

# Setting Up Hadoop Environment

## What is Big Data ?

Bernard Marr, defines Big Data as the digital trace that we are generating in this digital era. The Digital trace is made up of all the data that is captured when we use digital technology.

Ernst Young, refers Big Data to the dynamic large and disparate volume of data being created by people, tools and machines.



## V's of Big Data

- **Velocity**-> Speed of data at which data accumulates.
- **Volume**-> Scale of data or increase in amount of data store.
- **Variety**-> Diversity of data means Structured, Unstructured.
- **Veracity**-> Conformity to facts and accuracy.



## Characteristics of Big Data

1. Recommendation Engines are a common application of big data.
2. Companies like Amazon, Netflix use the Algorithm of Big Data.
3. SIRI personal assistants of Apple devices use Big Data to answer the infinite number of questions.
4. Google makes recommendations based on Big Data.

## What is Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

## Features of Hadoop

- Hadoop is Open Source
- Hadoop cluster is Highly Scalable
- Hadoop provides Fault Tolerance
- Hadoop provides High Availability
- Hadoop is very Cost-Effective
- Hadoop is Faster in Data Processing
- Hadoop is based on the Data Locality concept
- Hadoop provides Feasibility

## KO1: Able to install Hadoop

1.Hadoop is running using Java so first download java if it is not installed using the below link.

[Download Java](#)

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	59.1 MB	<a href="#"> jdk-8u291-linux-aarch64.rpm</a>
Linux ARM 64 Compressed Archive	70.79 MB	<a href="#"> jdk-8u291-linux-aarch64.tar.gz</a>
Linux ARM 32 Hard Float ABI	73.5 MB	<a href="#"> jdk-8u291-linux-arm32-vfp-hflt.tar.gz</a>
Linux x86 RPM Package	109.05 MB	<a href="#"> jdk-8u291-linux-i586.rpm</a>
Linux x86 Compressed Archive	137.92 MB	<a href="#"> jdk-8u291-linux-i586.tar.gz</a>
Linux x64 RPM Package	108.78 MB	<a href="#"> jdk-8u291-linux-x64.rpm</a>
Linux x64 Compressed Archive	138.22 MB	<a href="#"> jdk-8u291-linux-x64.tar.gz</a>
macOS x64	207.42 MB	<a href="#"> jdk-8u291-macosx-x64.dmg</a>
Solaris SPARC A11-bit (SVR4 package)	122.40 MB	<a href="#"> jdk-8u291-solaris-sparcv9.tar.Z</a>

## 2.Download Hadoop from apache.org

https://www.oracle.com/in/java/technologies/javase-jdk8-downloads.html#license-lightbox

We suggest the following mirror site for your download:

<https://downloads.apache.org/hadoop/common/>

Other mirror sites 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).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA\* etc) -- or if no other mirrors are working.

## HTTP

<https://downloads.apache.org/hadoop/common/>

## BACKUP SITES

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA\* etc) -- or if no other mirrors are working.

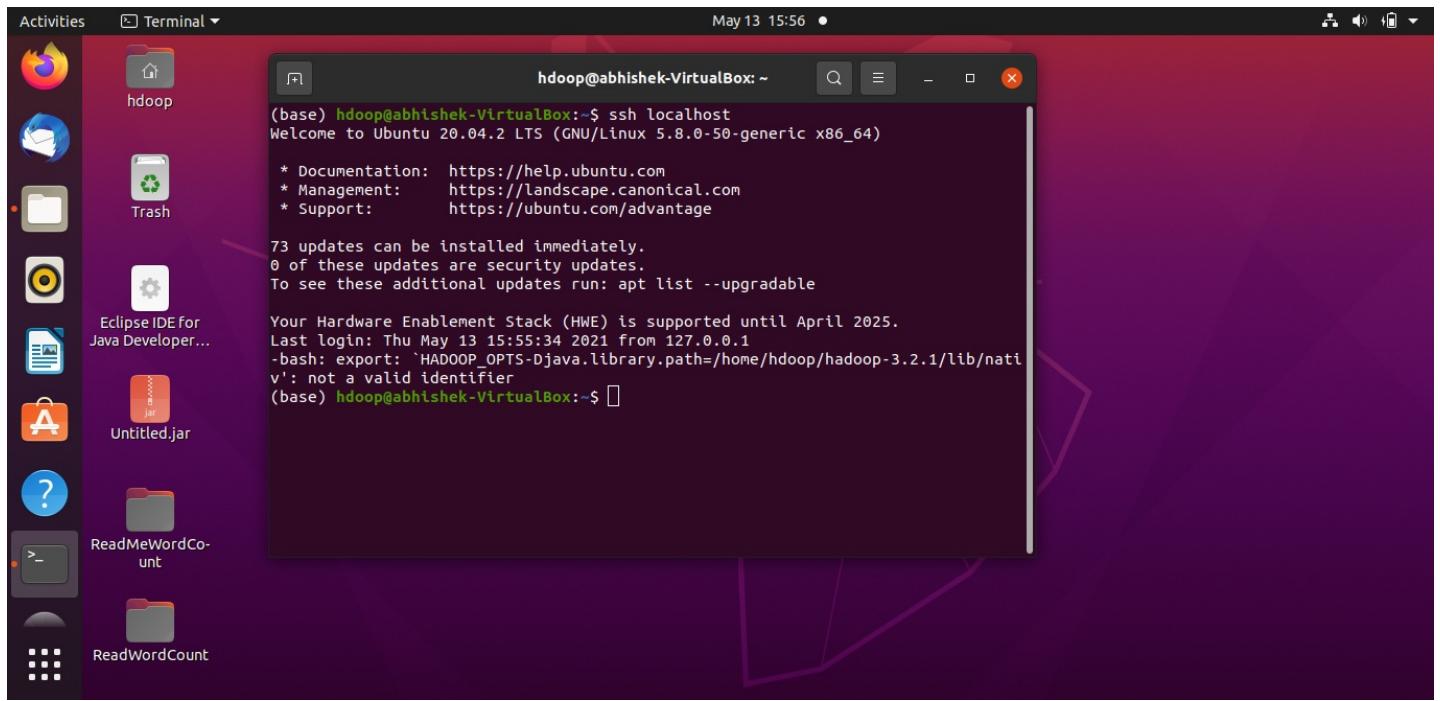
<https://downloads.apache.org/hadoop/common/>

The full listing of mirror sites is also available.

