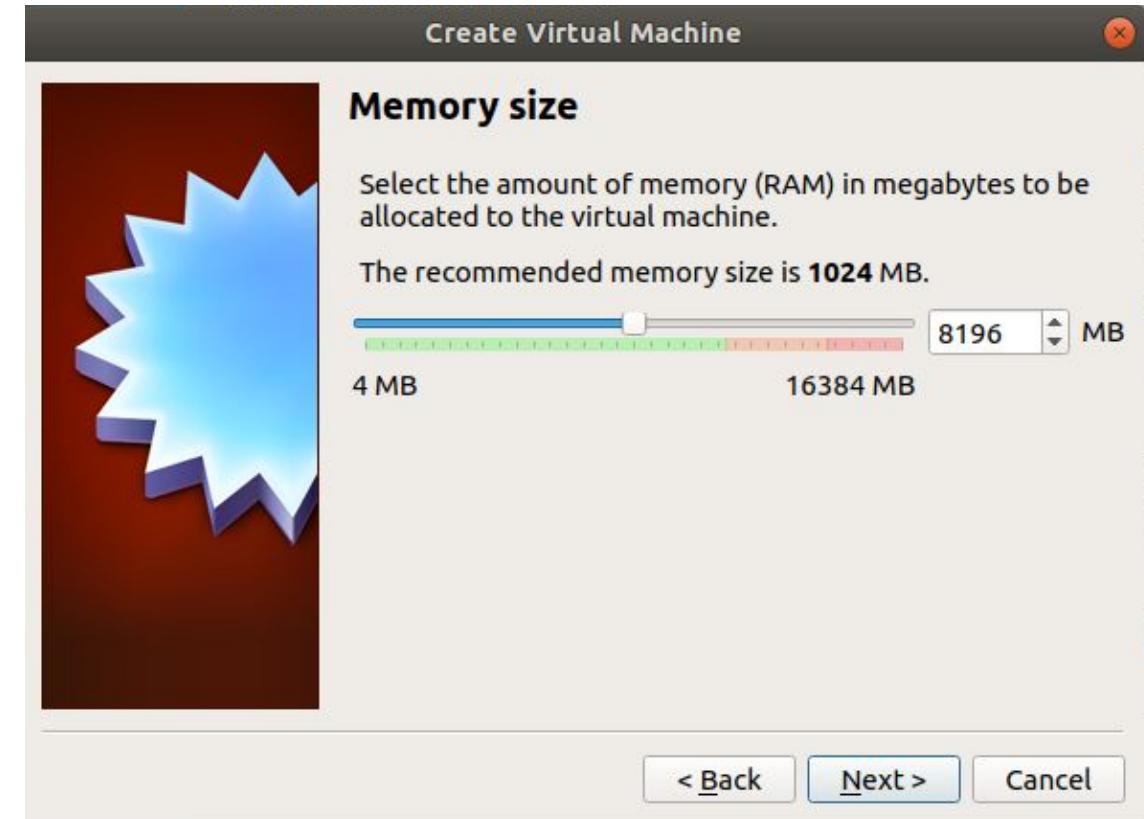
# Installation

- Click on 
- Enter Ubuntu
- Click on Continue



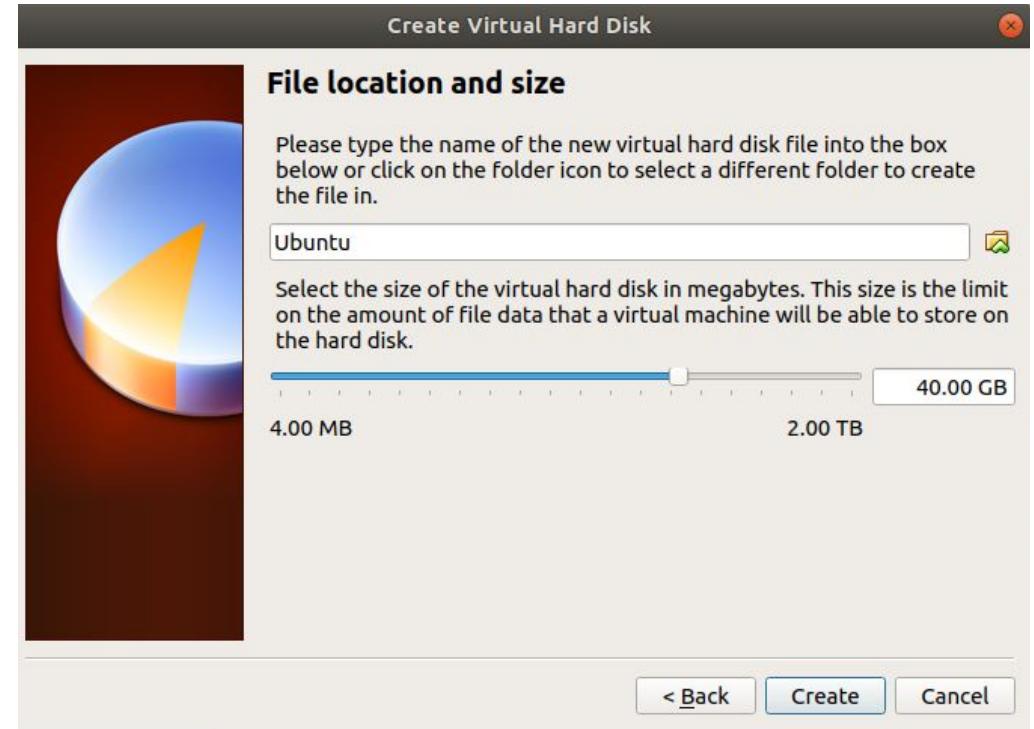
# Installation

- Set memory size to be either 4096 or 8192 MB
- Here we choose 8192 MB for better performance.
- Click on Next
- Click on **Create** to “Create a virtual hard disk now”
- Keep “VDI” and click on Next
- Keep “Dynamically allocated” and click Next



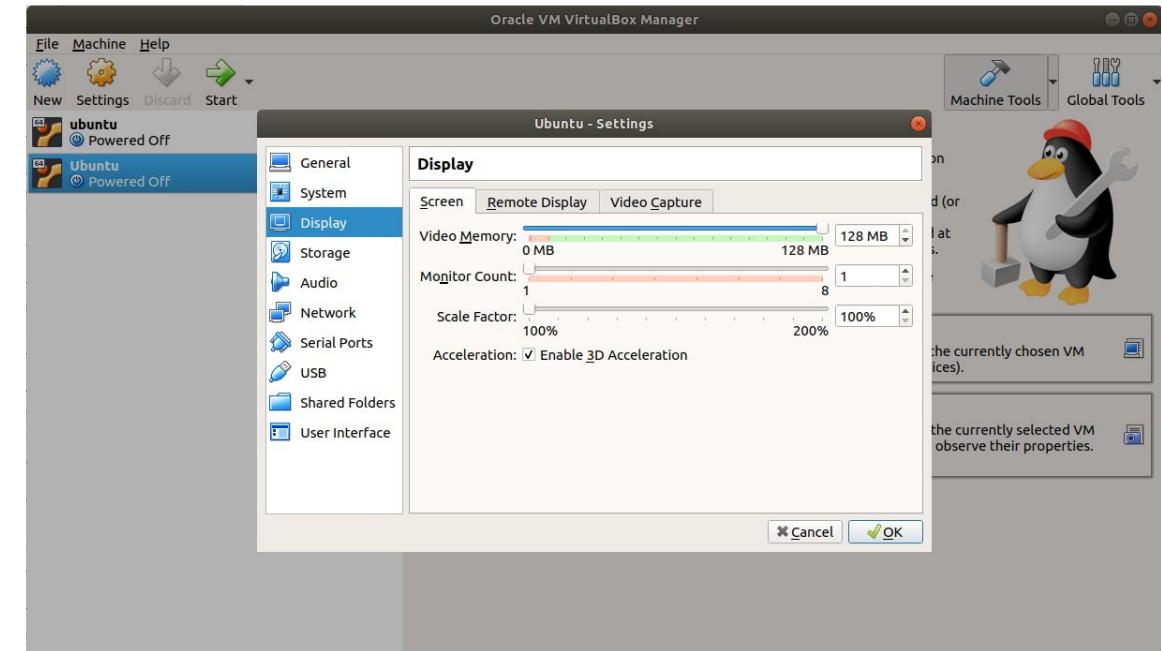
# Installation

- Setup disk size you prefer (**at least 40 GB**)
- Click on Create



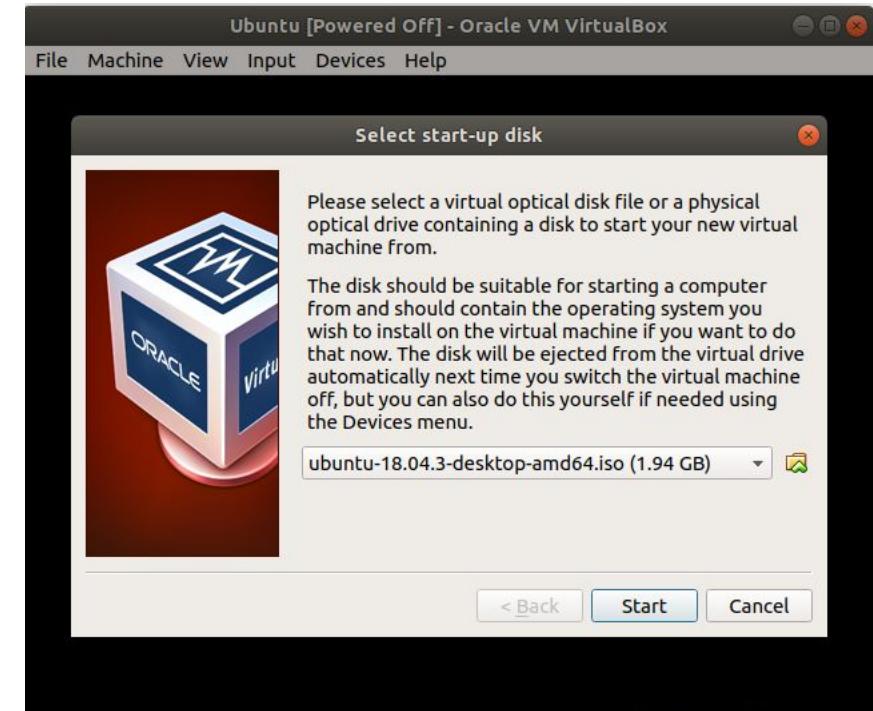
# Installation

- You will see the VM you just created in the VirtualBox Manager
- Select your VM, click on Settings and then Display
- Increase Video Memory to 128 MB
- Check Enable 3D Acceleration
- Click on OK
- Click on Start to start virtual machine



