Lab 01: Setup Environment for run Hadoop System

Group name: Bigdata boi

List of members:

- 19127276 – Nguyen Dang Thi
- 19127592 – Le Minh Tri
- 19127639 – Phan Vi Giai
- 19127642 – Vo Nhat Huy

Table content

Lab 01: Setup Environment for run Hadoop System 1

I. Introduction to Hadoop Ecosystem 2

II. Installation 2

1. Setup Hadoop run Pseudo-Distributed operation mode 2

2. Setup Hadoop multi-nodes run on Docker environment 12

III. Demonstration functionalities 22

1. Hadoop commands 22

2. Use Map-Reduce on Hadoop DFS with Pseudo-Distributed operation mode 27

IV. Self-evaluation 31

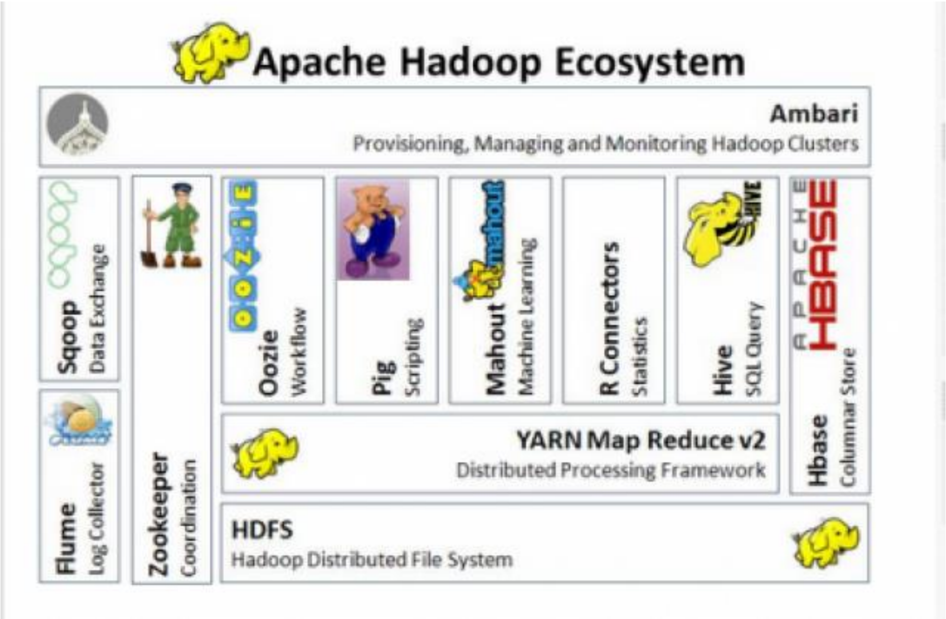
V. References 31

I. Introduction to Hadoop Ecosystem

The Apache Hadoop software library is an open-source framework maintained by the ASF that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is written in Java and optimized for massive amounts of data through distribution.

The Apache Hadoop framework is designed to scale up from single machine to thousands of machines which is integrated a Hadoop cluster, each offering local computation and storage. Further rely on hardware to possess high-availability, the architecture of framework is designed to resilient to failures at the application layer, so that delivering a high-availability service on top of a cluster of computers.

Beside the main purpose of data storage, several services such as processing, indexing and manipulating can be easily integrated on top a Hadoop cluster through Yet Another Resource Negotiator (>= Hadoop 2.0). These services help to solve any problems on Big Data in the biggest business companies.



Hadoop ecosystem (https://opensource.com/life/14/8/intro-apache-hadoop-big-data)

II. Installation

1. Setup Hadoop run Pseudo-Distributed operation mode

Overview:

"Hadoop can be installed in 3 different modes: Standalone mode, Pseudo-Distributed mode and Fully-Distributed mode.

...

***Pseudo-distributed mode** is also known as a **single-node cluster** where both NameNode and DataNode will reside on the same machine."*

https://medium0.com/analytics-vidhya/hadoop-single-node-cluster-setup-b11b957681f2

With Pseudo-distributed mode, Hadoop daemon is allowed to run as a single Java process.

Requirements and versions:

- Java engine version 8
- Hadoop version 2.10.1

Follow step by step installation:

Note: *Reports on installations in this section performed on Linux (Ubuntu 20.04 LTS)*

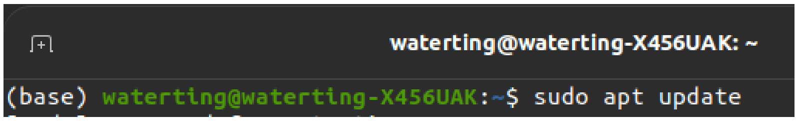
Step 1. Install OpenJDK

Hadoop is written in Java. So, it would require the JRE (Java Runtime Environment) at the very least. However, Hadoop is a program,Ong framework (for running map/reduce jobs or supporting YARN applications across a cluster).

To install OpenJDK, refer here: https://openjdk.java.net/install/

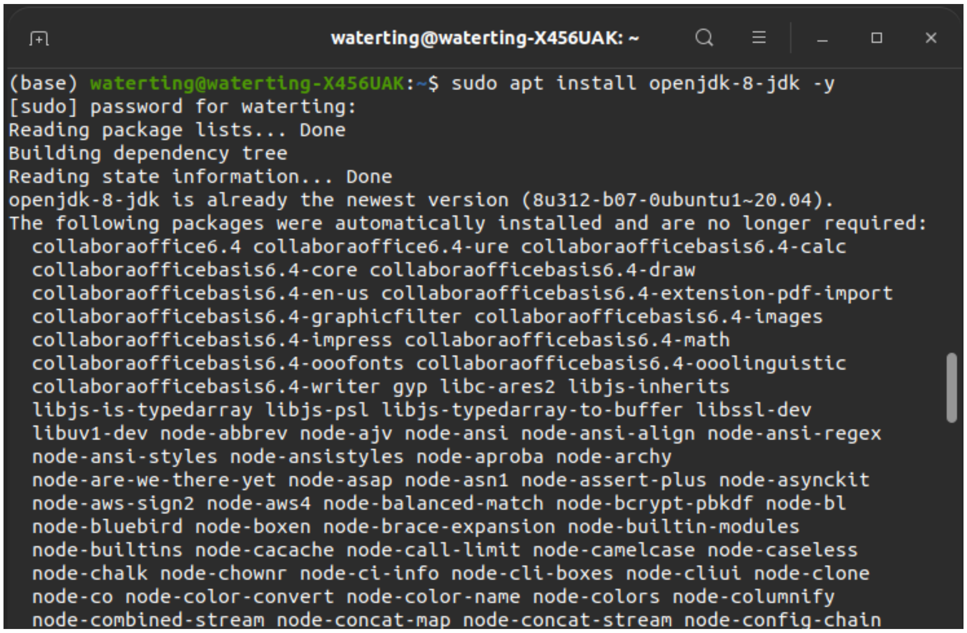
On ubuntu, we can install OpenJDK by following steps:

Update system before initiating a new installation:



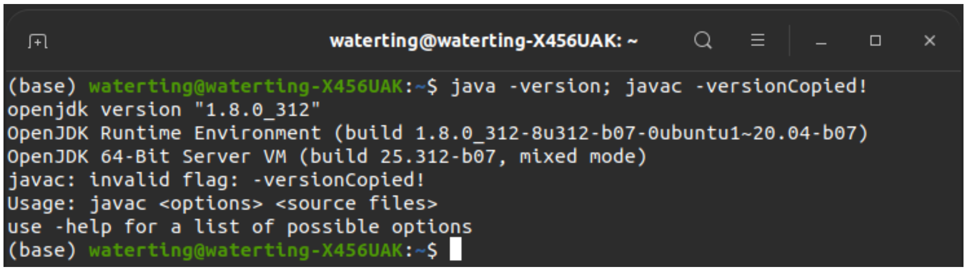
```
(base) watering@watering-X456UAK:~$ sudo apt update
```

Type the following command in your terminal to install OpenJDK 8:



```
(base) watering@watering-X456UAK:~$ sudo apt install openjdk-8-jdk -y
[sudo] password for watering:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-8-jdk is already the newest version (8u312-b07-0ubuntu1~20.04).
The following packages were automatically installed and are no longer required:
  collaboraoffice6.4 collaboraoffice6.4-ure collaboraofficebasis6.4-calc
  collaboraofficebasis6.4-core collaboraofficebasis6.4-draw
  collaboraofficebasis6.4-en-us collaboraofficebasis6.4-extension-pdf-import
  collaboraofficebasis6.4-graphicfilter collaboraofficebasis6.4-images
  collaboraofficebasis6.4-impress collaboraofficebasis6.4-math
  collaboraofficebasis6.4-ooofonts collaboraofficebasis6.4-ooolinguistic
  collaboraofficebasis6.4-writer gyp libc-ares2 libjs-inherits
  libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libssl-dev
  libuv1-dev node-abbrev node-ajv node-ansi node-ansi-align node-ansi-regex
  node-ansi-styles node-ansistyles node-aproba node-archy
  node-are-we-there-yet node-asap node-asn1 node-assert-plus node-asynckit
  node-aws-sign2 node-aws4 node-balanced-match node-bcrypt-pbkdf node-bl
  node-bluebird node-boxen node-brace-expansion node-builtin-modules
  node-builtins node-cacache node-call-limit node-camelcase node-caseless
  node-chalk node-chownr node-ci-info node-cli-boxes node-cliui node-clone
  node-co node-color-convert node-color-name node-colors node-columnify
  node-combined-stream node-concat-map node-concat-stream node-config-chain
```

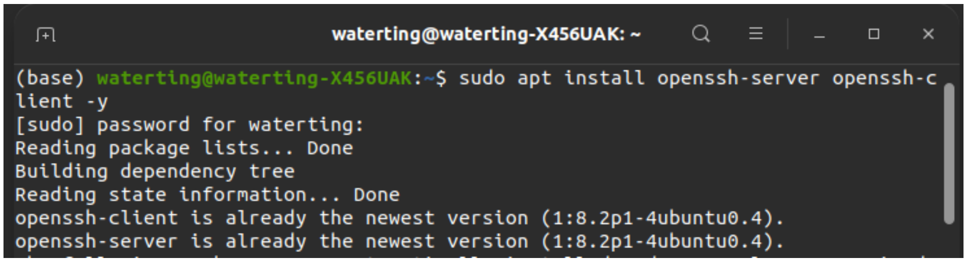
Check current Java version:



```
(base) watering@watering-X456UAK:~$ java -version; javac -versionCopied!
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1~20.04-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
javac: invalid flag: -versionCopied!
Usage: javac <options> <source files>
use -help for a list of possible options
(base) watering@watering-X456UAK:~$ 
```

Step 2. Set Up a Non-Root User for Hadoop Environment

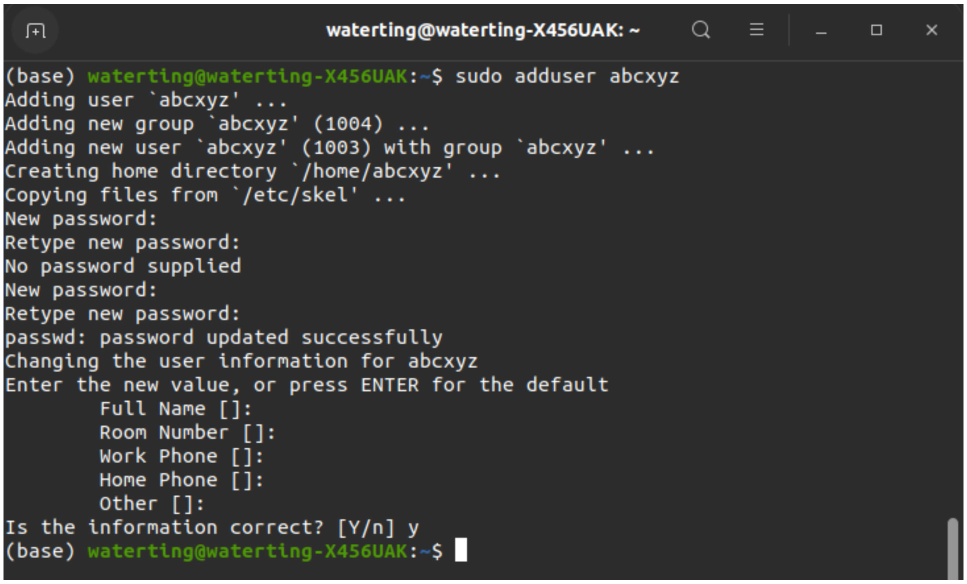
Install OpenSSH



```
(base) waterting@waterting-X456UAK:~$ sudo apt install openssh-server openssh-c
lient -y
[sudo] password for waterting:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:8.2p1-4ubuntu0.4).
openssh-server is already the newest version (1:8.2p1-4ubuntu0.4).
```

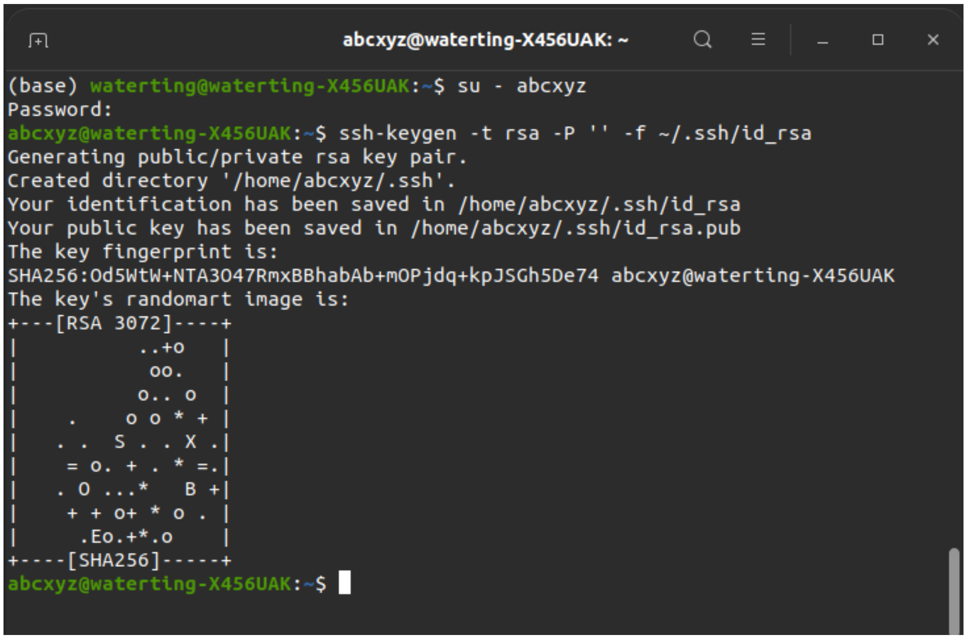
Create Hadoop user

Use **adduser** to create a new Hadoop user:

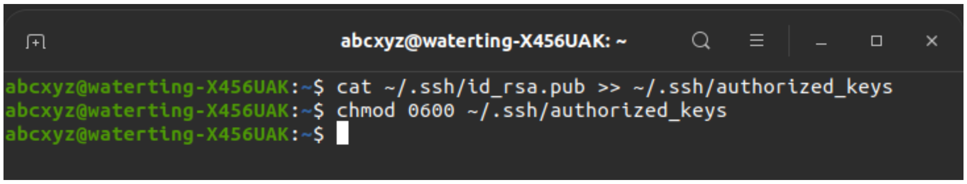


```
(base) waterting@waterting-X456UAK:~$ sudo adduser abcxyz
Adding user `abcxyz' ...
Adding new group `abcxyz' (1004) ...
Adding new user `abcxyz' (1003) with group `abcxyz' ...
Creating home directory `/home/abcxyz' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
No password supplied
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for abcxyz
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
(base) waterting@waterting-X456UAK:~$
```

Enable Passwordless SSH for Hadoop User



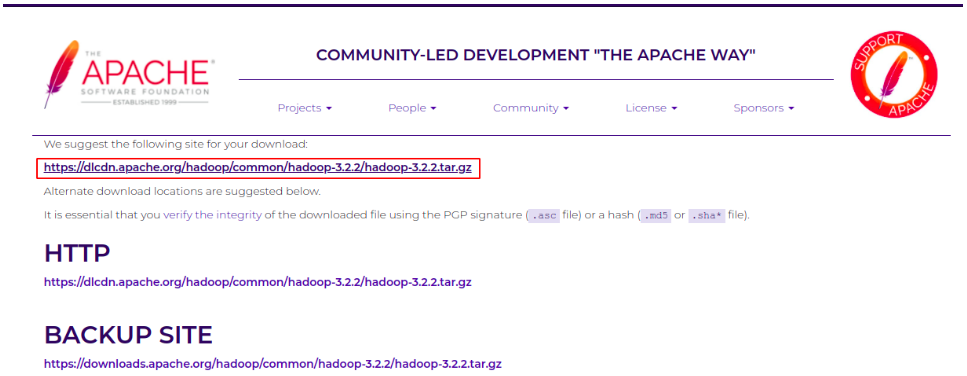
```
(base) waterting@waterting-X456UAK:~$ su - abcxyz
Password:
abcxyz@waterting-X456UAK:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/abcxyz/.ssh'.
Your identification has been saved in /home/abcxyz/.ssh/id_rsa
Your public key has been saved in /home/abcxyz/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Od5WtW+NTA3O47RmxBBhabAb+mOPjdq+kpJSGh5De74 abcxyz@waterting-X456UAK
The key's randomart image is:
+---[RSA 3072]----+
|          ..+o   |
|           oo.   |
|          o.. o  |
|    .    o o * + |
|   . .  S . . X .|
|    = o. + . * =.|
|   . O ...*   B +|
|    + + o+ * o . |
|     .Eo.+*.o    |
+----[SHA256]-----+
abcxyz@waterting-X456UAK:~$
```



```
abcxyz@waterting-X456UAK:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
abcxyz@waterting-X456UAK:~$ chmod 0600 ~/.ssh/authorized_keys
abcxyz@waterting-X456UAK:~$ 
```

Step 3. Download and Install Hadoop

Visit the Apache Hadoop homepage and select a Hadoop version to install: https://hadoop.apache.org/releases.html



We suggest the following site for your download:

https://dlcdn.apache.org/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz

Alternate download locations are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (.asc file) or a hash (.md5 or .sha* file).

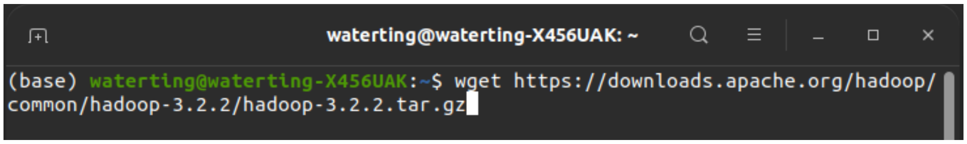
HTTP

https://dlcdn.apache.org/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz

BACKUP SITE

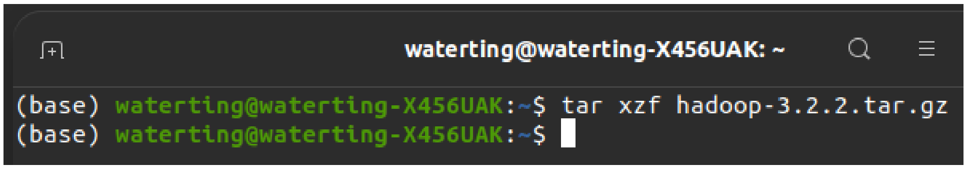
https://downloads.apache.org/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz

You can click on the link contains tar.gz file or copy link and run below command to download:



```
(base) waterting@waterting-X456UAK:~$ wget https://downloads.apache.org/hadoop/
common/hadoop-3.2.2/hadoop-3.2.2.tar.gz
```

Move to the folder containing the downloaded file and extract it



```
(base) waterting@waterting-X456UAK:~$ tar xzf hadoop-3.2.2.tar.gz
(base) waterting@waterting-X456UAK:~$ 
```

Remember to replace with the version you want to download accordingly

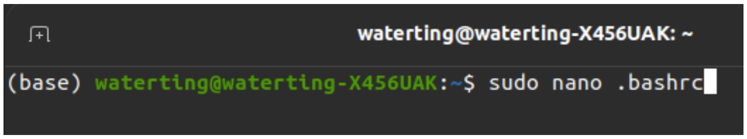
Step 4. Setup Hadoop run Pseudo-Distributed operation mode

To setup Hadoop for Pseudo-distributed mode, set up the following file:

- .bashrc
- hadoop-env.sh
- yarn-site.xml
- core-site.xml
- hdfs-site.xml
- mapred-site.xml

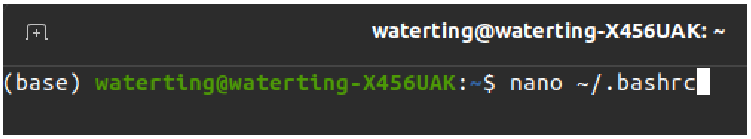
.bashrc

Using nano for configure the .bashrc shell. You can use another text editor to do the same.



```
(base) watering@watering-X456UAK:~$ sudo nano .bashrc
```

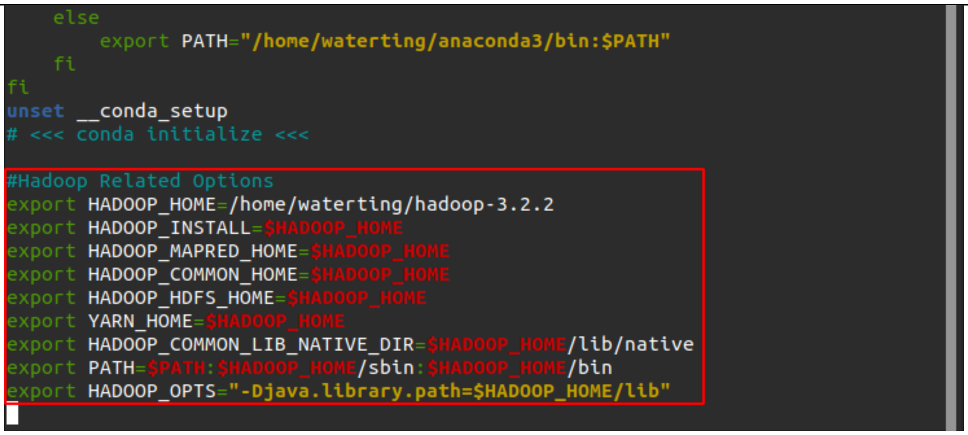
Or



```
(base) watering@watering-X456UAK:~$ nano ~/.bashrc
```

Put these lines into the end of .bashrc

```
#Hadoop Related Options
export HADOOP_HOME=<path to installed hadoop folder>
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:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```



```
    else
        export PATH="/home/waterting/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<

#Hadoop Related Options
export HADOOP_HOME=/home/waterting/hadoop-3.2.2
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:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

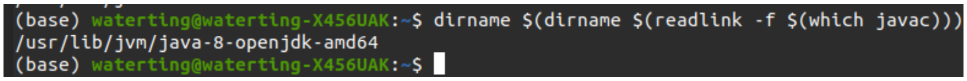
.bashrc

Note: You have to edit the path to the hadoop folder (HADOOP_HOME) based on your path.

hadoop-env.sh

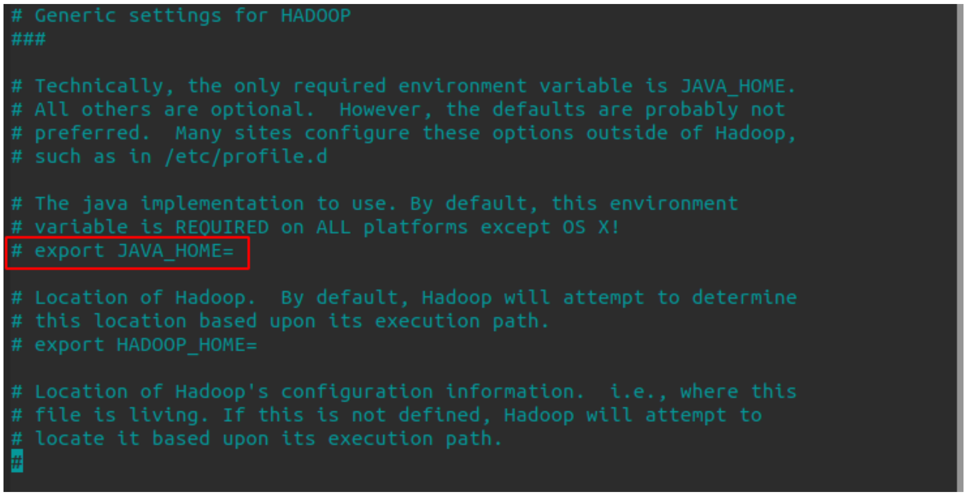
The hadoop-env.sh file is a file that contains some environment variable settings used by Hadoop. It is also a prerequisite for configuring settings related to YARN, DHFS and MapReduce.

In the hadoop-env.sh file, we need to uncomment the **$JAVA_HOME** variable and add the full path to the OpenJDK installation on your system. To check the correct path, you can refer here: https://www.baeldung.com/find-java-home



```
(base) waterting@waterting-X456UAK:~$ dirname $(dirname $(readlink -f $(which javac)))
/usr/lib/jvm/java-8-openjdk-amd64
(base) waterting@waterting-X456UAK:~$
```

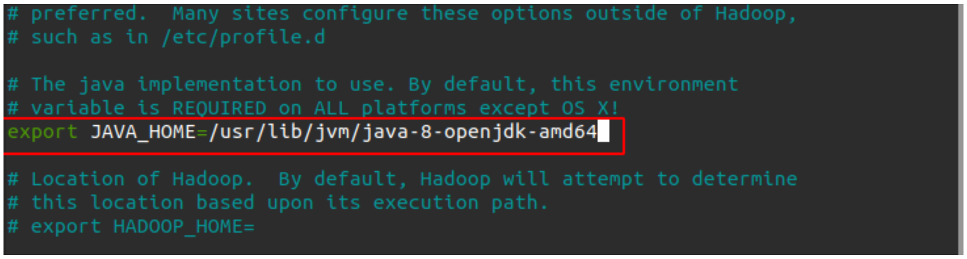
The file editing process will go from like this:



```
# Generic settings for HADOOP
###

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional.  However, the defaults are probably not
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
# export JAVA_HOME=

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information.  i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#
```

hadoop-env.sh

To this:



```
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=
```

hadoop-env.sh

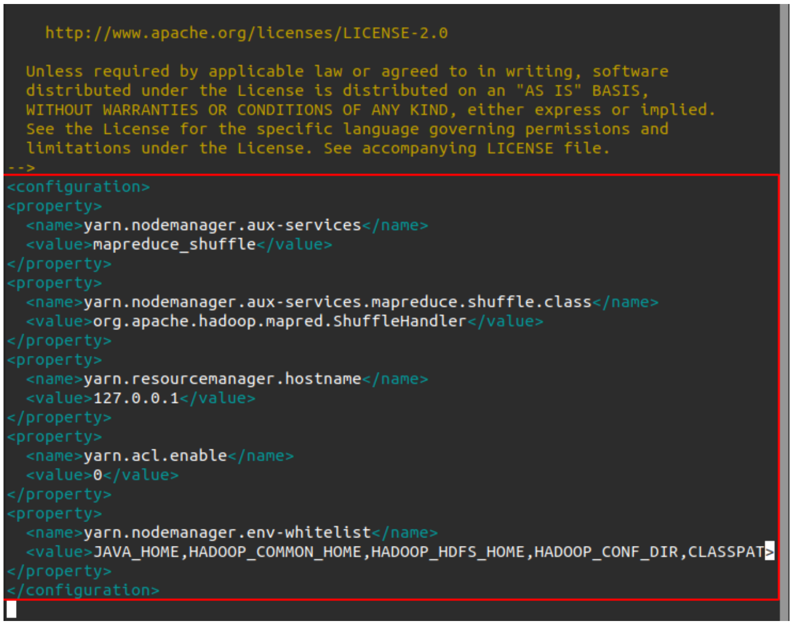
yarn-site.xml

The yarn-site.xml file is used to define YARN related settings. It contains configurations for Node Manager, Resource Manager, Containers and Application Master.

Open the yarn-site.xml file in a text editor, then append the following lines into the end of file:

```
<configuration>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>127.0.0.1</value>
</property>
<property>
    <name>yarn.acl.enable</name>
    <value>0</value>
</property>
<property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
```

```
</property>
</configuration>
```



```
    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.
-->
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPAT>
</property>
</configuration>
```

yarn-site.xml

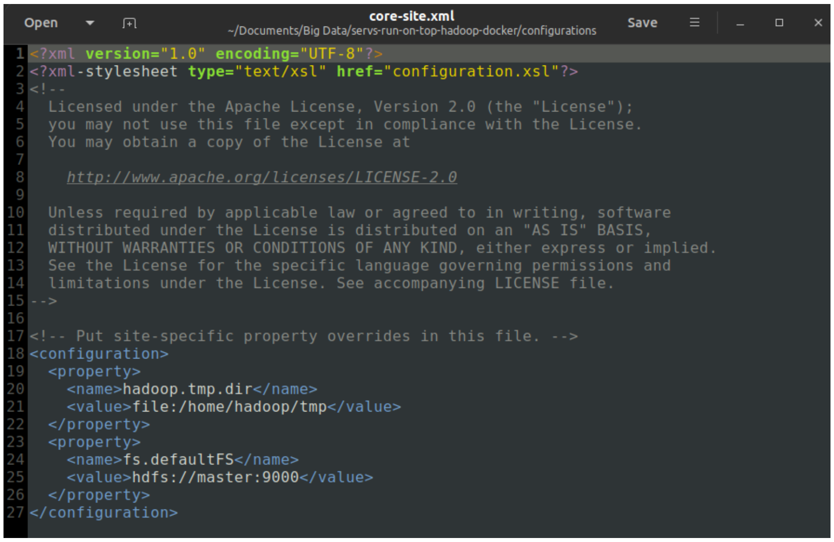
core-site.xml

The core-site. xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.

