Exercises Preparation

Setup Hadoop, HDFS and Yarn manually (standalone)

Install and Setup Java

1. Install OpenJDK (JDK 8):

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

2. Verify installation:

```
java -version
openjdk version "1.8.0_275"
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~20.04-b01)
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)
```

2. SET *JAVA_HOME* and *JRE_HOME*:

```
sudo vi /etc/environment
```

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
JRE_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"
```


Setup Hadoop User

1. Create User:

```
sudo adduser hadoop
sudo passwd hadoop
```

2. Switch To User:

```
sudo su hadoop
```

3. Switch Back To Root user:

```
exit
```


Setup SSH (needed by Hadoop components)

1. Install SSH and PDSH:

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

2. Create Private/Public Keypair for hadoop user (*without passphrase*):

```
sudo su hadoop
cd
ssh-keygen -t rsa -N "" -f /home/hadoop/.ssh/id_rsa
```

3. Add Public Key To Authorized Keys file (to enable passwordless ssh login)

```
cat /home/hadoop/.ssh/id_rsa.pub >> /home/hadoop/.ssh/authorized_keys
chmod 0600 /home/hadoop/.ssh/authorized_keys
```


Setup SSH (needed by Hadoop components)

4. Check If SSH Is Working

```
hadoop@big-data:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:YEUFliBVczkz2rvKWnYU9hB2ix2jnhBqLlbsJQfuBpE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1044-gcp x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
  System information as of Sat Oct 12 15:01:56 UTC 2019
  System load:  0.0               Processes:           117
  Usage of /:   5.8% of 28.90GB   Users logged in:     1
  Memory usage: 2%                IP address for ens4: 10.156.0.6
  Swap usage:   0%
30 packages can be updated.
17 updates are security updates.
Last login: Sat Oct 12 14:49:27 2019 from 80.144.211.195

hadoop@big-data:~$ exit
logout
Connection to localhost closed.

hadoop@big-data:~$
```


Install Hadoop

1. Download Hadoop (v3.1.1):

```
wget https://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2.tar.gz
```

2. Extract Binaries:

```
tar -xvzf hadoop-3.1.2.tar.gz
```

3. Move Binaries:

```
mv hadoop-3.1.2 hadoop
```


Configure Hadoop

1. Set Up **UNIX** Environment Variables

```
vi .bashrc
```

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export PDSH_RCMD_TYPE=ssh
```

```
source .bashrc
```


Configure Hadoop

2. Add **Hadoop** Environment Variables *(hadoop-env.sh)*

```
vi /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Configure Hadoop

3. Set Up **CORE** Variables *(core-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/core-site.xml
```

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

Configure Hadoop

4. Set Up **HDFS** Variables *(hdfs-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/hdfs-site.xml
```

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

      <property>
            <name>dfs.name.dir</name>
            <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
      </property>

      <property>
           <name>dfs.data.dir</name>
           <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
      </property>
</configuration>
```

Configure Hadoop

5. Set Up **MapReduce** Variables *(mapred-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/mapred-site.xml
```

```
<configuration>
      <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
      </property>
      <property>
            <name>yarn.app.mapreduce.am.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.map.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.reduce.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
</configuration>
```


Configure Hadoop

6. Set Up **YARN** Variables *(yarn-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/yarn-site.xml
```

```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.resource.memory-mb</name>
                <value>16384</value>
        </property>
</configuration>
```