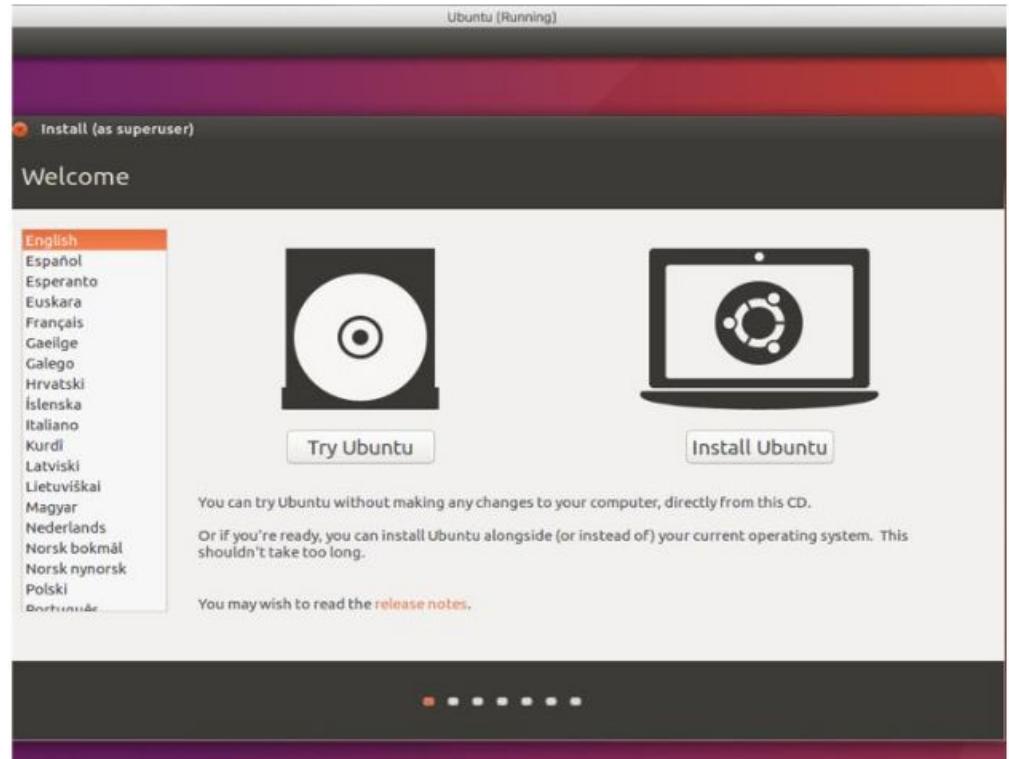
# Installation

- A window “Select start-up disk” is prompt out.
- Select the Ubuntu disk image you just download.
- Click on Start



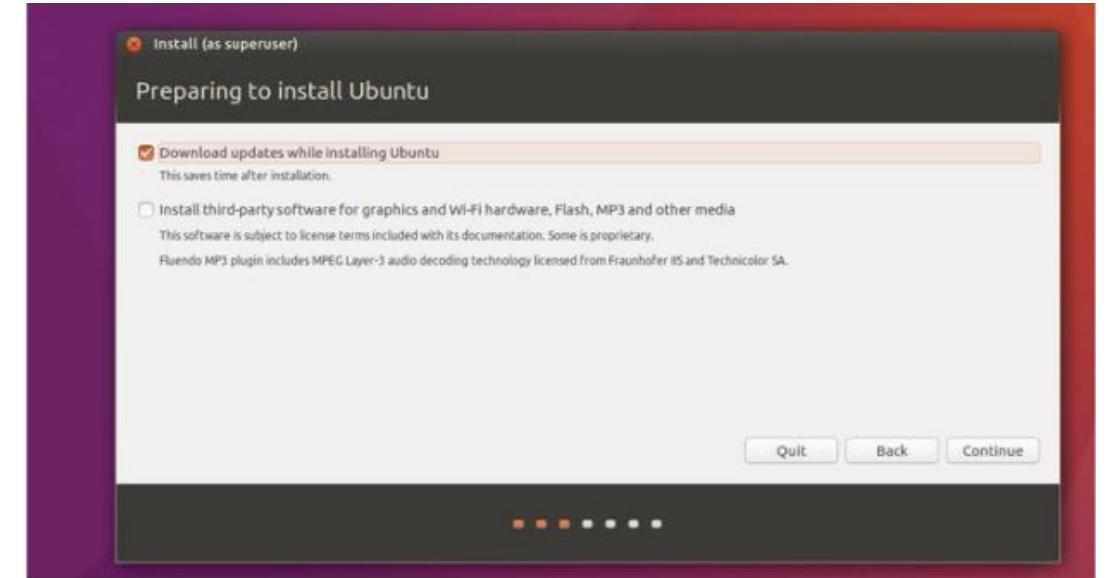
# Installation

- You are going to see a window like right image
- Click on install Ubuntu



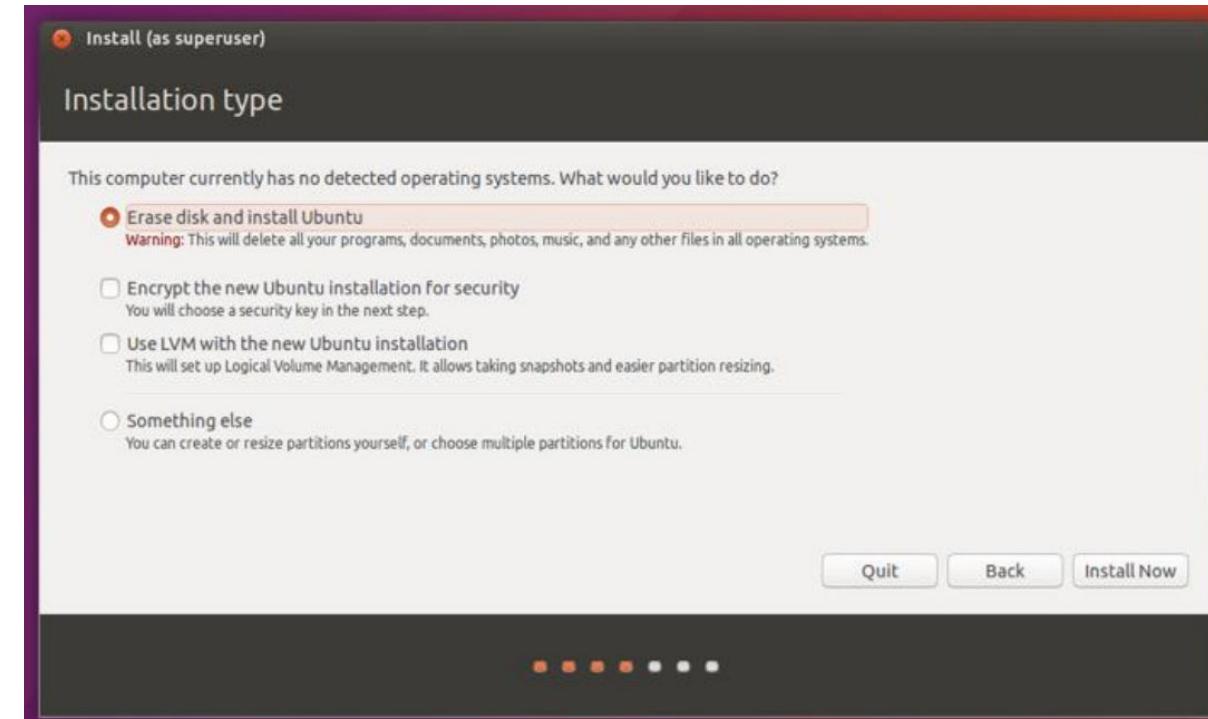
# Installation

- Check Download update while installing Ubuntu.
- You may also check the other one. It doesn't matter.
- From now, it is just installing Linux Ubuntu based on your own preferences...



# Installation

- For easy setup, click on “Erase disk and install Ubuntu” (**recommended**)
- You may also choose “Something else” to format disk by yourself.
- Here is my setting:
  - /boot 500 MB
  - /swap 2048 MB
  - / 40 GB
  - /home 60 GB



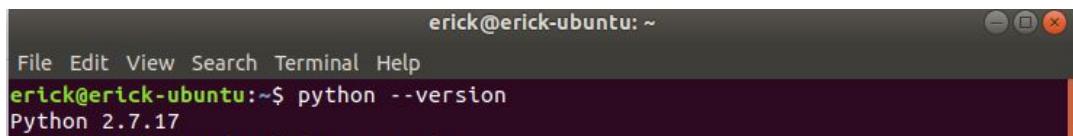
# Installation

- Click on continue
- Create your account and password (**remember it for future use**)
- Installation is going to be automatically executed.
- Wait until completes!

# Check python and java

- After Installation, restart the virtual machine
- Open terminal (Ctrl + Alt + T)
- Type following command to check your python and java version

```
$ python --version  
$ java -version
```



A screenshot of a terminal window titled "erick@erick-ubuntu: ~". The window shows the following text:  
File Edit View Search Terminal Help  
erick@erick-ubuntu:~\$ python --version  
Python 2.7.17



---

# Java Installation

---

Installation Guide

# Learning Outcomes

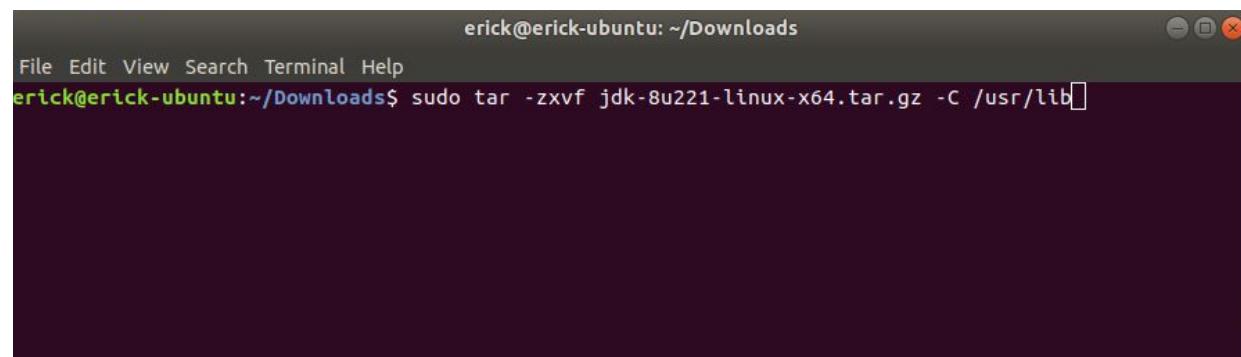
*Upon completion of this lesson, students will be able to:*

- Install JAVA

**Note:** Hadoop is built on Java, so this step is required to be done **before** installing Hadoop.

# JAVA Installation

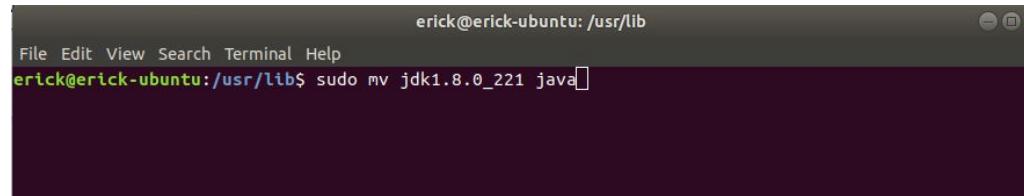
- Go to <http://www.oracle.com/technetwork/java/javase/downloads/index.html> to download Java SE Development Kit (my version is *jdk-8u221-linux-x64.tar.gz*).
- Remember to accept agreement and select the desired jdk file (e.g. *jdk-8u221-linux-x64.tar.gz*)
- Go to directory where you store the file and enter the command as shown below



The screenshot shows a terminal window titled "erick@erick-ubuntu: ~/Downloads". The window has a dark background and light-colored text. At the top, there's a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, the terminal prompt is "erick@erick-ubuntu:~/Downloads\$". A command is being typed into the terminal: "sudo tar -zvxf jdk-8u221-linux-x64.tar.gz -C /usr/lib". The cursor is positioned at the end of the command line.