Open the core-site.xml in text editor, then add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>file:/home/hadoop/tmp</value>
</property>
<property>
<name>fs.defaultFS</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
```

You can customize the name and value, remember to create a directory in the location you specified for your temporary data.


Open
core-site.xml
~/Documents/Big Data/servs-run-on-top-hadoop-docker/configurations
Save

```
<?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>hadoop.tmp.dir</name>
    <value>file:/home/hadoop/tmp</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

core-site.xml

hdfs-site.xml

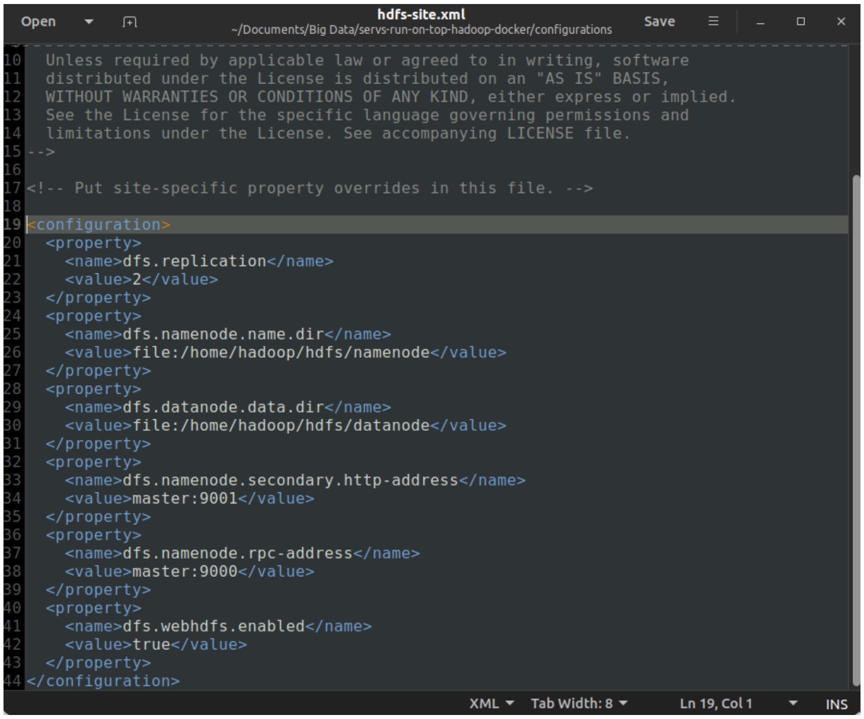
The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes. ... xml to specify default block replication and permission checking on HDFS. The actual number of replications can also be specified when the file is created.

Additionally, the default dfs.replication value of 3 needs to be changed to 1 to match the single node setup.

Open the hdfs-site.xml file for editing:

Add the following configuration to the file and adjust the NameNode and DataNode directories to your custom locations:

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>2</value>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home/hadoop/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home/hadoop/hdfs/datanode</value>
 </property>
 <property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>master:9001</value>
 </property>
 <property>
  <name>dfs.namenode.rpc-address</name>
  <value>master:9000</value>
 </property>
 <property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
 </property>
</configuration>
```


Open
hdfs-site.xml
~/Documents/Big Data/servs-run-on-top-hadoop-docker/configurations
Save
XML
Tab Width: 8
Ln 19, Col 1
INS

```
  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>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/hadoop/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/hadoop/hdfs/datanode</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>master:9001</value>
  </property>
  <property>
    <name>dfs.namenode.rpc-address</name>
    <value>master:9000</value>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
</configuration>
```

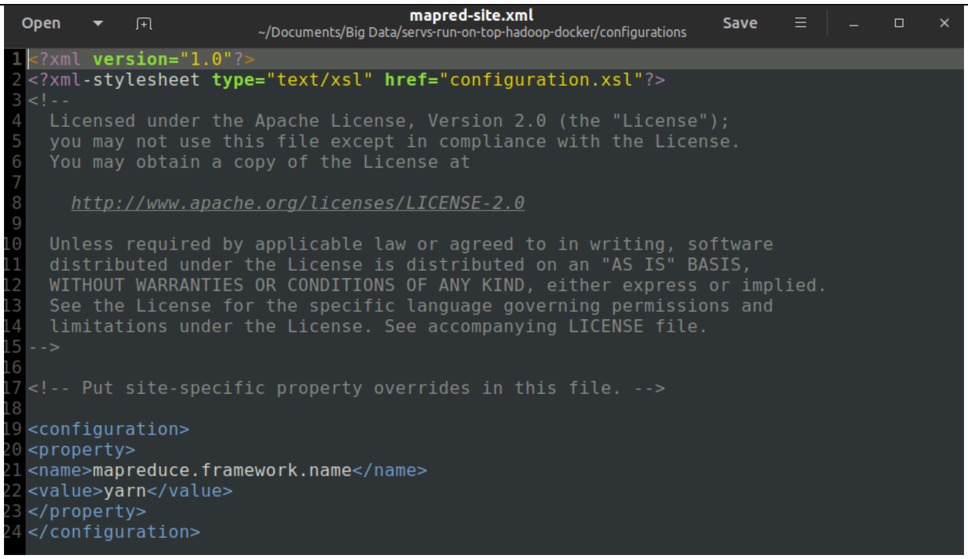
hdfs-site.xml

mapred-site.xml

The mapred-site. xml file contains the configuration settings for MapReduce daemons; the job tracker and the task-trackers.

Add the following configuration to change the default MapReduce framework name value to yarn:

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

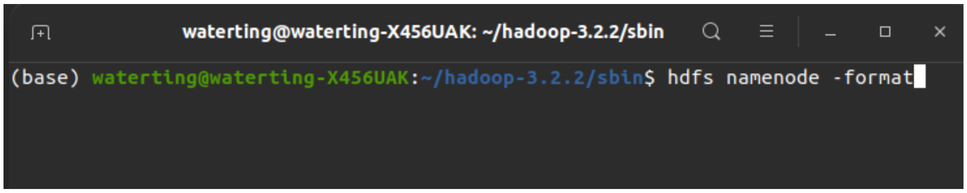

Open
mapred-site.xml
~/Documents/Big Data/servs-run-on-top-hadoop-docker/configurations
Save

```
<?xml version="1.0"?>
<?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>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

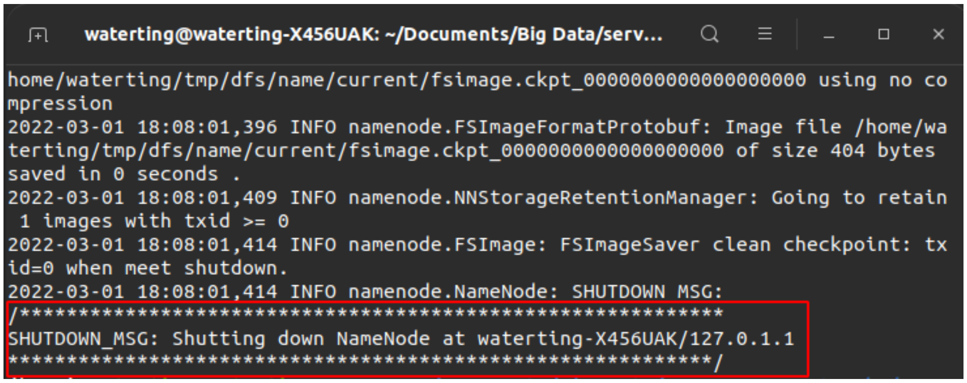
mapred-site.xml

Format HDFS NameNode

Before starting Hadoop service for the fist time, we need to format HDFS NameNode



```
(base) waterting@waterting-X456UAK:~/hadoop-3.2.2/sbin$ hdfs namenode -format
```

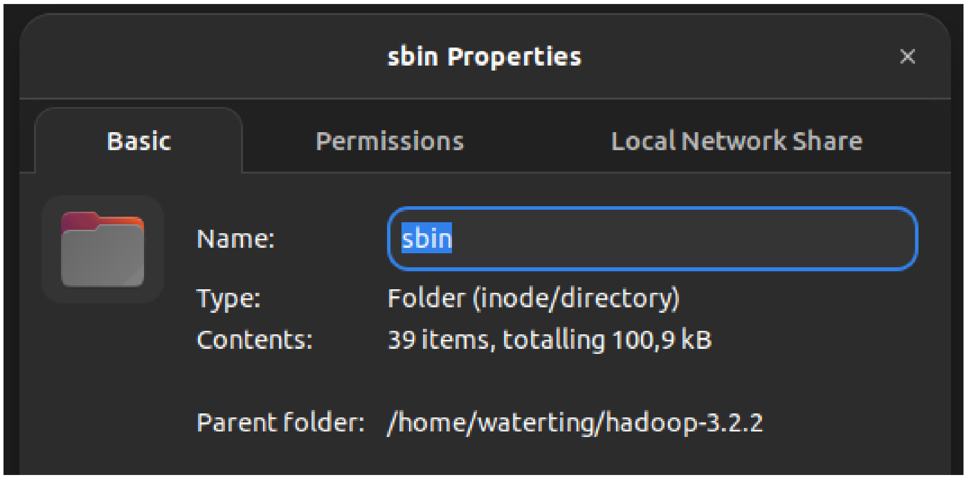


```
home/waterting/tmp/dfs/name/current/fsimage.ckpt_0000000000000000000 using no co
mpression
2022-03-01 18:08:01,396 INFO namenode.FSImageFormatProtobuf: Image file /home/wa
terting/tmp/dfs/name/current/fsimage.ckpt_0000000000000000000 of size 404 bytes
saved in 0 seconds .
2022-03-01 18:08:01,409 INFO namenode.NNStorageRetentionManager: Going to retain
 1 images with txid >= 0
2022-03-01 18:08:01,414 INFO namenode.FSImage: FSImageSaver clean checkpoint: tx
id=0 when meet shutdown.
2022-03-01 18:08:01,414 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at waterting-X456UAK/127.0.1.1
************************************************************/
```

format HDFS NameNode

Step 5. Start Hadoop Cluster

Open terminal in sbin folder inside hadoop folder was installed before.



Name: sbin

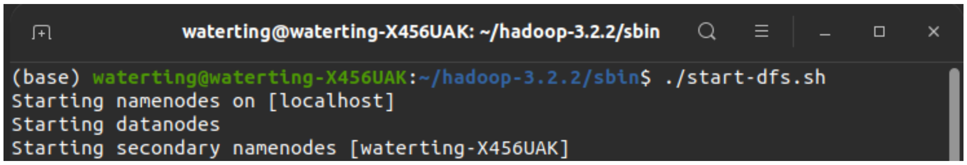
Type: Folder (inode/directory)

Contents: 39 items, totalling 100,9 kB

Parent folder: /home/waterting/hadoop-3.2.2

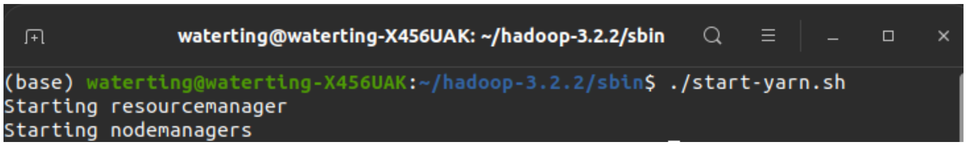
sbin folder location

Execute 2 files start-dfs.sh and start-yarn.sh, it may take a few moments.



```
(base) waterting@waterting-X456UAK:~/hadoop-3.2.2/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [waterting-X456UAK]
```

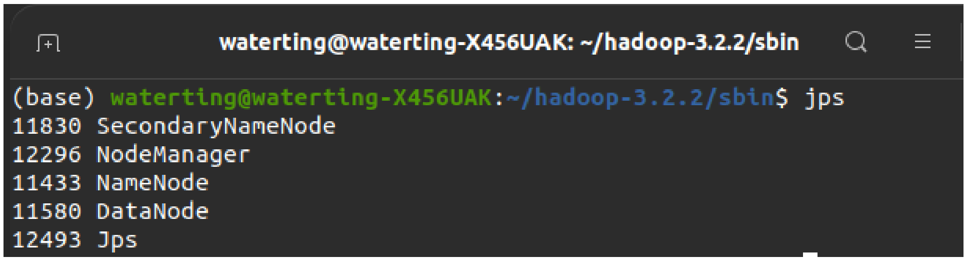
Execute start-dfs.sh



```
(base) waterting@waterting-X456UAK:~/hadoop-3.2.2/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

Execute start-yarn.sh

Use *jps* command to check if all the daemons are active and running as Java processes:



```
(base) waterting@waterting-X456UAK:~/hadoop-3.2.2/sbin$ jps
11830 SecondaryNameNode
12296 NodeManager
11433 NameNode
11580 DataNode
12493 Jps
```

2. Setup Hadoop multi-nodes run on Docker environment

Overview:

Docker is an open-source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications and have become increasingly popular as organizations shift to cloud-native development and hybrid multicloud environments.

Based on the idea of Docker running several containers like virtual machines, we take advantage of this to simulate a Hadoop cluster in a single computer. Our virtual Hadoop cluster includes 3 containers corresponding to 1 master and 2 slaves.

To connect these containers (virtual machines), we need docker container management to handle this problem. Docker compose is the key solution – a tool for defining, running and connect multi-container via a virtual network.

Full configurations for this installation can be found there.

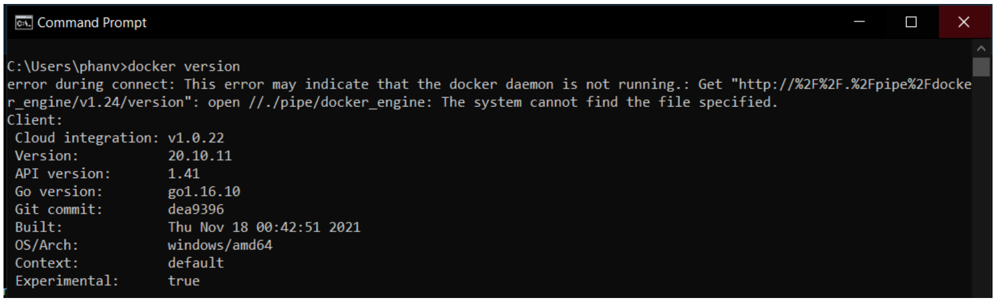
Requirements and versions:

- Docker engine and docker compose:
  - https://docs.docker.com/desktop/windows/install/
  - https://docs.docker.com/compose/install/
- Java engine version 8
- Hadoop version 2.10.1

Follow step by step installation:

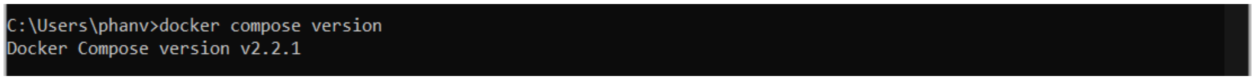
Step 1. Install Docker environment

- First, we need to install docker and docker compose environment through links in the requirement section
- After installing docker, we check whether it is installed by commands:
  - *docker version*
  - *docker compose version*



