

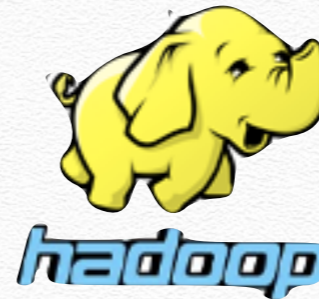
Hadoop 3 Configuration and First Examples

Big Data - 25/03/2020

Apache Hadoop & YARN

❖ Apache Hadoop (1.X)

- * De facto **Big Data** open source platform



- * Running for about 5 years in production at hundreds of companies like Yahoo, Ebay and Facebook

❖ Hadoop 2.X

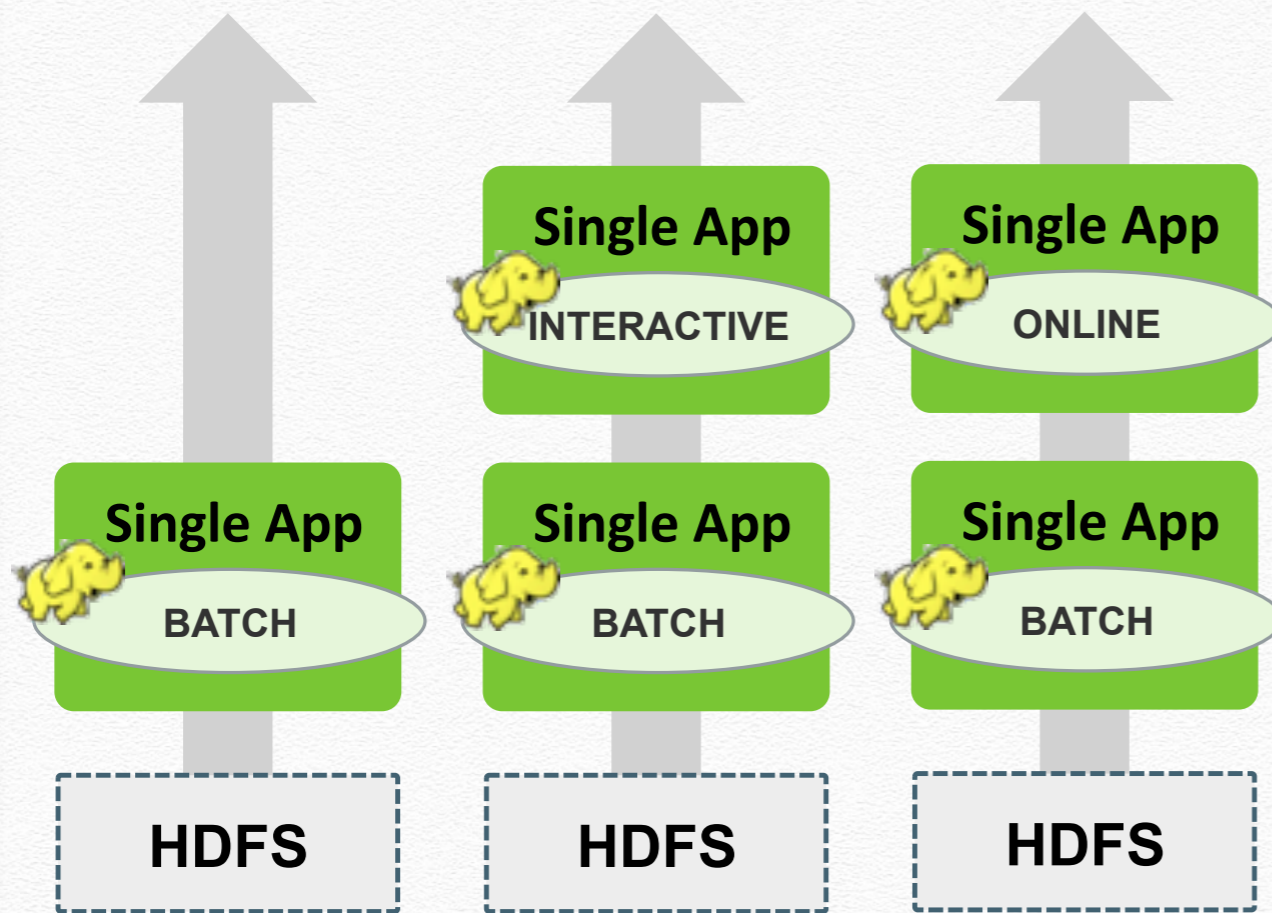


- * Significant **improvements** in **HDFS** distributed storage layer. High Availability, NFS, Snapshots
- * **YARN** – next generation compute framework for Hadoop designed from the ground up based on experience gained from Hadoop 1
- * **YARN** running in production at Yahoo for about a year

1st Generation Hadoop: Batch Focus

HADOOP 1.0

Built for Web-Scale Batch Apps



All other usage patterns
MUST leverage same
infrastructure

Forces Creation of Silos to
Manage Mixed Workloads

Hadoop 1 Architecture

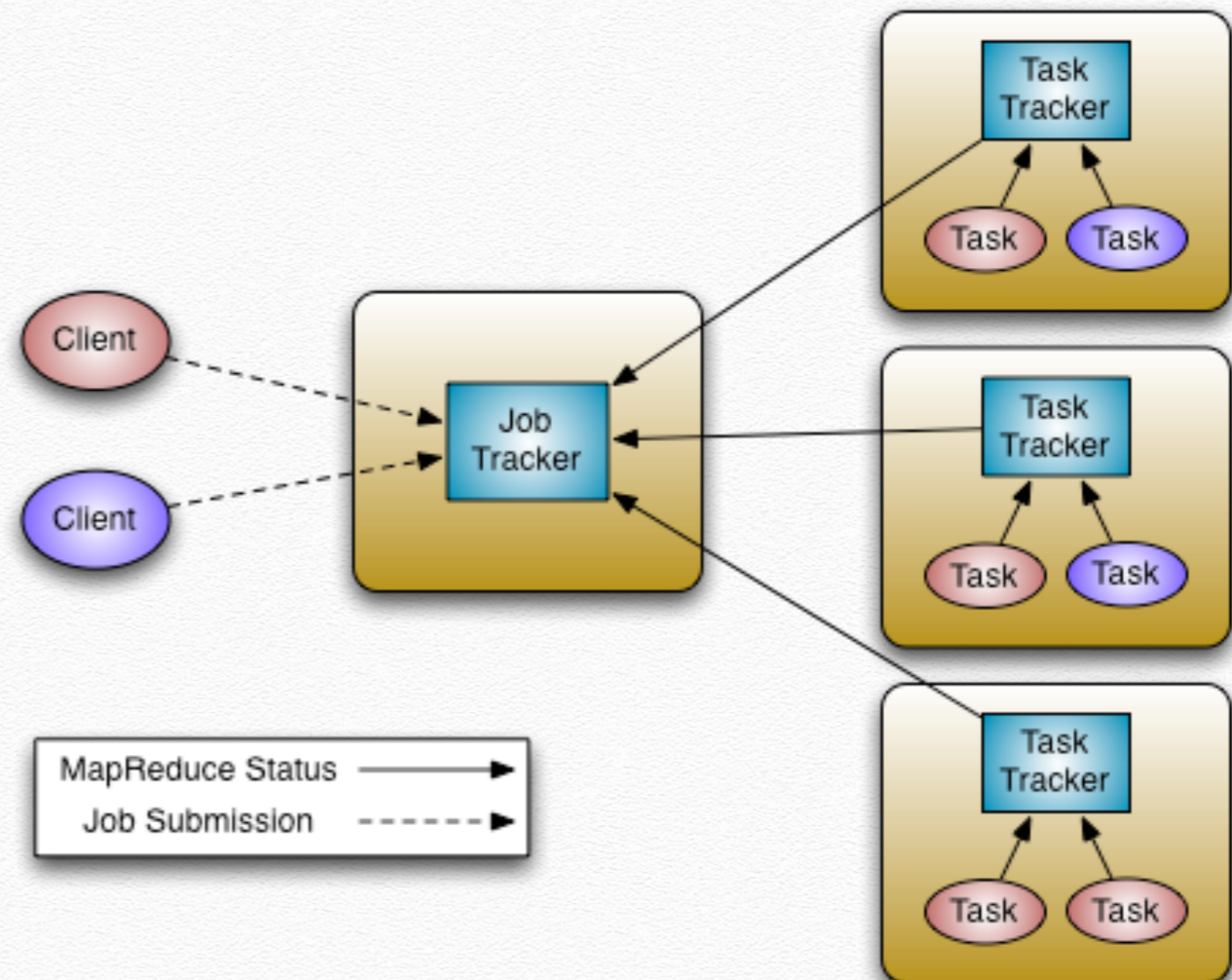
JobTracker

Manage Cluster Resources & Job Scheduling

TaskTracker

Per-node agent

Manage Tasks



Hadoop 1 Limitations

Lacks Support for Alternate Paradigms and Services

Force everything needs to look like Map Reduce

Iterative applications in MapReduce are 10x slower

Scalability

Max Cluster size ~5,000 nodes

Max concurrent tasks ~40,000

Availability

Failure Kills Queued & Running Jobs

Hard partition of resources into map and reduce slots

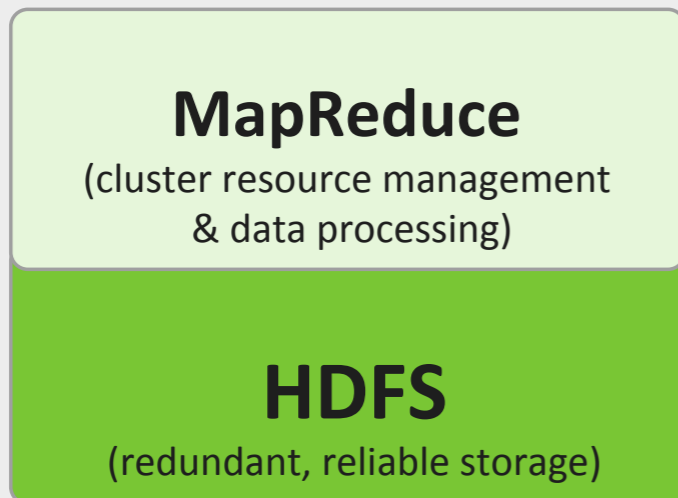
Non-optimal Resource Utilization

Hadoop as Next-Gen Platform

Single Use System

Batch Apps

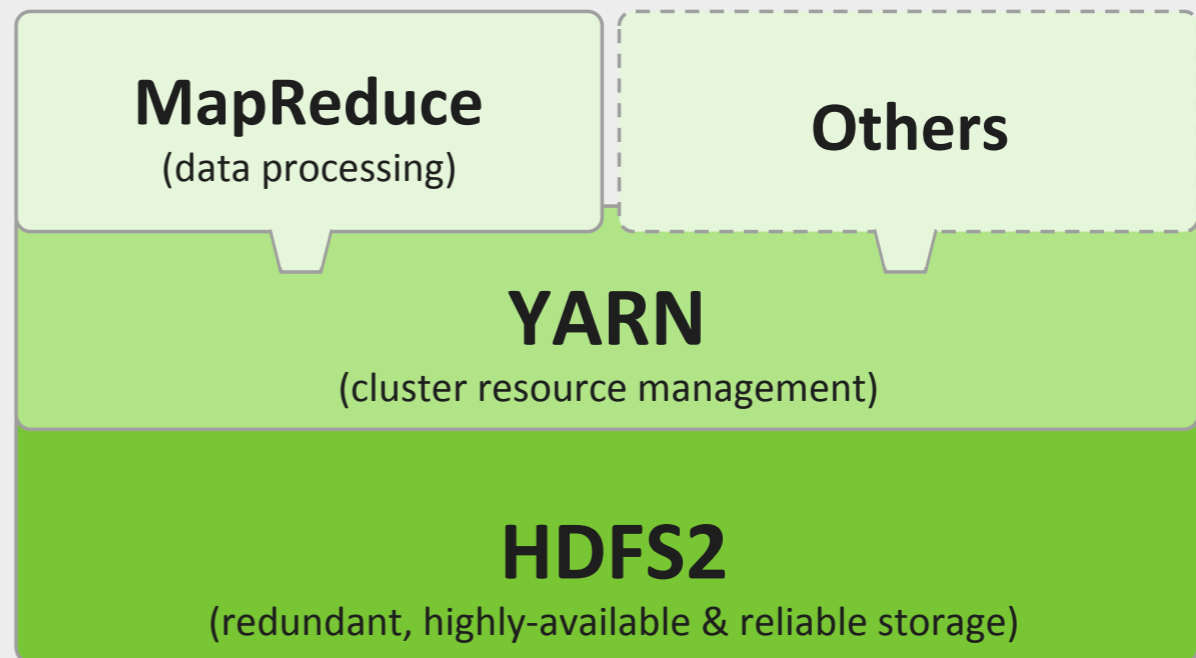
HADOOP 1.0



Multi Purpose Platform

Batch, Interactive, Online, Streaming, ...

HADOOP 2.0



Hadoop 2 - YARN Architecture

ResourceManager (RM)

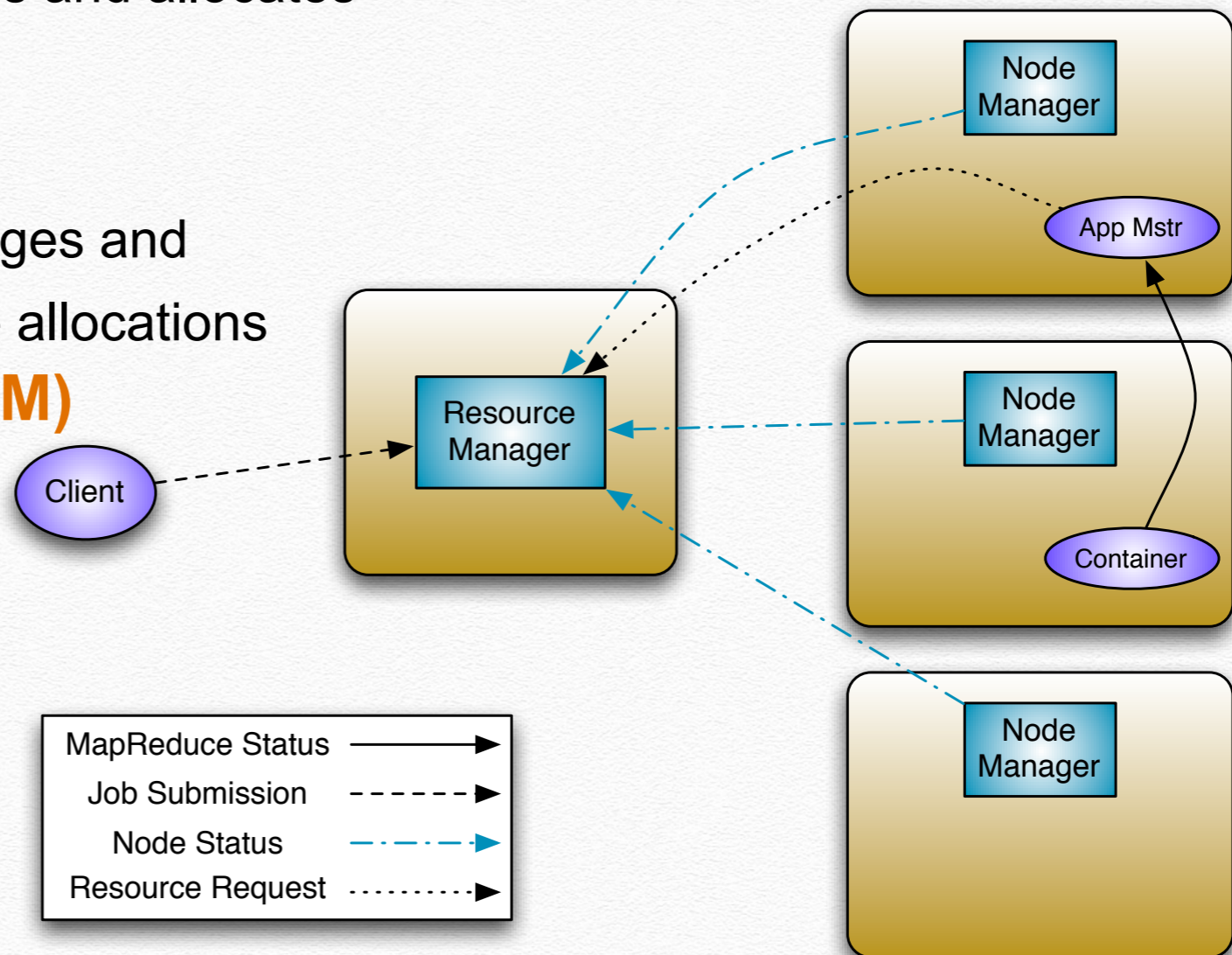
Central agent - Manages and allocates cluster resources