## BECOMING A MIRROR

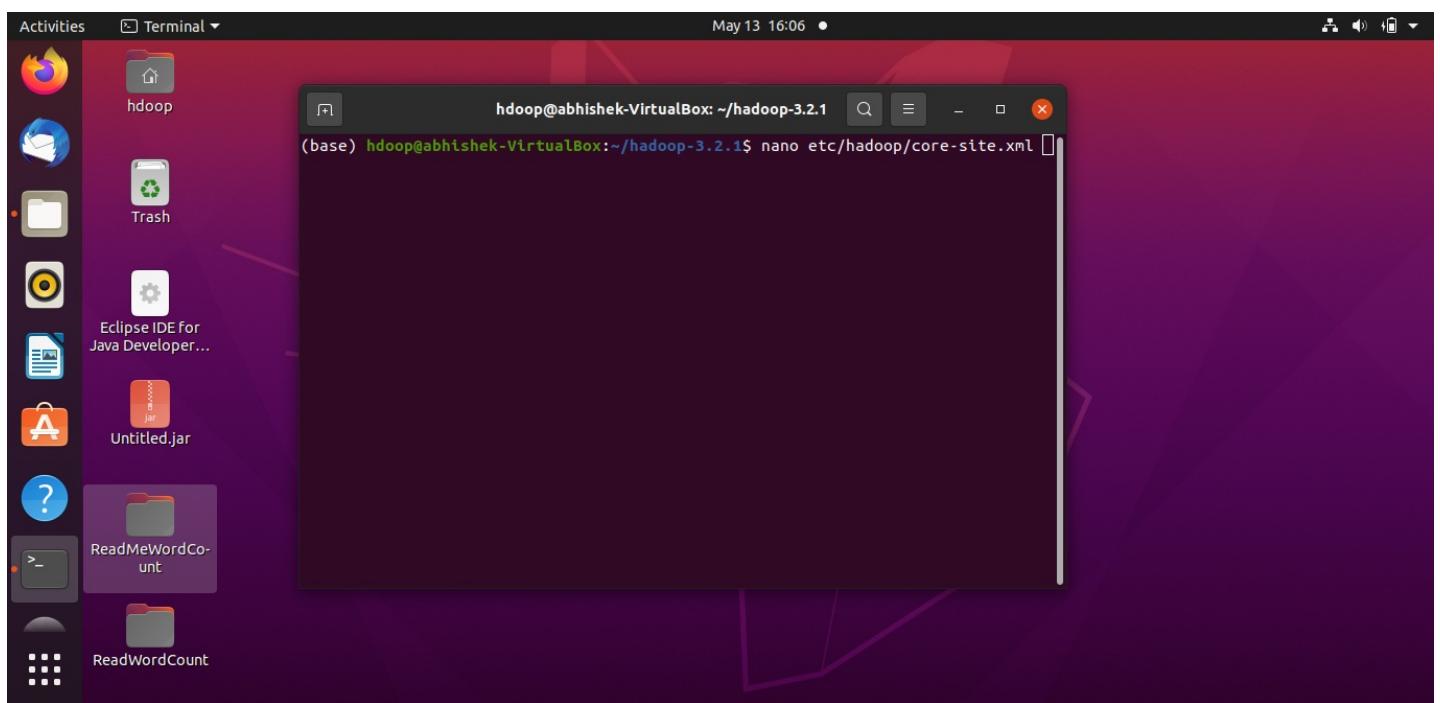
https://downloads.apache.org/hadoop/common/

3.Check the localhost connection .



4.Unzip the downloaded Hadoop file. The Unzipped folder contains many subfolders.Make the changes in the etc folder.

5.Open the file core-site.xml



Add two properties in this file.

- temporary directory
- default file system

Activities Terminal May 13 16:06

```
GNU nano 4.8      etc/hadoop/core-site.xml
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>/home/hadoop/tmpdata</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
<description>The name of the default file system</description>
</property>
</configuration>
```

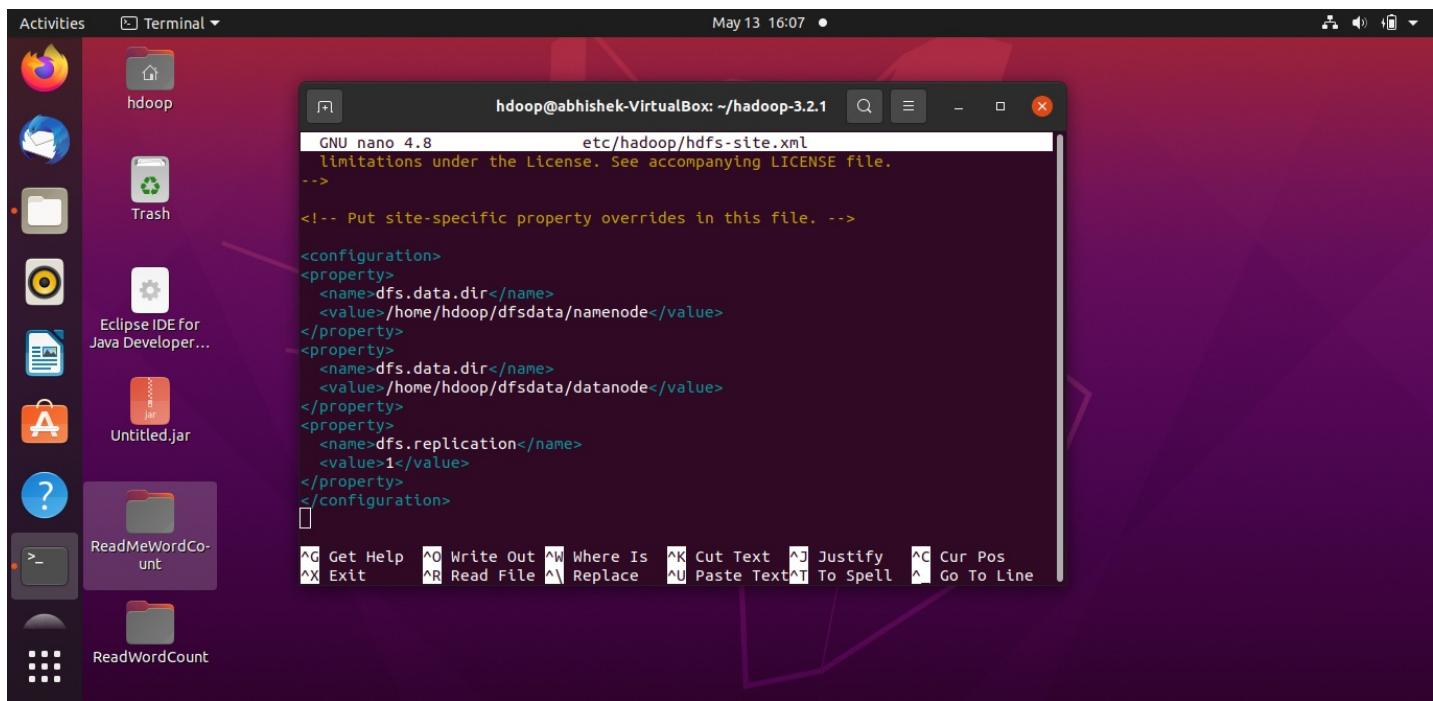
7. Now open the file `hdfs-site.xml`.