```
C:\Users\phanv>docker version
error during connect: This error may indicate that the docker daemon is not running.: Get "http://%2F%2F.%2Fpipe%2Fdocke
r_engine/v1.24/version": open //./pipe/docker_engine: The system cannot find the file specified.
Client:
 Cloud integration: v1.0.22
 Version:           20.10.11
 API version:       1.41
 Go version:        go1.16.10
 Git commit:        dea9396
 Built:             Thu Nov 18 00:42:51 2021
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true
```

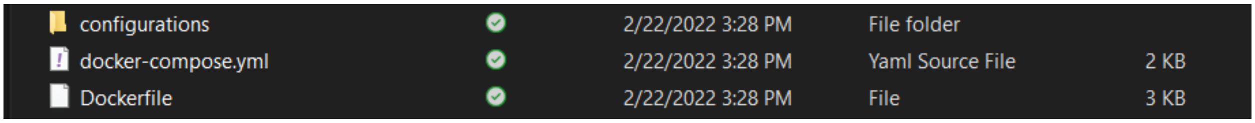
***docker** is installed*



```
C:\Users\phanv>docker compose version
Docker Compose version v2.2.1
```

***docker compose** is installed*

Step 2. Setup folder structure

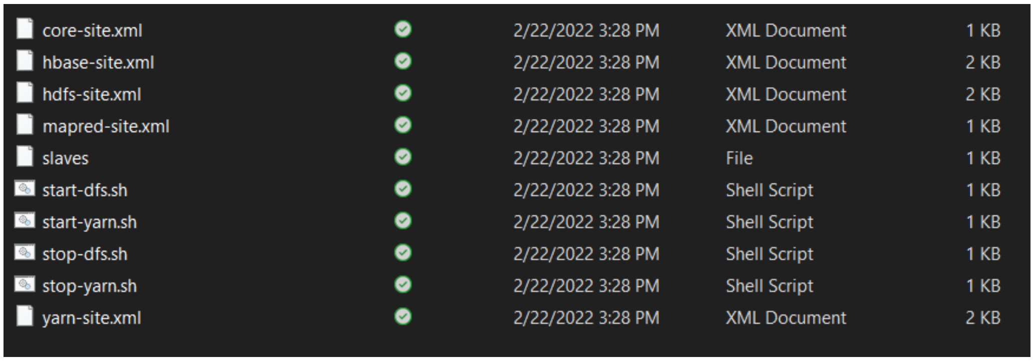


configurations	✓	2/22/2022 3:28 PM	File folder	
docker-compose.yml	✓	2/22/2022 3:28 PM	Yaml Source File	2 KB
Dockerfile	✓	2/22/2022 3:28 PM	File	3 KB

Folder structure

- **configurations**: The folder contains all configurations for Hadoop and other services if needed
- **docker-compose.yml**: The file defines our container configurations such as container name, ram, network ip, command, ports, … for each container.
- **Dockerfile**: The file contains all commands that set up our Hadoop and other services like downloading package, decompress, add configurations to Hadoop, **…**

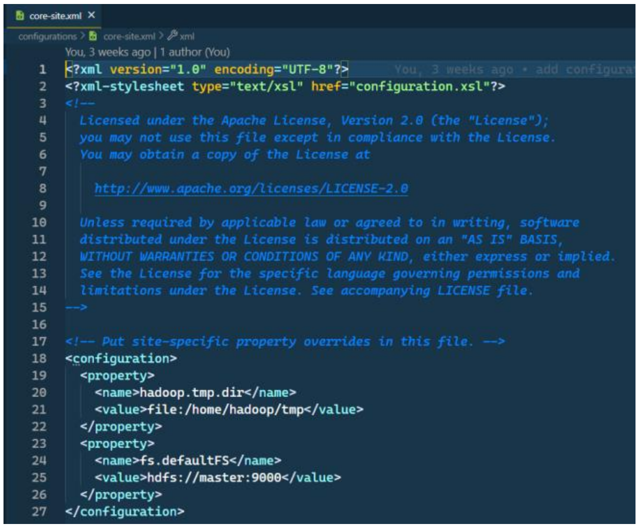
Step 3. Create files and configurations



core-site.xml	✓	2/22/2022 3:28 PM	XML Document	1 KB
hbase-site.xml	✓	2/22/2022 3:28 PM	XML Document	2 KB
hdfs-site.xml	✓	2/22/2022 3:28 PM	XML Document	2 KB
mapred-site.xml	✓	2/22/2022 3:28 PM	XML Document	1 KB
slaves	✓	2/22/2022 3:28 PM	File	1 KB
start-dfs.sh	✓	2/22/2022 3:28 PM	Shell Script	1 KB
start-yarn.sh	✓	2/22/2022 3:28 PM	Shell Script	1 KB
stop-dfs.sh	✓	2/22/2022 3:28 PM	Shell Script	1 KB
stop-yarn.sh	✓	2/22/2022 3:28 PM	Shell Script	1 KB
yarn-site.xml	✓	2/22/2022 3:28 PM	XML Document	2 KB

configurations folder

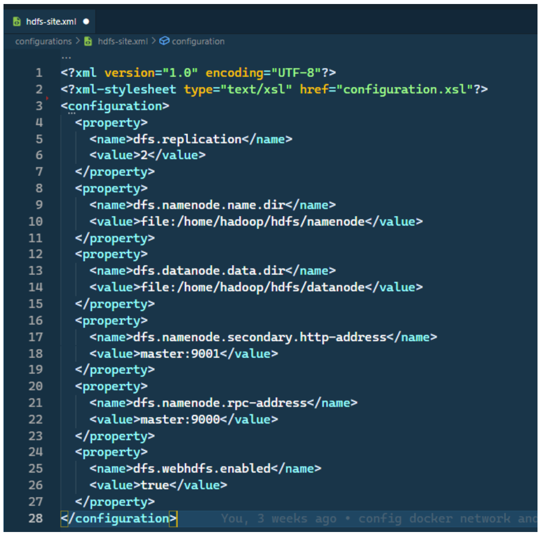
configurations folder contains several files for running Hadoop, that all files we needed to create:


core-site.xml ×
configurations > core-site.xml > xml
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configura

```xml
<?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>hadoop.tmp.dir</name>
    <value>file:/home/hadoop/tmp</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

core-site.xml

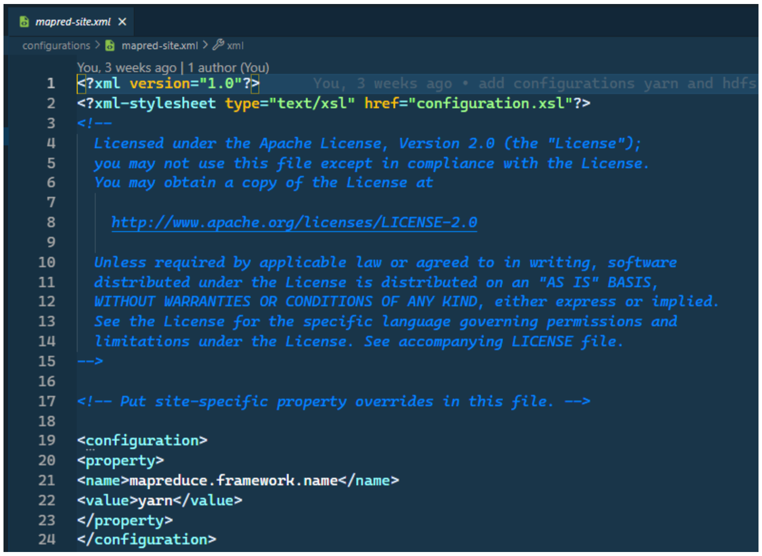
- **core-site.xml:** The file defines temporary directory for Hadoop and master host
  - **hadoop.tmp.dir:** direct to temporary directory which is manually created
  - **fs.defaultFS:** specify an address that allows dfs commands without providing full site name in the command


hdfs-site.xml ●
configurations > hdfs-site.xml > configuration
You, 3 weeks ago • config docker network anc

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/hadoop/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/hadoop/hdfs/datanode</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>master:9001</value>
  </property>
  <property>
    <name>dfs.namenode.rpc-address</name>
    <value>master:9000</value>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
</configuration>
```

hdfs-site.xml

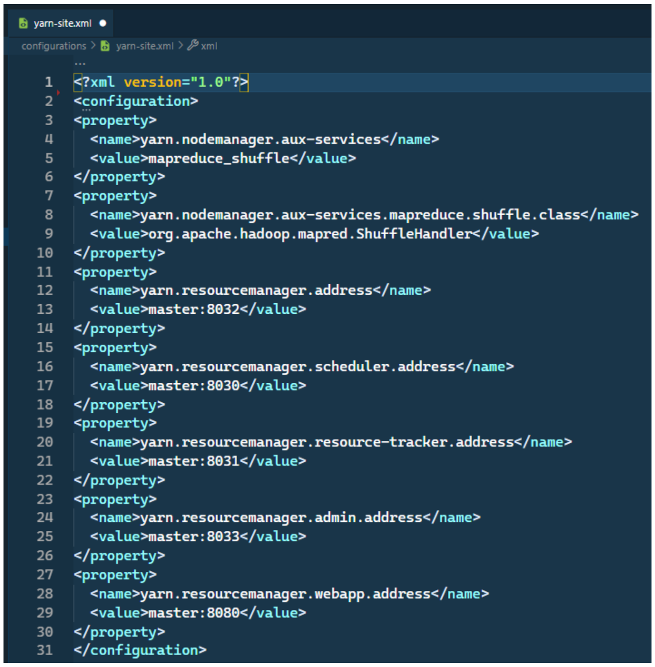
- **hdfs-site.xml:** The file configures Hadoop DFS properties like replication factors, directory for namenode, directory for datanode, and other properties
  - **dfs.replication:** we set 2 for 2 datanodes
  - **dfs.namenode.name.dir:** working directory for namenode
  - **dfs.datanode.data.dir:** working directory for datanode
  - **dfs.namenode.secondary.http-address:** The secondary namenode http server address and port.
  - **dfs.namenode.rpc-address:** RPC address that handles all client's requests
  - **dfs.webhdfs.enabled:** Enable WebHDFS (REST API) in Namenodes and Datanodes


mapred-site.xml ×
configurations > mapred-site.xml > xml
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configurations yarn and hdfs

```xml
<?xml version="1.0"?>
<?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>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

mapred-site.xml

- **mapred-site.xml:** The file use Yarn as Resource Manager for Map-Reduce application on Hadoop
  - **mapreduce.framework.name:** Specify *yarn* value for framework name as using Map-Reduce version 2

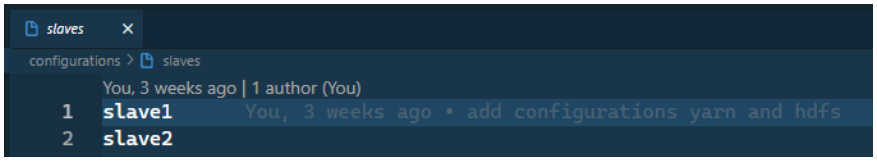


```xml
<?xml version="1.0"?>
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>master:8032</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>master:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>master:8031</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>master:8033</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value>master:8080</value>
</property>
</configuration>
```

yarn-site.xml

- **yarn-site.xml:** The file configures some property of Yarn ResourceManager
  - **yarn.nodemanager.aux-services:** use default shuffle and sort of Map-Reduce
  - **yarn.nodemanager.axu-services.mapreduce.shuffle.class:** The auxiliary service class to use as default value
  - **yarn.resourcemanager.address:** The address of the applications manager interface in the ResourceManager
  - **yarn.resourcemanager.scheduler.adress:** The address of the scheduler interface

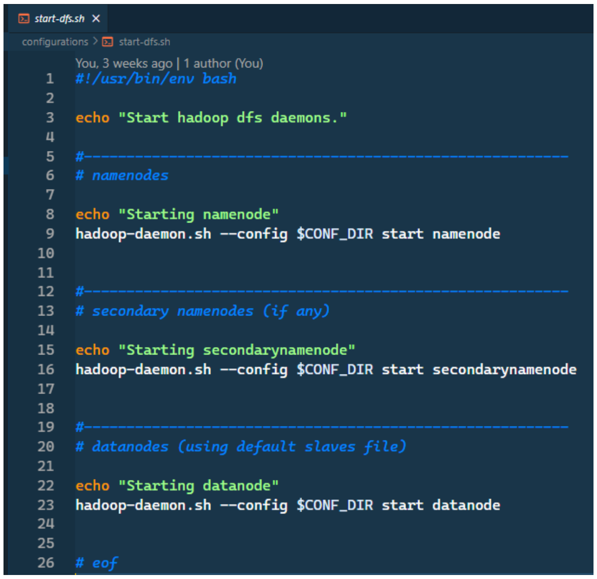
- **yarn.resourcemanager.resource-tracker.address:** The address of the resource-tracker interface
- **yarn.resourcemanager.admin.address:** The address of the admin interface
- **yarn.resourcemanager.webapp.address:** The http address of the RM web application


slaves
configurations > slaves
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configurations yarn and hdfs

```
slave1
slave2
```

slaves

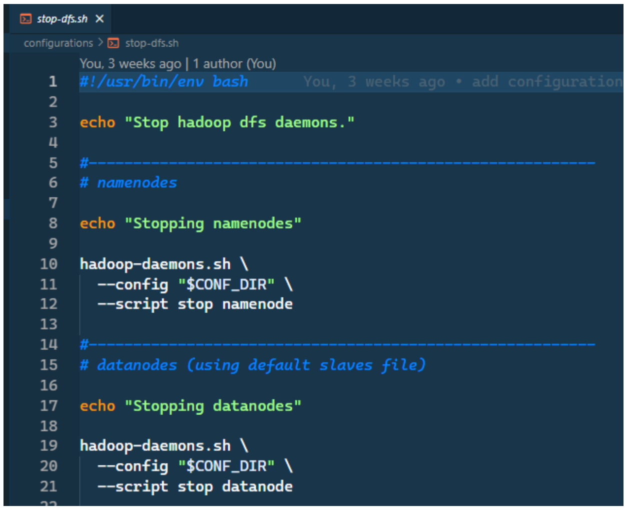
- **slaves:** Define name of slaves which are slave1 for datanode 1 and slave2 for datanode 2


start-dfs.sh
configurations > start-dfs.sh
You, 3 weeks ago | 1 author (You)

```
#!/usr/bin/env bash

echo "Start hadoop dfs daemons."

#---------------------------------------------------------
# namenodes

echo "Starting namenode"
hadoop-daemon.sh --config $CONF_DIR start namenode


#---------------------------------------------------------
# secondary namenodes (if any)

echo "Starting secondarynamenode"
hadoop-daemon.sh --config $CONF_DIR start secondarynamenode


#---------------------------------------------------------
# datanodes (using default slaves file)

echo "Starting datanode"
hadoop-daemon.sh --config $CONF_DIR start datanode


# eof
```

start-dfs.sh

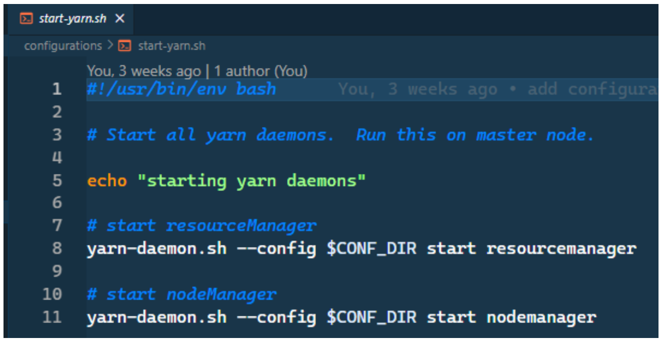
- **start-dfs.sh:** That is just a default file when we install Hadoop locally. The following command starts demons like starting namenode, secondary namenode (if any), datanodes.