# JAVA Installation

- Rename jdk folder following commands below



A screenshot of a terminal window titled "erick@erick-ubuntu: /usr/lib". The window has a dark theme with white text. The user is at the prompt "erick@erick-ubuntu: /usr/lib\$". They are typing the command "sudo mv jdk1.8.0\_221 java". The "java" part of the command is highlighted with a cursor.

- Install vim:

*\$ sudo apt-get install vim*

# JAVA Installation

- Enter the following command to setup the environment variables:

\$ vim *~/.profile*

- Enter All “**export**” commands as in the setup below at the tail of *.profile*
- Press ESC, then “:”
- Enter “wq”, press enter to save changes

```
File Edit View Search Terminal Help
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

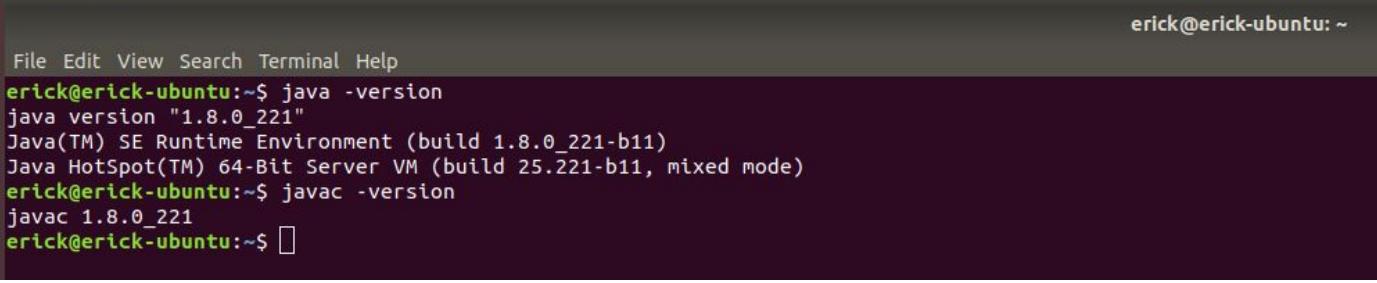
#PATH="$HOME/bin:$HOME/.local/bin:$PATH"
export JAVA_HOME=/usr/lib/java
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$CLASSPATH:.:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

# JAVA Installation

- Source *.profile*

```
$ source ~/.profile
```

- Verify your installation



A screenshot of a terminal window titled "erick@erick-ubuntu: ~". The window shows the following command-line session:

```
File Edit View Search Terminal Help
erick@erick-ubuntu:~$ java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
erick@erick-ubuntu:~$ javac -version
javac 1.8.0_221
erick@erick-ubuntu:~$
```

- Restart Ubuntu



---

# Hadoop Setup

---

Installation Guide

# Learning Outcomes

*Upon completion of this lesson, students will be able to:*

- Install Hadoop
- Setup all configurations to run pseudo-distributed environment

# Pseudo-distributed Cluster

- **Pseudo-Distributed Operation:** Hadoop can also be run on a single-node in a **pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.**
- This document describes how to setup and configure a single-node Hadoop installation so that you can quickly perform simple operations using Hadoop MapReduce and the Hadoop Distributed File System (HDFS).
- Developing and testing your Hadoop program on your own machine before submitting to Hadoop cluster is **highly recommended**.
- Refer to <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html> for installation documentation

# Hadoop Setup

- Install **ssh** and **pdsh**:

```
$ sudo apt-get install ssh  
$ sudo apt-get install pdsh
```

- Download Hadoop (my version is hadoop-2.8.1.tar.gz):

<https://archive.apache.org/dist/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.tar.gz>

- Unzip this tar.gz file through command:

```
$ tar -xvzf hadoop-2.8.1.tar.gz -C ~/Documents/
```

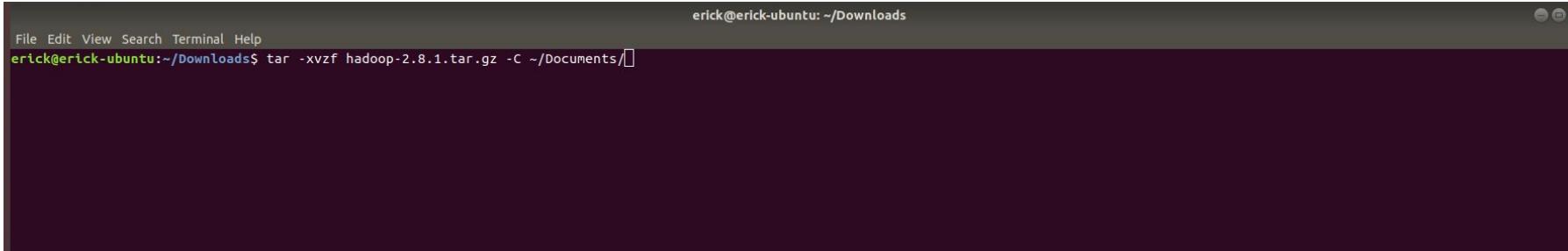
- Go to installation directory and rename the folder to hadoop

```
$ mv ~/Documents/hadoop-2.8.1/ ~/Documents/hadoop
```

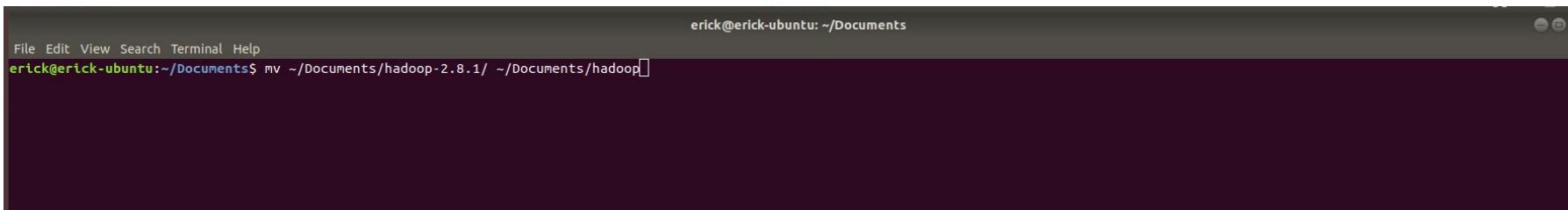
## Hadoop Installation

---

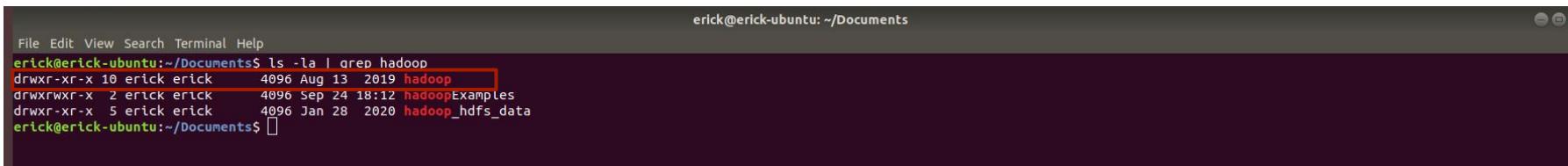
# Hadoop Setup



```
erick@erick-ubuntu:~/Downloads$ tar -xvf hadoop-2.8.1.tar.gz -C ~/Documents/
```



```
erick@erick-ubuntu:~/Documents$ mv ~/Documents/hadoop-2.8.1/ ~/Documents/hadoop
```



```
erick@erick-ubuntu:~/Documents$ ls -la | grep hadoop
drwxr-xr-x 10 erick erick 4096 Aug 13 2019 hadoop
drwxrwxr-x  2 erick erick 4096 Sep 24 18:12 hadoopExamples
drwxr-xr-x  5 erick erick 4096 Jan 28 2020 hadoop_hdfs_data
erick@erick-ubuntu:~/Documents$
```

# Hadoop Installation

- Enter the following command to setup the environment variables:

```
$ vim ~/.profile
```

- Add the “export” line marked by red line at the bottom of the file
- Press ESC, then “:”
- Enter “wq”, press enter to save changes
- Source *.profile*

```
$ source ~/.profile
```

```
File Edit View Search Terminal Help
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