Activities Terminal May 13 16:07

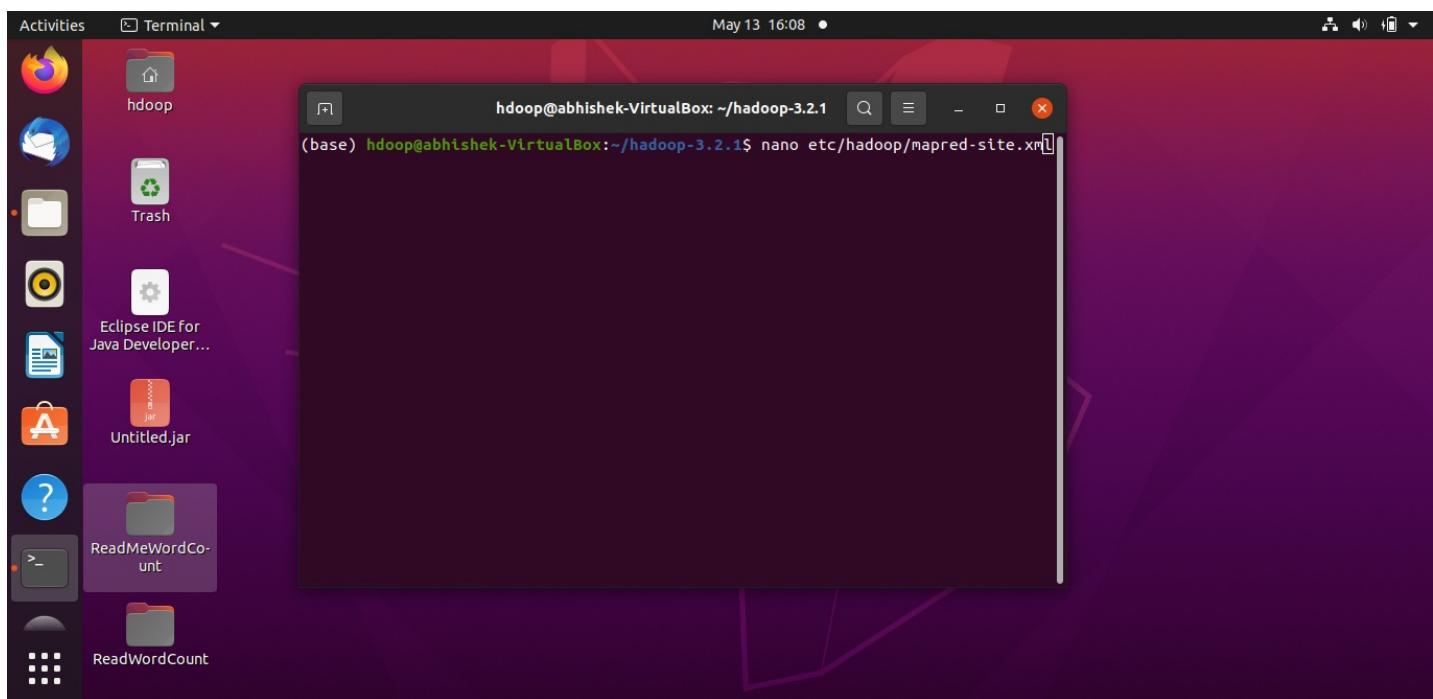
```
(base) hdoop@abhishek-VirtualBox:~/hadoop-3.2.1$ nano etc/hadoop/hdfs-site.xml
```

Add three properties in this file.

- datanode
- namenode
- duplicated values



8.Now open the file mapred-site.xml.



Add one properties in this file.

- yarnframework

Activities Terminal May 13 16:08

```
GNU nano 4.8          etc/hadoop/mapred-site.xml
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>
```

9. Now open the file `yarn-site.xml`.

Activities Terminal May 13 16:09

```
(base) hdoop@abhishek-VirtualBox:~/hadoop-3.2.1$ nano etc/hadoop/yarn-site.xml
```

Add five properties in this file.

- services
- class
- hostname
- enable acl
- add whitelist

Activities Terminal May 13 16:10

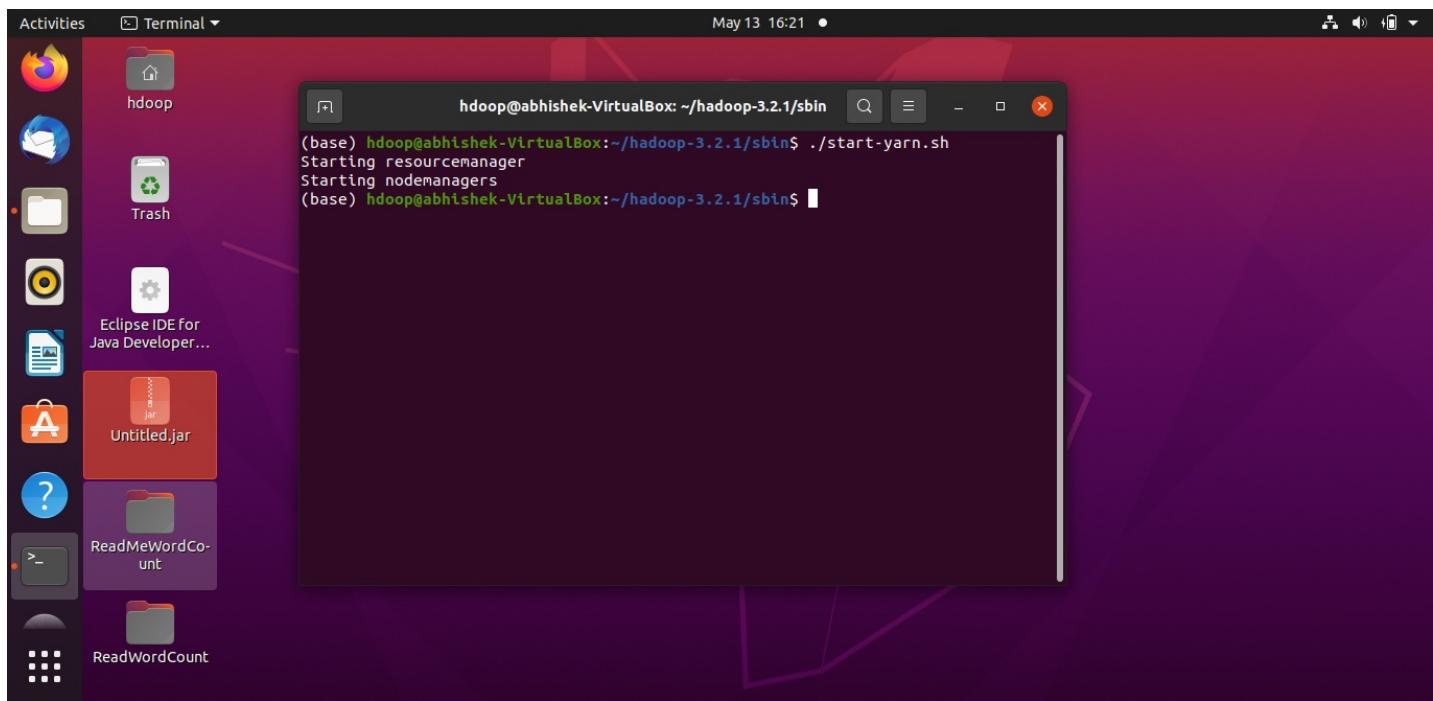
```
GNU nano 4.8          etc/hadoop/yarn-site.xml
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</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$JAVA_HOME</value>
</property>
```