stop-dfs.sh
configurations > stop-dfs.sh
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configuration

```bash
#!/usr/bin/env bash

echo "Stop hadoop dfs daemons."

#-----------------------------------------------------------
# namenodes

echo "Stopping namenodes"

hadoop-daemons.sh \
  --config "$CONF_DIR" \
  --script stop namenode

#-----------------------------------------------------------
# datanodes (using default slaves file)

echo "Stopping datanodes"

hadoop-daemons.sh \
  --config "$CONF_DIR" \
  --script stop datanode
```

stop-dfs.sh

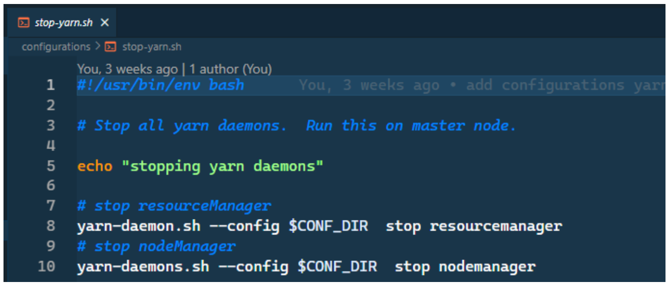
- **stop-dfs.sh:** That is also just a default file when we install Hadoop locally. The following commands stop demons like stopping namenode, secondary namenode (if any), datanodes.


start-yarn.sh
configurations > start-yarn.sh
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configura

```bash
#!/usr/bin/env bash

# Start all yarn daemons.  Run this on master node.

echo "starting yarn daemons"

# start resourceManager
yarn-daemon.sh --config $CONF_DIR start resourcemanager

# start nodeManager
yarn-daemon.sh --config $CONF_DIR start nodemanager
```

start-yarn.sh

- **start-yarn.sh:** The file is starting Yarn as a Resource Manager and Node Manager.


stop-yarn.sh
configurations > stop-yarn.sh
You, 3 weeks ago | 1 author (You)
You, 3 weeks ago • add configurations yarn

```bash
#!/usr/bin/env bash

# Stop all yarn daemons.  Run this on master node.

echo "stopping yarn daemons"

# stop resourceManager
yarn-daemon.sh --config $CONF_DIR  stop resourcemanager
# stop nodeManager
yarn-daemons.sh --config $CONF_DIR  stop nodemanager
```

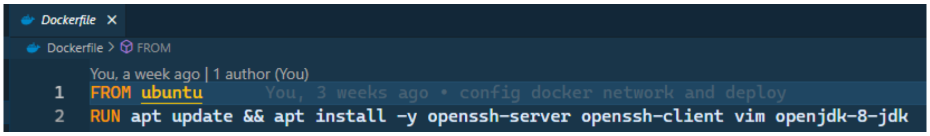
stop-yarn.sh

- **stop-yarn.sh:** The file is a default file when installing Hadoop. The following commands are stopping Yarn daemons.

Step 4. Setup Dockerfile and docker-compose.yml

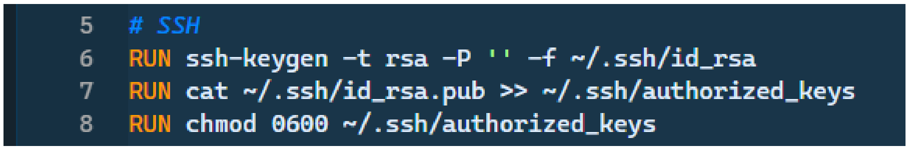
We need to create 2 files that are **Dockerfile** and **docker-compose.yml**

First, **Dockerfile** contains all command needed to download package like Hadoop or something you want to integrate with Hadoop system, then setup folder like we installed Hadoop in local mode and add configuration files to Hadoop. All those commands are executed in containers.


Dockerfile ×
Dockerfile > FROM
You, a week ago | 1 author (You)
You, 3 weeks ago • config docker network and deploy

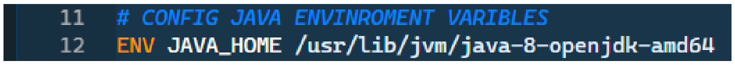
```
FROM ubuntu
RUN apt update && apt install -y openssh-server openssh-client vim openjdk-8-jdk
```

- We use **ubuntu** operating system then install **openssh-client** for remote login with **SSH** protocol and **openjdk-8-jdk** for both the runtime environment and development kit of Java 8



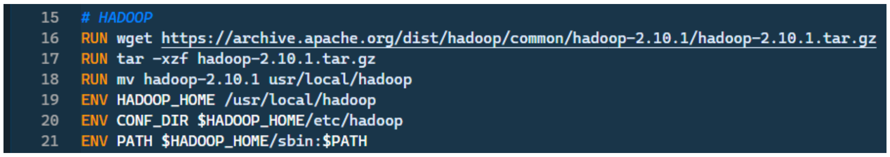
```
# SSH
RUN ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
RUN cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
RUN chmod 0600 ~/.ssh/authorized_keys
```

- To enable passwordless **SSH** for Hadoop user
    - First, we need to generate an SSH key pair and define the location to store it
    - Copy value to the *authorized_keys* folder
    - Set the permissions for your user with the *chmod* command



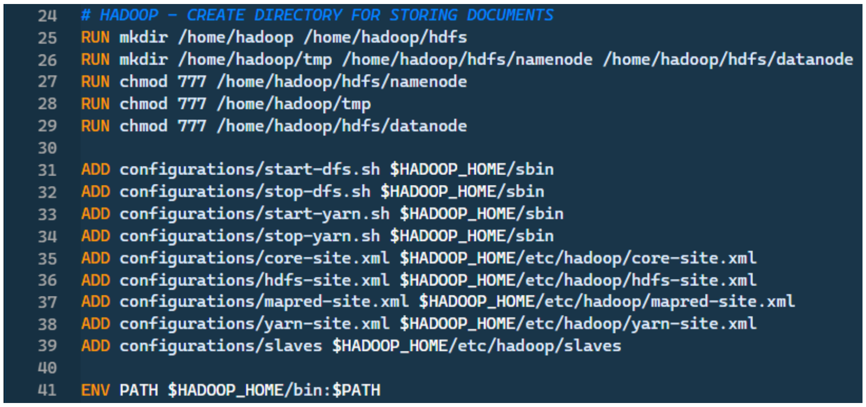
```
# CONFIG JAVA ENVINROMENT VARIBLES
ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64
```

- Configures global Java environment variables



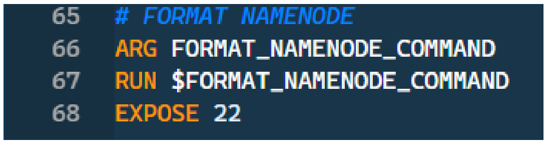
```
# HADOOP
RUN wget https://archive.apache.org/dist/hadoop/common/hadoop-2.10.1/hadoop-2.10.1.tar.gz
RUN tar -xzf hadoop-2.10.1.tar.gz
RUN mv hadoop-2.10.1 usr/local/hadoop
ENV HADOOP_HOME /usr/local/hadoop
ENV CONF_DIR $HADOOP_HOME/etc/hadoop
ENV PATH $HADOOP_HOME/sbin:$PATH
```

- Download Hadoop package
    - First, we download Hadoop package from repository
    - Decompress package
    - Move folder after decompressing to our specific location and rename it for shorthand
    - Configure some Hadoop environment variables such as $HADOOP_HOME, $CONF_DIR, $PATH in order to use these variables in short later



```
# HADOOP - CREATE DIRECTORY FOR STORING DOCUMENTS
RUN mkdir /home/hadoop /home/hadoop/hdfs
RUN mkdir /home/hadoop/tmp /home/hadoop/hdfs/namenode /home/hadoop/hdfs/datanode
RUN chmod 777 /home/hadoop/hdfs/namenode
RUN chmod 777 /home/hadoop/tmp
RUN chmod 777 /home/hadoop/hdfs/datanode

ADD configurations/start-dfs.sh $HADOOP_HOME/sbin
ADD configurations/stop-dfs.sh $HADOOP_HOME/sbin
ADD configurations/start-yarn.sh $HADOOP_HOME/sbin
ADD configurations/stop-yarn.sh $HADOOP_HOME/sbin
ADD configurations/core-site.xml $HADOOP_HOME/etc/hadoop/core-site.xml
ADD configurations/hdfs-site.xml $HADOOP_HOME/etc/hadoop/hdfs-site.xml
ADD configurations/mapred-site.xml $HADOOP_HOME/etc/hadoop/mapred-site.xml
ADD configurations/yarn-site.xml $HADOOP_HOME/etc/hadoop/yarn-site.xml
ADD configurations/slaves $HADOOP_HOME/etc/hadoop/slaves

ENV PATH $HADOOP_HOME/bin:$PATH
```

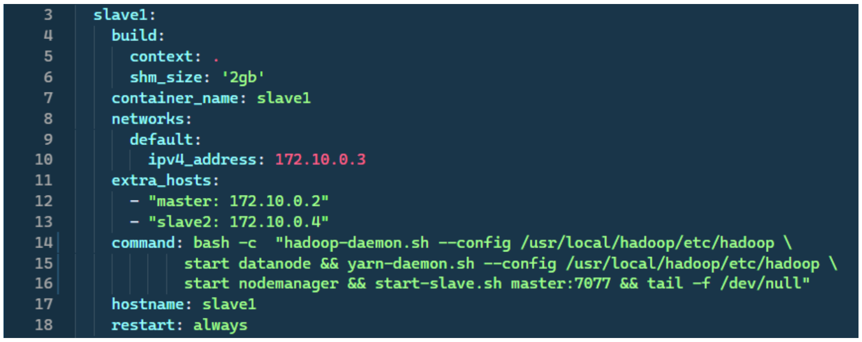
- Setup directory and configurations
  - We create working spaces for namenode and datanodes
  - Give these folders permission to read, change, delete and execute files with **chmod 777** mode
  - We add all files which are created in Step 3 are files in **configurations folder** to Hadoop configurations



```
# FORMAT NAMENODE
ARG FORMAT_NAMENODE_COMMAND
RUN $FORMAT_NAMENODE_COMMAND
EXPOSE 22
```

- Finally, run format namenode command
  - *hdfs namenode -format*

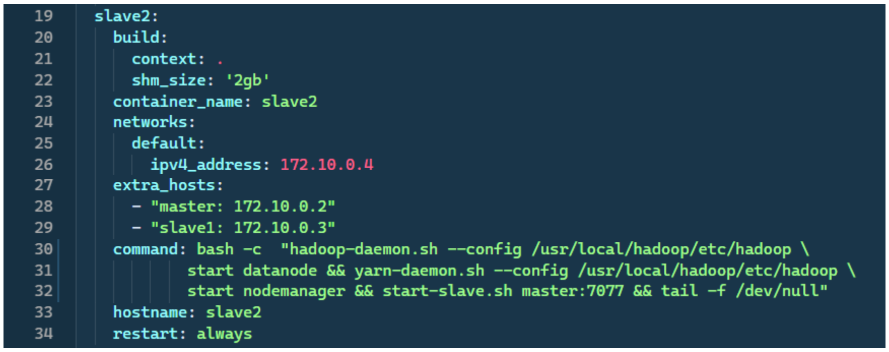
Second, **docker-compose.yml** file is used to configure our application's services. In this context, three containers are needed that are one master and two slaves. Here are the following steps:



```
  slave1:
    build:
      context: .
      shm_size: '2gb'
    container_name: slave1
    networks:
      default:
        ipv4_address: 172.10.0.3
    extra_hosts:
      - "master: 172.10.0.2"
      - "slave2: 172.10.0.4"
    command: bash -c  "hadoop-daemon.sh --config /usr/local/hadoop/etc/hadoop \
            start datanode && yarn-daemon.sh --config /usr/local/hadoop/etc/hadoop \
            start nodemanager && start-slave.sh master:7077 && tail -f /dev/null"
    hostname: slave1
    restart: always
```

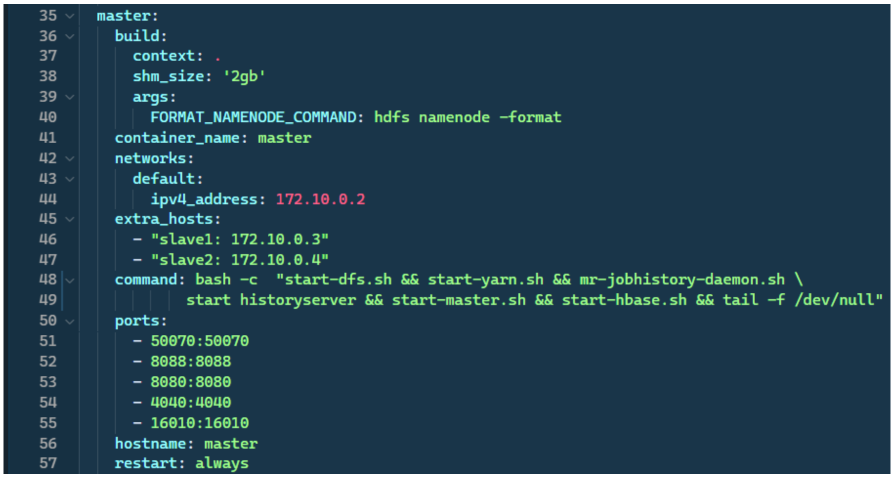
- Configure for slave1 (datanode1):
  - build
    - *context: .* : It's a path to the directory that contains **Dockerfile.** In our case, **Dockerfile** file exists in the current directory (in the same folder of the **docker-compose.yml** file).
    - *shm_size: '2gb'*: Set the size of the /dev/shm partition for this build's containers.
  - container_name: Define *slave1* for container's name
  - networks: Set virtual ip_address for containers to connect others
  - extra_hosts: Add hostname mappings at build-time. Here are our two other containers such as master and slave2

- command: Client command to start Yarn, Hadoop daemons but not namenode
- hostname: Hostname is that computers on the network know each other and thus communicate between themselves. For *container slave1,* we set hostname value is *slave1*
- restart: Always restart container if started fail



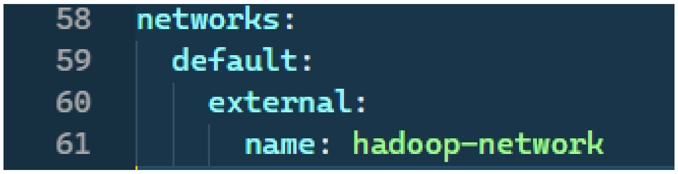
```
slave2:
  build:
    context: .
    shm_size: '2gb'
  container_name: slave2
  networks:
    default:
      ipv4_address: 172.10.0.4
  extra_hosts:
    - "master: 172.10.0.2"
    - "slave1: 172.10.0.3"
  command: bash -c  "hadoop-daemon.sh --config /usr/local/hadoop/etc/hadoop \
          start datanode && yarn-daemon.sh --config /usr/local/hadoop/etc/hadoop \
          start nodemanager && start-slave.sh master:7077 && tail -f /dev/null"
  hostname: slave2
  restart: always
```