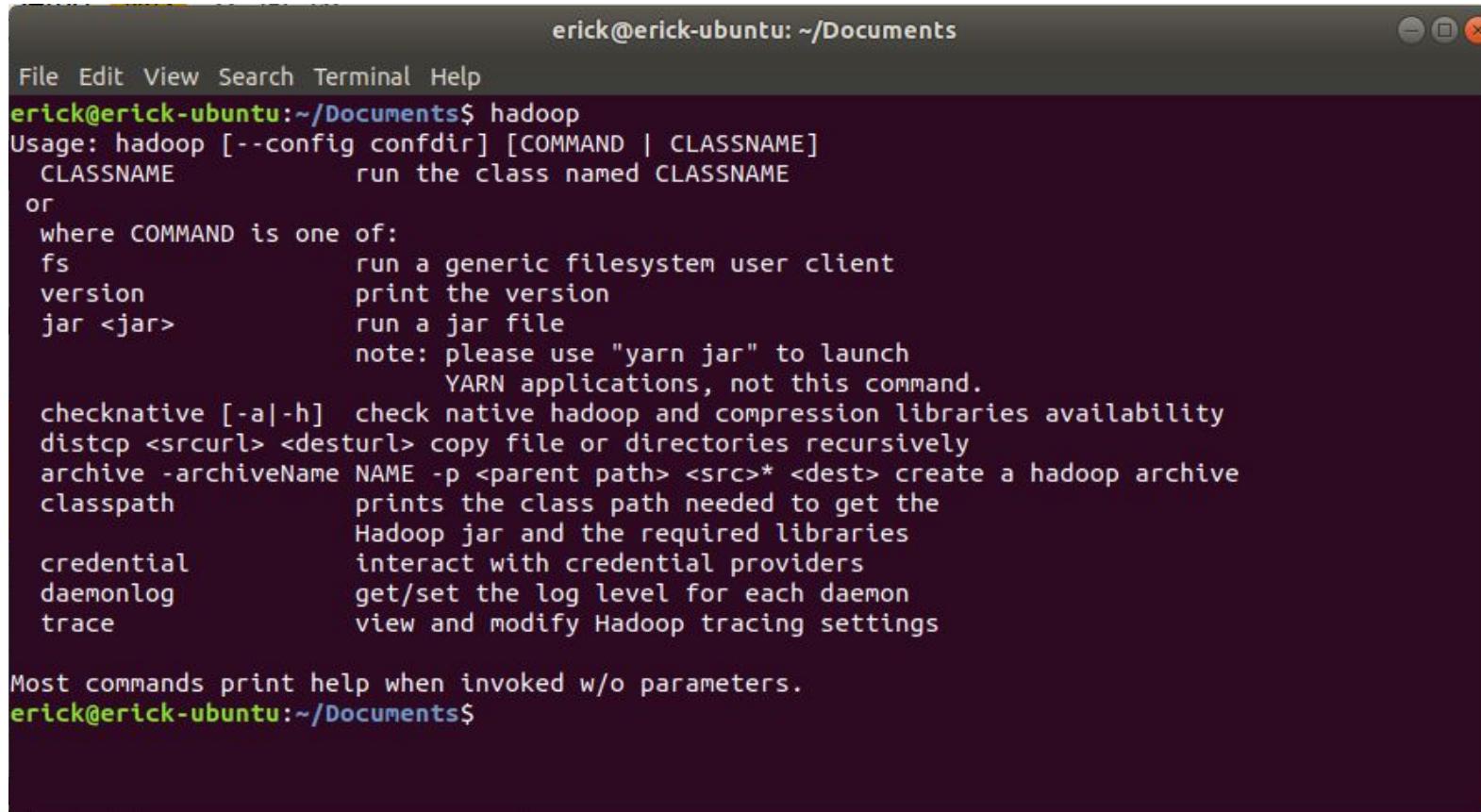
# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

#PATH="$HOME/bin:$HOME/.local/bin:$PATH"
export JAVA_HOME=/usr/lib/java
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$CLASSPATH:.:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
export PATH=$PATH:~/Documents/hadoop/bin:~/Documents/hadoop/sbin
```

# Check setup

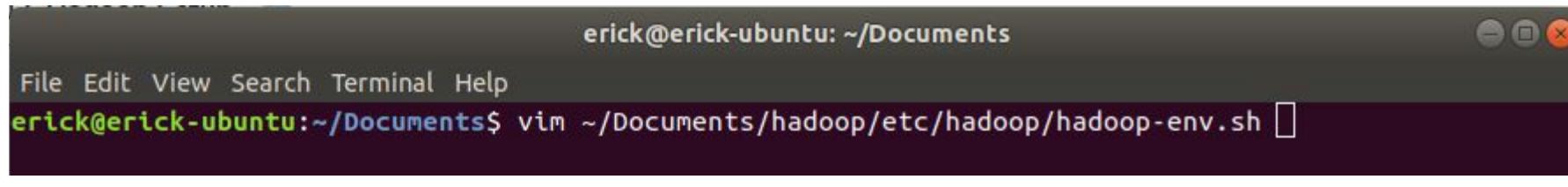


The screenshot shows a terminal window titled "erick@erick-ubuntu: ~/Documents". The window contains the Hadoop command-line usage information. The text is as follows:

```
erick@erick-ubuntu:~/Documents$ hadoop
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]
  CLASSNAME          run the class named CLASSNAME
or
  where COMMAND is one of:
    fs                  run a generic filesystem user client
    version             print the version
    jar <jar>            run a jar file
                          note: please use "yarn jar" to launch
                          YARN applications, not this command.
    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
    credential          interact with credential providers
    daemonlog           get/set the log level for each daemon
    trace               view and modify Hadoop tracing settings

  Most commands print help when invoked w/o parameters.
erick@erick-ubuntu:~/Documents$
```

# Change JAVA\_HOME in hadoop-env.sh



A screenshot of a vim editor window titled "erick@erick-ubuntu: ~/Documents". The file "hadoop-env.sh" is open. The line "export JAVA\_HOME=/usr/lib/java" is highlighted with a red rectangle. The rest of the file contains standard Hadoop configuration comments and environment variable definitions.

```
#  
#      http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
  
# Set Hadoop-specific environment variables here.  
  
# The only required environment variable is JAVA_HOME. All others are  
# optional. When running a distributed configuration it is best to  
# set JAVA_HOME in this file, so that it is correctly defined on  
# remote nodes.  
  
# The java implementation to use.  
export JAVA_HOME=/usr/lib/java  
  
# The jsvc implementation to use. Jsvc is required to run secure datanodes  
# and secondary namenodes. It is also required to run the Java process  
# which runs the metrics collector.
```

Same as we did before:

- Press ESC, then ":"
- Enter "wq", press enter to save changes

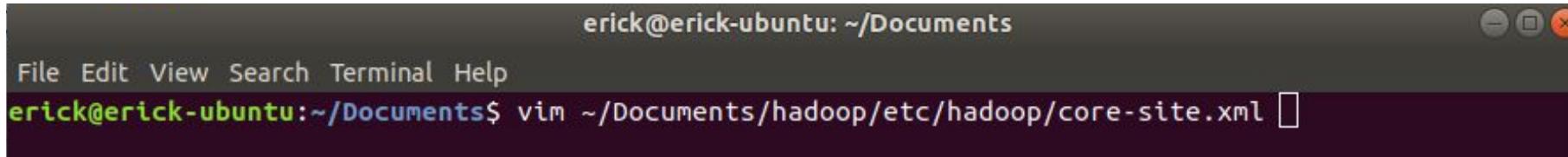
# Pseudo-distributed mode

- Go to Hadoop folder, open **core-site.xml** and add the following configuration:

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

- Enter the content above in **core-site.xml**, save and exit (see next slide).

# Pseudo-distributed mode



A screenshot of a vim editor window titled "erick@erick-ubuntu: ~/Documents". The file "core-site.xml" is open. The code shown is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

**Same old to save core-site.xml:**  
press ESC, press “：“, enter “wq”,  
and press enter to save changes

# Pseudo-distributed mode

- Now, we need to create a directory in the local environment for hosting datanodes and namenodes (meta)data:

\$ *mkdir <directory for hosting metadata for hadoop>*

\*In my case: \$ ***mkdir /home/erick/Documents/hadoop\_hdfs\_data***

- Keep this path!! We will use it in the next step...

# Pseudo-distributed mode

- Go to Hadoop folder, open **hdfs-site.xml** and add the following configuration:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/data</value>
  </property>
  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/namesecondary</value>
  </property>
</configuration>
```

Input here the path you defined as directory to host hadoop metadata in previous step.

# Pseudo-distributed mode

```
erick@erick-ubuntu: ~/Documents
File Edit View Search Terminal Help
erick@erick-ubuntu:~/Documents$ vim ~/Documents/hadoop/etc/hadoop/hdfs-site.xml
```

```
erick@erick-ubuntu: ~/Documents
File Edit View Search Terminal Help
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/name</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/data</value>
  </property>

  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>/home/erick/Documents/hadoop_hdfs_data/namesecondary</value>
  </property>
</configuration>
```

**Same old to save hdfs-site.xml:**  
press ESC, press “:”, enter “wq”, and press enter to save changes

# Pseudo-distributed mode

- ssh to localhost with the following command:

```
$ ssh localhost
```

- Format the namenode with the following command:

```
$ hdfs namenode -format
```

- Start hadoop deamons with the following command(s):

```
$ start-all.sh
```

or

```
$ start-dfs.sh  
$ start-yarn.sh
```

# Pseudo-distributed mode

**start-dfs.sh** command will start NameNode and DataNode deamons.  
Note: use command jps to check which deamons are running.

**start-yarn.sh** command will start NodeManager and ResourceManager deamons.  
Again: use command jps to check which deamons are running.

```
erick@erick-ubuntu:~/Documents$ start-dfs.sh
Starting namenodes on [localhost]
erick@localhost's password:
localhost: starting namenode, logging to /home/erick/Documents/hadoop/logs/hadoop-erick-namenode-erick-ubuntu.out
erick@localhost's password:
localhost: starting datanode, logging to /home/erick/Documents/hadoop/logs/hadoop-erick-datanode-erick-ubuntu.out
Starting secondary namenodes [0.0.0.0]
erick@0.0.0.0's password:
0.0.0.0: starting secondarynamenode, logging to /home/erick/Documents/hadoop/logs/hadoop-erick-secondarynamenode-erick-ubuntu.out
erick@erick-ubuntu:~/Documents$ jps
5604 SecondaryNameNode
5718 Jps
5177 NameNode
5340 DataNode

erick@erick-ubuntu:~/Documents$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/erick/Documents/hadoop/logs/yarn-erick-resourcemanager-erick-ubuntu.out
erick@localhost's password:
localhost: starting nodemanager, logging to /home/erick/Documents/hadoop/logs/yarn-erick-nodemanager-erick-ubuntu.out
erick@erick-ubuntu:~/Documents$ jps
6114 NodeManager
5604 SecondaryNameNode
5177 NameNode
6155 Jps
5340 DataNode
5774 ResourceManager
erick@erick-ubuntu:~/Documents$ 
```

**Make sure all deamons** NameNode, DataNode, NodeManager and ResourceManager are running.

# Pseudo-distributed mode

- Browse the web interface for the NameNode:
  - by default it is available at: <http://localhost:50070>  
(or <http://localhost:9870> in the case you are using *Hadoop 3.0.0 or above*).

The screenshot shows a web browser window with the URL `localhost:50070/dfshealth.html#tab-overview`. The page has a green header bar with tabs for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Overview tab is active. Below the tabs, there's a section titled "Overview 'localhost:9000' (active)". This section contains several status boxes:

Started:	Sun Feb 07 20:19:34 -0600 2021
Version:	2.8.1, r20fe530490fcfcf5a18053c389e43cd26f7a70fe
Compiled:	Fri Jun 02 01:14:00 -0500 2017 by vinodkv from branch-2.8.1-private
Cluster ID:	CID-e2e27460-3b26-4390-b0b1-936d74484a57
Block Pool ID:	BP-195117247-127.0.1.1-1980231746472

Below this is a "Summary" section with the following information:

Security is off.  
Safemode is off.