NodeManager (NM)

Per-Node agent - Manages and enforces node resource allocations

ApplicationMaster (AM)

Per-Application –
Manages application
lifecycle and task
scheduling

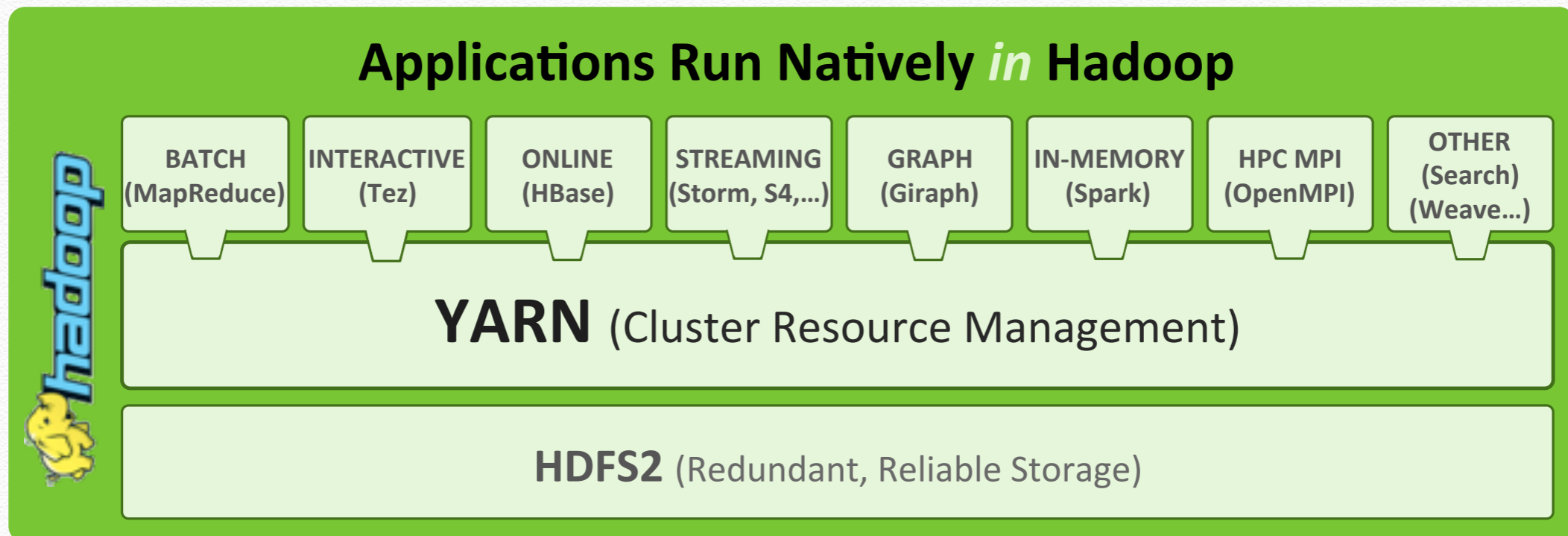


YARN: Taking Hadoop Beyond Batch

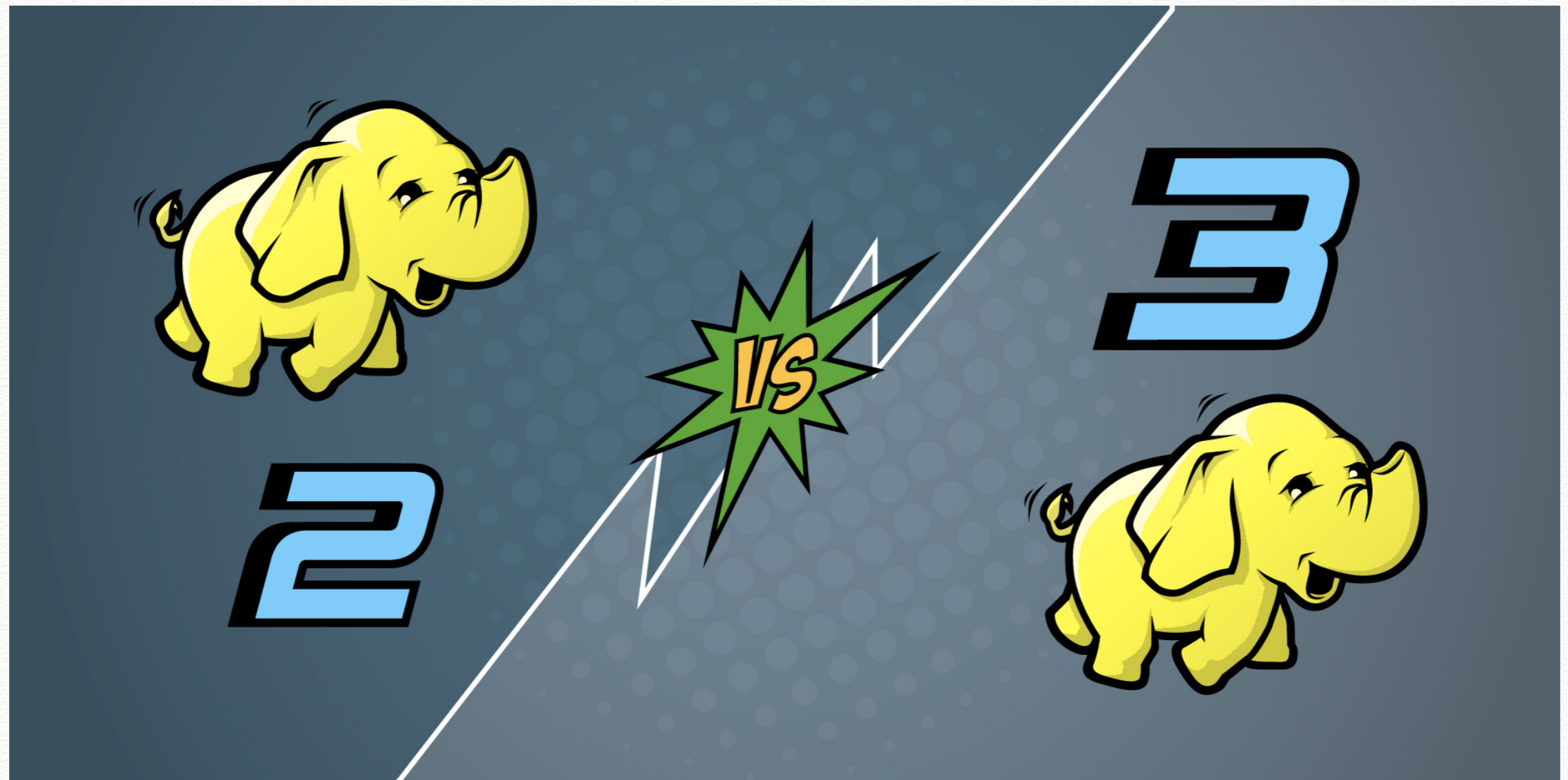
Store ALL DATA in one place...

Interact with that data in MULTIPLE WAYS

with Predictable Performance and Quality of Service



How Apache Hadoop 3 Adds Value Over Apache Hadoop 2



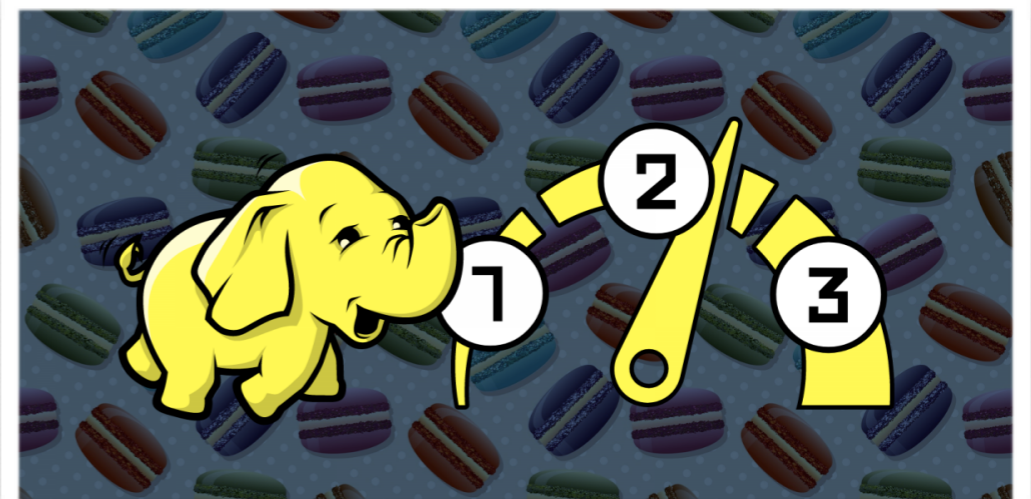
How Apache Hadoop 3 Adds Value Over Apache Hadoop 2

Attributes	Hadoop 2.x	Hadoop 3.x
Handling Fault-tolerance	Through replication	Through erasure coding
Storage	Consumes 200% in HDFS	Consumes just 50%
Scalability	Limited	Improved
File System	DFS, FTP and Amazon S3	All features plus Microsoft Azure Data Lake File System
Manual Intervention	Not needed	Not needed
Scalability	Up to 10,000 nodes in a cluster	Over 10,000 nodes in a cluster
Cluster Resource Management	Handled by YARN	Handled by YARN
Data Balancing	Uses HDFS balancer for this purpose	Uses Intra-data node balancer

How Apache Hadoop 3 Adds Value Over Apache Hadoop 2

Look at

- ✓ <https://techvidvan.com/tutorials/hadoop-2-x-vs-hadoop-3-x/>
- ✓ <http://www.adaltas.com/en/2018/07/25/clusters-workloads-migration-hadoop-2-to-3/>
- ✓ <https://data-flair.training/blogs/hadoop-2-x-vs-hadoop-3-x-comparison/>



Environment variables

- ❖ In the **bash_profile** export all needed **environment variables**

```
[Air-di-Roberto:~ roberto$ cd  
Air-di-Roberto:~ roberto$ nano .bash_profile
```



```
GNU nano 2.0.6      File: .bash_profile  
  
export PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin  
export JAVA_HOME=$(/usr/libexec/java_home)  
export HADOOP_HOME=/Users/roberto/Documents/hadoop-3.2.1  
export PATH=$PATH:$HADOOP_HOME/bin
```



Setup passphraseless ssh

- ❖ check that you can **ssh** to the **localhost** without a **passphrase**:

```
$:~ ssh localhost
```

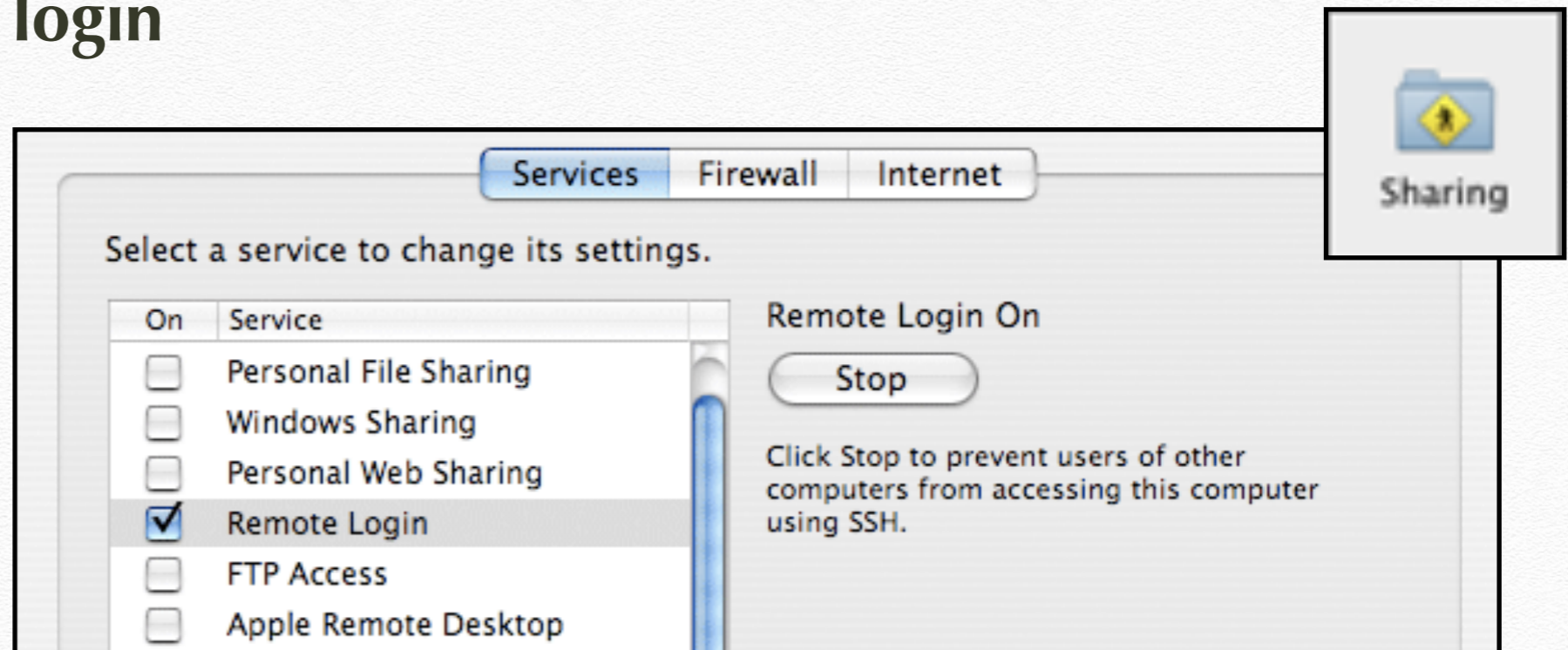
- ❖ If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$:~ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa  
$:~ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
$:~ chmod 0600 ~/.ssh/authorized_keys
```



Setup passphraseless ssh

❖ Allow remote login



```
$:~ ssh-keygen -t rsa -P ""  
$:~ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

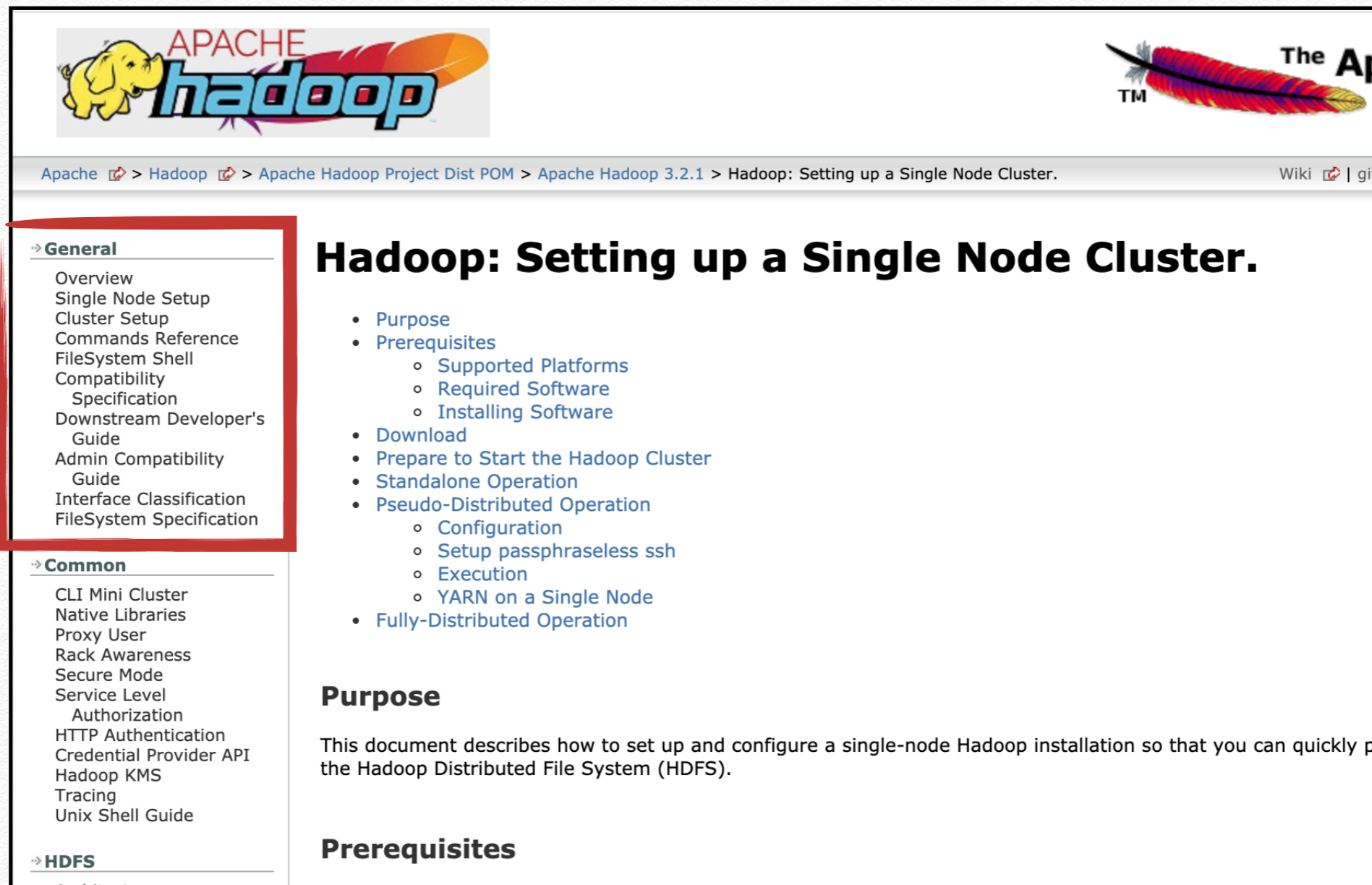
Hadoop 3 Configuration

- ❖ Download the **binary release** of **apache hadoop**:
- ❖ **hadoop-3.2.1.tar.gz**