10.Start the dfs services

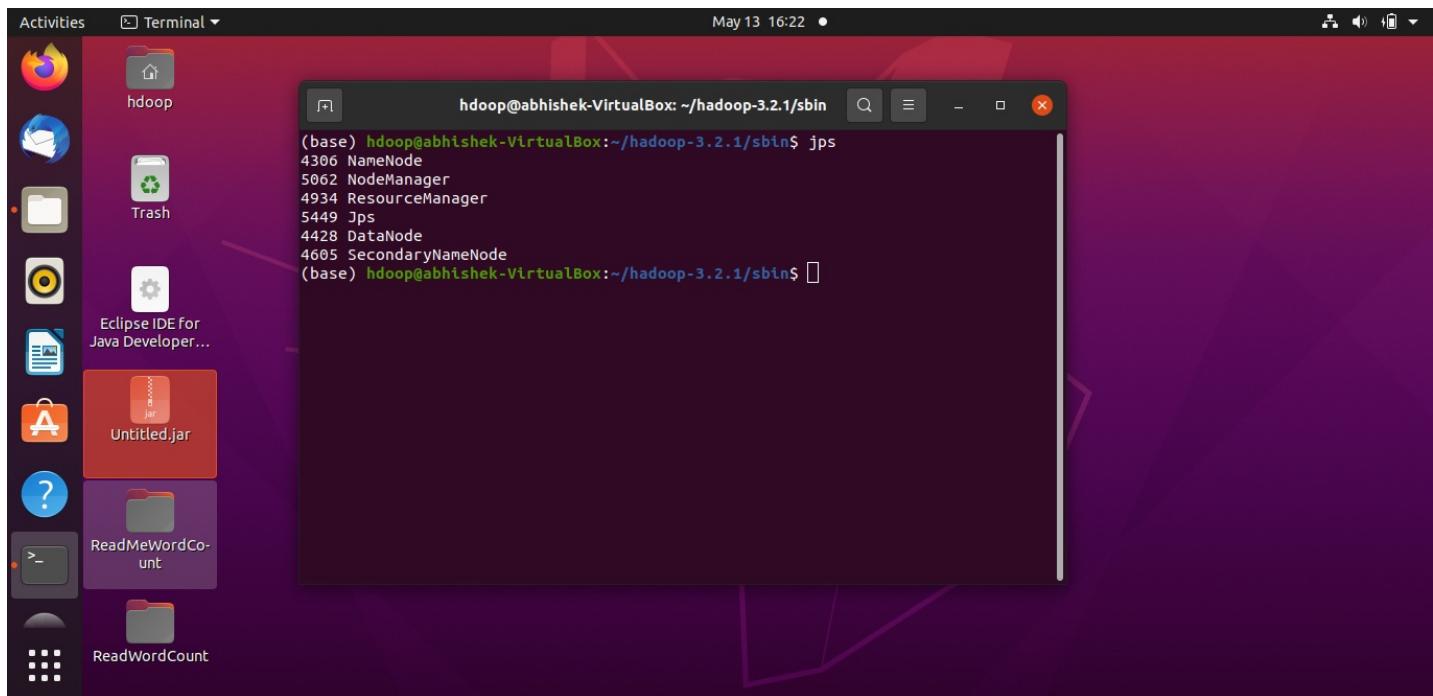
Activities Terminal May 13 16:16

```
(base) hdoop@abhishek-VirtualBox:~/hadoop-3.2.1$ cd sbin
(base) hdoop@abhishek-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [abhishek-VirtualBox]
2021-05-13 16:16:23,633 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(base) hdoop@abhishek-VirtualBox:~/hadoop-3.2.1/sbin$
```

11.Start the yarn services



12.Check that all the services of dfs and yarn is started by running **jps** command.



- All the services has started.Hadoop is successfully installed we are good to go further.

## KO2: Able to store data in HDFS

### What is HDFS

Hadoop Distributed file system is used to store and retrieve files. \* HDFS is built on commodity hardware \* Highly fault tolerant,hardware failure is the norm. \* Suited to Batch processing data access has high throughput rather than low latency. \* Supports very large data sets.

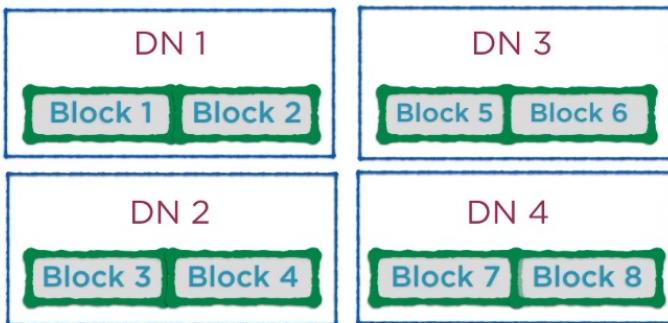
### Master And Slave Nodes

There are many nodes and hdfs choose among all a **Master** and all other **Slave**. Namenode is the master node which contains all metadata of all other slave nodes. Datanode is the node which physically stores the data.