42 files and directories, 23 blocks = 65 total filesystem object(s).  
Heap Memory used 40.27 MB of 121.88 MB Heap Memory. Max Heap Memory is 966.69 MB.  
Non Heap Memory used 44.08 MB of 44.91 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	39.12 GB
DFS Used:	80.45 MB (0.2%)
Non DFS Used:	27.29 GB
DFS Remaining:	9.74 GB (24.89%)
Block Pool Used:	80.45 MB (0.2%)
DataNodes usages% (Min/Median/Max/stdDev):	0.20% / 0.20% / 0.20% / 0.00%

# Pseudo-distributed mode

- To stop your hadoop system, you can execute the following command(s):

\$ stop-all.sh

or

\$ stop-dfs.sh  
\$ stop-yarn.sh

# WordCount Example

---



Designing our first Hadoop code

# Learning Outcomes

*Upon completion of this lesson, students will be able to:*

- Design and code our first Hadoop job

# Pseudo-code

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
        emit(w, "1");

reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    key = output_key
    int result = 0;
    for each v in intermediate_values:
        result = result + ParseInt(v);
    emit(key, String(result));
```

# Map code in WordCount.java

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}
```

- **map** class implements a public map method, that processes one line at a time from input data;
- Splits each line into **tokens separated by whitespaces**.
- It **emits a key-value pair** of <<Word>, 1>, written to the Context.
- **Note:** Context object allows the Mapper to interact with the rest of the Hadoop system.

# Reduce code in WordCount.java

```
public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {  
    private IntWritable result = new IntWritable();  
  
    public void reduce(Text key, Iterable<IntWritable> values,  
                      Context context) throws IOException, InterruptedException {  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
        result.set(sum);  
        context.write(key, result);  
    }  
}
```

- The intermediate key-value pair gets grouped by based on the key and each group comes at a time to a reducer
- Reducer.reduce( ) is called once per key
- Reducer.reduce( ) receives an **Iterable** which returns all values associated with that key and here we sum all those values.
- Finally, emits output with through Context.write() with all the results summarized/aggregated for each key.

# Driver setup in WordCount.java

```
// Usage: wordcount <in> <out>
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);

    job.setMapperClass(TokenizerMapper.class);

    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

# Running Hadoop Program

---



Running WordCount Job

# Learning Outcomes

*Upon completion of this lesson, students will be able to:*

- Install and setup IDE
  - Eclipse
  - IntelliJ
- Create a Hadoop program
- Run a simple MapReduce Program

# Introduction

- So far from previous tutorials, we have a pseudo-distributed cluster running on our machine and learned how to write our first map-reduce program
- Next Questions is
  - How to get the jar of the map-reduce program?
  - How to run the jar file in Hadoop?

# The Process

- We assume that you already have Hadoop on your own machine.
- In the following, we will discuss steps in details

# Preparing IDE

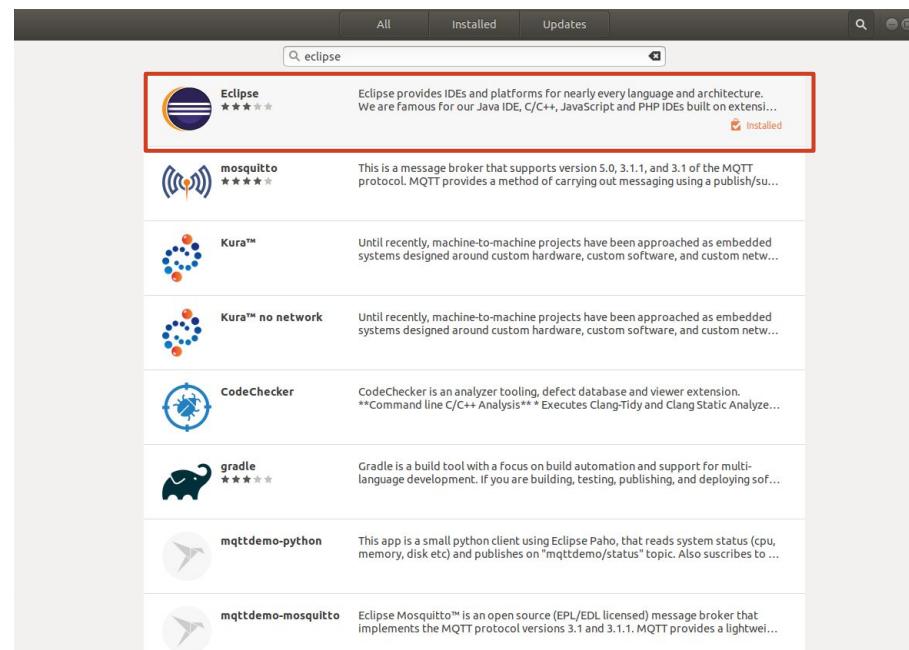
- Hadoop programs are Java programs. You may use any Java IDE such as Eclipse, NetBeans, IntelliJ IDEA to develop your Map-Reduce program.
- We are going to use Eclipse/IntelliJ in this tutorial.
- If you have Eclipse/IntelliJ on your own machine, you can skip this section.

# Install Eclipse

- To install Eclipse, run this command in shell

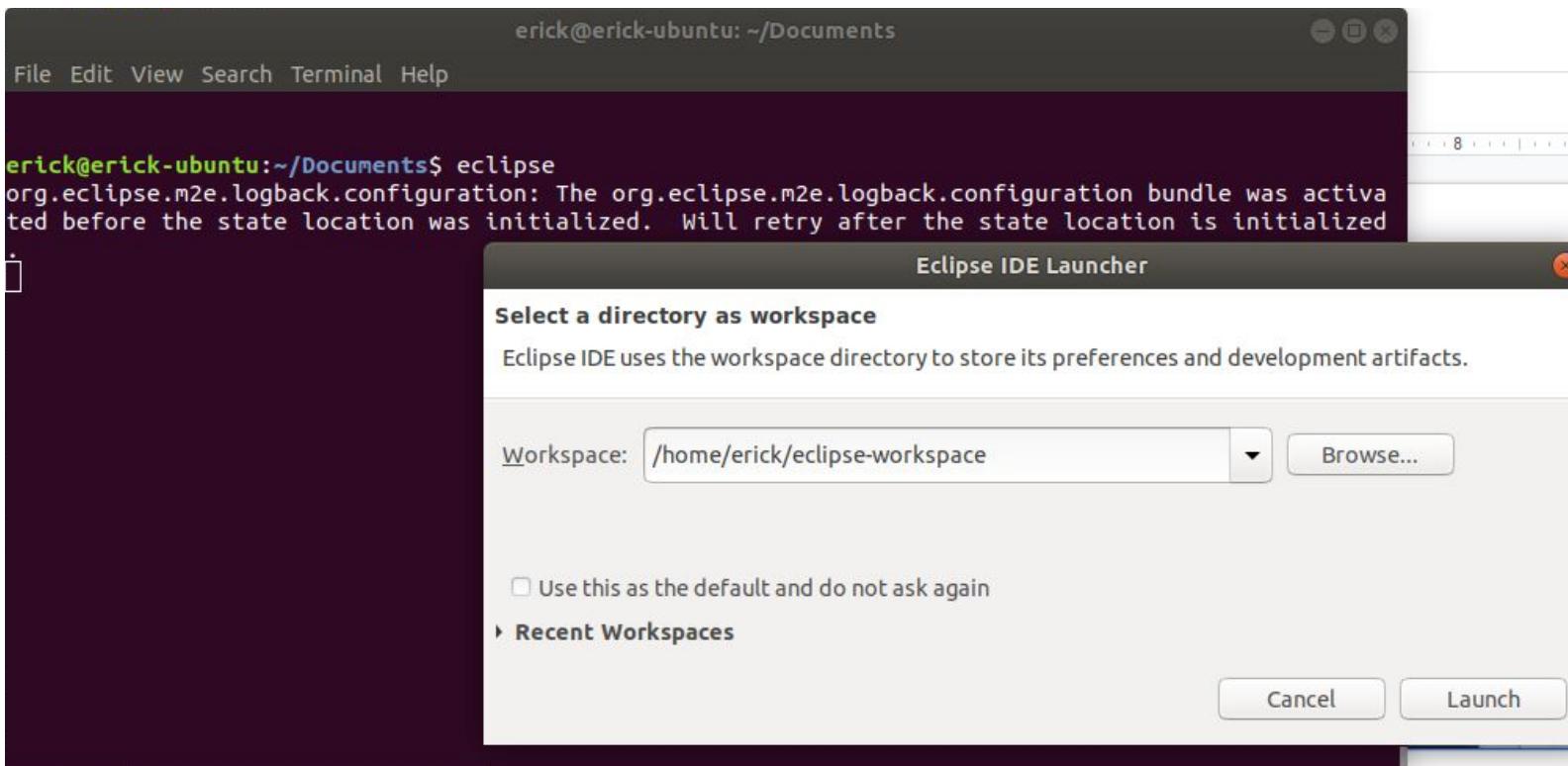
```
$ sudo apt-get install eclipse
```

... or simply use the Ubuntu Software  for that:



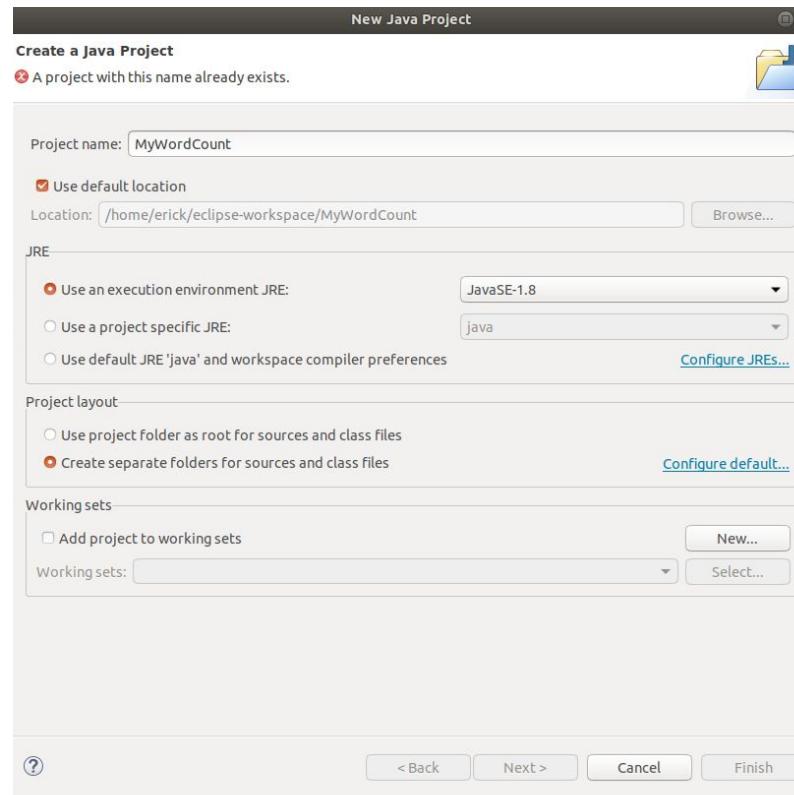
# Start Eclipse

Run command “\$ eclipse” to start Eclipse:



# Create Project in Eclipse

File >> New >> Java Project to create your new project...



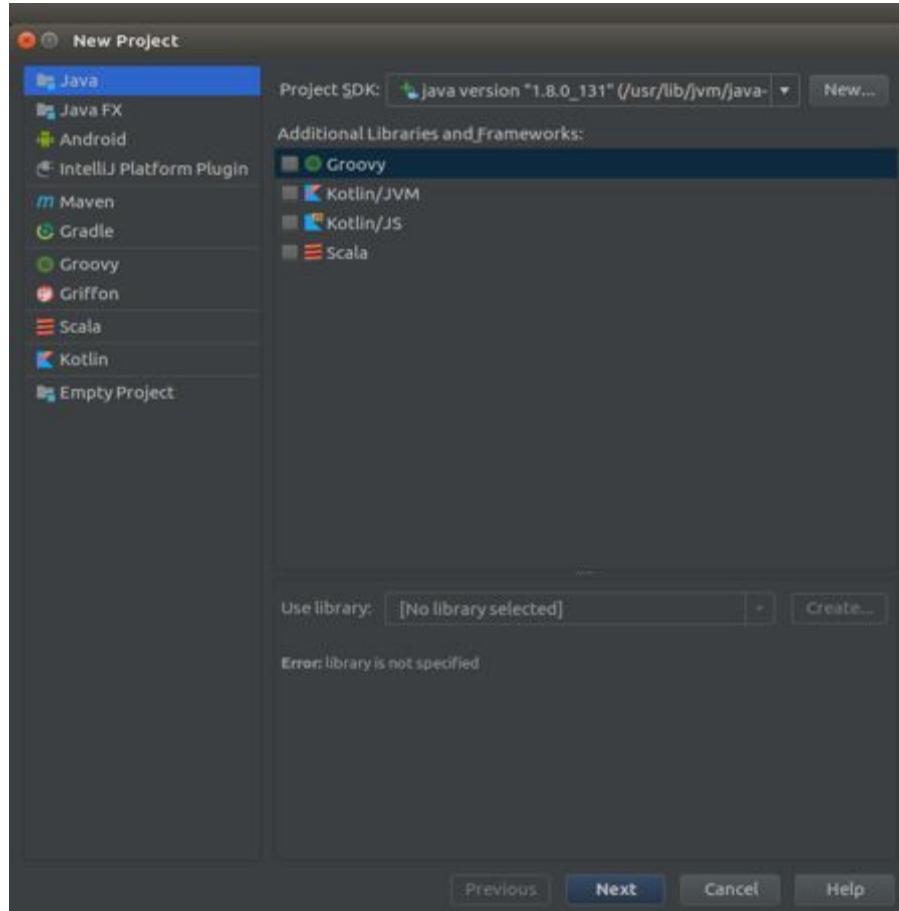
# Install IntelliJ

- Download intelliJ community version  
<https://www.jetbrains.com/idea/download/#section=linux>
- Double click on the downloaded tar.gz file, unzip it to a folder
- Go to that folder then go to the bin folder (e.g. idea-IC-172.4155.36/bin)
- run the command “./idea.sh” in shell
- Remember to install scala plugin if you use IntelliJ
- For IntelliJ, you can also use Ubuntu Software 

# Start IntelliJ



# Create Project in IntelliJ



Click on next until done...

# Create your JAVA Class

- Create a new java class
- Copy all the code in **WordCount.java** into this file

# Create your JAVA Class

- Create a new java class
- Copy all the code in the attached **WordCount.java** into this new class file