Version	Release date	Source download	Binary download
2.10.0	2019 Oct 29	source (checksum signature)	binary (checksum signature)
3.1.3	2019 Oct 21	source (checksum signature)	binary (checksum signature)
3.2.1	2019 Sep 22	source (checksum signature)	binary (checksum signature)
3.1.2	2019 Feb 6	source (checksum signature)	binary (checksum signature)
2.9.2	2018 Nov 19	source (checksum signature)	binary (checksum signature)

Hadoop 3 Configuration

- ❖ At <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html> you can find a **WIKI** about Hadoop 3



The screenshot shows the Apache Hadoop Wiki page for "Setting up a Single Node Cluster". The page features the Apache Hadoop logo (a yellow elephant) and the Apache feather logo. The breadcrumb trail is: Apache > Hadoop > Apache Hadoop Project Dist POM > Apache Hadoop 3.2.1 > Hadoop: Setting up a Single Node Cluster. The page title is "Hadoop: Setting up a Single Node Cluster." The left sidebar contains a navigation menu with sections: General, Common, and HDFS. The main content area lists a table of contents for the page, including Purpose, Prerequisites, Download, Prepare to Start the Hadoop Cluster, Standalone Operation, Pseudo-Distributed Operation, and Fully-Distributed Operation. The "Purpose" section states: "This document describes how to set up and configure a single-node Hadoop installation so that you can quickly p the Hadoop Distributed File System (HDFS)." The "Prerequisites" section is also visible.

APACHE hadoop

The Ap
TM

Apache > Hadoop > Apache Hadoop Project Dist POM > Apache Hadoop 3.2.1 > Hadoop: Setting up a Single Node Cluster. Wiki | gi

Hadoop: Setting up a Single Node Cluster.

- [Purpose](#)
- [Prerequisites](#)
 - [Supported Platforms](#)
 - [Required Software](#)
 - [Installing Software](#)
- [Download](#)
- [Prepare to Start the Hadoop Cluster](#)
- [Standalone Operation](#)
- [Pseudo-Distributed Operation](#)
 - [Configuration](#)
 - [Setup passphraseless ssh](#)
 - [Execution](#)
 - [YARN on a Single Node](#)
- [Fully-Distributed Operation](#)

Purpose

This document describes how to set up and configure a single-node Hadoop installation so that you can quickly p the Hadoop Distributed File System (HDFS).

Prerequisites

General

- Overview
- Single Node Setup
- Cluster Setup
- Commands Reference
- FileSystem Shell
- Compatibility Specification
- Downstream Developer's Guide
- Admin Compatibility Guide
- Interface Classification
- FileSystem Specification

Common

- CLI Mini Cluster
- Native Libraries
- Proxy User
- Rack Awareness
- Secure Mode
- Service Level Authorization
- HTTP Authentication
- Credential Provider API
- Hadoop KMS
- Tracing
- Unix Shell Guide

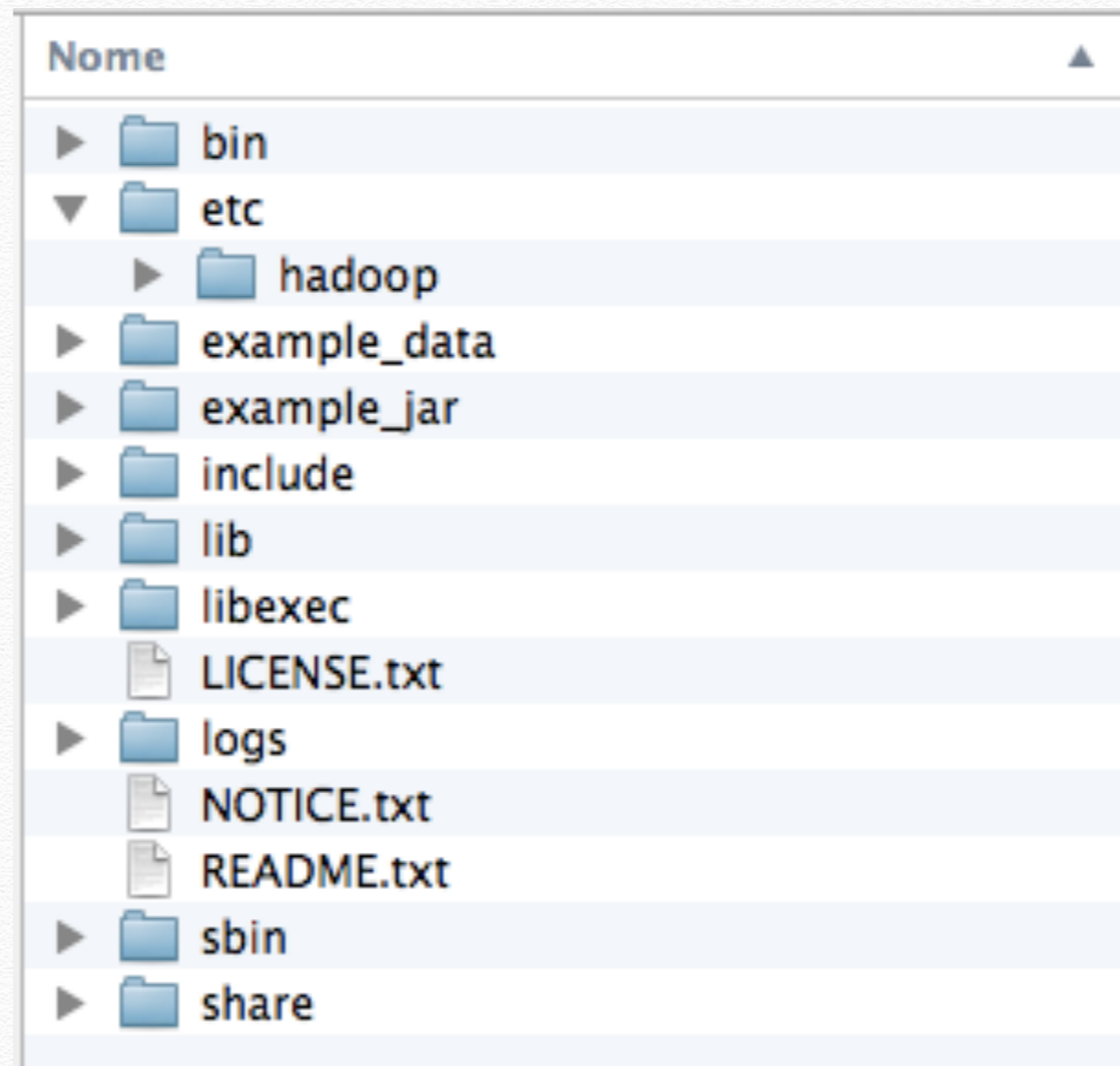
HDFS

Hadoop 3 Configuration: Pseudo-Distributed Operation

❖ In the **etc/hadoop** directory of the **hadoop-home** directory, set the following files

- **core-site.xml**

- **hdfs-site.xml**



Hadoop 3 Configuration: Pseudo-Distributed Operation

core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

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

hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

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

Hadoop 3 Configuration & Running

- ❖ Final configuration:

```
$:~ hadoop-*/bin/hdfs namenode -format
```

- ❖ **Running** hadoop:

```
$:~ hadoop-*/sbin/start-dfs.sh
```

Hadoop 3 Configuration & Running

- ❖ Check all running daemons in Hadoop using the command **jps**

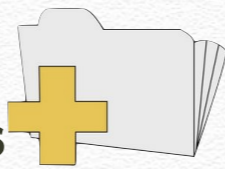
```
$:~ jps
```

```
20758 NameNode  
20829 DataNode  
20920 SecondaryNameNode  
21001 Jps
```

Hadoop 3: commands on hdfs

```
$: ~hadoop-*/bin/hdfs dfs <command> <parameters>
```

- create a directory in hdfs



```
$: ~hadoop-*/bin/hdfs dfs -mkdir input
```

- copy a local file in hdfs



```
$: ~hadoop-*/bin/hdfs dfs -put /tmp/example.txt input
```

- copy result files from hdfs to local file system

```
$: ~hadoop-*/bin/hdfs dfs -get output/result localoutput
```

- delete a directory in hdfs



```
$: ~hadoop-*/bin/hdfs dfs -rm -r input
```

Hadoop 3 Configuration & Running

- ❖ Final configuration:

```
$:~ hadoop-*/bin/hdfs namenode -format
```

- ❖ **Running** hadoop:

```
$:~ hadoop-*/sbin/start-dfs.sh
```

- ❖ **Make** the HDFS **directories** required to execute **MapReduce jobs**:

```
$:~ hadoop-*/bin/hdfs dfs -mkdir /user  
$:~ hadoop-*/bin/hdfs dfs -mkdir /user/<username>
```

- ❖ **Copy** the input files into the distributed filesystem:

```
$:~ hadoop-*/bin/hdfs dfs -put etc/hadoop input
```

Hadoop3: browse hdfs

❖ <http://localhost:9870/>

The screenshot shows the Hadoop 3 web interface. At the top, there is a navigation bar with tabs: Hadoop, Overview, Datanodes, Snapshot, Startup Progress, and Utilities. The Utilities tab is selected, and a dropdown menu is open, showing options for 'Browse the file system' and 'Logs'. A green arrow points from the 'Browse the file system' option to the 'Browse Directory' view below.

Overview 'localhost:9000' (active)

Started:	Wed Mar 25 16:49:08 CET 2015
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-1b20ec3f-9e75-4160-830c-3d6452564225
Block Pool ID:	

Browse Directory

/user/mac/input Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	mac	supergroup	4.33 KB	1	128 MB	capacity-scheduler.xml
-rw-r--r--	mac	supergroup	1.3 KB	1	128 MB	configuration.xml
-rw-r--r--	mac	supergroup	318 B	1	128 MB	container-executor.cfg
-rw-r--r--	mac	supergroup	884 B	1	128 MB	core-site.xml
-rw-r--r--	mac	supergroup	3.58 KB	1	128 MB	hadoop-env.cmd
-rw-r--r--	mac	supergroup	4.21 KB	1	128 MB	hadoop-env.sh
-rw-r--r--	mac	supergroup	2.43 KB	1	128 MB	hadoop-metrics.properties

Hadoop 3: **execute MR** application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **Word Count in MapReduce**

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/word.jar  
WordCount input/words.txt output/result
```

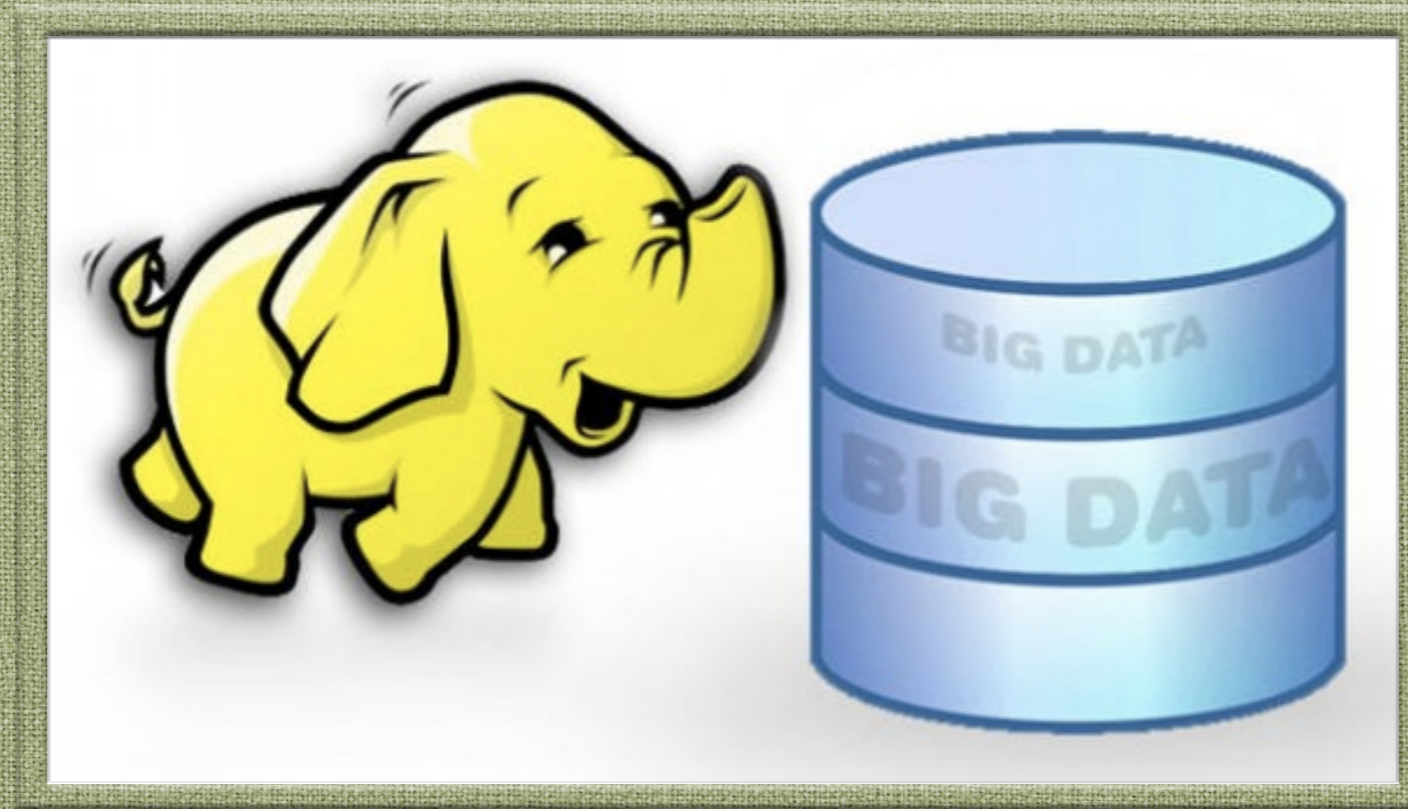
path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

<https://youtu.be/BungOXP3q5Q>

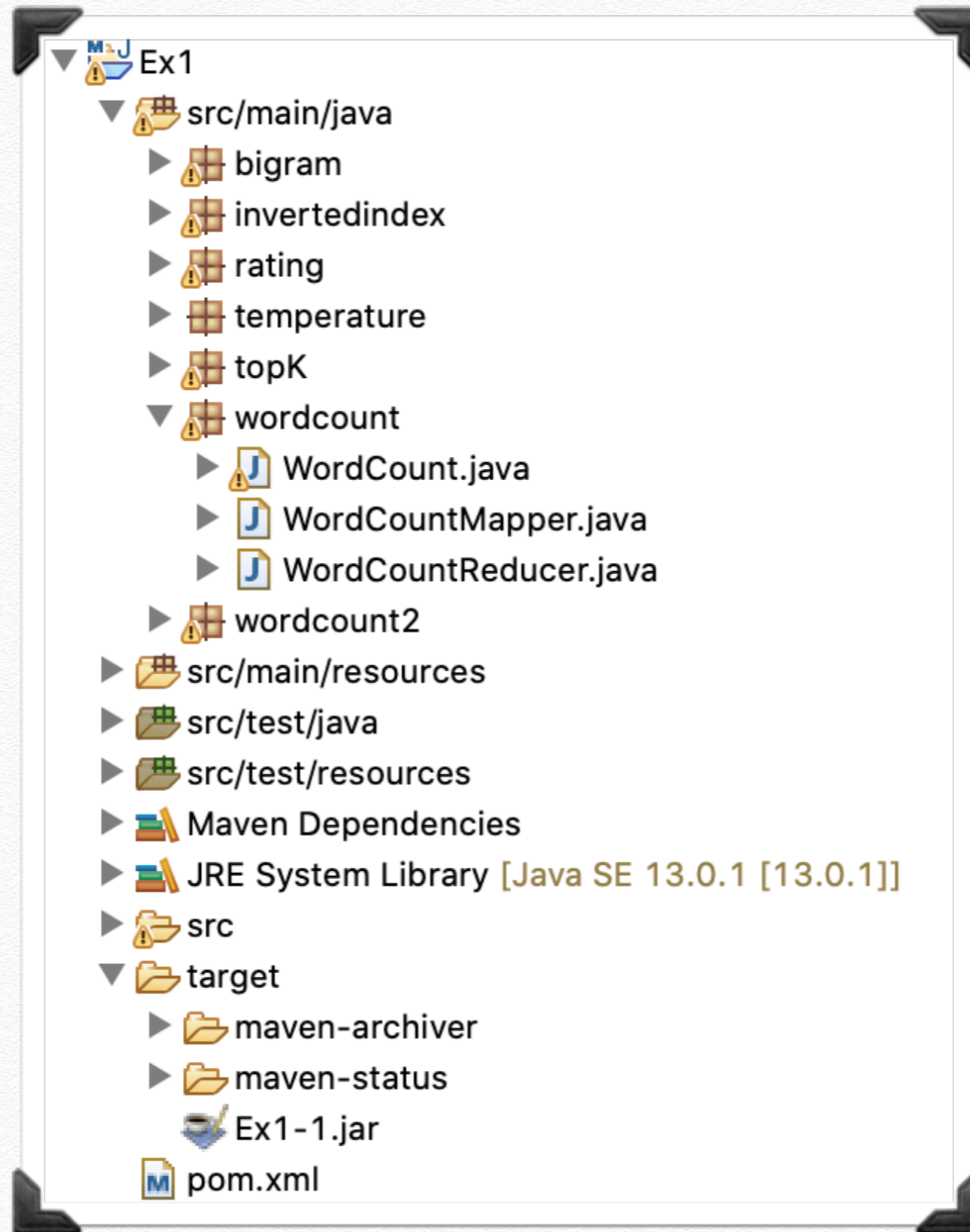
- ❖ Web video to **configure** Hadoop





Let's start with some examples!

Build a Maven Java Project



Build a Maven Java Project: .pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://  
www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://  
maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
```

```
  <groupId>BigData</groupId>  
  <artifactId>Ex1</artifactId>  
  <version>1</version>  
  <packaging>jar</packaging>
```

```
  <name>Ex1</name>  
  <url>http://maven.apache.org</url>
```

```
  <properties>  
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
  </properties>
```

```
  ...
```

```
</project>
```