### Storing File in HDFS

If there is a very large file say in Petabytes then file will split into blocks and these blocks are stored at different nodes. The size of each block is same 128 MB.

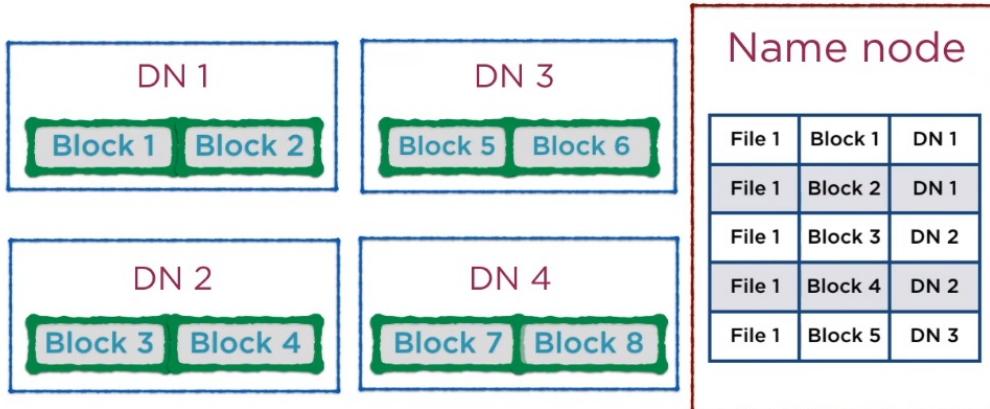
## Storing a File in HDFS



Each node contains a partition or a split of data

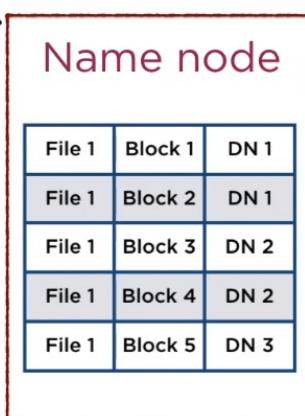
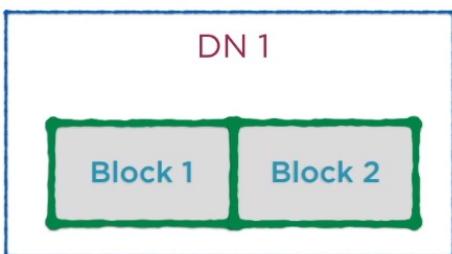