```
public static class Map
    extends Mapper<LongWritable, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text(); // type of output key

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] mydata = value.toString().split(" ");
        for (String data : mydata) {
            word.set(data); // set word as each input keyword
            context.write(word, one); // create a pair <keyword, 1>
        }
    }
}
```

# Create your JAVA Class

```
public static class Reduce
    extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int sum = 0; // initialize the sum for each keyword
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result); // create a pair <keyword, number of occurrences>
    }
}

// Driver program
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    // get all args
    if (otherArgs.length != 2) {
        System.err.println("Usage: WordCount <in> <out>");
        System.exit(2);
    }

    // create a job with name "wordcount"
    Job job = new Job(conf, "wordcount");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    // uncomment the following line to add the Combiner job.setCombinerClass(Reduce.class);

    // set output key type
    job.setOutputKeyClass(Text.class);
    // set output value type
    job.setOutputValueClass(IntWritable.class);
    //set the HDFS path of the input data
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    // set the HDFS path for the output
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    //Wait till job completion
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

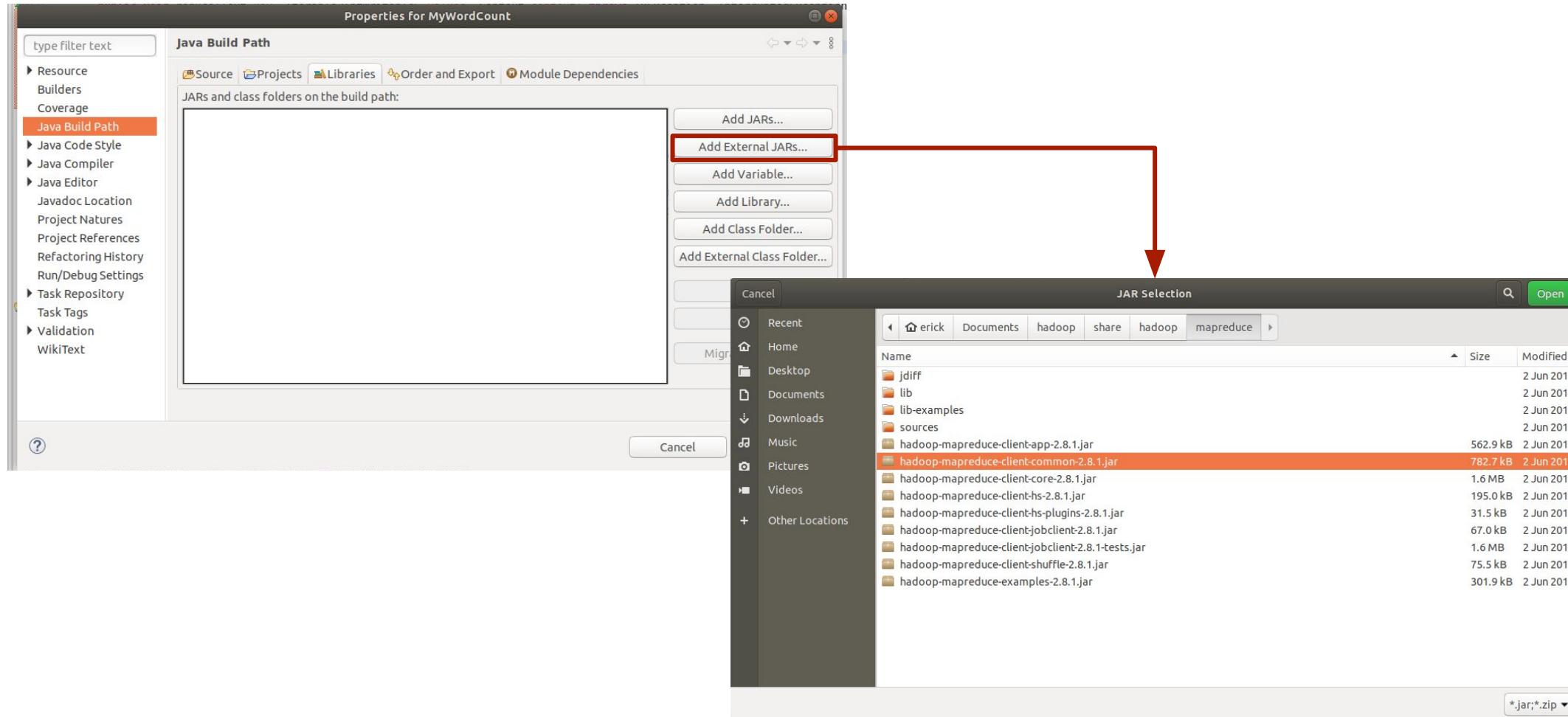
# Adding Hadoop libraries

In order to compile Hadoop projects, you need to add Hadoop libraries as a reference to your projects...

# Adding Hadoop libraries in Eclipse

- Right-click on the project:
  - Select “**Build Path**” >> “**Configure Build Path...**”
  - Select “**Libraries**” tab.
  - Click on “**Add External JARs...**” to continue.
- Go to your Hadoop installation path (my case is ~/Documents).
- Find “**hadoop-mapreduce-client-core-2.8.1.jar**” in /share/hadoop/mapreduce folder, and add it.
- Click on “Add External JARs...” again.
- Find “**hadoop-common-2.8.1.jar**” in /share/hadoop/common folder, and add it.
- You need also add “**commons-cli-1.2.jar**” in the folder /share/hadoop/common/lib folder.