Build a Maven Java Project: .pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://  
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://  
maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

...

```
<dependencies>
```

...

```
<dependency>  
  <groupId>org.apache.hadoop</groupId>  
  <artifactId>hadoop-common</artifactId>  
  <version>3.2.1</version>  
</dependency>  
  
<dependency>  
  <groupId>org.apache.hadoop</groupId>  
  <artifactId>hadoop-mapreduce-client-core</artifactId>  
  <version>3.2.1</version>  
</dependency>
```

```
<dependency>  
  <groupId>log4j</groupId>  
  <artifactId>log4j</artifactId>  
  <version>1.2.16</version>  
</dependency>  
</dependencies>
```

...

```
</project>
```

Build a Maven Java Project

The screenshot shows an IDE window with a Project Explorer on the left and a code editor on the right. The code editor displays the contents of a `pom.xml` file for a project named `Example1`. The XML content is as follows:

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>BigData</groupId>
4   <artifactId>Example1</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>jar</packaging>
7   <name>Example1</name>
8   <url>http://maven.apache.org</url>
9
10  <properties>
11    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12  </properties>
13
14  <dependencies>
15    <dependency>
16      <groupId>junit</groupId>
17      <artifactId>junit</artifactId>
18      <version>3.8.1</version>
19      <scope>test</scope>
20    </dependency>
21  </dependencies>
22
23  <build>
24    <plugins>
```

A context menu is open over the `pom.xml` file, with the `Run As` option selected. The `Run As` submenu is also open, showing various options. The `Maven install` option is highlighted in blue.

Option	Shortcut
1 Run on Server	⇧⌘X R
2 Java Applet	⌘X A
3 Java Application	⌘X J
4 JUnit Test	⌘X T
5 Maven build	⇧⌘X M
6 Maven build...	
7 Maven clean	
8 Maven generate-sources	
9 Maven install	
Maven test	
Run Configurations...	



The screenshot shows a file explorer view of the `target` directory. The directory structure is as follows:

- target
 - maven-archiver
 - maven-status
 - Ex1-1.jar**

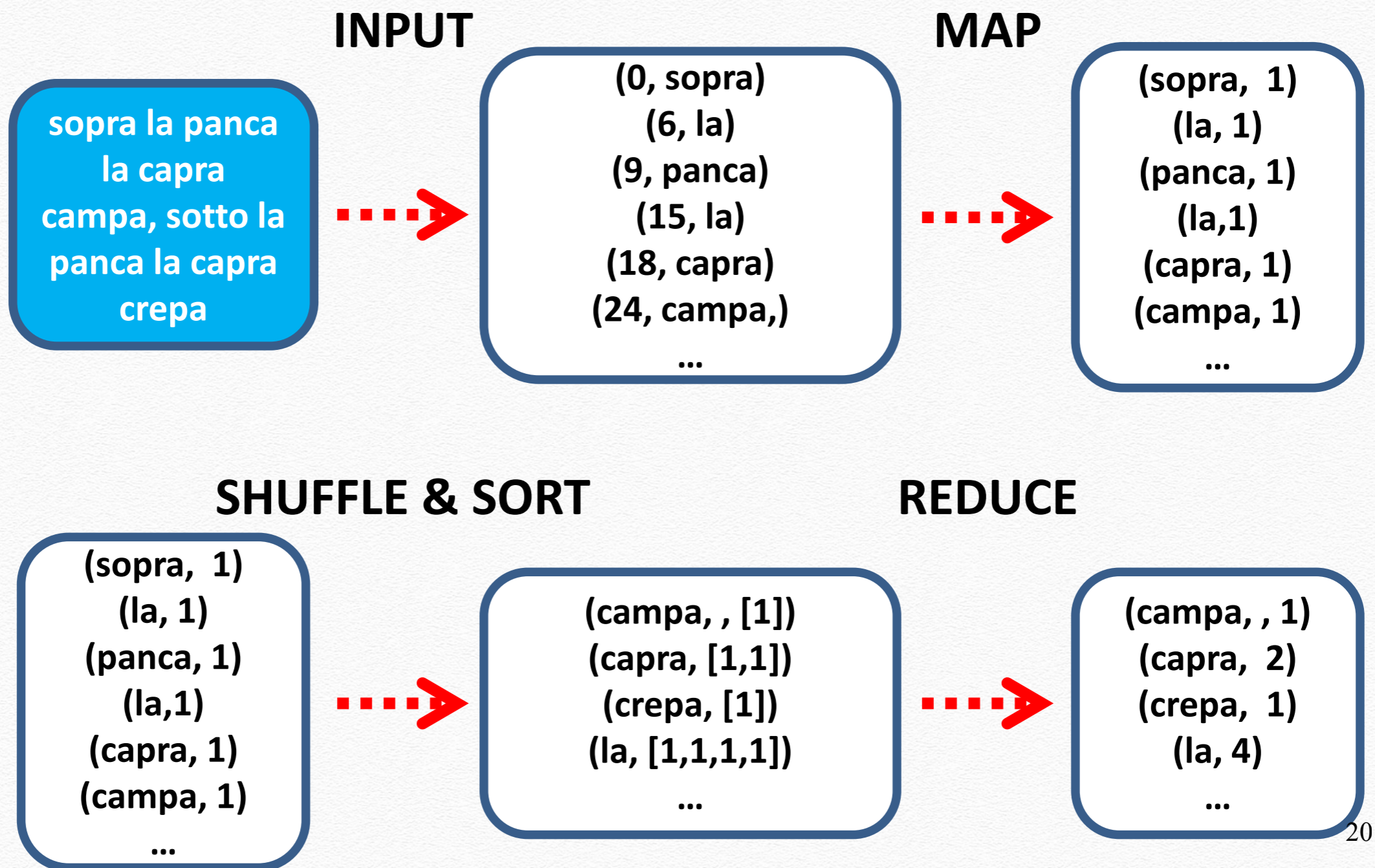
The `Ex1-1.jar` file is circled in red, indicating it is the output of the Maven install command.

<https://youtu.be/M2a5gTNgeqo>

- ❖ Web video to **build** a Maven Project



WordCount: logical data flow



WordCount: MAPPER

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

    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

WordCount: REDUCER

```
public class WordCountReducer 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));
    }
}
```

WordCount: JOB

```
public class WordCount {  
    public static void main(String[] args) throws Exception {  
        Job job = new Job(new Configuration(), "WordCount");  
  
        job.setJarByClass(WordCount.class);  
  
        job.setMapperClass(WordCountMapper.class);  
        // combiner use  
        // job.setCombinerClass(WordCountReducer.class);  
        job.setReducerClass(WordCountReducer.class);  
  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
  
        job.waitForCompletion(true);  
    }  
}
```

Hadoop 3: **execute** *WordCount* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **WordCount** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example1.jar  
wordcount/WordCount input/words.txt output/result_words
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

<https://youtu.be/q3HfIE1bLQ>

- ❖ Web video to **execute** a **Java** MapReduce job



WordCount: without Combiner

MAP OUTPUT

(wordA,1)
(wordA,1)
(wordB,1)
(wordB,1)
(wordB,1)

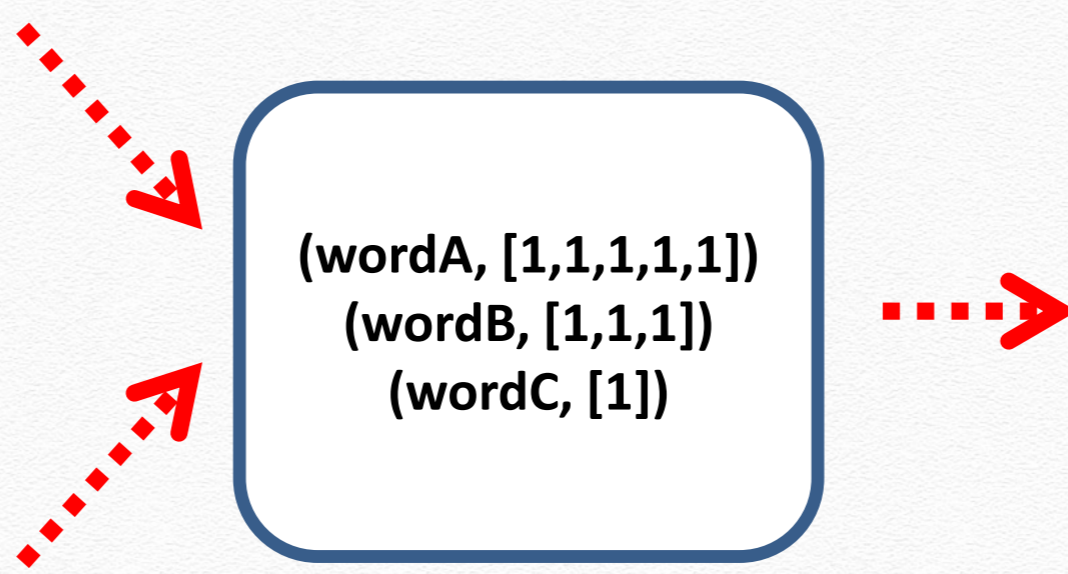
(wordA,1)
(wordA,1)
(wordA,1)
(wordB,1)
(wordC,1)

REDUCER INPUT

(wordA, [1,1,1,1,1])
(wordB, [1,1,1])
(wordC, [1])

OUTPUT

(wordA, 5)
(wordB, 3)
(wordC, 1)



WordCount: with Combiner

MAP OUTPUT

(wordA,1)
(wordA,1)
(wordB,1)
(wordB,1)
(wordB,1)

(wordA,1)
(wordA,1)
(wordA,1)
(wordB,1)
(wordC,1)

(wordA, 2)
(wordB, 3)

(wordA, 3)
(wordB, 1)
(wordC, 1)

REDUCER INPUT

(wordA, [2,3])
(wordB, [3,1])
(wordC, [1])

OUTPUT

(wordA, 5)
(wordB, 3)
(wordC, 1)

COMBINER OUTPUT

WordCount: with Combiner

In the job configuration: ...

```
job.setMapperClass(WordCountMapper.class);  
job.setCombinerClass(WordCountReducer.class);  
job.setReducerClass(WordCountReducer.class);  
...
```


Example: Temperature

We collected information from

<https://www.ncdc.noaa.gov/>

(National Climatic Data Center) about the atmosphere.

For each year (starting from 1763), for each month, for each day, for each hour, for each existing meteorological stations in the world we have an entry in a file that specifies:

data, time, id, air temperature, pressure, elevation, latitude, longitude...

Temperature: data format

...

position: 15-19

0057332130999999**1958**010103004+51317+028783FM-
12+017199999V0203201N00721004501CN0100001N
9-**0021**1-01391102681

87-92: position

value 9999 for missing air temperature

char + at position 87 is optional

0057332130999999**1960**010103004+51317+028783FM-
12+017199999V0203201N00721004501CN0100001N
9+**0054**1-01391102681

...

Year

Air Temperature in Degrees Celsius x10

Example: Temperature

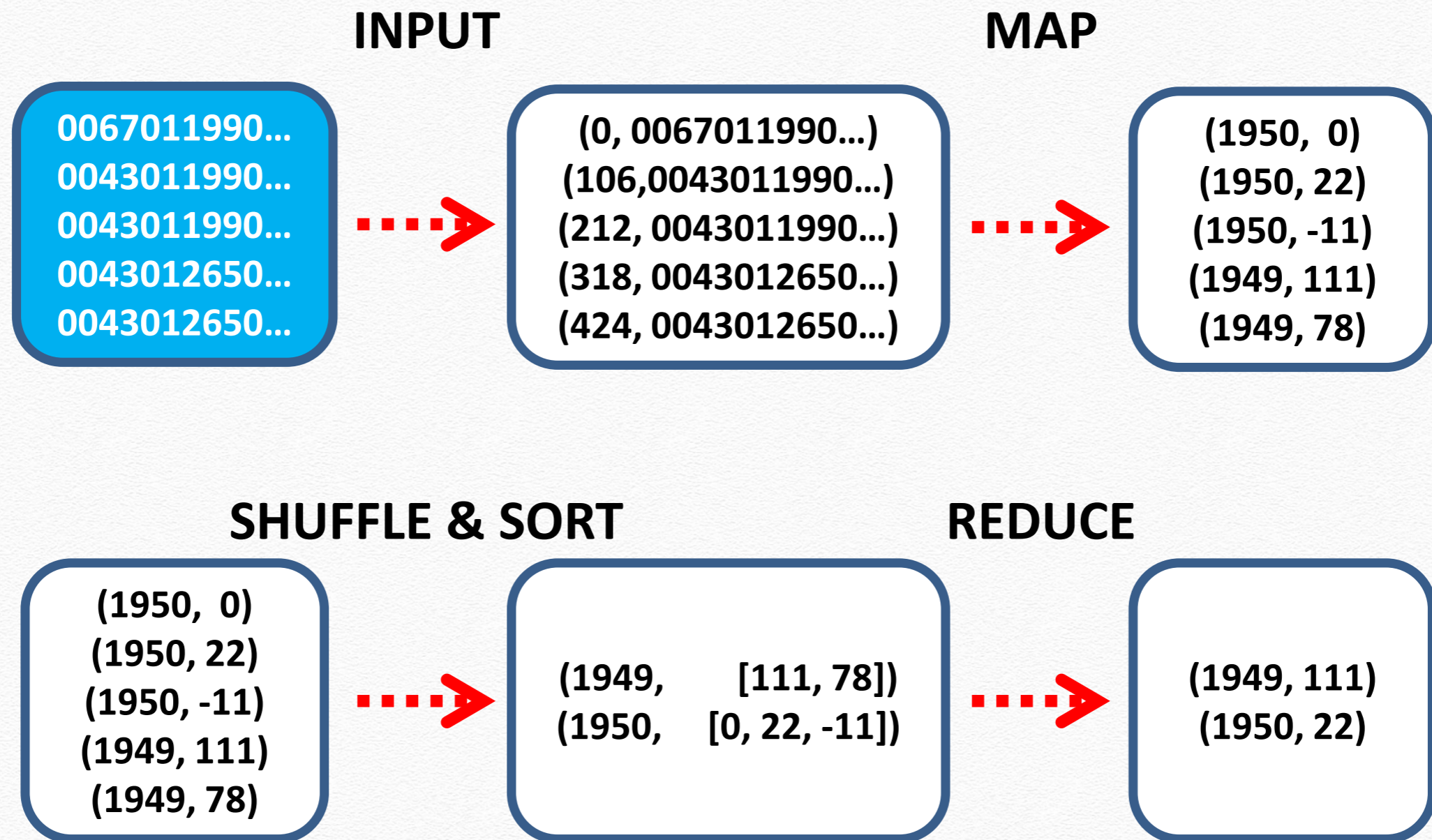
Return for each **Year** (*incrementally*) the **max Air**
Temperature:

1949 -> 111

1950 -> 22

...

Temperature: logical data flow



Temperature: MAPPER

map (k1, v1) -> [(k2, v2)]