## Storing a File in HDFS

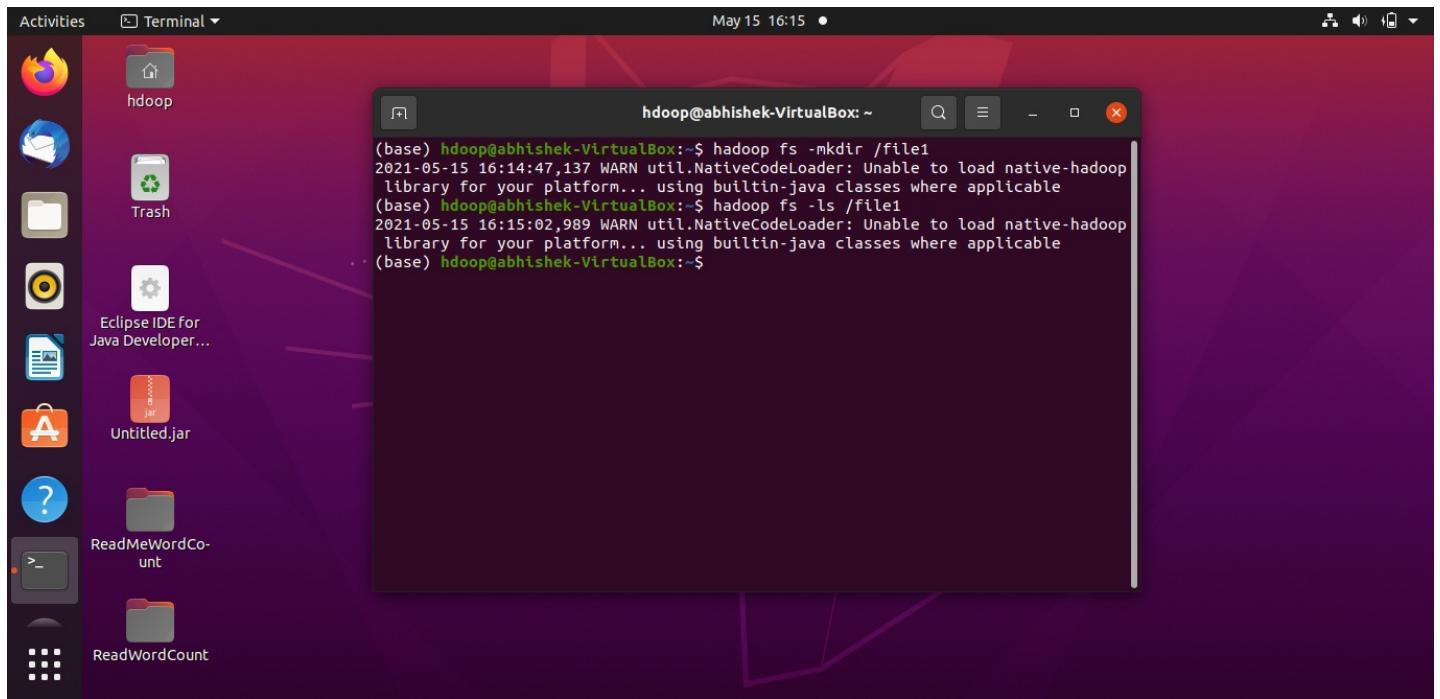


## Reading a File in HDFS

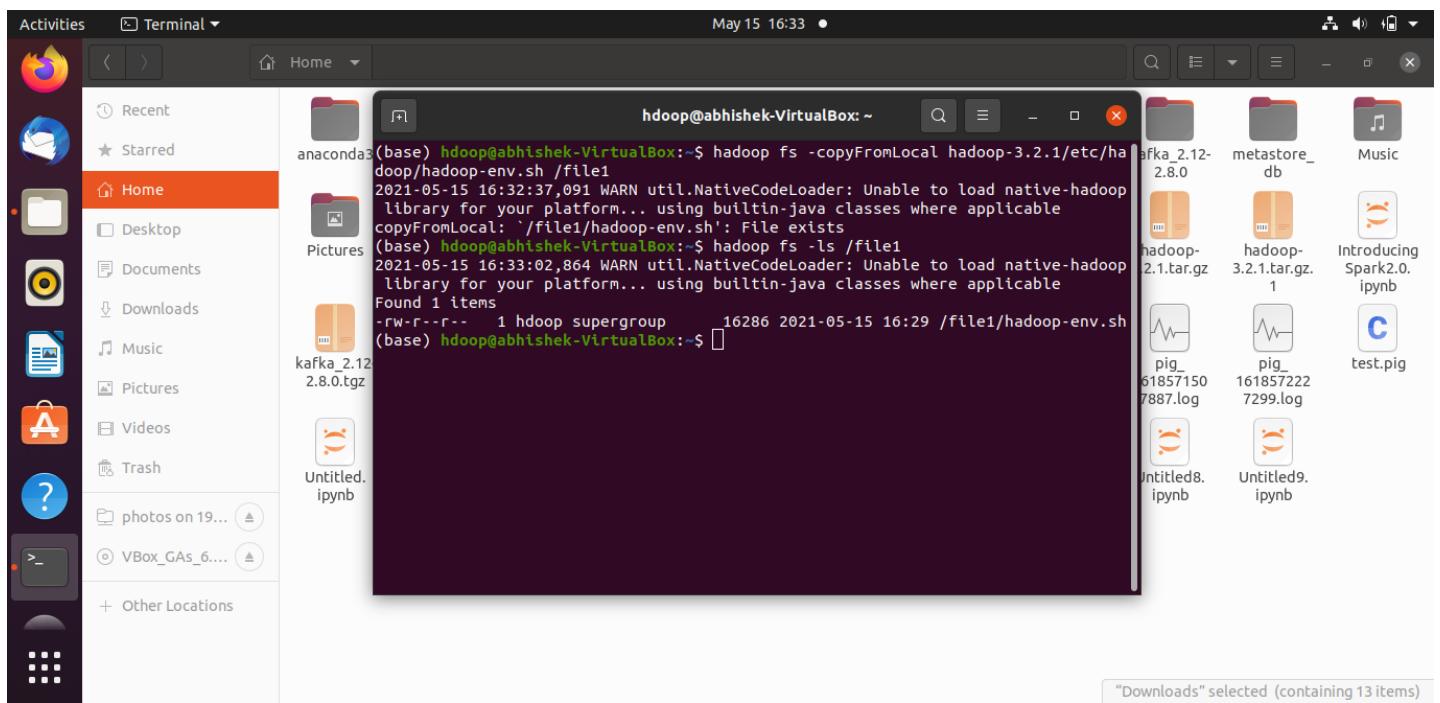
Request to the name node



Step-1) Make the directory



step-2) copy file from local to new directory



Another way to copy file

Activities Terminal May 15 16:36

```
hadoop@abhishek-VirtualBox:~$ hadoop fs -put hadoop-3.2.1/etc/hadoop/core-site.xml /file1
2021-05-15 16:35:42,534 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
put: '/file1/core-site.xml': File exists
(base) hadoop@abhishek-VirtualBox:~$ hadoop fs -ls /file1
2021-05-15 16:35:53,324 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 hadoop supergroup 1140 2021-05-15 16:35 /file1/core-site.xml
-rw-r--r-- 1 hadoop supergroup 16286 2021-05-15 16:29 /file1/hadoop-env.sh
(base) hadoop@abhishek-VirtualBox:~$ 
```

"Downloads" selected (containing 13 items)

step-3) copy one file into new file

Activities Terminal May 15 16:39

```
hadoop@abhishek-VirtualBox:~$ hadoop fs -cp /file1/* /file-dst/
2021-05-15 16:39:02,686 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-05-15 16:39:05,581 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-15 16:39:05,959 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-15 16:39:06,100 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
(base) hadoop@abhishek-VirtualBox:~$ 
```

"Downloads" selected (containing 13 items)

step-4) Get file from HDFS

A screenshot of a Linux desktop environment. On the left is a dock with icons for various applications like a browser, terminal, file manager, and system tools. A terminal window titled 'Screenshot from 2021-05-13 15-56-38.000' is open, showing Hadoop command-line interface (CLI) output:

```
hadoop@abhishek-VirtualBox:~$ hadoop fs -get /file1/ newfile
2021-05-15 16:44:40,266 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
get: 'newfile/file1/core-site.xml': File exists
get: 'newfile/file1/hadoop-env.sh': File exists
(base) hadoop@abhishek-VirtualBox:~$ ls newfile
file1
(base) hadoop@abhishek-VirtualBox:~$
```

## Fault Tolerance of Datanode

### Managing Failures in Data Nodes



Define a  
replication factor



### Replication

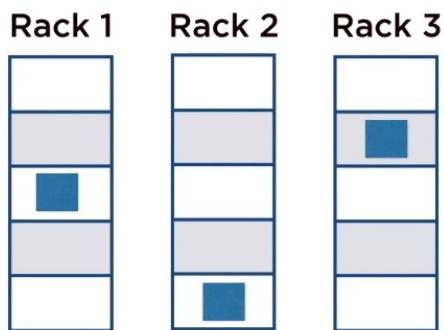
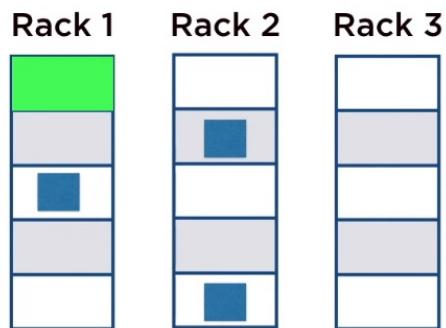
The replica locations are also stored in the name node

Name node

File 1	Block 1	DN 1
File 1	Block 2	DN 1
File 1	Block 3	DN 2
File 1	Block 4	DN 2
File 1	Block 5	DN 3
File 1	Block 1	DN 2



## Default Hadoop Replication Strategy



Maximize redundancy

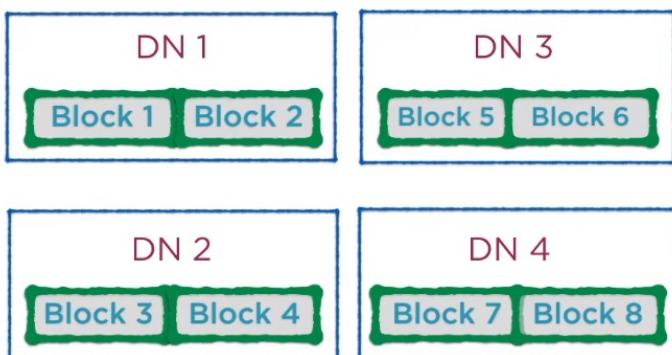
Store replicas “far away”  
i.e. on different nodes



## Fault Tolerance of Namenode

Namenode is the heart of the HDFS.