Configure Hadoop

7. Clear HDFS

```
hdfs namenode -format
```

8. Start HDFS:

```
start-dfs.sh
```

9. Start YARN:

```
start-yarn.sh
```


Check Hadoop/HDFS

10. Run Admin Status Report

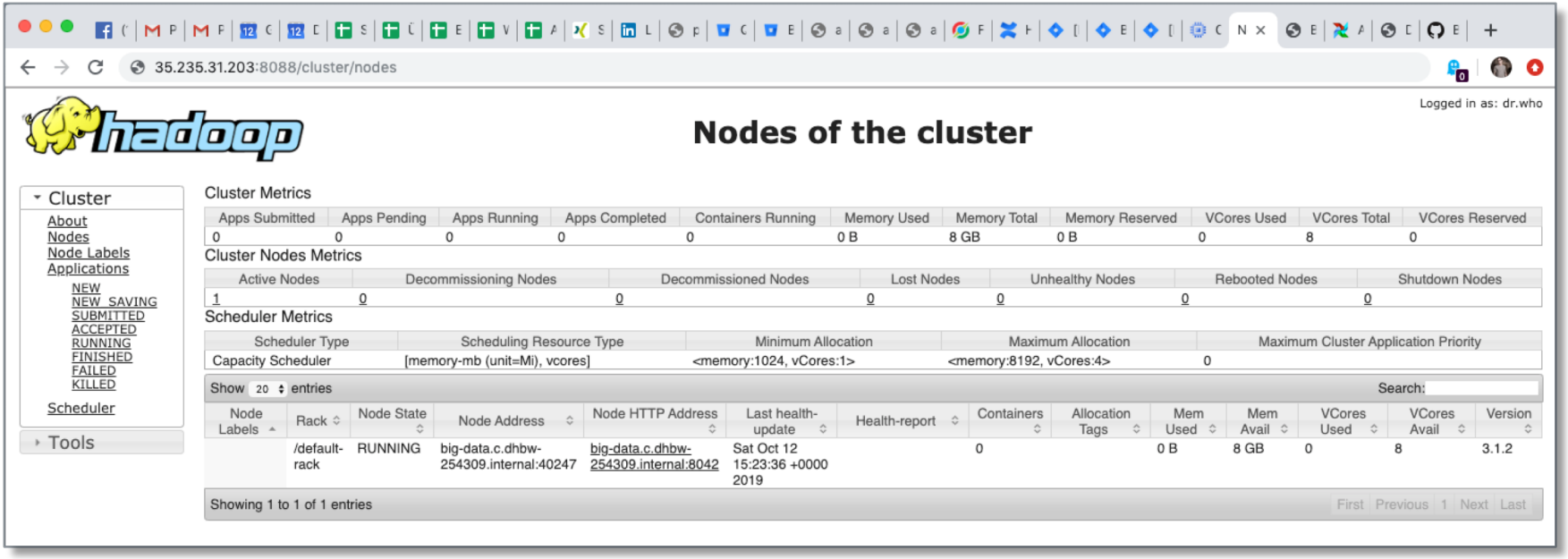
```
hdfs dfsadmin -report
```

```
Configured Capacity: 31035637760 (28.90 GB)
Present Capacity: 28187471872 (26.25 GB)
DFS Remaining: 28187447296 (26.25 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Replicated Blocks:
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0
Low redundancy blocks with highest priority to recover: 0
Pending deletion blocks: 0
Erasure Coded Block Groups:
Low redundancy block groups: 0
Block groups with corrupt internal blocks: 0
Missing block groups: 0
Low redundancy blocks with highest priority to recover: 0
Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (1):
Name: 127.0.0.1:9866 (localhost)
Hostname: big-data.c.dhbw-253679.internal
Decommission Status : Normal
Configured Capacity: 31035637760 (28.90 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 2831388672 (2.64 GB)
DFS Remaining: 28187447296 (26.25 GB)
DFS Used%: 0.00%
DFS Remaining%: 90.82%
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: 1
Last contact: Sat Oct 12 15:19:44 UTC 2019
Last Block Report: Sat Oct 12 15:18:29 UTC 2019
Num of Blocks: 0
```