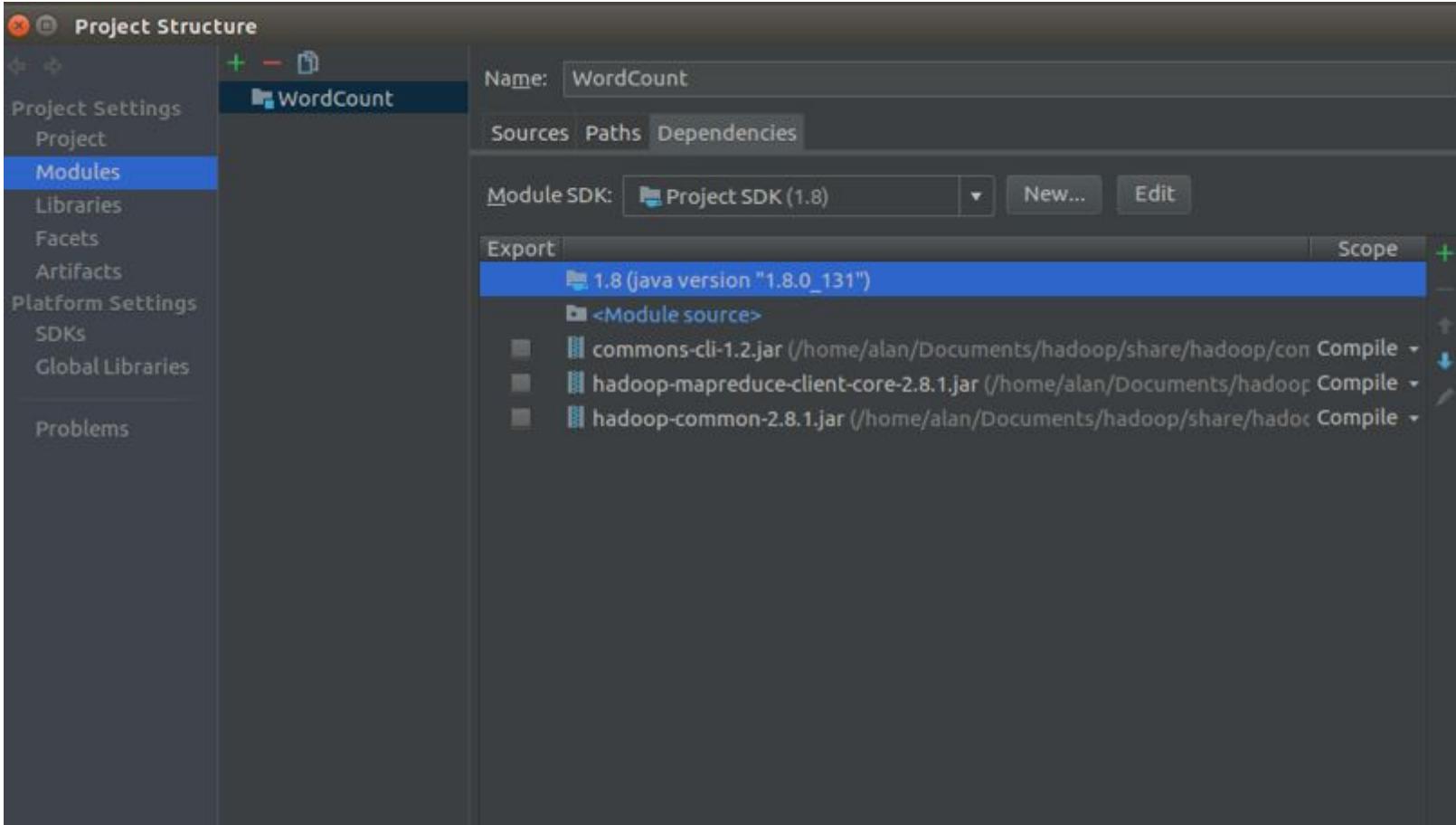
# Adding Hadoop libraries in Eclipse



# Adding Hadoop libraries in IntelliJ

- Click on File -> Project Structure -> Module
- On the larger right pane, choose the Dependencies tab
- Click on “+” and choose ”Jars or directories”
- Add all those jars mentioned above
- Click on OK to save and exit

# Adding Hadoop libraries in IntelliJ

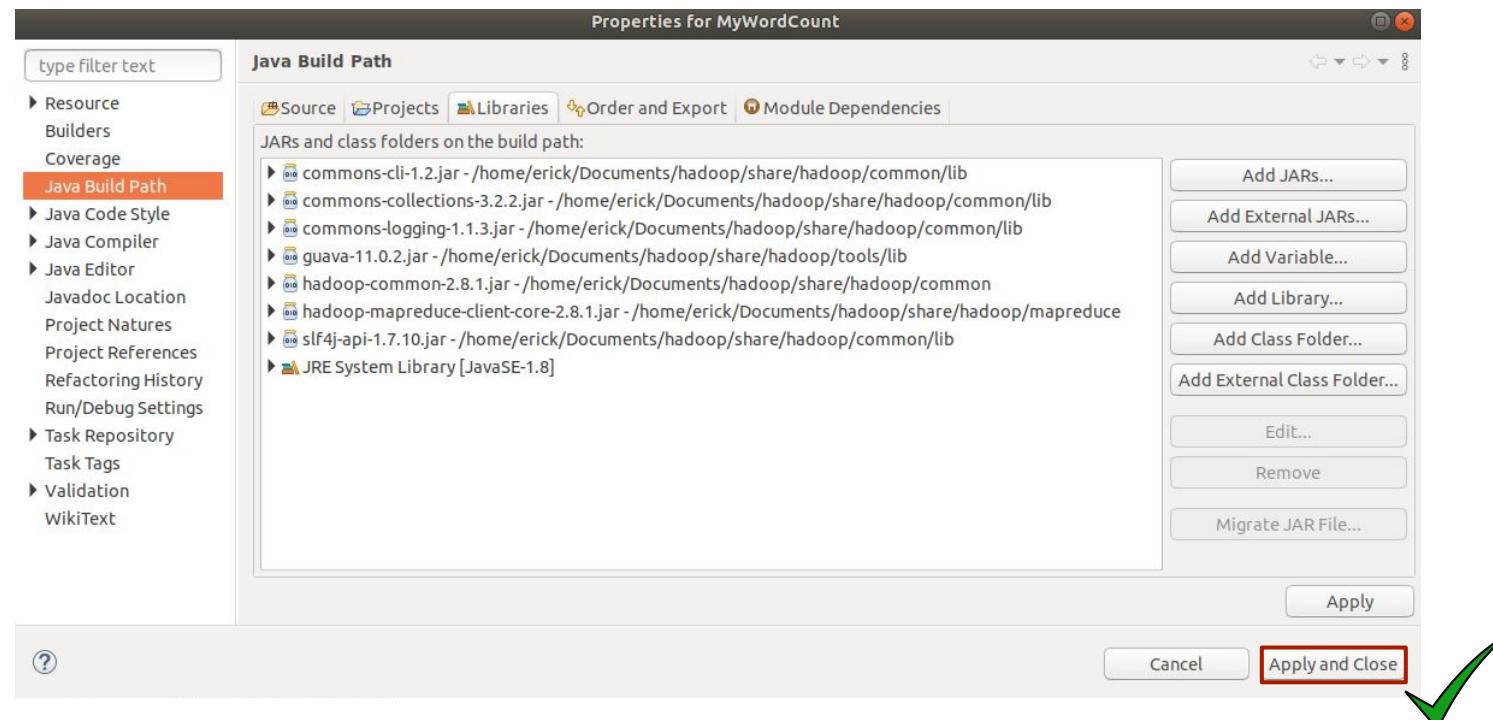


## Additional Hadoop libraries in both (Eclipse and IntelliJ)

- Add all the JARS below into build path (both Eclipse and IntelliJ)
  - **commons-collections-3.2.2.jar**
  - **slf4j-api-1.7.25.jar (or slf4j-api-1.7.10.jar)**
  - **commons-logging-1.1.3.jar** at /share/hadoop/common/lib
  - **guava-11.0.2.jar** at /share/hadoop/tools/lib

# Additional Hadoop libraries in both (Eclipse and IntelliJ)

- Your final build path configuration should be similar to this now:



- Note: the version of the libraries may vary depending on your hadoop version

## Creating the JAR file for Hadoop

- Now, all you need to do now is to create the JAR file and run it in Hadoop...

## Creating the JAR file for Hadoop in Eclipse

- Right click on the project, and choose “**Export...**”
- Then use choose “**JAR file**” under “**Java**” and click **Next.**
- Specify the name of the export file:
  - For example, you may use: “~/Documents/hadoopExamples/MyWordCount.jar”
  - Click and Finish

## Creating the JAR file for Hadoop in IntelliJ

- File -> Project Structure -> Artifacts -> “+” -> Jar -> From modules with dependencies
- Then select main class and click on OK.
- Next, go to Build >> Build Artifacts >> Build. It will be in out/artifacts folder

## Data manipulation in HDFS

- Before running our JAR file in Hadoop, we need to make sure our input file is available in HDFS system
- Remember: HDFS is build on the top of Linux!
  - Therefore, the commands you run in HDFS will be similar to those that you would run in usual shell script in your Ubuntu
- Next you see some couple of commands that you will run on HDFS to handle data and the directory file structure...

## Data manipulation in HDFS

- To create a directory:

*\$ hdfs dfs -mkdir <directory name you want to create>*

- To list/display a directory:

*\$ hdfs dfs -ls <directory name you want to look inside>*

- To remove a directory or a file:

*\$ hdfs dfs -rm -r <directory name you want to delete/remove>*

*\$ hdfs dfs -rm <file name you want to delete/remove>*

# Data manipulation in HDFS

```
erick@erick-ubuntu:~/Documents$ hdfs dfs -mkdir /user
erick@erick-ubuntu:~/Documents$ hdfs dfs -mkdir /user/erick
erick@erick-ubuntu:~/Documents$ hdfs dfs -ls /user
Found 1 items
drwxr-xr-x - erick supergroup 0 2021-02-08 09:11 /user/erick
erick@erick-ubuntu:~/Documents$ hdfs dfs -ls /
Found 11 items
drwxr-xr-x - erick supergroup 0 2020-09-24 17:39 /input
drwxr-xr-x - erick supergroup 0 2020-02-19 18:31 /input_old
drwxr-xr-x - erick supergroup 0 2020-08-27 20:21 /outputCountWords
drwxr-xr-x - erick supergroup 0 2020-01-29 17:51 /outputFilter
drwxr-xr-x - erick supergroup 0 2020-01-29 18:58 /outputFilter1
drwxr-xr-x - erick supergroup 0 2020-01-29 19:27 /outputFilterByDate
drwxr-xr-x - erick supergroup 0 2020-09-24 18:19 /outputInMemory
drwxr-xr-x - erick supergroup 0 2020-09-24 18:15 /outputReduceSide
drwxr-xr-x - erick supergroup 0 2020-09-24 16:31 /outputRelativeOccurrence
drwxr-xr-x - erick supergroup 0 2020-02-19 18:32 /outputYelp
drwxr-xr-x - erick supergroup 0 2021-02-08 09:11 /user
erick@erick-ubuntu:~/Documents$ hdfs dfs -rm -r /user
Deleted /user
erick@erick-ubuntu:~/Documents$ hdfs dfs -ls /
Found 10 items
drwxr-xr-x - erick supergroup 0 2020-09-24 17:39 /input
drwxr-xr-x - erick supergroup 0 2020-02-19 18:31 /input_old
drwxr-xr-x - erick supergroup 0 2020-08-27 20:21 /outputCountWords
drwxr-xr-x - erick supergroup 0 2020-01-29 17:51 /outputFilter
drwxr-xr-x - erick supergroup 0 2020-01-29 18:58 /outputFilter1
drwxr-xr-x - erick supergroup 0 2020-01-29 19:27 /outputFilterByDate
drwxr-xr-x - erick supergroup 0 2020-09-24 18:19 /outputInMemory
drwxr-xr-x - erick supergroup 0 2020-09-24 18:15 /outputReduceSide
drwxr-xr-x - erick supergroup 0 2020-09-24 16:31 /outputRelativeOccurrence
drwxr-xr-x - erick supergroup 0 2020-02-19 18:32 /outputYelp
erick@erick-ubuntu:~/Documents$ 
```

Creating directory  
**/user**

Creating directory  
**/user/erick**

Listing all contents of directory  
**/user**

Listing all contents of directory **/** (root directory)  
Note that I already had many other directories in root besides **/user**

Removing recursively the directory  
**/user**

Listing again all contents of directory **/** (root directory)

Note that directory **/user** is not there anymore

## Data manipulation in HDFS

- Transferring/copying data files from local file system to HDFS:  
*\$ hdfs dfs -put <filename in local file system/source> <target directory in HDFS>*
- Transferring/copying data files from HDFS to local file system:  
*\$ hdfs dfs -get <source file in HDFS> <target directory in local file system>*

**Note 1:** these are only some few commands that you can run over HDFS. Eventually you may need to do other operations.

**Note 2:** Remember that although you see it seems to be a local system, HDFS is a distributed file system. This is really interesting!!!