- Configure for slave2 (datanode2): The same configurations as slave1 (datanode1)



```
master:
  build:
    context: .
    shm_size: '2gb'
    args:
      FORMAT_NAMENODE_COMMAND: hdfs namenode -format
  container_name: master
  networks:
    default:
      ipv4_address: 172.10.0.2
  extra_hosts:
    - "slave1: 172.10.0.3"
    - "slave2: 172.10.0.4"
  command: bash -c  "start-dfs.sh && start-yarn.sh && mr-jobhistory-daemon.sh \
          start historyserver && start-master.sh && start-hbase.sh && tail -f /dev/null"
  ports:
    - 50070:50070
    - 8088:8088
    - 8080:8080
    - 4040:4040
    - 16010:16010
  hostname: master
  restart: always
```

- Configure for master (namenode): some attributes are same configurations of slave
  - build
    - args: Specify arguments using for command in **Dockerfile**
      - *FORMAT_NAMENODE_COMMAND: hdfs namenode -format* command use to format namenode before starting Hadoop services for the first time
  - command: Client command to start Yarn, Hadoop daemons but not datanode
  - ports: Activates the container to listen for specified ports from the world outside of the docker and accessible world inside docker
    - Resource Manager WebUI port
    - Namenode WebUI port
    - Datanode port



```
networks:
  default:
    external:
      name: hadoop-network
```

- networks: Create network for allowing to connect all containers together

Step 5. Run Hadoop cluster on Docker environment

Build & Run project

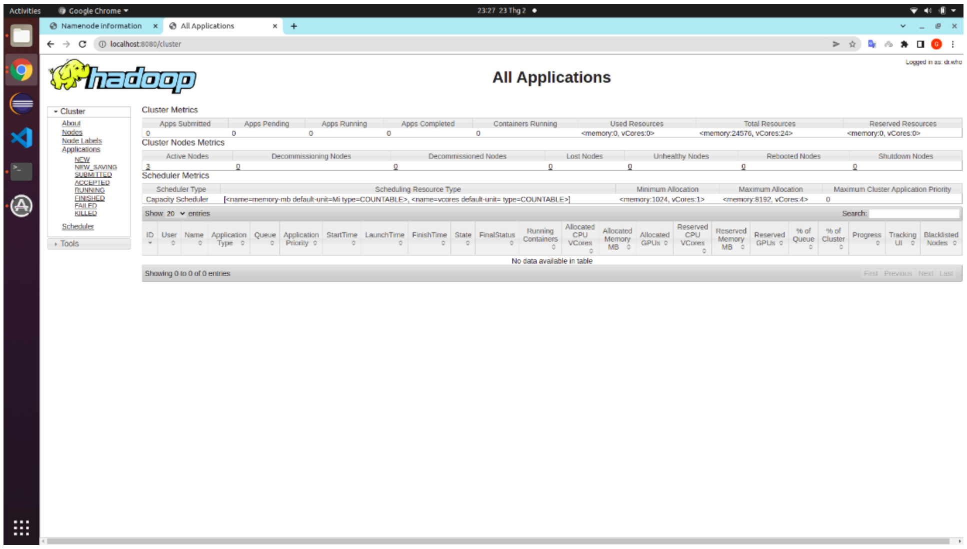
- Create subnets and bridge for *hadoop-network*
  - *docker network create --driver bridge hadoop-network --subnet=172.10.0.0/16*
- Build image and run our containers
  - *docker-compose up*
- Check whether containers are running which are daemons of Hadoop cluster
  - *docker container ls*

If want to end sessions, run

- *docker-compose down*

Manipulate with daemons

- Attach node of HDFS cluster
  - *docker exec -it master/slave1/slave2/... /bin/bash*
- Webapp daemons UI
  - For Resource Manager (YARN): http://localhost:8080/
  - For Master Management (Namenode): http://localhost:50070/



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources
0	0	0	0	0	<memory:0, vCores:0>	<memory:24576, vCores:24>	<memory:0, vCores:0>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	0	0	0	0

Scheduler Metrics

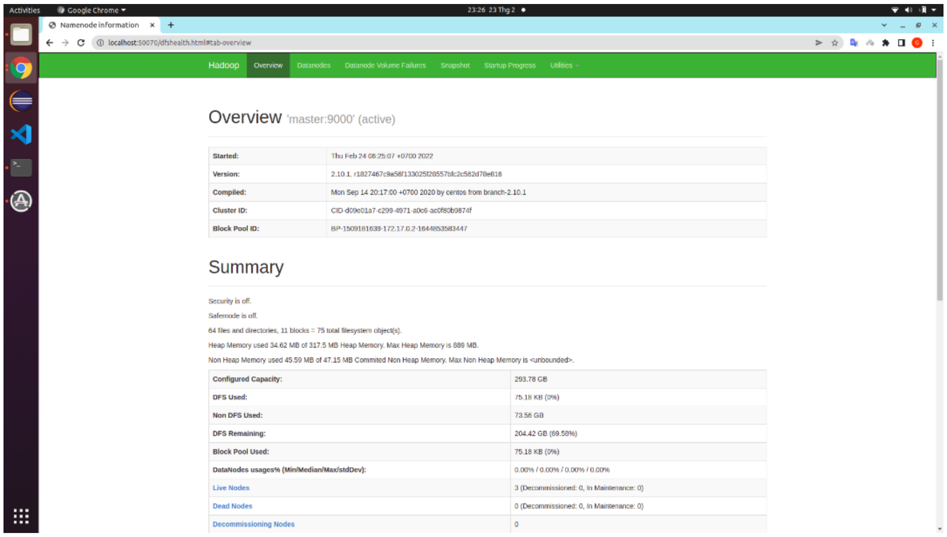
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																						

Showing 0 to 0 of 0 entries First Previous Next Last

Yarn Resource Manager



Overview 'master:9000' (active)

Started:	Thu Feb 24 06:25:07 +0700 2022
Version:	2.10.1, r1827467c9a56f133025f28557bfc2c562d78e816
Compiled:	Mon Sep 14 20:17:00 +0700 2020 by centos from branch-2.10.1
Cluster ID:	CID-d09e01a7-c299-4971-a0c6-ac0f80b9874f
Block Pool ID:	BP-1509181639-172.17.0.2-1644853583447

Summary

Security is off.

Safemode is off.

64 files and directories, 11 blocks = 75 total filesystem object(s).

Heap Memory used 34.62 MB of 317.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 45.59 MB of 47.15 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	293.78 GB
DFS Used:	75.18 KB (0%)
Non DFS Used:	73.56 GB
DFS Remaining:	204.42 GB (69.58%)
Block Pool Used:	75.18 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0

Namenode

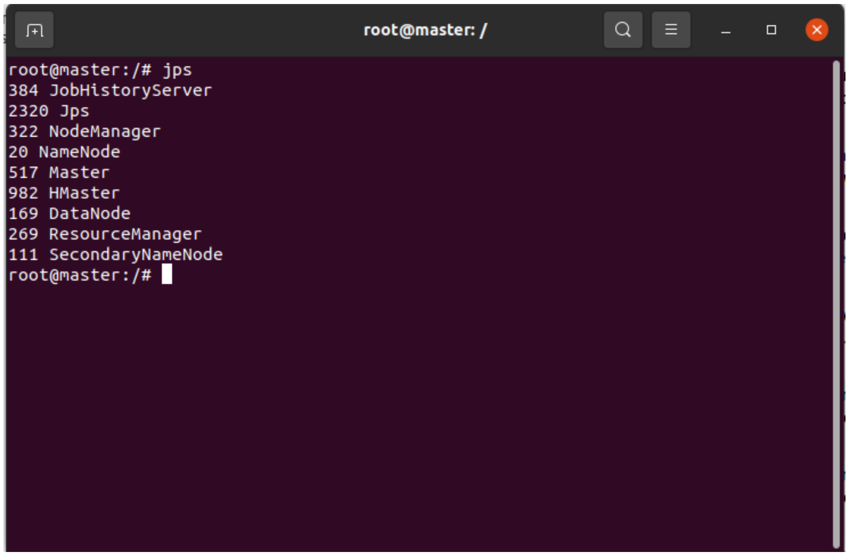
III. Demonstration functionalities

Section III focuses on demonstrating functionalities of Hadoop Distributed Filesystem.

This section has two parts:

- Hadoop command: To illustrate interaction between master and slaves in reading, writing files and other management functions of Hadoop system
- Framework operation: Writing Map-Reduce application that is running on Hadoop DFS

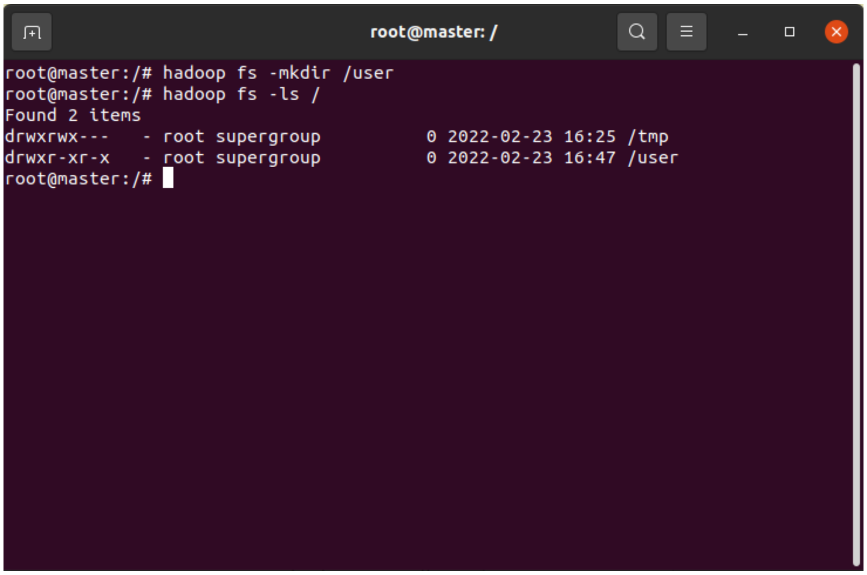
1. Hadoop commands



```
root@master:/# jps
384 JobHistoryServer
2320 Jps
322 NodeManager
20 NameNode
517 Master
982 HMaster
169 DataNode
269 ResourceManager
111 SecondaryNameNode
root@master:/# 
```

All the instrumental hotspots that the JVM is running in the system

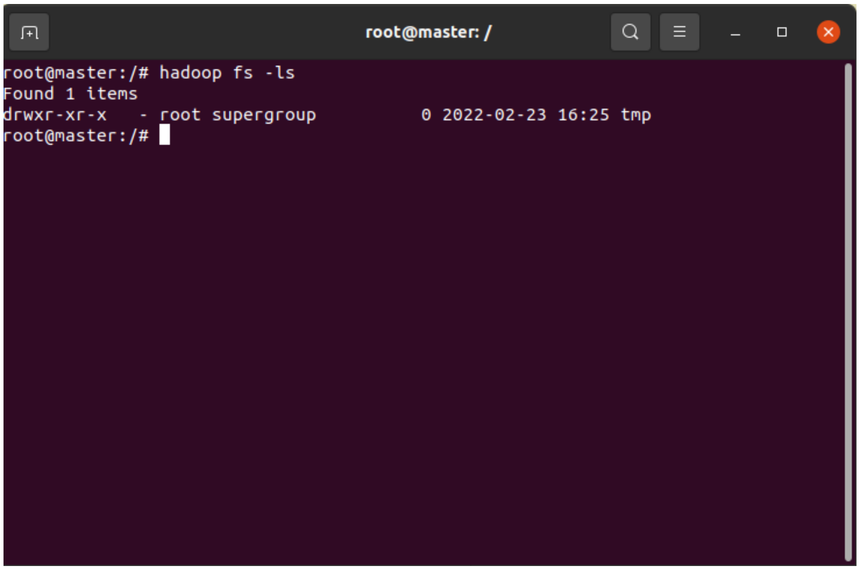
- jps*:* Java Virtual Machine Process Status Tool use to check all JVM process running in the system



```
root@master:/# hadoop fs -mkdir /user
root@master:/# hadoop fs -ls /
Found 2 items
drwxrwx---   - root supergroup          0 2022-02-23 16:25 /tmp
drwxr-xr-x   - root supergroup          0 2022-02-23 16:47 /user
root@master:/# 
```

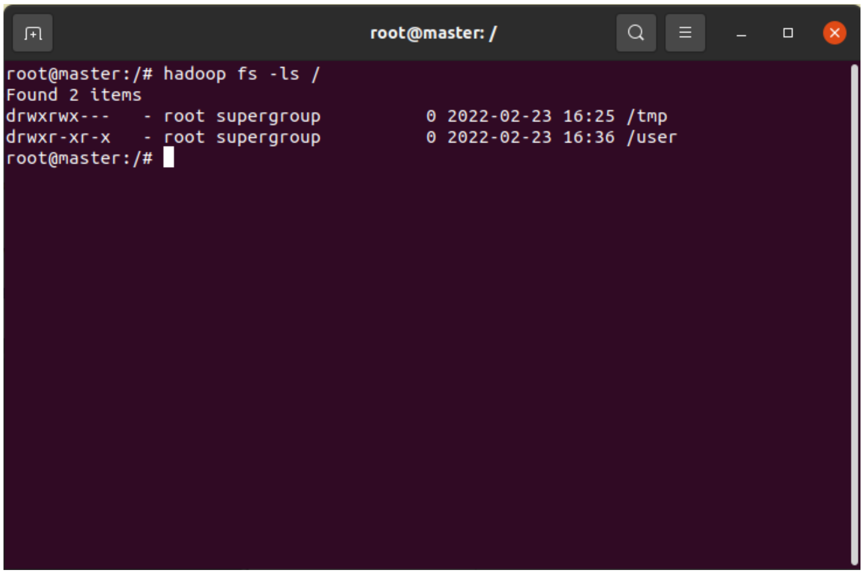
Create directory /user

- -mkdir: Command use to create directory in Hadoop DFS



```
root@master:/# hadoop fs -ls
Found 1 items
drwxr-xr-x   - root supergroup          0 2022-02-23 16:25 tmp
root@master:/# 
```

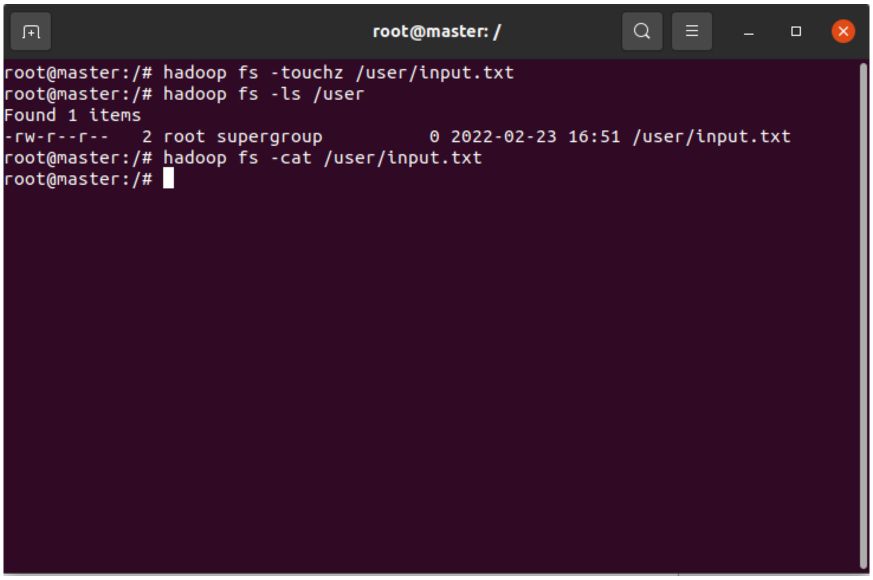
List file in home location



```
root@master:/# hadoop fs -ls /
Found 2 items
drwxrwx---   - root supergroup          0 2022-02-23 16:25 /tmp
drwxr-xr-x   - root supergroup          0 2022-02-23 16:36 /user
root@master:/# 
```