```
public class MaxTemperatureMapper
    extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private static final int MISSING = 9999;

    public void map(LongWritable key, Text value, OutputCollector<Text,
        IntWritable> output, Reporter reporter)
        throws IOException {

        String line = value.toString();
        String year = line.substring(15, 19);

        int airTemperature;

        if (line.charAt(87) == '+') {
            airTemperature = Integer.parseInt(line.substring(88, 92));
        } else {
            airTemperature = Integer.parseInt(line.substring(87, 92));
        }

        if ( airTemperature != MISSING ) {
            output.collect(new Text(year), new IntWritable(airTemperature));
        }
    }
}
```

Temperature: REDUCER

reduce (k2, [v2]) -> [(k3, v3)]

```
public class MaxTemperatureReducer
    extends MapReduceBase implements Reducer<Text, IntWritable, Text,
        DoubleWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, DoubleWritable> output,
        Reporter reporter)
        throws IOException {

        int maxValue = Integer.MIN_VALUE;

        while (values.hasNext()) {
            maxValue = Math.max(maxValue, values.next().get());
        }

        output.collect(key, new DoubleWritable(((double)maxValue)/10));
    }
}
```

Temperature: JOB

```
public class MaxTemperature {  
    public static void main(String[] args) throws IOException {  
  
        JobConf conf = new JobConf(MaxTemperature.class);  
        conf.setJobName("Max temperature");  
  
        FileInputFormat.addInputPath(conf, new Path(args[0]));  
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
  
        conf.setMapperClass(MaxTemperatureMapper.class);  
        conf.setReducerClass(MaxTemperatureReducer.class);  
  
        conf.setMapOutputKeyClass(Text.class);  
        conf.setMapOutputValueClass(IntWritable.class);  
  
        conf.setOutputKeyClass(Text.class);  
        conf.setOutputValueClass(DoubleWritable.class);  
  
        JobClient.runJob(conf);  
    }  
}
```

Hadoop 3: **execute** *MaxTemperature* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **MaxTemperature** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/temperature.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example1.jar  
temperature/MaxTemperature input/temperature.txt output/result_temperature
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

Bigram: count

“A bigram is a couple of two consecutive words”

In this sentence there are 8 bigrams:

- ❖ *A bigram*
- ❖ *bigram is*
- ❖ *is a*
- ❖ *a couple*
- ❖ *...*

Bigram: count

sopra la panca la
capra camp,
sotto la panca la
capra crepa



camp, sotto 1
capra camp, 1
capra crepa 1
la capra 2
la panca 2
panca la 2
sopra la 1
sotto la 1

To represent a Bigram we want to use a custom Writable type (BigramWritable)

Bigram: count

```
public class BigramWritable implements WritableComparable<BigramWritable> {

    private Text leftBigram;
    private Text rightBigram;

    public BigramWritable() {
    }

    public BigramWritable(Text left, Text right) {
        this.leftBigram = left;
        this.rightBigram = right;
    }

    public void readFields(DataInput in) throws IOException {
        leftBigram = new Text(in.readUTF());
        rightBigram = new Text(in.readUTF());
    }

    public void write(DataOutput out) throws IOException {
        out.writeUTF(leftBigram.toString());
        out.writeUTF(rightBigram.toString());
    }

    public void set(Text prev, Text cur) {
        leftBigram = prev;
        rightBigram = cur;
    }
}
```

Bigram: count

```
@Override
public String toString() {
    return leftBigram.toString() + " " + rightBigram.toString();
}

@Override
public int hashCode() {
    return leftBigram.hashCode() + rightBigram.hashCode();
}

@Override
public boolean equals(Object o) {
    if (o instanceof BigramWritable) {
        BigramWritable bigram = (BigramWritable) o;
        return leftBigram.equals(bigram.leftBigram)
            && rightBigram.equals(bigram.rightBigram);
    }
    return false;
}

public int compareTo(BigramWritable tp) {
    int cmp = leftBigram.compareTo(tp.leftBigram);
    if (cmp != 0) {
        return cmp;
    }
    return rightBigram.compareTo(tp.rightBigram);
}
}
```

Bigram: MAPPER

```
public class BigramCountMapper extends Mapper<LongWritable, Text, BigramWritable,
IntWritable> {
    private static final IntWritable ONE = new IntWritable(1);
    private static final BigramWritable BIGRAM = new BigramWritable();

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        String line = value.toString();
        String prev = null;
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            String cur = itr.nextToken();
            if (prev != null) {
                BIGRAM.set(new Text(prev), new Text(cur));
                context.write(BIGRAM, ONE);
            }
            prev = cur;
        }
    }
}
```

Bigram: REDUCER

```
public class BigramCountReducer extends Reducer<BigramWritable,  
IntWritable, Text, IntWritable> {  
  
    private final static IntWritable SUM = new IntWritable();  
  
    @Override  
    public void reduce(BigramWritable key, Iterable<IntWritable> values,  
Context context) throws IOException, InterruptedException {  
        int sum = 0;  
        Iterator<IntWritable> iter = values.iterator();  
        while (iter.hasNext()) {  
            sum += iter.next().get();  
        }  
        SUM.set(sum);  
        context.write(new Text(key.toString()), SUM);  
    }  
}
```

Bigram: JOB

```
public class BigramCount extends Configured implements Tool {

    private static final Logger LOG = Logger.getLogger(BigramCount.class);

    private BigramCount() {
    }

    private static final String INPUT = "input";
    private static final String OUTPUT = "output";
    private static final String NUM_REDUCERS = "numReducers";

    @SuppressWarnings({ "static-access" })
    public int run(String[] args) throws Exception {
        Options options = new Options();

        options.addOption(OptionBuilder.withArgName("path").hasArg()
            .withDescription("input path").create(INPUT));
        options.addOption(OptionBuilder.withArgName("path").hasArg()
            .withDescription("output path").create(OUTPUT));
        options.addOption(OptionBuilder.withArgName("num").hasArg()
            .withDescription("number of reducers").create(NUM_REDUCERS));
    }
}
```

Bigram: JOB

```
CommandLine cmdLine;
CommandLineParser parser = new GnuParser();

try {
    cmdLine = parser.parse(options, args);
} catch (ParseException exp) {
    System.err.println("Error parsing command line: "
        + exp.getMessage());
    return -1;
}

if (!cmdLine.hasOption(INPUT) || !cmdLine.hasOption(OUTPUT)) {
    System.out.println("args: " + Arrays.toString(args));
    HelpFormatter formatter = new HelpFormatter();
    formatter.setWidth(120);
    formatter.printHelp(this.getClass().getName(), options);
    ToolRunner.printGenericCommandUsage(System.out);
    return -1;
}

String inputPath = cmdLine.getOptionValue(INPUT);
String outputPath = cmdLine.getOptionValue(OUTPUT);
int reduceTasks = cmdLine.hasOption(NUM_REDUCERS) ? Integer
    .parseInt(cmdLine.getOptionValue(NUM_REDUCERS)) : 1;

LOG.info("Tool name: " + BigramCount.class.getSimpleName());
LOG.info(" - input path: " + inputPath);
LOG.info(" - output path: " + outputPath);
LOG.info(" - num reducers: " + reduceTasks);
```


Bigram: JOB

```
Job job = new Job(getConf());
    job.setJobName(BigramCount.class.getSimpleName());
    job.setJarByClass(BigramCount.class);

    job.setNumReduceTasks(reduceTasks);

FileInputFormat.setInputPaths(job, new Path(inputPath));
FileOutputFormat.setOutputPath(job, new Path(outputPath));

//     FileInputFormat.addInputPath(job, new Path(args[0]));
//     FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapOutputKeyClass(BigramWritable.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    job.setMapperClass(BigramCountMapper.class);
    job.setReducerClass(BigramCountReducer.class);

    Path outputDir = new Path(outputPath);
    FileSystem.get(getConf()).delete(outputDir, true);

    long startTime = System.currentTimeMillis();
    job.waitForCompletion(true);

    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0
        + " seconds");

    return 0;
}

public static void main(String[] args) throws Exception {
    ToolRunner.run(new BigramCount(), args);
}
}
```

Hadoop 3: **execute** *BigramCount* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **BigramCount** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example1.jar  
    bigram/BigramCount input/words.txt output/result_bigram 1
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

Average Word Length by initial letter

sopra la panca
la capra campa,
sotto la panca la
capra crepa

INPUT



(0, sopra)
(6, la)
(9, panca)
(15, la)
(18, capra)
(24, campa,)
...

MAP



(s, 5)
(l, 2)
(p, 5)
(l, 2)
(c, 5)
(c, 6)
...

(s, 5)
(l, 2)
(p, 5)
(l, 2)
(c, 5)
(c, 5)
...

SHUFFLE & SORT



(c, [5,6])
(l, [2,2])
(p, [5])
(s, [5])
...

REDUCE



(c, 5.25)
(l, 2)
(p, 5)
(s, 5)
...

Hadoop 3: **execute** *AverageWordLength* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: *AverageWordLength* in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example1.jar  
    avgword/AverageWordLength input/words.txt output/result_avg_word
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

Inverted Index

there is a tab (\t)

- 1 if you prick us do we not bleed,
- 2 if you tickle us do we not laugh,
- 3 if you poison us do we not die and,
- 4 if you wrong us shall we not revenge

and,	3
bleed,	1
die	3
do	1,2,3
if	1,4,3,2
laugh,	2
not	3,4,1,2
poison	3
prick	1
revenge	4
shall	4
tickle	2
us	2,4,3,1
we	2,4,1,3
wrong	4
you	1,3,4,2

Inverted Index

- 1 if you prick us do we not bleed,
- 2 if you tickle us do we not laugh,
- 3 if you poison us do we not die and,
- 4 if you wrong us shall we not revenge

↓ INPUT

(1, if you prick ...)
(2, if you tickle ...)
(3, if you poison ...)
(4, if you wrong ...)

MAP
→

(if, 1)
(you, 1)
(prick, 1)
...
(if, 2)
(you, 2)
(tickle, 2)
...

REDUCE
→

(if, [1,2,...])
(you, [1,2,...])
(prick, [1])
(tickle, [2])
...

Hadoop 3: **execute** *InvertedIndex* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **InvertedIndex** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

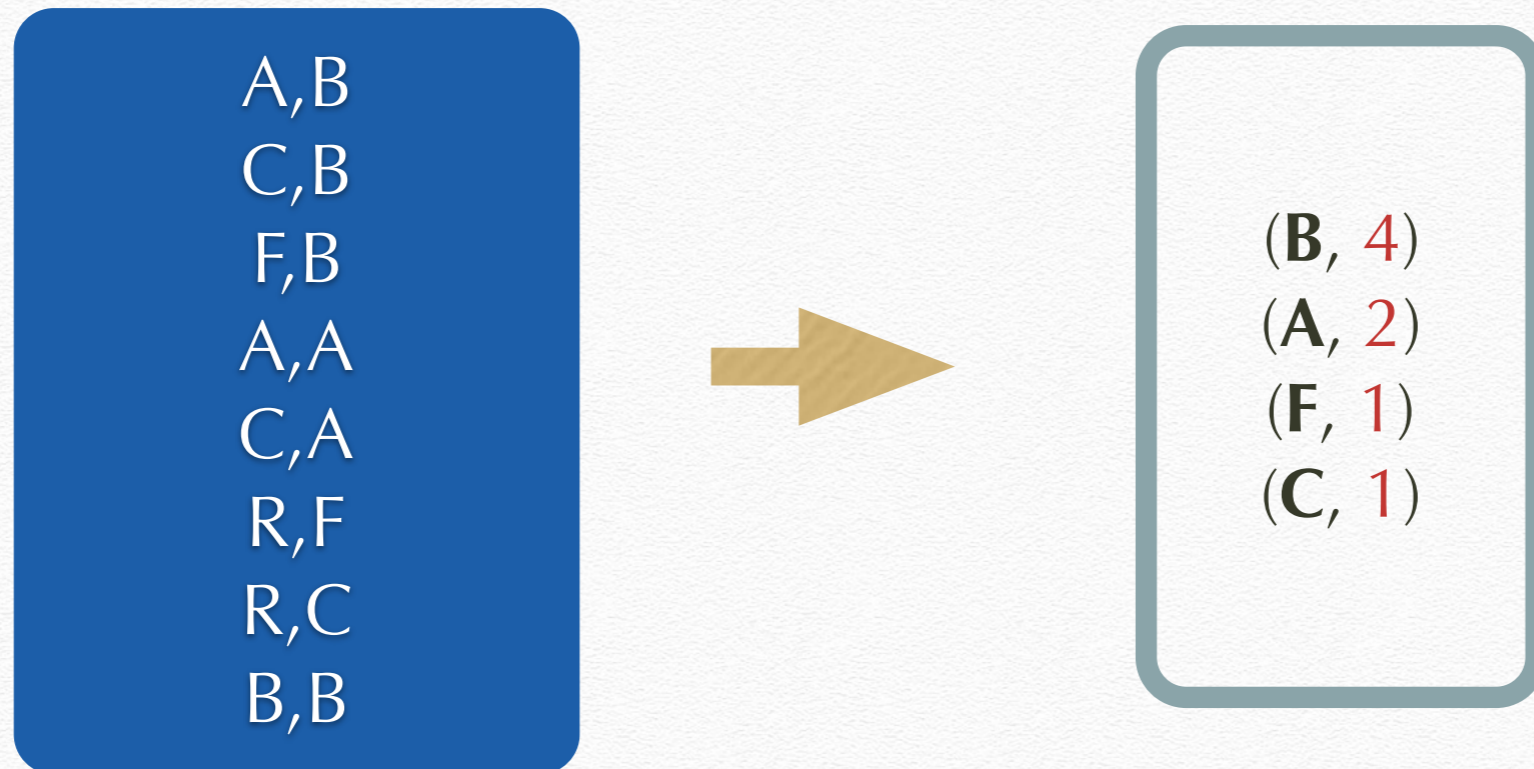
```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words2.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example.jar  
    invertedindex/InvertedIndex input/words2.txt output/result_ii
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