# Data manipulation in HDFS

```
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -mkdir /user  
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -mkdir /user/erick  
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -mkdir /user/erick/input  
erick@erick-ubuntu:~/Documents/hadoopExamples$ ls -la  
total 46488  
drwxrwxr-x 2 erick erick 4096 Feb 8 09:36 .  
drwxr-xr-x 14 erick erick 4096 Aug 25 14:00 ..  
-rw-rw-r-- 1 erick erick 6488666 Aug 18 2019 big.txt  
-rw-rw-r-- 1 erick erick 3027486 Feb 19 2020 business.csv  
-rw-rw-r-- 1 erick erick 1150 Feb 12 2020 cust_details.csv  
-rw-rw-r-- 1 erick erick 5272 Feb 19 2020 InMemoryJoin.jar  
-rw-rw-r-- 1 erick erick 79205 Jan 24 2020 inputWeather.txt  
-rw-r--r-- 1 erick erick 4198 Feb 8 08:52 MyWordCount.jar  
-rw-rw-r-- 1 erick erick 4198 Aug 27 20:10 MyWordCount.jar_old  
-rw-rw-r-- 1 erick erick 4198 Aug 25 13:57 MyWordCount.jar_old2  
-rw-rw-r-- 1 erick erick 5348 Feb 12 2020 ReduceSideJoin.jar  
-rw-rw-r-- 1 erick erick 8288 Feb 12 2020 RelativeOccurrence.jar  
-rw-rw-r-- 1 erick erick 25085396 Feb 19 2020 review.csv  
-rw-rw-r-- 1 erick erick 4053 Jan 24 2020 texasWeather.txt  
-rw-r--r-- 1 erick erick 3467 Sep 24 12:10 textInput.txt  
-rw-rw-r-- 1 erick erick 40630 Feb 12 2020 transaction_checking.ods  
-rw-rw-r-- 1 erick erick 32707 Feb 12 2020 transaction_details.csv  
-rw-rw-r-- 1 erick erick 12729122 Feb 19 2020 user.csv  
-rw-rw-r-- 1 erick erick 4110 Jan 24 2020 Weather1.jar  
-rw-rw-r-- 1 erick erick 23316 Jan 27 2020 weatherAnalytics.jar  
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -put big.txt /user/erick/input  
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -ls /user/erick/input  
Found 1 items  
-rw-r--r-- 1 erick supergroup 6488666 2021-02-08 09:39 /user/erick/input/big.txt  
erick@erick-ubuntu:~/Documents/hadoopExamples$ 
```

Creating directory **/user/erick/input**

*big.txt* is the file in my local system that I want to transfer to HDFS (into **/user/erick/input**)

Copying *big.txt* from my local system to HDFS.

There it is !!!

## Running MyWordCount.jar in Hadoop

- Now, lets execute the WordCount program in Hadoop...
- First, please make sure your hadoop system is running fine!
  - Remember: to start it, run the command below in shell
    - \$ start-all.sh
    - or
    - \$ start-dfs.sh
    - \$ start-yarn.sh

## Running MyWordCount.jar in Hadoop

- Transfer the file you want to count the words as input into the HDFS (like we did two slides ago)

```
$ hdfs dfs -put <filename in local file system/source> <target directory in HDFS>
```

\*In my case: \$ hdfs dfs -put big.txt /user/erick/input

- Run the JAR file with following command:

```
$ hadoop jar <path of the JAR file> <class name> <input path of file in HDFS> <directory you want to output>
```

\*In my case: hadoop jar MyWordCount.jar WordCount /user/erick/input/big.txt /outputCountWords

## Running Hadoop Program

---

# Running MyWordCount.jar in Hadoop

```
erick@erick-ubuntu:~/Documents/hadoopExamples$ hadoop jar MyWordCount.jar WordCount /user/erick/input/big.txt /outputCountWords
21/02/08 10:07:26 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
21/02/08 10:07:26 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
21/02/08 10:07:27 INFO input.FileInputFormat: Total input files to process : 1
21/02/08 10:07:27 INFO mapreduce.JobSubmitter: number of splits:1
21/02/08 10:07:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1612623958_0001
21/02/08 10:07:28 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
21/02/08 10:07:28 INFO mapreduce.Job: Running job: job_local1612623958_0001
21/02/08 10:07:28 INFO mapred.LocalJobRunner: OutputCommitter set in config null
21/02/08 10:07:28 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/02/08 10:07:28 INFO output.FileOutputCommitter: Fileoutputcommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
21/02/08 10:07:28 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
21/02/08 10:07:28 INFO mapred.LocalJobRunner: Waiting for map tasks
21/02/08 10:07:28 INFO mapred.LocalJobRunner: Starting task: attempt_local1612623958_0001_m_000000_0
21/02/08 10:07:28 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/02/08 10:07:28 INFO output.FileOutputCommitter: Fileoutputcommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
21/02/08 10:07:28 INFO mapred.Task: TaskAttempt 1 has failed
```

• • •

```
21/02/08 10:07:34 INFO mapred.LocalJobRunner: reduce > reduce
21/02/08 10:07:34 INFO mapred.Task: task 'attempt_local1612623958_0001_r_000000_0' done.
21/02/08 10:07:34 INFO mapred.LocalJobRunner: Final status: attempt_local1612623958_0001_r_000000_0
21/02/08 10:07:34 INFO mapred.LocalJobRunner: reduce task executor complete.
21/02/08 10:07:34 INFO mapreduce.Job: map 100% reduce 100%
21/02/08 10:07:34 INFO mapreduce.Job: Job job_local1612623958_0001 completed successfully
21/02/08 10:07:34 INFO mapreduce.Job: Counters: 35
  File System Counters
    FILE: Number of bytes read=26959920
    FILE: Number of bytes written=1075547
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=12977332
    HDFS: Number of bytes written=937095
    HDFS: Number of read operations=13
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Map-Reduce Framework
    Map input records=128457
    Map output records=1164553
    Map output bytes=1146463
    Map output materialized bytes=13475575
    Input split bytes=111
    Combine input records=0
    Combine output records=0
    Reduce input groups=81410
    Reduce shuffle bytes=13475575
    Reduce input records=1164553
    Reduce output records=81410
    Spilled Records=2329106
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=57
    Total committed heap usage (bytes)=465313792
  shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=498666
  File Output Format Counters
    Bytes Written=937095
erick@erick-ubuntu:~/Documents/hadoopExamples$
```

# Running MyWordCount.jar in Hadoop

- The job was successfully run! Now we can see the output of the job and copy it to your local machine...

```
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -ls /outputCountWords
Found 2 items
-rw-r--r--  1 erick supergroup          0 2021-02-08 10:07 /outputCountWords/_SUCCESS
-rw-r--r--  1 erick supergroup  937095 2021-02-08 10:07 /outputCountWords/part-r-00000
erick@erick-ubuntu:~/Documents/hadoopExamples$ hdfs dfs -get /outputCountWords/part-r-00000 /home/erick/Documents/hadoopExamples
erick@erick-ubuntu:~/Documents/hadoopExamples$ tail -25 /home/erick/Documents/hadoopExamples/part-r-00000
|Missouri      1
|Montana       1
|Nebraska      1
|Nevada        1
|New            4
|North          2
|Ohio           1
|Oklahoma       1
|Oregon         1
|Pennsylvania   1
|Rhode          1
|South          2
|Tennessee     1
|Texas          1
|United         1
|Utah           1
|Vermont        1
|Virginia       1
|Washington     1
|West           1
|Wisconsin      1
|Wyoming        1
|but            1
|had            1
|not            1
```

## Running Hadoop Program

# Running MyWordCount.jar in Hadoop

- Remember: you can always use the browser interface to navigate and check your files in HDFS:

The screenshot shows the Hadoop Web UI's Overview page for a cluster running on 'localhost:9000'. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, Utilities (with a dropdown menu for 'Browse the file system' and 'Logs'), and a user icon. Below the navigation is a summary table with the following data:

Started:	Sun Feb 07 20:19:34 -0600 2021
Version:	2.8.1, r20fe5304904fc2f5a18053c389e43cd26f7a70fe
Compiled:	Fri Jun 02 01:14:00 -0500 2017 by vinodkv from branch-2.8.1-private
Cluster ID:	CID-e2e27460-3b26-4390-b0b1-936d74484a57
Block Pool ID:	BP-1951117247-127.0.1.1-1580231746472

The screenshot shows the 'Browse Directory' page for the '/user/erick/input' directory. The top navigation bar is identical to the Overview page. The main area displays a table of file entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	erick	supergroup	6.19 MB	Feb 08 09:39	1	128 MB	big.txt

Showing 1 to 1 of 1 entries.

The screenshot shows the 'Browse Directory' page for the '/outputCountWords' directory. The top navigation bar is identical to the Overview page. The main area displays a table of file entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	erick	supergroup	0 B	Feb 08 10:07	1	128 MB	_SUCCESS
-rw-r--r--	erick	supergroup	915.13 KB	Feb 08 10:07	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries.

Browse Directory

Browse Directory

*Thank you*

For questions about the  
material, please come to  
office hours or contact me:  
**[exs172930@utdallas.edu](mailto:exs172930@utdallas.edu)**