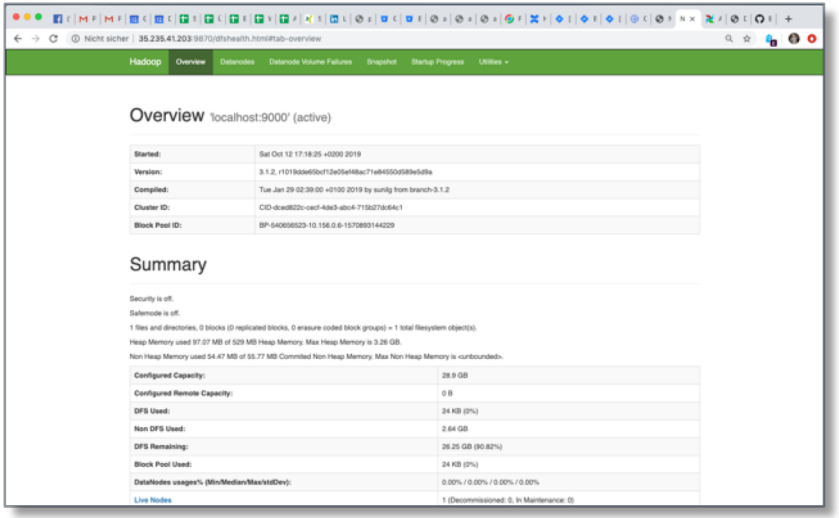
Check Hadoop/HDFS

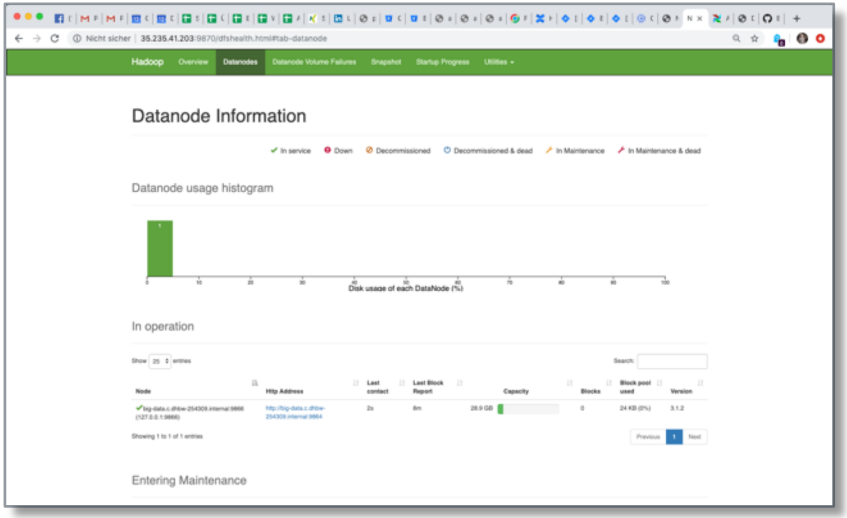
11. Check Ressource Manager Landing Page (http://XXX.XXX.XXX.XXX:8088/cluster):



Check Hadoop/HDFS

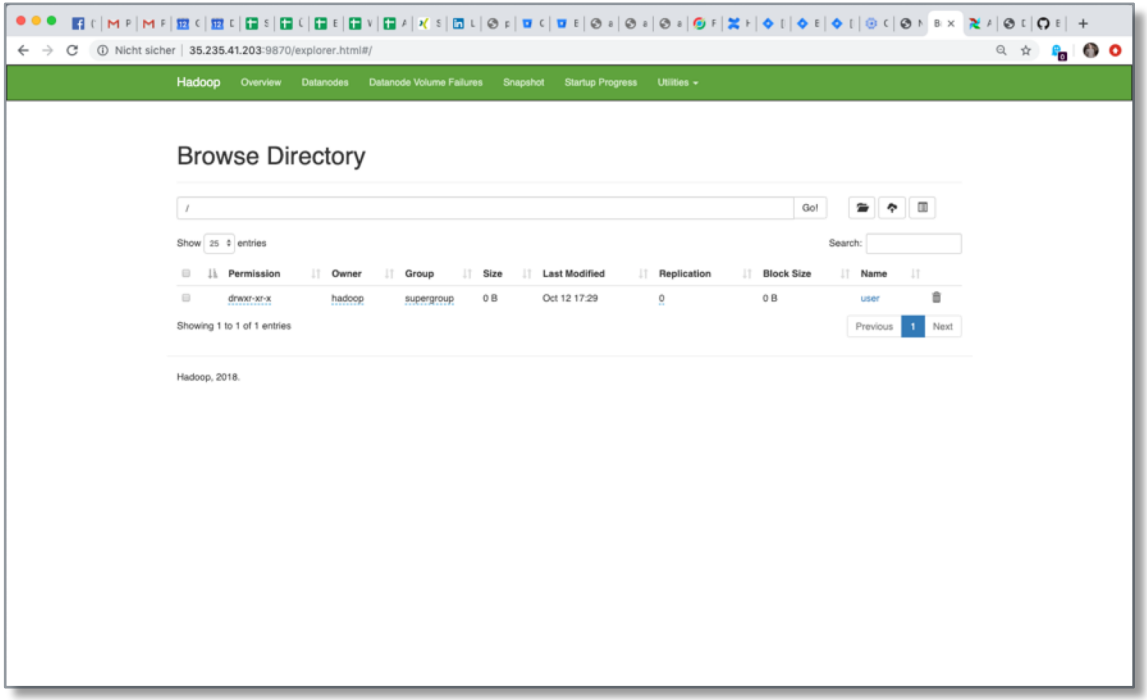
12. Check NameNode Landing and Status Page (http://XXX.XXX.XXX.XXX:9870):





Check Hadoop/HDFS

13. Check HDFS File Browser (http://XXX.XXX.XXX.XXX:9870/explorer.html#/)



Working with HDFS

1. Create User Directory (***on HDFS***):

```
hadoop fs -mkdir /user
hadoop fs -mkdir /user/hadoop
```

2. List Directories (*on HDFS*):

```
hadoop@big-data:~$ hadoop fs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2019-10-12 15:29 /user
hadoop@big-data:~$
```