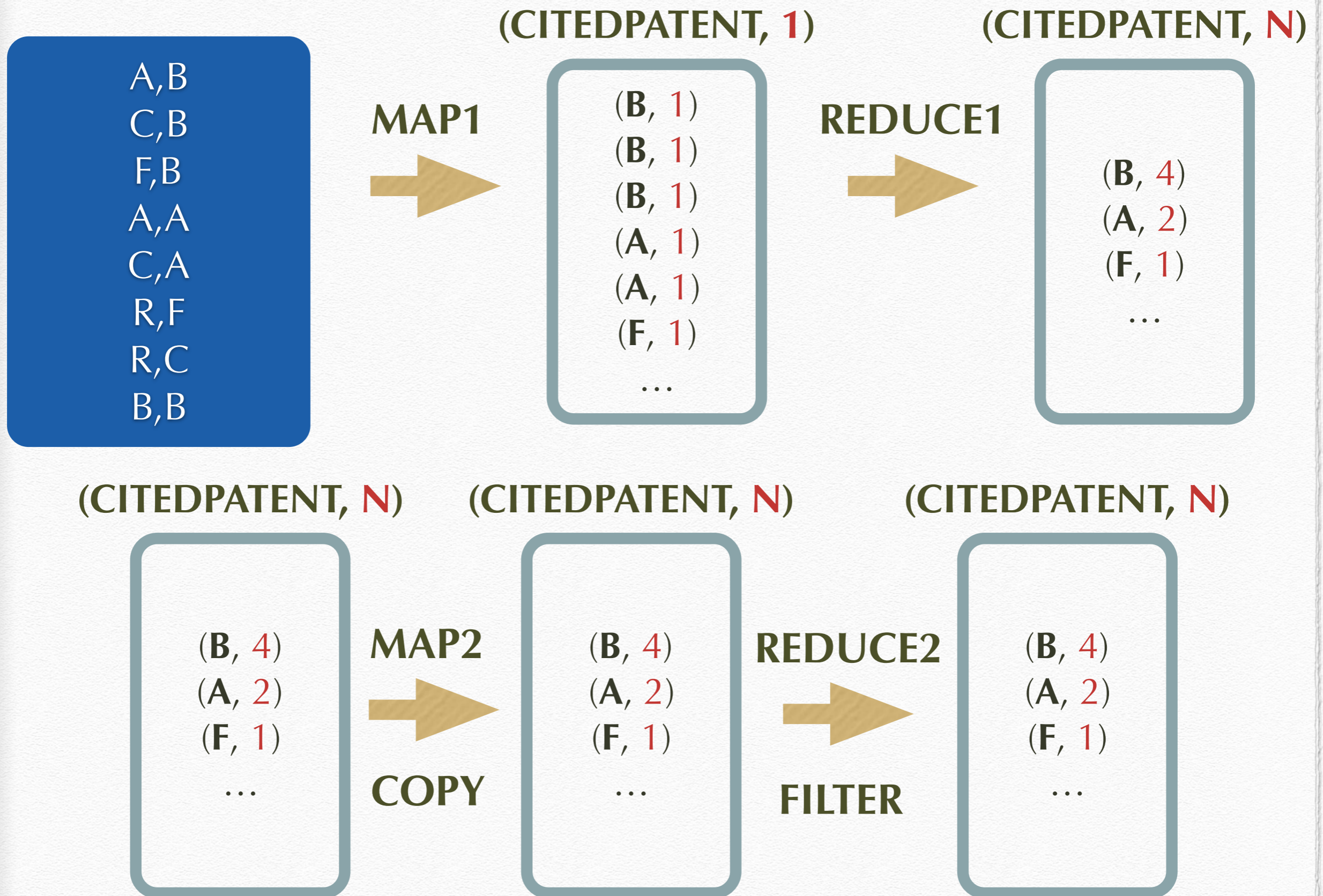
Top K records (citations)



The objective is to find the **top-10** most frequently cited patents in descending order.

The format of the input data is **CITING_PATENT,CITED_PATENT**.

Top K records (citations)



Hadoop 3: **execute** *TopKRecords* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **TopKRecords** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/citations.txt input
```

```
$:~hadoop-*/bin/hadoop jar /example_jar/Example1.jar  
topK/TopKRecords input/citations.txt output/result_topk
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

Hadoop 3: **clean** *DisjointSelection* application

```
$:~hadoop-*/bin/hadoop jar <path-jar> <jar-MainClass> <jar-parameters>
```

❖ Example: **DisjointSelection** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -rm -r output
```

Build a Python Project



Hadoop 3 streaming

- ❖ At <https://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html#Streaming Command Options> you can find a **WIKI** about Hadoop 3 streaming

Hadoop Streaming

- [Hadoop Streaming](#)
 - [Hadoop Streaming](#)
 - [How Streaming Works](#)
 - [Streaming Command Options](#)
 - [Specifying a Java Class as the Mapper/Reducer](#)
 - [Packaging Files With Job Submissions](#)
 - [Specifying Other Plugins for Jobs](#)
 - [Setting Environment Variables](#)
 - [Generic Command Options](#)
 - [Specifying Configuration Variables with the -D Option](#)
 - [Specifying Directories](#)
 - [Specifying Map-Only Jobs](#)
 - [Specifying the Number of Reducers](#)
 - [Customizing How Lines are Split into Key/Value Pairs](#)
 - [Working with Large Files and Archives](#)
 - [Making Files Available to Tasks](#)
 - [Making Archives Available to Tasks](#)
 - [More Usage Examples](#)
 - [Hadoop Partitioner Class](#)
 - [Hadoop Comparator Class](#)
 - [Hadoop Aggregate Package](#)
 - [Hadoop Field Selection Class](#)
 - [Frequently Asked Questions](#)
 - [How do I use Hadoop Streaming to run an arbitrary set of \(semi\) independent tasks?](#)
 - [How do I process files, one per map?](#)
 - [How many reducers should I use?](#)
 - [If I set up an alias in my shell script, will that work after -mapper?](#)
 - [Can I use UNIX pipes?](#)
 - [What do I do if I get the "No space left on device" error?](#)
 - [How do I specify multiple input directories?](#)
 - [How do I generate output files with gzip format?](#)
 - [How do I provide my own input/output format with streaming?](#)

→ General

- Overview
- Single Node Setup
- Cluster Setup
- Commands Reference
- FileSystem Shell
- Compatibility
 - Specification
- Downstream Developer's Guide
- Admin Compatibility Guide
- Interface Classification
- FileSystem Specification

→ Common

- CLI Mini Cluster
- Native Libraries
- Proxy User
- Rack Awareness
- Secure Mode
- Service Level
 - Authorization
- HTTP Authentication
- Credential Provider API
- Hadoop KMS
- Tracing
- Unix Shell Guide

→ HDFS

- Architecture
- User Guide
- Commands Reference
- NameNode HA With QJM
- NameNode HA With NFS
- Federation
- ViewFs
- Snapshots

Hadoop 3 streaming

- ❖ Download **hadoop-streaming-3.2.1.jar** at <https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming/3.2.1/source-code>

Download hadoop-streaming JAR 3.2.1 with all dependencies

These are the files of the artifact **hadoop-streaming** version **3.2.1** from the group **org.apache.hadoop**.

Apache Hadoop MapReduce Streaming

 Download hadoop-streaming (3.2.1)

Artifact hadoop-streaming

Group org.apache.hadoop

Version 3.2.1

Last update 10. September 2019

Newest version No

Tags: streaming hadoop apache mapreduce

Organization not specified

URL Not specified

License not specified

Dependencies amount 0

Dependencies No dependencies

There are maybe transitive dependencies!

✓ **The newest version!**

Maven

Gradle


Ivy


SBT

```
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-streaming</artifactId>
  <version>3.2.1</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->
```

 Show more of this group

 Show more artifacts with this name

 Show all versions of hadoop-streaming

 Show documentation

 **Add to Project**

Python: WordCount Mapper

❖ mapper.py

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

Python: WordCount Reducer

❖ reducer.py

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```


Hadoop 3: **execute** *WordCount* application

```
$:~hadoop-*/bin/hadoop jar <path-streaming-jar> <jar-parameters>
```

❖ Example: (Python) **WordCount** in MapReduce

```
$:~hadoop-*/bin/hdfs dfs -mkdir output
```

```
$:~hadoop-*/bin/hdfs dfs -put /example_data/words.txt input
```

```
$:~hadoop-*/bin/hadoop jar hadoop-streaming-3.2.1.jar  
-mapper mapper.py -reducer reducer.py  
-input /user/roberto/input/words.txt  
-output /user/roberto/output/result_words
```

path on hdfs to reach the file

path on hdfs of the directory to generate
to store the result

<https://youtu.be/dFM-aSjJ86E>

- ❖ Web video to **execute** a **Python** MapReduce job

A hand holding a yellow pencil is pointing to a snippet of Python code on a blackboard. The code is color-coded and includes a class method and several instance methods.

```
if path:
    self.file = open(os.path.join(
        self.file, path))
    self.fingerprints.add(fp)

@classmethod
def from_settings(cls, settings):
    debug = settings.get('debug', False)
    return cls(job_dir=settings.get('job_dir',
        os.path.join(settings.get('base_dir',
            '/tmp'), 'mapreduce')))

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + '\n')

def request_fingerprint(self, request):
    return hashlib.md5(request.url.encode('utf-8')).hexdigest()
```

Stopping the Hadoop 3 DFS

❖ **Stop** hadoop:

```
$:~ hadoop-*/sbin/stop-dfs.sh
```

That's all! Happy Map Reducing!

Hadoop 3 on AMAZON

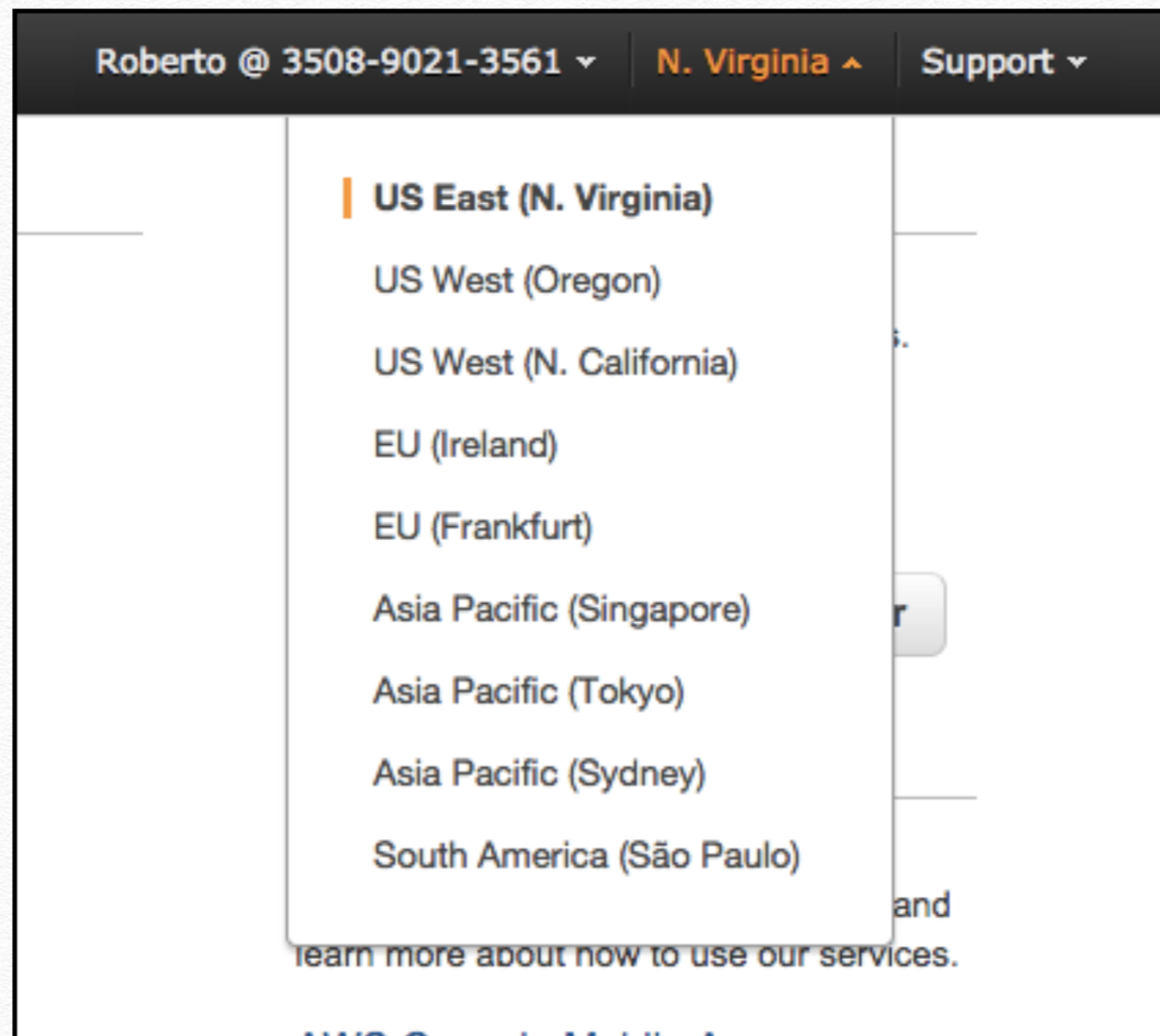


Hadoop 3 on AMAZON

The screenshot displays the AWS Management Console interface. At the top, there is a navigation bar with the AWS logo, 'Services', 'Resource Groups', and user information (Roberto, N. Virginia, Support). The main content area is titled 'AWS services' and includes a search bar with the placeholder text 'Find a service by name (for example, EC2, S3, Elastic Beanstalk)'. Below the search bar, there are sections for 'Recently visited services' and 'All services'. The 'All services' section is organized into columns of service categories, each with an icon and a list of services. The categories and their respective services are: Compute (EC2, EC2 Container Service, Lightsail, Elastic Beanstalk, Lambda, Batch), Storage (S3, EFS, Glacier, Storage Gateway), Database (partially visible), Developer Tools (CodeCommit, CodeBuild, CodeDeploy, CodePipeline, X-Ray), Management Tools (CloudWatch, CloudFormation, CloudTrail, Config, OpsWorks, Service Catalog, Trusted Advisor), Internet of Things (AWS IoT), Contact Center (Amazon Connect), Game Development (Amazon GameLift), and Mobile Services (Mobile Hub, Cognito, Device Farm, Mobile Analytics). On the right side, there is a 'Featured next steps' section with two items: 'Manage your costs' (Get real-time billing alerts based on your cost and usage budgets. [Start now](#)) and 'Get best practices' (Use AWS Trusted Advisor for security, performance, cost and availability best practices. [Start now](#)). Below this is a 'What's new?' section with two announcements: 'Announcing AWS Batch' (Now generally available, AWS Batch enables developers, scientists, and engineers to process large-scale batch jobs with ease. [Learn more](#)) and 'Announcing Amazon Lightsail' (See how this new service allows you to launch and manage your).

Hadoop 3 on AMAZON

Regions



The screenshot shows the AWS Management Console interface. At the top, there is a dark navigation bar with three items: 'Roberto @ 3508-9021-3561' with a dropdown arrow, 'N. Virginia' with an upward arrow, and 'Support' with a dropdown arrow. Below this, a dropdown menu is open, displaying a list of AWS regions. The first item, 'US East (N. Virginia)', is highlighted with an orange vertical bar on its left. The other regions listed are 'US West (Oregon)', 'US West (N. California)', 'EU (Ireland)', 'EU (Frankfurt)', 'Asia Pacific (Singapore)', 'Asia Pacific (Tokyo)', 'Asia Pacific (Sydney)', and 'South America (São Paulo)'. At the bottom of the dropdown menu, there is a link that says 'Learn more about how to use our services.'.

Roberto @ 3508-9021-3561 ▾ N. Virginia ▲ Support ▾

- US East (N. Virginia)**
- US West (Oregon)
- US West (N. California)
- EU (Ireland)
- EU (Frankfurt)
- Asia Pacific (Singapore)
- Asia Pacific (Tokyo)
- Asia Pacific (Sydney)
- South America (São Paulo)

Learn more about how to use our services.

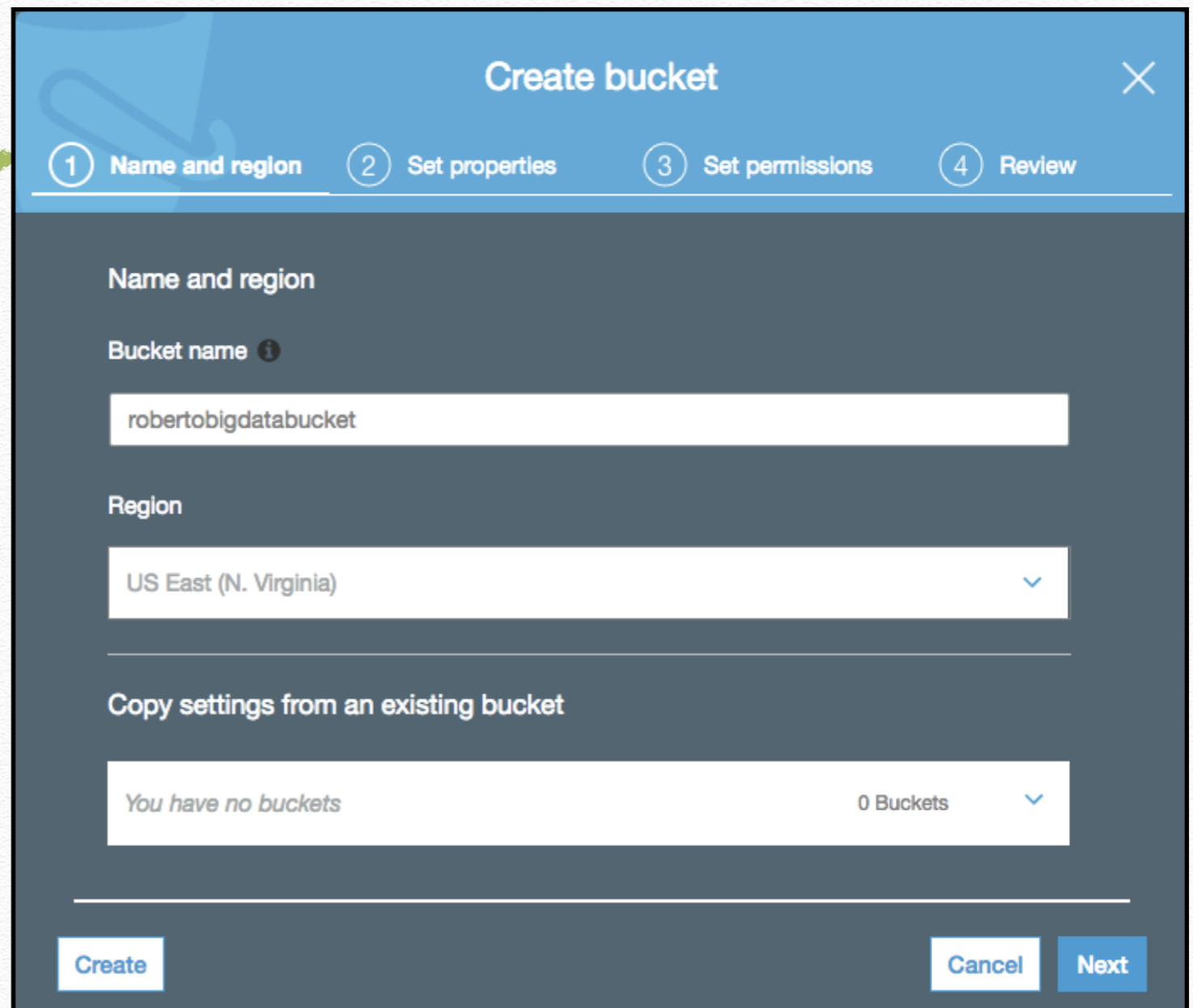
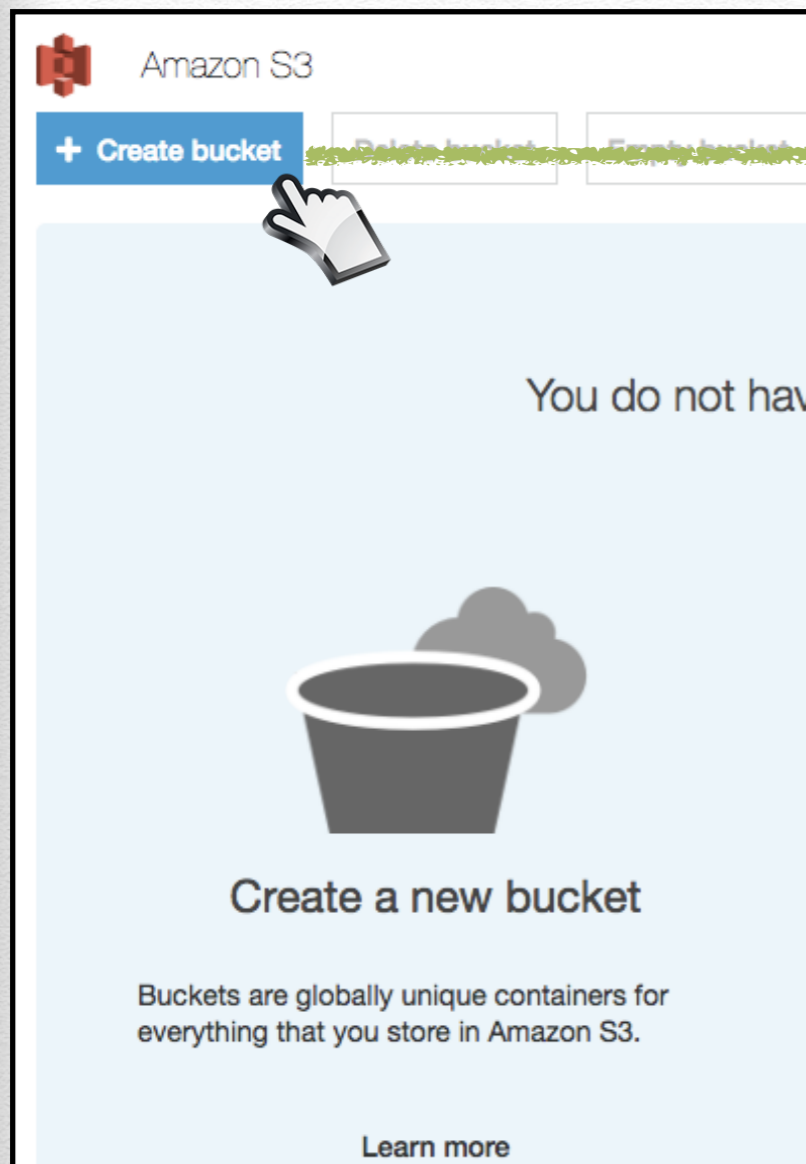
Hadoop 3 on AMAZON

S3 and buckets



Hadoop 3 on AMAZON

S3 and buckets



Hadoop 3 on AMAZON

S3 and buckets

Create bucket ✕

1 Name and region ✓ 2 Set properties ✓ 3 Set permissions 3 4 Review 4

▼ Manage users

User ID ?	Objects ?	Object permissions ?	
rde79(Owner)	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write	✕

▼ Manage public permissions

Group ?	Objects ?	Object permissions ?	
Any authenticated AWS user	<input type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write	
Everyone	<input type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write	

[Previous](#) [Next](#)

Hadoop 3 on AMAZON

S3 and buckets

Create bucket ✕

Name and region Set properties Set permissions **4** Review

Name and region Edit

Bucket name robertobigdatabucket **Region** US East (N. Virginia)

Properties Edit

Versioning	Disabled
Logging	Disabled
Tagging	0 Tags

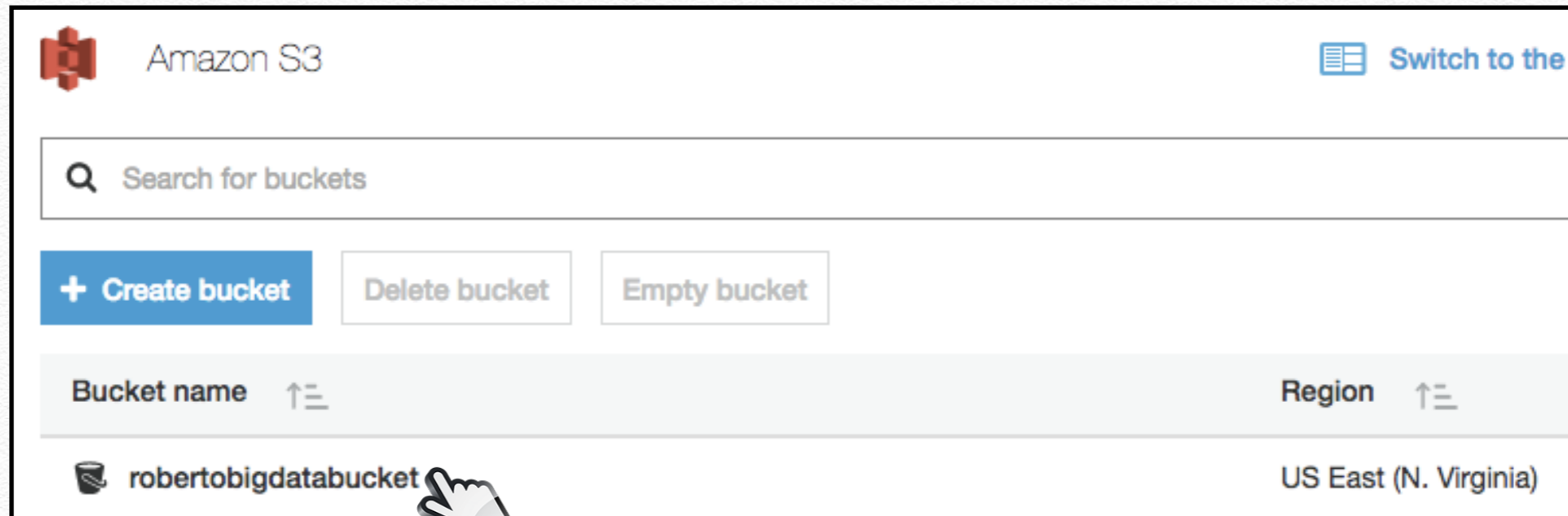
Permissions Edit

Users	1
Public permissions	Disabled

Previous Create bucket

Hadoop 3 on AMAZON

S3 and buckets

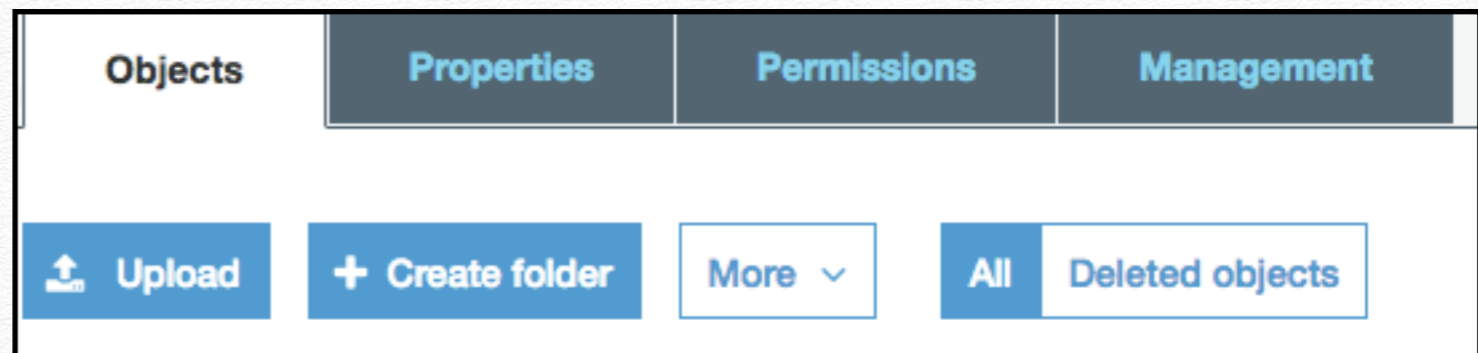


Amazon S3

Search for buckets

+ Create bucket Delete bucket Empty bucket

Bucket name	Region
robertobigdatabucket	US East (N. Virginia)

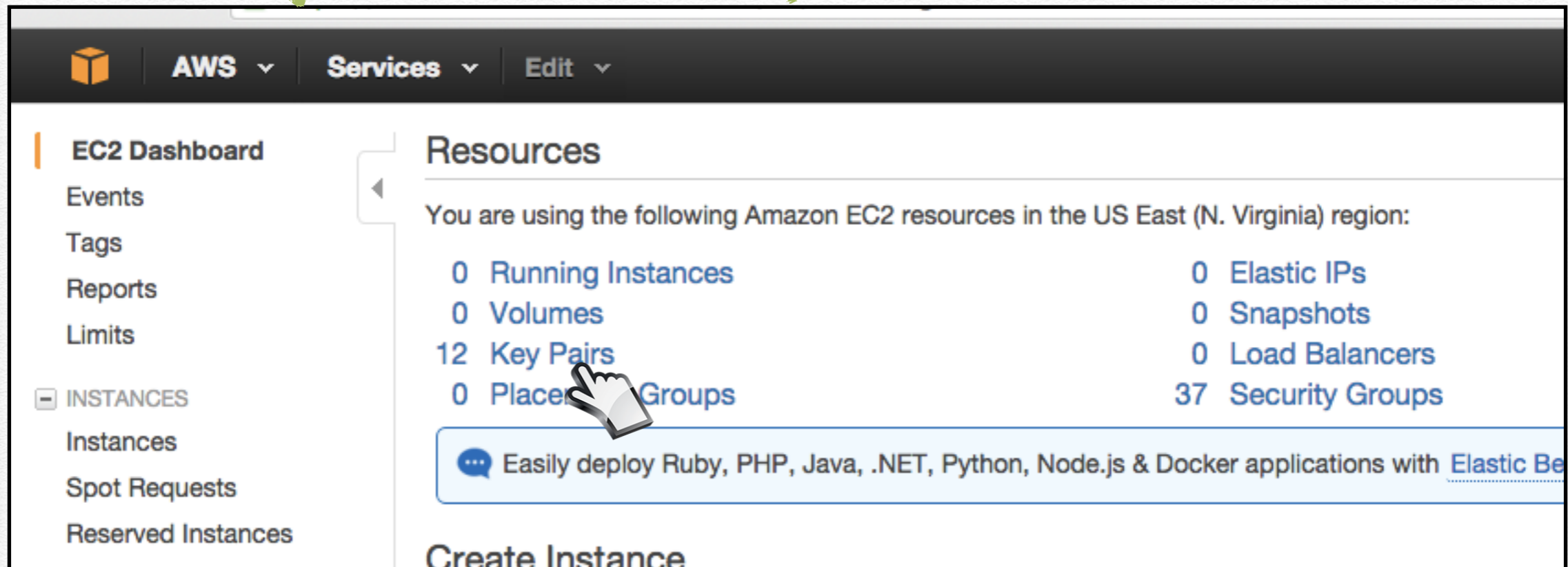
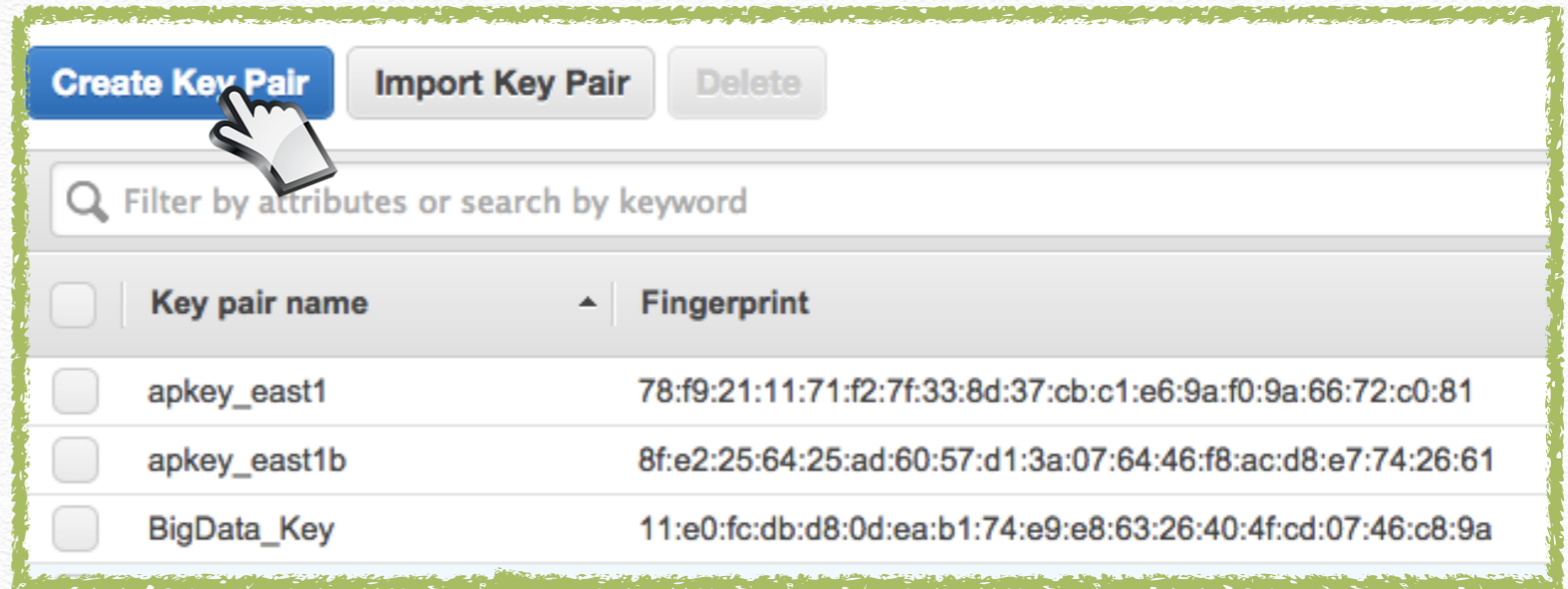
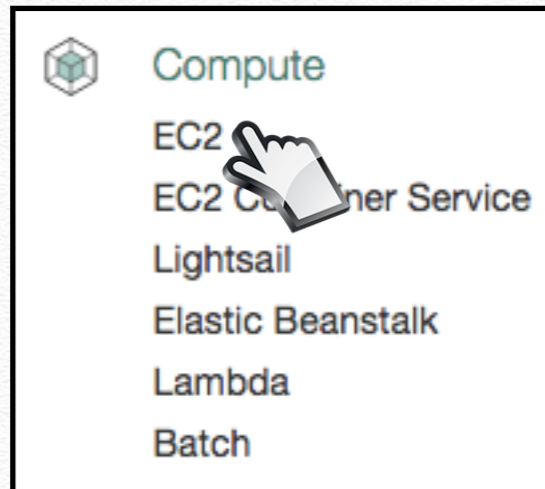


Objects Properties Permissions Management

Upload + Create folder More All Deleted objects

Hadoop 3 on AMAZON

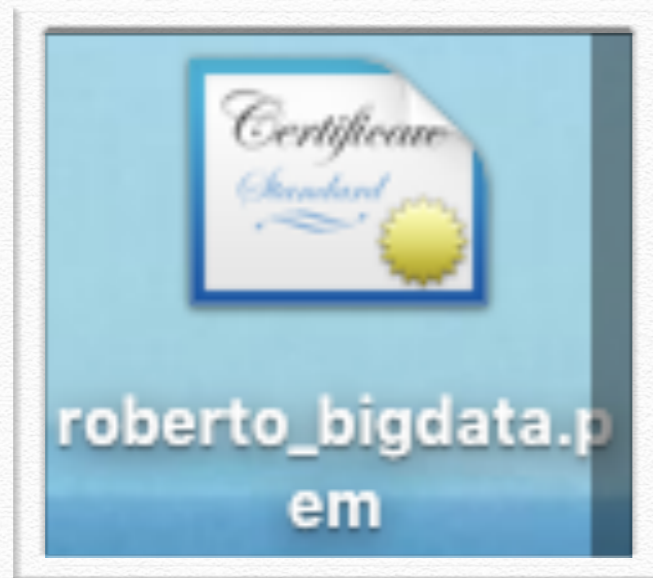
EC2 virtual servers



Hadoop 3 on AMAZON

Key pair name	Fingerprint
<input checked="" type="checkbox"/> roberto_bigdata	40:1d:a1:c1:25:49:d7:74:20:16:6d:54:88:42:45:9d:85:7b:f6:48

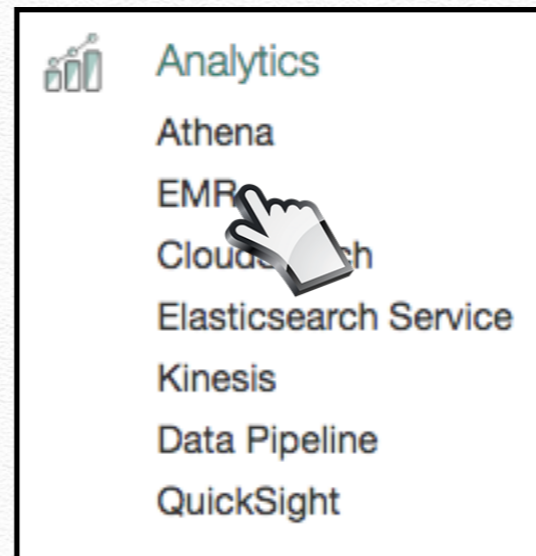
file .pem