List files in root location

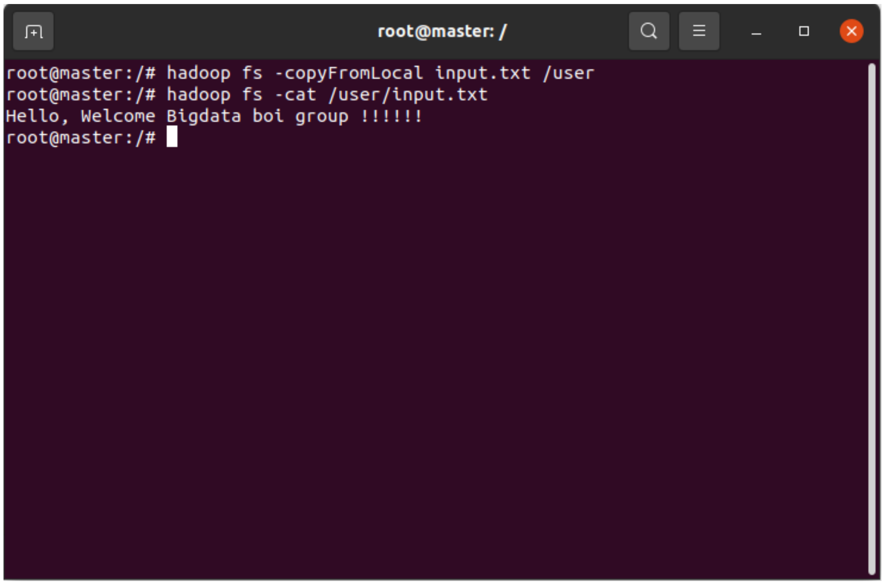
- -ls : List files in Hadoop DFS. This command has an optional parameter:
    - If no param: List files in home location
    - If / : List files in root location
    - If directory: List files in this directory



```
root@master:/# hadoop fs -touchz /user/input.txt
root@master:/# hadoop fs -ls /user
Found 1 items
-rw-r--r--   2 root supergroup          0 2022-02-23 16:51 /user/input.txt
root@master:/# hadoop fs -cat /user/input.txt
root@master:/# 
```

Touch empty file input.txt in folder user

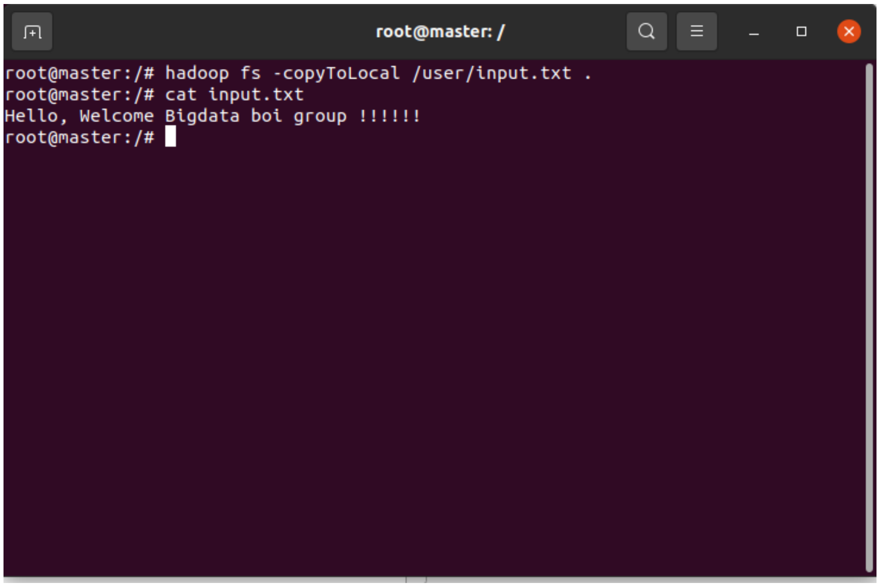
- -touchz: Create empty file in specific directory



```
root@master:/# hadoop fs -copyFromLocal input.txt /user
root@master:/# hadoop fs -cat /user/input.txt
Hello, Welcome Bigdata boi group !!!!!!
root@master:/#
```

Copy input.txt from master container to /user folder

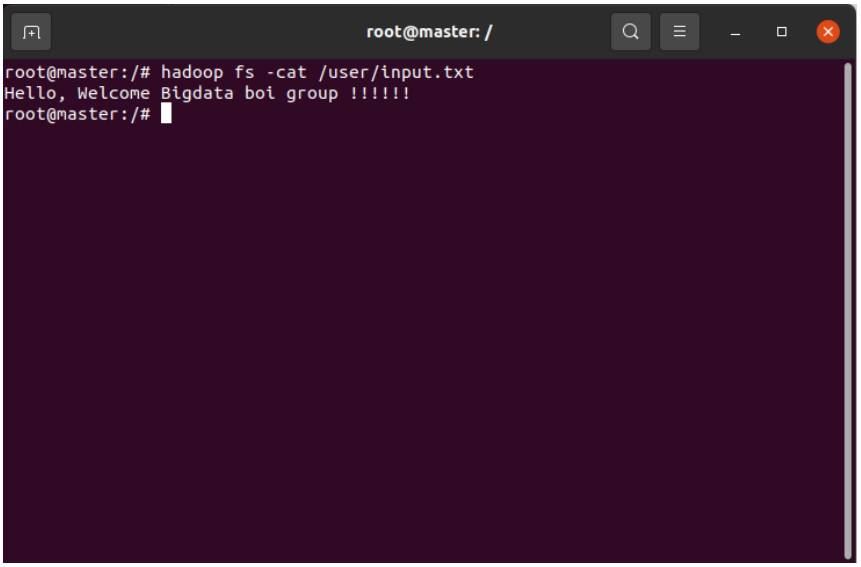
- -copyFromLocal: Copy file from local to destination in Hadoop DFS



```
root@master:/# hadoop fs -copyToLocal /user/input.txt .
root@master:/# cat input.txt
Hello, Welcome Bigdata boi group !!!!!!
root@master:/#
```

Copy file input.txt to local master container

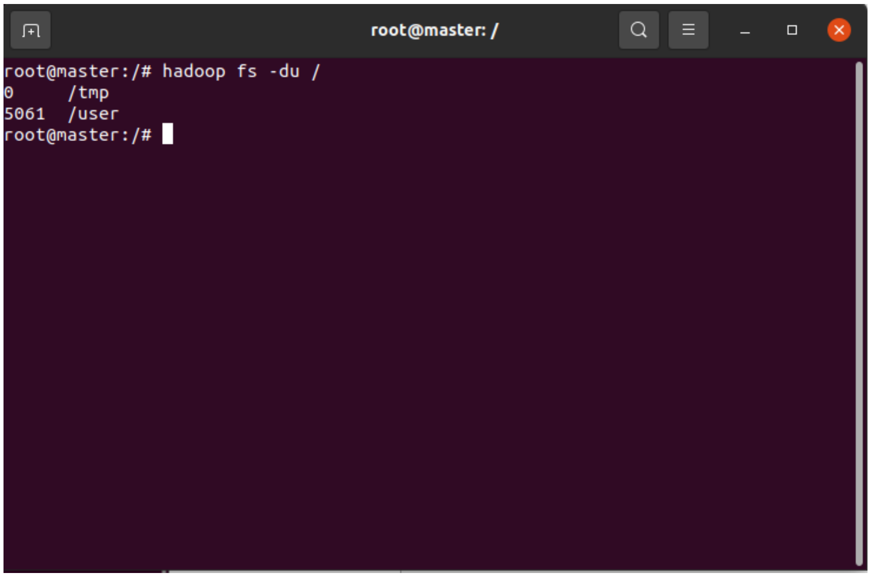
- -copyToLocal: Copy file from Hadoop DFS to local machine



```
root@master:/# hadoop fs -cat /user/input.txt
Hello, Welcome Bigdata boi group !!!!!!
root@master:/#
```

Concatenate input.txt

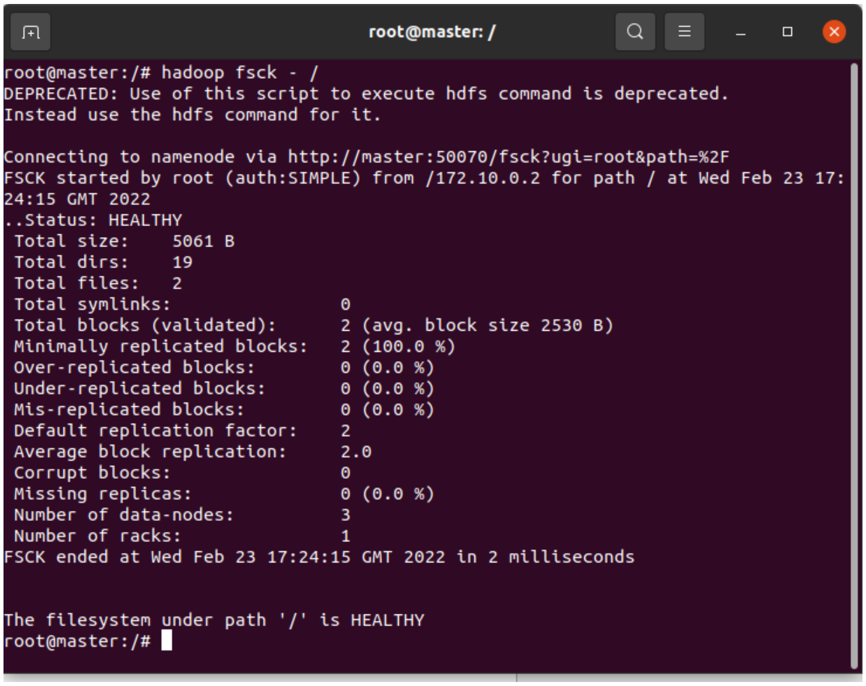
- -cat: Concatenate file (read) in location



```
root@master:/# hadoop fs -du /
0     /tmp
5061  /user
root@master:/#
```

Disk usage of root location

- -du: Show total bytes of directories in specific location of Hadoop DFS



```
root@master:/# hadoop fsck - /
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Connecting to namenode via http://master:50070/fsck?ugi=root&path=%2F
FSCK started by root (auth:SIMPLE) from /172.10.0.2 for path / at Wed Feb 23 17:
24:15 GMT 2022
..Status: HEALTHY
 Total size:    5061 B
 Total dirs:    19
 Total files:   2
 Total symlinks:                0
 Total blocks (validated):      2 (avg. block size 2530 B)
 Minimally replicated blocks:   2 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     2.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          3
 Number of racks:               1
FSCK ended at Wed Feb 23 17:24:15 GMT 2022 in 2 milliseconds


The filesystem under path '/' is HEALTHY
root@master:/#
```

Healthy report root location

- fsck – directory: Used to check the health of directory in Hadoop DFS

2. Use Map-Reduce on Hadoop DFS with Pseudo-Distributed operation mode

Overview:

We take advantage of the distributed-computing framework which is designed to Map phase and Reduce phase calling MapReduce framework.

To demonstrate how MapReduce framework works on top distributed filesystem Hadoop, we play around with simple MapReduce application. The application is called **Sum Operation** that calculates summarization of even numbers and summarization of odd numbers in a very long sequence number stored in text file.

Requirements and versions:

- Linux system
- Java engine version 8
- Hadoop version 2.10.1

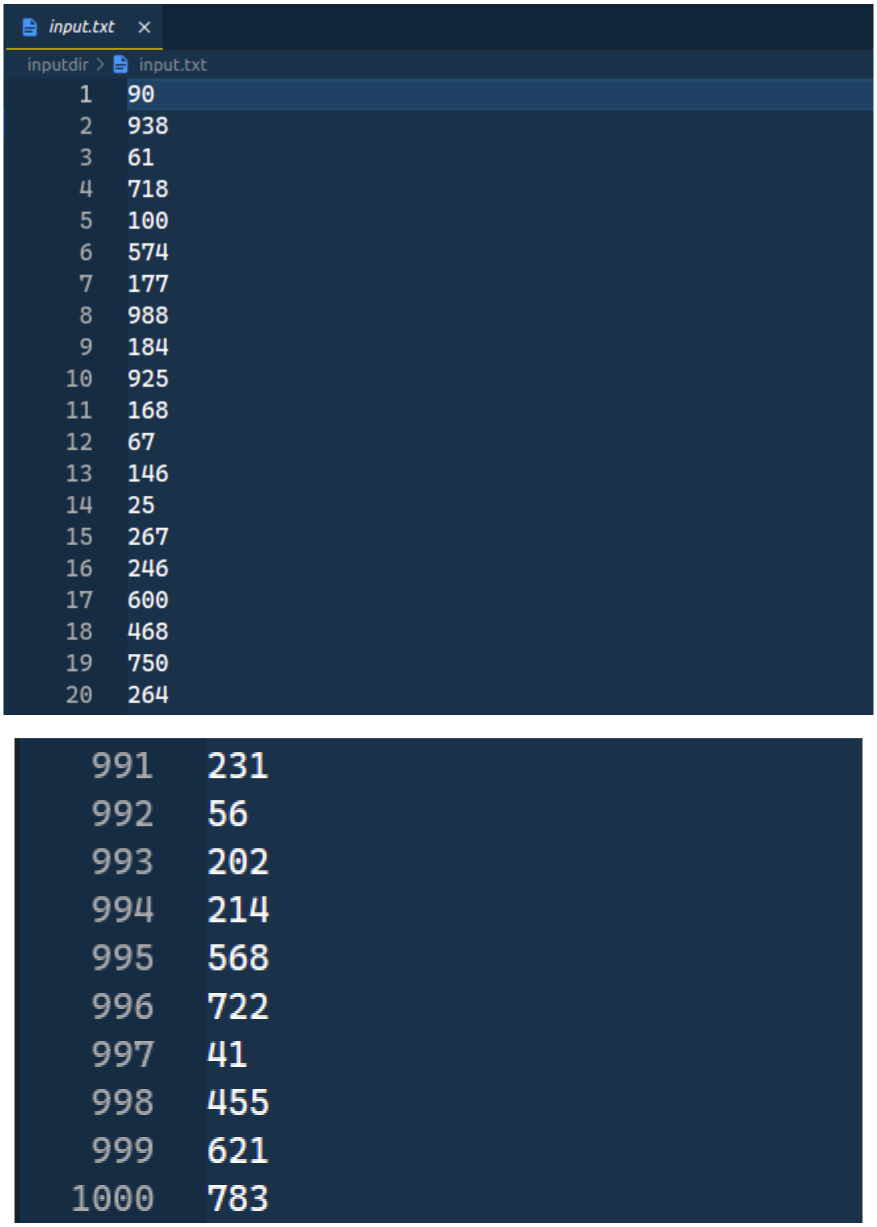
Follow step by step implementation:

Step 1. Start Hadoop on local:

- We need to start Hadoop on a local machine with Pseudo-Distributed mode which serves datanode, namenode and yarn daemons in one machine.