Working with HDFS

3. Copy File (just a *random log file*) from local directory to HDFS:

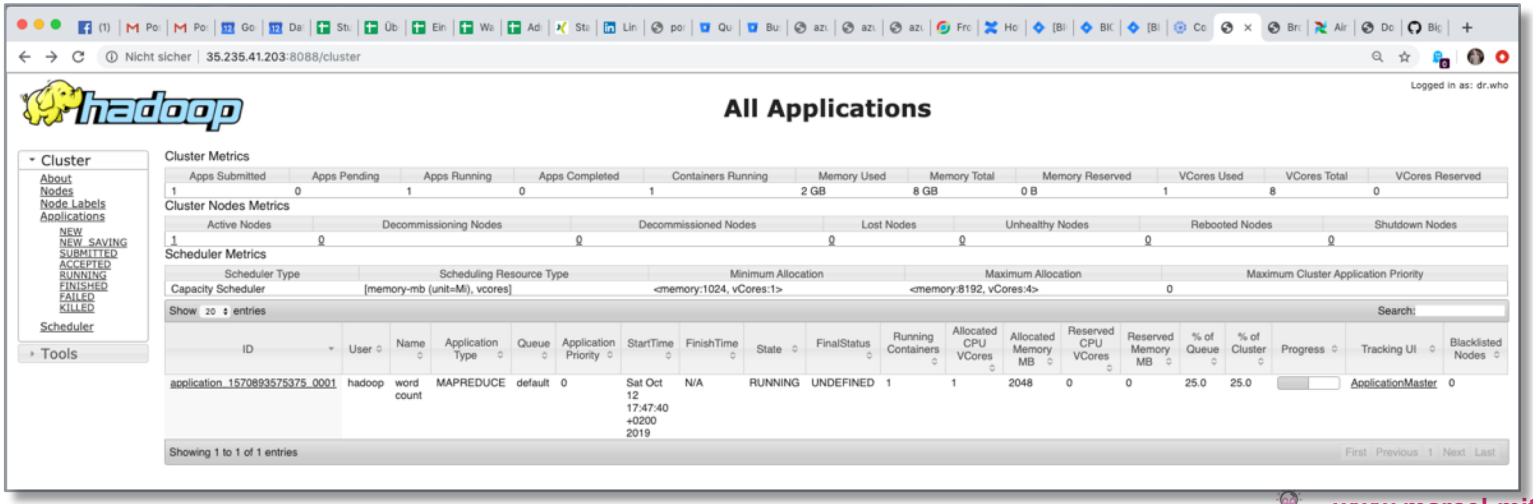
```
hadoop fs -put /var/log/dpkg.log /user/hadoop/dpkg.log
```


Run Example MapReduce Job

1. Using MapReduce WordCount Jar provided by Hadoop to count words within file *dpkg.log*

```
hadoop jar hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar
wordcount /user/hadoop/dpkg.log /user/hadoop/test_output
```