## Name Node Failures



This data is  
worthless  
without the  
name node



# fsimage edits

Metadata Files

Two files that store  
the filesystem  
metadata



2. Secondary Namenode

Secondary Name Node

Name Node  
**fsimage edits**

Secondary Name Node  
**fsimage edits**



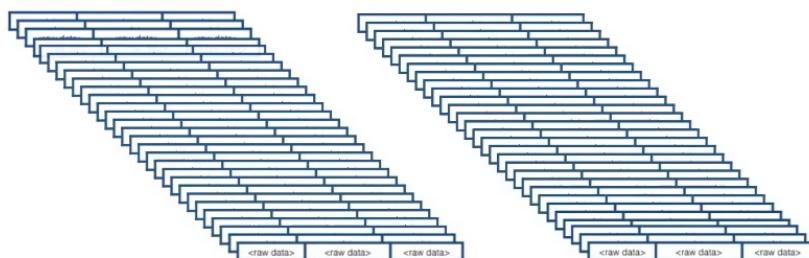
## KO3: Able to Process data with MapReduce

### 1.What is MapReduce

- It is a programming paradigm that follows Distributed Programming Approach.
- In today's world a lot of data is generated every minute. so, mapreduce process data in two phases.
  1. **Map** Runs on multiple nodes of the clusters. So there are multiple Map.
  2. **Reduce** Takes the output of the map and further process it to give final result.

So, for developer there are only two operations to be performed. Rest of the work is done by the Hadoop behind the scenes.

MapReduce



A task of this scale is processed in  
two stages

map

reduce



**map** A step that can be performed in parallel

**reduce** A step to combine the intermediate results



## 2. Data Flow in MapReduce

- MapReduce count the number of words from the file in petabytes.

### Counting Word Frequencies

Consider a large text file

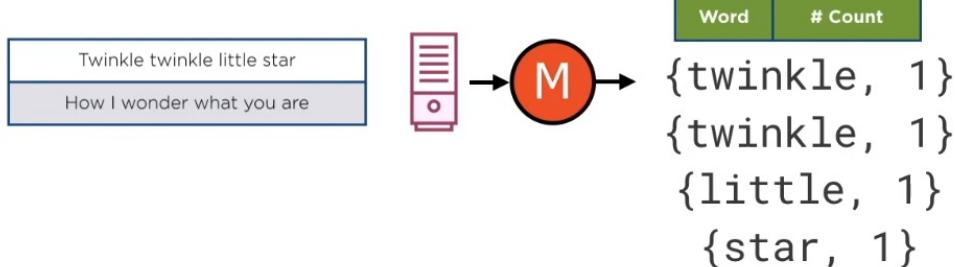
Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle twinkle little star
How I wonder what you are
.....

Word	Frequency
above	14
are	20
how	21
star	22
twinkle	32
...	..



- In Map flow each row emits {Key,Value} pairs.

### Map Flow

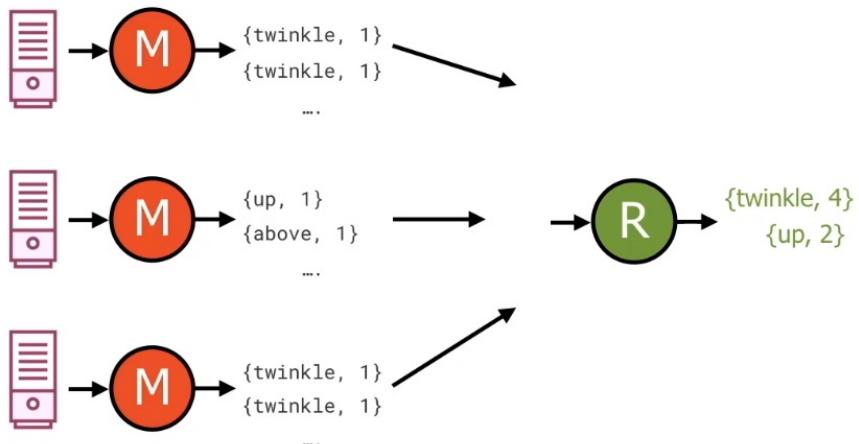


Each row emits {key, value} pairs

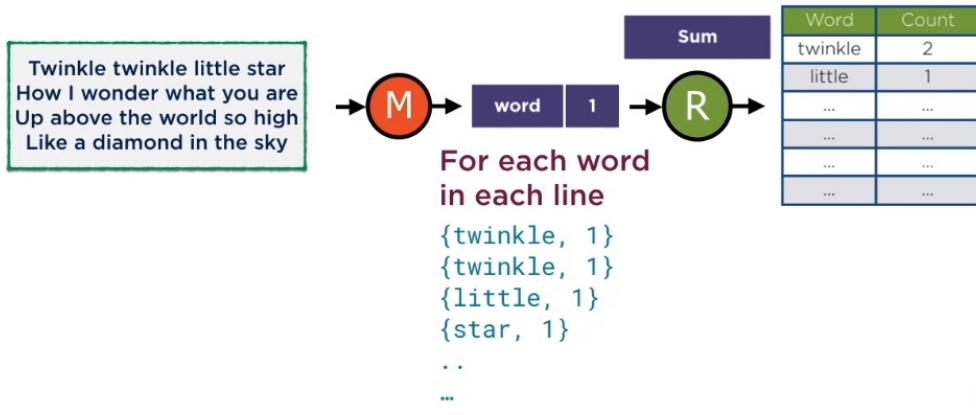


- In Reduce flow it produces the meaningful result.

## Reduce Flow



## Counting Word Frequencies



### 3. Implement MapReduce in java

- It uses the Map and Reduce class where logic is implemented.