Step 2. MapReduce implementation

- To run a MapReduce application, we need a Java program for writing Map function and Reduce function to submit to Hadoop for creating Map task(s) and Reduce task(s)

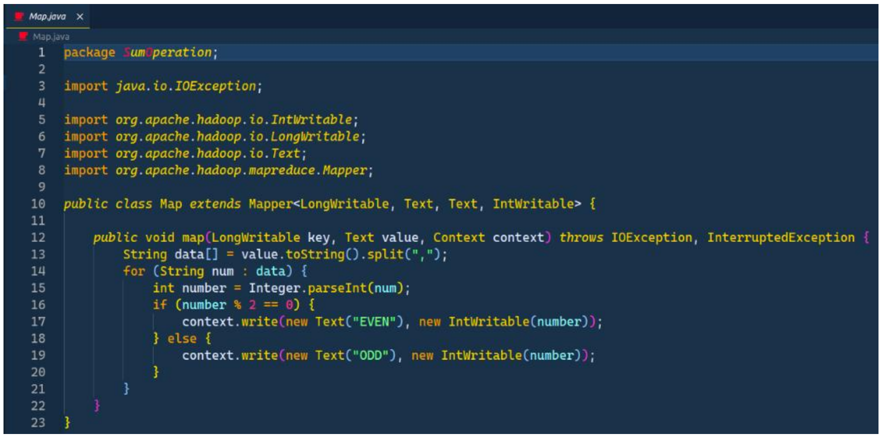


```
1     90
2     938
3     61
4     718
5     100
6     574
7     177
8     988
9     184
10    925
11    168
12    67
13    146
14    25
15    267
16    246
17    600
18    468
19    750
20    264
```

```
991   231
992   56
993   202
994   214
995   568
996   722
997   41
998   455
999   621
1000  783
```

Our input file

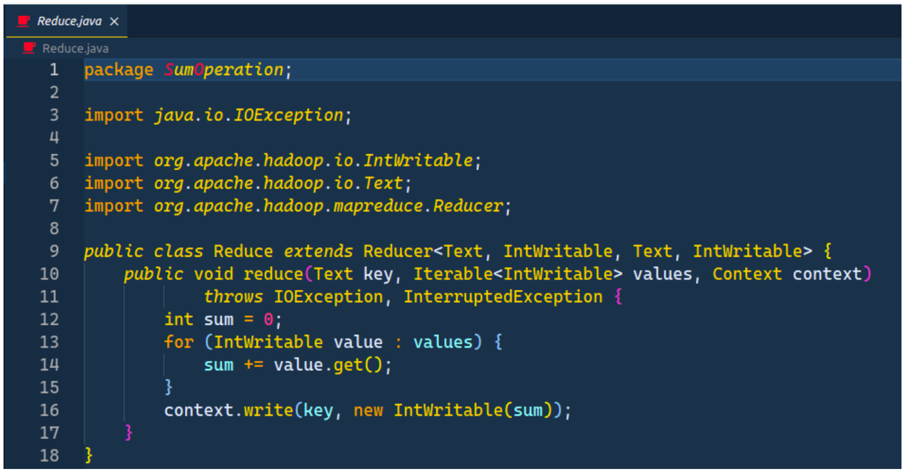
- Input file has a long sequence number. But to fit with the performance of our machine, input file contains 1000 rows, each row contains one number.



```java
package SumOperation;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

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

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String data[] = value.toString().split(",");
        for (String num : data) {
            int number = Integer.parseInt(num);
            if (number % 2 == 0) {
                context.write(new Text("EVEN"), new IntWritable(number));
            } else {
                context.write(new Text("ODD"), new IntWritable(number));
            }
        }
    }
}
```

Map function

- The strategy of map function is:
  - Convert number character to Hadoop internal integer type
  - If number % 2 == 0, label number key with *even*
  - Else, label number key with *odd*
  - We do that give each value has specific 'odd' or 'even' key and send these pairs to Reduce phase



```java
package SumOperation;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

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

Reduce function

- The strategy of reduce function is:
  - With each pair has the same key, we accumulate these value of pairs
  - Return result for 'odd' and 'even' key with values

Step 3. Build MapReduce application

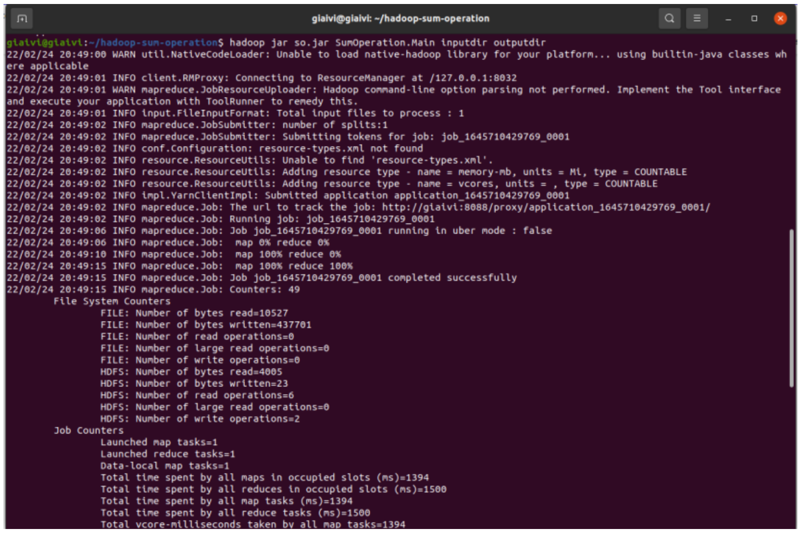
- Export Hadoop classpath
  - *export HADOOP_CLASSPATH=$(hadoop classpath)*
  - *echo $HADOOP_CLASSPATH*
- Create compiled directory
  - *mkdir container*
- Compile application to JARs packages
  - *javac -cp ${HADOOP_CLASSPATH}:container/:. -d container/ Map.java*

  - *javac -cp ${HADOOP_CLASSPATH}:container/:. -d container/ Reduce.java*
  - *javac -cp ${HADOOP_CLASSPATH}:container/:. -d container/ Main.java*
- Create project JARs file
  - jar -cvf so.jar -C container/ .

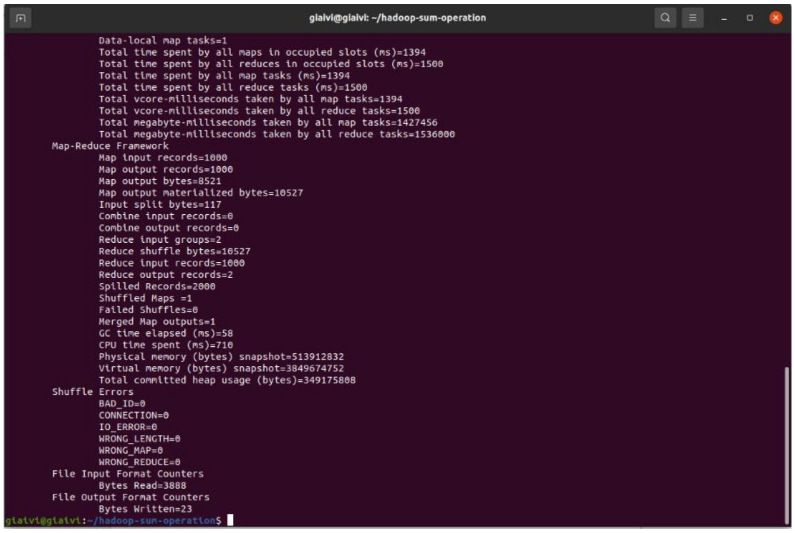
Step 4. Run MapReduce application

- Create Hadoop input directory for storing our data
  - *hadoop fs -mkdir inputdir*
- Copy our local data to Hadoop HDFS directory
  - *hadoop fs -put inputdir/* inputdir*
- Run MapReduce job in Hadoop
  - *hadoop jar so.jar SumOperation.Main inputdir outputdir*
- View MapReduce result
  - *hadoop fs -cat outputdir/part-r-00000*

Demonstration

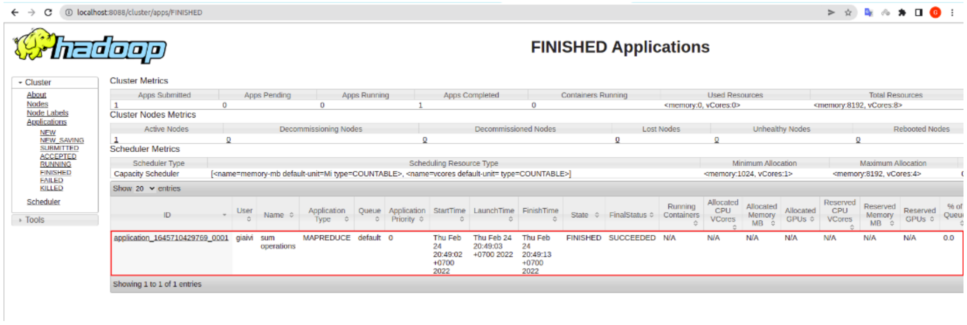


```
giaivi@giaivi:~/hadoop-sum-operation$ hadoop jar so.jar SumOperation.Main inputdir outputdir
22/02/24 20:49:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
22/02/24 20:49:01 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
22/02/24 20:49:01 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
and execute your application with ToolRunner to remedy this.
22/02/24 20:49:01 INFO input.FileInputFormat: Total input files to process : 1
22/02/24 20:49:02 INFO mapreduce.JobSubmitter: number of splits:1
22/02/24 20:49:02 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1645710429769_0001
22/02/24 20:49:02 INFO conf.Configuration: resource-types.xml not found
22/02/24 20:49:02 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/02/24 20:49:02 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
22/02/24 20:49:02 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
22/02/24 20:49:02 INFO impl.YarnClientImpl: Submitted application application_1645710429769_0001
22/02/24 20:49:02 INFO mapreduce.Job: The url to track the job: http://giaivi:8088/proxy/application_1645710429769_0001/
22/02/24 20:49:02 INFO mapreduce.Job: Running job: job_1645710429769_0001
22/02/24 20:49:06 INFO mapreduce.Job: Job job_1645710429769_0001 running in uber mode : false
22/02/24 20:49:06 INFO mapreduce.Job:  map 0% reduce 0%
22/02/24 20:49:10 INFO mapreduce.Job:  map 100% reduce 0%
22/02/24 20:49:15 INFO mapreduce.Job:  map 100% reduce 100%
22/02/24 20:49:15 INFO mapreduce.Job: Job job_1645710429769_0001 completed successfully
22/02/24 20:49:15 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=10527
                FILE: Number of bytes written=437701
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4005
                HDFS: Number of bytes written=23
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1394
                Total time spent by all reduces in occupied slots (ms)=1500
                Total time spent by all map tasks (ms)=1394
                Total time spent by all reduce tasks (ms)=1500
                Total vcore-milliseconds taken by all map tasks=1394
```



```
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1394
                Total time spent by all reduces in occupied slots (ms)=1500
                Total time spent by all map tasks (ms)=1394
                Total time spent by all reduce tasks (ms)=1500
                Total vcore-milliseconds taken by all map tasks=1394
                Total vcore-milliseconds taken by all reduce tasks=1500
                Total megabyte-milliseconds taken by all map tasks=1427456
                Total megabyte-milliseconds taken by all reduce tasks=1536000
        Map-Reduce Framework
                Map input records=1000
                Map output records=1000
                Map output bytes=8521
                Map output materialized bytes=10527
                Input split bytes=117
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=10527
                Reduce input records=1000
                Reduce output records=2
                Spilled Records=2000
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=58
                CPU time spent (ms)=710
                Physical memory (bytes) snapshot=513912832
                Virtual memory (bytes) snapshot=3849674752
                Total committed heap usage (bytes)=349175808
        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=3888
        File Output Format Counters
                Bytes Written=23
giaivi@giaivi:~/hadoop-sum-operation$
```

MapReduce Job report



FINISHED Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
1	0	0	1	0	<memory:0, vCores:0>	<memory:8192, vCores:8>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	0	0	0

Scheduler Metrics

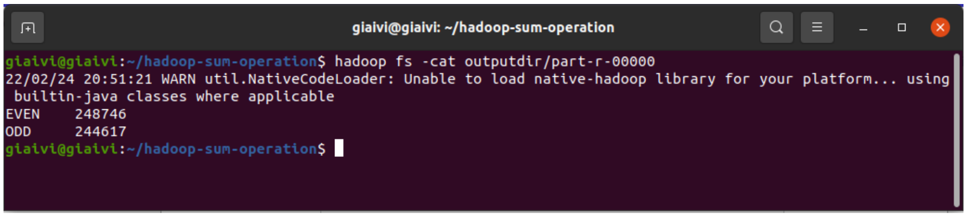
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queu
application_1645710429769_0001	giaivi	sum operations	MAPREDUCE	default	0	Thu Feb 24 20:49:02 +0700 2022	Thu Feb 24 20:49:03 +0700 2022	Thu Feb 24 20:49:13 +0700 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0

Showing 1 to 1 of 1 entries

Yarn Resource Manager manage MR Job



```
giaivi@giaivi:~/hadoop-sum-operation$ hadoop fs -cat outputdir/part-r-00000
22/02/24 20:51:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
EVEN    248746
ODD     244617
giaivi@giaivi:~/hadoop-sum-operation$ 
```

MapReduce application result

IV. Self-evaluation

LAB STATUS	Installation: • Install Pseudo-distributed mode (Single Node Cluster) on local machine: FINISHED • Install Fully-Distributed Mode (Multi-Node Cluster) on Docker environment: FINISHED Test run process: • Demo functionalities of Multi-Node Cluster in docker environments: FINISHED
PROBLEM	• Teamworking in remote • Unfamiliar with Docker container, network, … • The way connects each container together is complicated and even more difficult is how to configure Hadoop cluster on those containers • Some Hadoop ecosystem documents are complicated • Inconsistent OS working environment between members causes several unexpected errors • Installations is flaky, it sometimes causes error, however when reinstalling, its run normally
SOLUTION	• Take advantage of online working tools to guarantee communication between members • Research document from Docker website • Try many times as well as search for solutions on the internet, YouTube, tutorial, … • Research from many sources about Hadoop ecosystem • Use containerization platform (Docker) to guarantee Hadoop's configuration in team • Hold weekly meetings to review, evaluate each member's work or assist members who are in trouble

V. References

[1] https://hadoop.apache.org/

[2] https://www.cloudera.com/products/open-source/apache-hadoop.html

[3] https://www.docker.com/get-started

[4] https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

[5] https://www.geeksforgeeks.org/hdfs-commands/

[6] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[7] https://www.baeldung.com/find-java-home