2. View Running MapReduce Job:



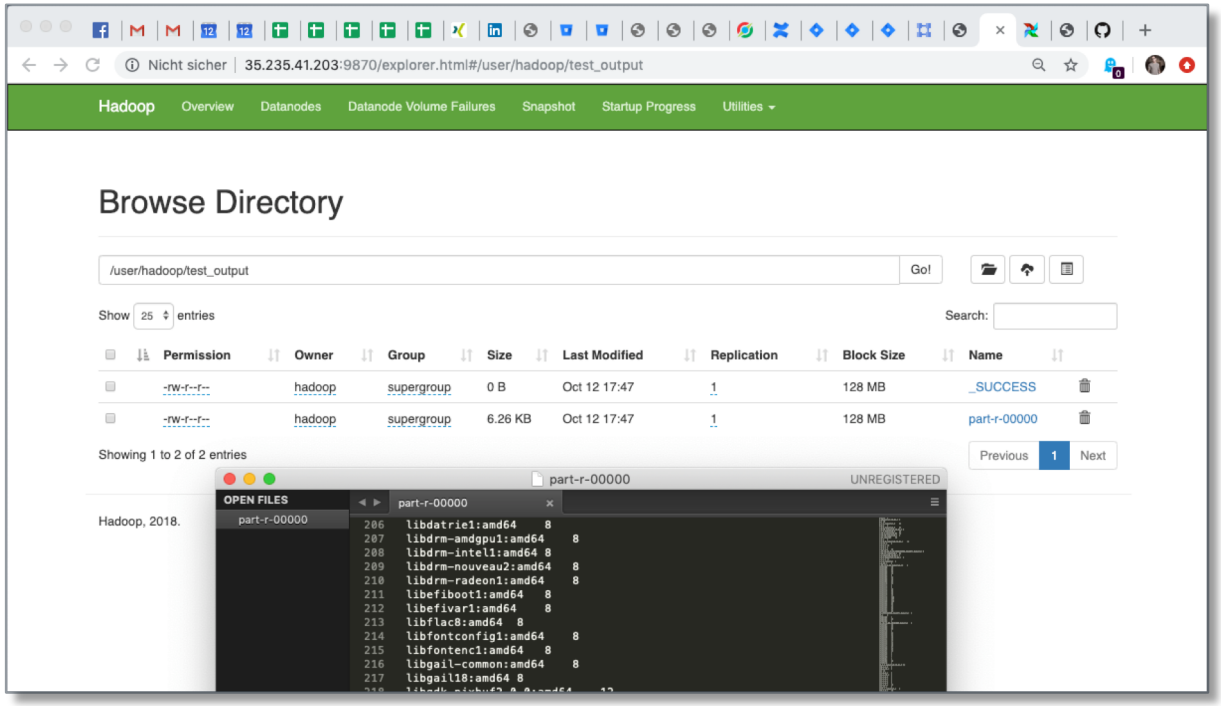
Run Example MapReduce Job

3. Take A Look At The Output/Result (*via Bash*):

```
hadoop@big-data:~$ hadoop fs -cat /user/hadoop/test_output/part-r-00000
…
libglx0:amd64 8
libgraphite2-3:amd64 8
libgtk2.0-0:amd64 8
libgtk2.0-bin:amd64 8
libgtk2.0-common:all 9
libharfbuzz0b:amd64 8
libice-dev:amd64 8
libice6:amd64 8
libjbig0:amd64 8
libjpeg-turbo8:amd64 8
libjpeg8:amd64 8
libnss3:amd64 8
libogg0:amd64 8
libpango-1.0-0:amd64 8
libpangocairo-1.0-0:amd64 8
…
```


Run Example MapReduce Job

4. Take A Look At The Output/Result (*via Web HDFS File Browser*):



Exercises I

Hadoop, HDFS, Yarn

Exercises

1. **Clone git repo** (to get sample data):

```
git clone https://github.com/marcelmittelstaedt/BigData.git
```

2.

- **Copy sample file** (*/BigData/exercises/winter_semester_2020-2021/01_hadoop/sample_data/Faust_1.txt*) from Git Repo **to HDFS**.
- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar*) **to calculate wordcount** for text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.

3.

- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar*) to **get the count of occurences** of the exact string ***‚Faust‘*** within text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.
- **Tip:** don‘t use *wordcount* part of jar but another MapReduce program on next slide.

Help

MapReduce Examples within *hadoop-mapreduce-examples-3.1.1.jar*:

aggregatewordcount:	An Aggregate based mapreduce program that counts the words in the input files.
aggregatewordhist:	An Aggregate based mapreduce program that computes the histogram of the words in the input files.
bbp:	A mapreduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
dbcount:	An example job that counts the pageview logs stored in a database.
distbbp:	A mapreduce program that uses a BBP-type formula to compute exact bits of Pi.
grep:	A mapreduce program that counts the matches of a regex in the input.
join:	A job that performs a join over sorted, equally partitioned datasets.
multifilewc:	A job that counts words from several files.
pentomino:	A mapreduce tile laying program to find solutions to pentomino problems.
pi:	A mapreduce program that estimates Pi using a quasi-Monte Carlo method.
randomtextwriter:	A mapreduce program that writes 10 GB of random textual data per node.
randomwriter:	A mapreduce program that writes 10 GB of random data per node.
secondarysort:	An example defining a secondary sort to the reduce phase.
sort:	A mapreduce program that sorts the data written by the random writer.
sudoku:	A sudoku solver.
teragen:	Generate data for the terasort.
terasort:	Run the terasort.
teravalidate:	Checking results of terasort.
wordcount:	A mapreduce program that counts the words in the input files.
wordmean:	A mapreduce program that counts the average length of the words in the input files.
wordmedian:	A mapreduce program that counts the median length of the words in the input files.
wordstandarddeviation:	A mapreduce program that counts the standard deviation of the length of the words in the input files.

Stop Your VM Instance

DON'T FORGET TO STOP YOUR VM INSTANCE!

```
gcloud compute instances stop big-data
```