## Implementing in Java



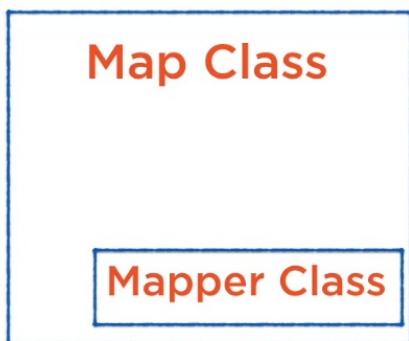
A class where the map logic is implemented

A class where the reduce logic is implemented

A driver program that sets up the job

- Map Step

## Map Step



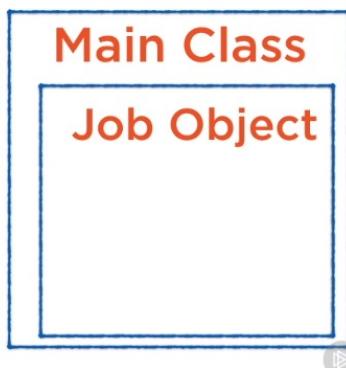
The map logic is implemented in a class that extends the Mapper Class



- Setting up the Job

## Setting up the Job

The Mapper and Reducer classes are used by a Job that is configured in the Main Class



## 4. Set up the Map Reduce

\*Create the Map class

A screenshot of an IDE showing the code for the Map.java class. The code implements the Mapper interface, specifically for the wordcount example. It reads a line of text from the input, splits it into words, and emits each word as a key-value pair where the value is the integer 1.

```
package wordcount;

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

import java.io.IOException;

public class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        String[] words = line.split(" ");

        for (String word : words) {
            context.write(new Text(word), new IntWritable(1));
        }
    }
}
```



\*Create the Reduce class

```

package wordcount;

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

import java.io.IOException;

public class Reduce extends Reducer<Text, IntWritable, Text, LongWritable> {

    @Override
    public void reduce(Text key,
                       Iterable<IntWritable> values,
                       Context context)
        throws IOException, InterruptedException {
        long count = 0;
        for (IntWritable value : values) {
            count++;
        }
        context.write(key, new LongWritable(count));
    }
}

```

\*Create the Main class

```

import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class Main extends Configured implements Tool {
    @Override
    public int run(String[] args) throws Exception{
        Job job = Job.getInstance(getConf());
        job.setJobName("wordcount");
        job.setJarByClass(Main.class);

        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        Path inputFilePath = new Path(args[0]);
        Path outputPath = new Path(args[1]);

        FileInputFormat.addInputPath(job, inputFilePath);
        FileOutputFormat.setOutputPath(job, outputPath);

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new Main(), args);
        System.exit(exitCode);
    }
}

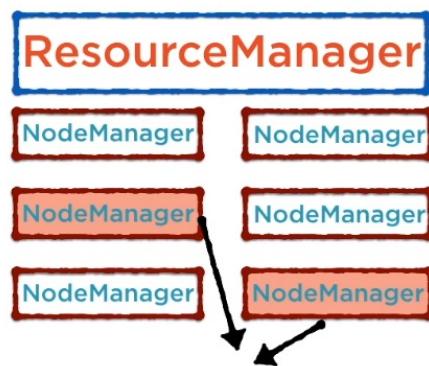
```

## KO4: Able to Schedule and Manage data with MapReduce

### YARN

In 2013, hadoop architecture was changed and MapReduce was divided into two MapReduce AND Yarn. \* MapReduce used for what operations we want to define. \* Yarn was responsible how these operations run across cluster.

### YARN



It has two components : 1. Resource Manager 2. Node Manager

## YARN



Co-ordinates tasks running on the cluster

Assigns new nodes in case of failure



There are Three Scheduling Policies

- 1. FIFO
- 2. Capacity
- 3. Fair

\*\* FIFO Scheduler\*\*

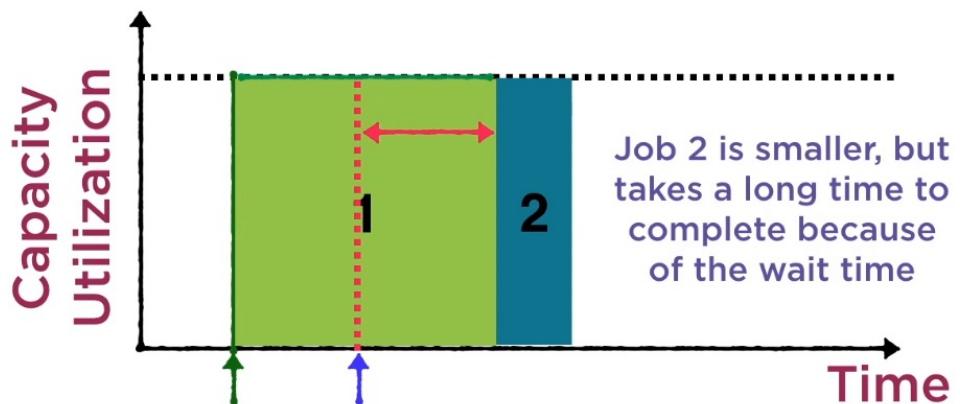
FIFO Scheduler

The Building Blocks of Hadoop - HDFS, MapReduce, and YARN  
By Janani Ravi

Table of Contents Notes

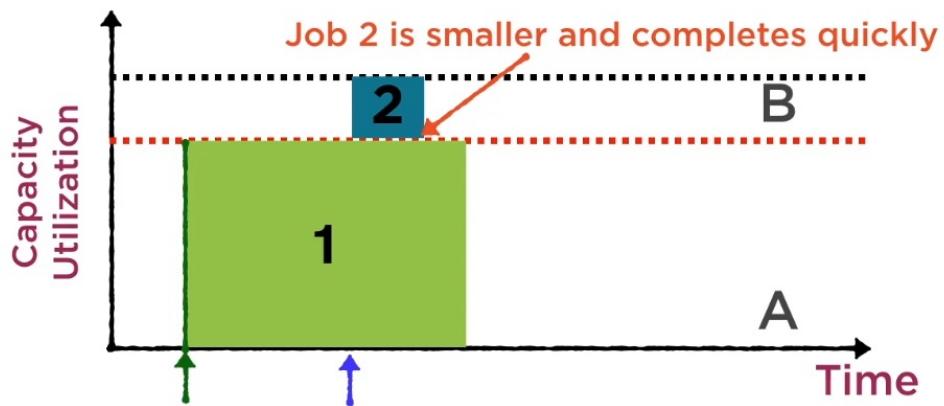
- ④ Storing Data with HDFS 34m 8s
- ⑤ Processing Data with MapReduce 26m 19s
- ⑥ Scheduling and Managing Tasks with YARN 22m 39s
  - ✓ Anatomy of a Job Run in YARN 6m 14s
  - ✓ The First in First out Scheduler 4m 18s
  - ✓ The Capacity Scheduler 3m 34s
  - ✓ The Fair Scheduler 2m 31s
  - ✓ Running Jobs on a Specific Qu... 6m 0s

### FIFO Scheduler



\*\* Capacity Scheduler\*\*

### Capacity Scheduler



\*\* Fair Scheduler

### Fair Scheduler