```
-----BEGIN RSA PRIVATE KEY-----
MIIeOwIBAAKCAQEA1y1mGpoREC945ImOEqM01VxOX3ICShOOrZTzf1tC9Oj4ReqChRuuF3X7ndvo
Y8LLTWKc4p/qFHjya9jjvIkOTcu4RuB8h8cP/HZhOZu1tZy1/XobA+eIOPKBuhQACATXbs0ERIDD
fC6sLvwurMJHU0OznElbEJsMN/nEK8IYv4uu2gqfzbgoxHibHd6jiHFEk17ymZIB8D0dQhFGwBII
B29grlmyUCC1miA1pp8L3eHBqHXs0Vlw94tLM+ktLgeA4mdP4jvADRLt0quozmp89dj2ZHO8npGq
DMkDUJsXJIBcfHT7dmGBIxATuLdAjwHt56nwchA9dGXCVI7YFIjmtwIDAQABAolBAHkVrAJGNInk
TOVa+c7VFnMF+XhSVRI4RS66xfch1ODahHNbjsz2kZXUJ55iVDhnMI4+osglcwHIOaqkpyq9+VWF
OPfZdVo2k3Fe8EEptSwYnnSgFLmyzdDTBrs+a/IXP1+zcLZXuymTXgMml+FRhi99xoGo6dzDUzHg
lBqHo8OM/wuFk3zStDIRJta1q6T12LyCyxQ5IDeWo5MJIBfiKuTQOE573UILHp2tyIEeYcQKHpBO
BW9jkafue/QramwsyZmyBtRMcZm4wDmRqm0YjfGRP+gNCYD/bXhofqzHlsjFsuxVIMhLIQrYZm9A
iFzdZBSavWw0rxlc2zUdgj3FAskCgYEA/aExJIQwvSdvHb0AedqpTvfmbj9iAKJ1xYwbz04grnph
Nw1eWt40vjVRZaVCaNk33QjIHZJBr5w7BFqcdLuoes2kGMnOFANK7kQ0HAnPDvQtt3gqlvcn6
Eq23C7Dhj+3qKSI2g4b7qSW43eys/FiYMvWwXNfo1L0PDmtGJ+MCgYEA2TA1uMX4Vu3SPDDLXIV
cExRw5RY60dUwnfbqeaITf+VNui+9XZuXSBzQj0782xmeJwzbFHSQFzeZzMABWYL8tUufVutoGD
1SkC0uxcYq/5URD3zw5dgybwIjZMzcTpN3Ug7u6oeNPRYQ9FDcfMMMPeyBjFM2JzGn2KHmcSNh0C
gYADfaj1hDQ/hjlrSVymULif2h11pAWvSi0iaOIBEyWz+o83+MEhswk6zPUP1xRw6Pxx2ESQ/8/F
ff2D4kF6zPQH3zlr1tcemaMOnEtb+Z2zC0Xfv7FlavQZBpyggS+Rw593laklY8koSkVQcKp4s3c2
CeoEWY7xiT40JnqcuOGTQKBgEhXQ32RCz/BjSaBRHdt2iR2d0Gctv9fGf9QNu1naP2B+ie/pJdn
UQbxpl5rrIH1nHNANij9KvZJMjOZLCegLtiqYzrD/5gM04bw+IHclo4rP1wfmOMKd+WZ6MjDN/4
94IOTTLocVsVioces+x8ISobJT/U6FJON3KaYBfyUuGNAoGBAKmZhyvN4M7Dkcj3nCRFahyrMnaz
okfHacmb3vQ3Q0wSP51k8YMM4qXlqGhCWm3wY8jsufUt+Lk5lcfNjTLyAMP98WY5LppZv/80IAZ5
Wltx8fAuMRtoR5ojqp3wo1ZUNzp+0saPPVQ7n6CBWjj3TbzU5w6bR4UI2kiPomt40rEx
-----END RSA PRIVATE KEY-----
```

Hadoop 3 on AMAZON

Elastic Map Reduce



A screenshot of the AWS Elastic MapReduce 'Cluster List' interface. The interface includes a 'Create cluster' button, 'View details', 'Clone', and 'Terminate' buttons. A mouse cursor is pointing at the 'Create cluster' button. Below the buttons is a filter section with a dropdown menu set to 'All clusters' and a search box containing 'Filter clusters ...'. To the right of the search box, it says '1 cluster (all loaded)'. Below the filter section is a table with two columns: 'Name' and 'ID'.

Name	ID
------	----


Hadoop 3 on AMAZON

EMR cluster

General Configuration

Cluster name

Logging ⓘ

S3 folder 

Launch mode **Cluster** ⓘ **Step execution** ⓘ

Hadoop 3 on AMAZON

EMR cluster

cluster

Logging ⓘ

S3 folder

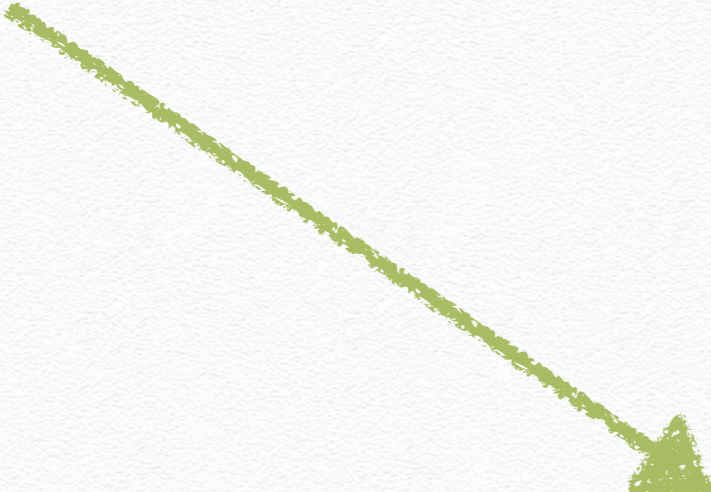
Cluster ⓘ Step execution ⓘ

Select S3 Folder

< > URL:

robertobigdatabucket/

Cancel Select




Hadoop 3 on AMAZON

EMR cluster

General Configuration

Cluster name

Logging ⓘ

S3 folder 

Launch mode **Cluster** ⓘ

With Cluster, EMR creates a cluster with a set of specified applications.


Hadoop 3 on AMAZON



EMR cluster

General Configuration

Cluster name

Logging 

S3 folder 

Launch mode Cluster  Step execution 

With Step execution, EMR will create a cluster, execute added steps and terminate when done.

Hadoop 3 on AMAZON

EMR cluster

Software configuration

Vendor Amazon MapR

Release  

- Applications**
- Core Hadoop: Hadoop 2.7.3 with Ganglia 3.7.2, Hive 2.1.1, Hue 3.11.0, Mahout 0.12.2, Pig 0.16.0, and Tez 0.8.4
 - HBase: HBase 1.3.0 with Ganglia 3.7.2, Hadoop 2.7.3, Hive 2.1.1, Hue 3.11.0, Phoenix 4.9.0, and ZooKeeper 3.4.9
 - Presto: Presto 0.166 with Hadoop 2.7.3 HDFS and Hive 2.1.1 Metastore
 - Spark: Spark 2.1.0 on Hadoop 2.7.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.0

Hadoop 3 on AMAZON

EMR cluster

Software configuration

Vendor Amazon MapR

Release MapR - M3 - 4.0.2

Applications All applications: Hadoop 2.4.0, Hive 0.13.1, and Pig 0.12.0



<http://doc.mapr.com/display/MapR/MapR+Overview>

MapR 5.0 Documentation / Home Tools ▾

MapR Overview

MapR is a complete enterprise-grade distribution for Apache Hadoop. The MapR Distribution for Apache Hadoop has been engineered to improve Hadoop's reliability, performance, and ease of use. The MapR distribution provides a full Hadoop stack that includes the MapR File System (MapR-FS), MapReduce, a complete Hadoop ecosystem, and the MapR Control System user interface. You can use MapR with Apache Hadoop, HDFS, and MapReduce APIs.

The following image displays a high-level view of the MapR Distribution for Apache Hadoop:

The MapR distribution provides several unique features that address common concerns with Apache Hadoop:

Hadoop 3 on AMAZON

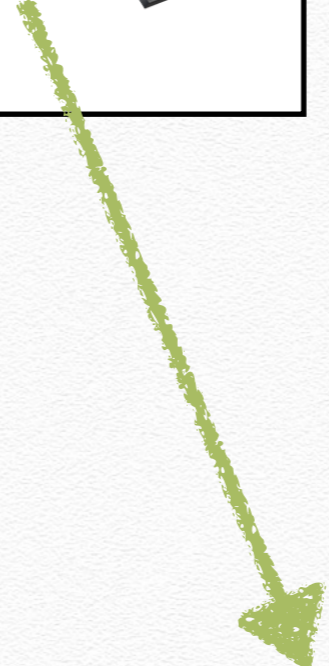
EMR cluster

Hardware configuration

Instance type 

Number of instances (1 master and 2 core nodes)

- m2.xlarge
- m2.2xlarge
- m2.4xlarge
- Storage Optimized
- d2.xlarge
- d2.2xlarge
- d2.4xlarge
- d2.8xlarge
- hs1.8xlarge
- i2.xlarge
- i2.2xlarge
- i2.4xlarge
- i2.8xlarge
- GPU Instances
- g2.2xlarge
- Storage Optimized (Previous Generation)
- hi1.4xlarge
- General Purpose (Previous Generation)
- m1.medium
- m1.large
- m1.xlarge
- General Purpose
- ✓ m3.xlarge
- m3.2xlarge
- m4.large
- m4.xlarge
- m4.2xlarge
- m4.4xlarge
- m4.10xlarge
- m4.16xlarge
- Memory Optimized
- r3.xlarge
- r3.2xlarge
- r3.4xlarge
- r3.8xlarge
- r4.xlarge
- r4.2xlarge
- r4.4xlarge
- r4.8xlarge
- r4.16xlarge



Hadoop 3 on AMAZON

EMR cluster

Security and access


EC2 key pair   [Learn how to create an EC2 key pair.](#)

Permissions Default Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [EMR_DefaultRole](#) 

EC2 instance profile [EMR_EC2_DefaultRole](#) 



✓ Choose an option
Proceed without an EC2 key pair
roberto_bigdata

Hadoop 3 on AMAZON


EMR cluster

Cluster: My cluster **Starting** Configuring cluster software

Connections: [Enable Web Connection](#) – Hue, Ganglia, Resource Manager ... (View All)

Master public DNS: ec2-54-159-62-237.compute-1.amazonaws.com [SSH](#)

Tags: -- [View All / Edit](#)

Summary	Configuration Details
ID: j-47LWBSL8AKPS	Release label: emr-5.4.0
Creation date: 2017-04-04 11:55 (UTC+2)	Hadoop distribution: Amazon 2.7.3
Elapsed time: 2 minutes	Applications: Ganglia 3.7.2, Hive 2.1.1, Hue 3.11.0, Mahout 0.12.2, Pig 0.16.0, Tez 0.8.4
Auto-terminate: No	Log URI: s3://robertobigdatabucket/ 
Termination protection: Off Change	EMRFS consistent view: Disabled

Network and Hardware	Security and Access
Availability zone: us-east-1c	Key name: roberto_bigdata
Subnet ID: --	EC2 instance profile: EMR_EC2_DefaultRole
Master: Bootstrapping 1 m3.xlarge	EMR role: EMR_DefaultRole
Core: Provisioning 2 m3.xlarge	Visible to all users: All Change
Task: --	Security groups for sg-98da73f3 (ElasticMapReduce-Master: master)

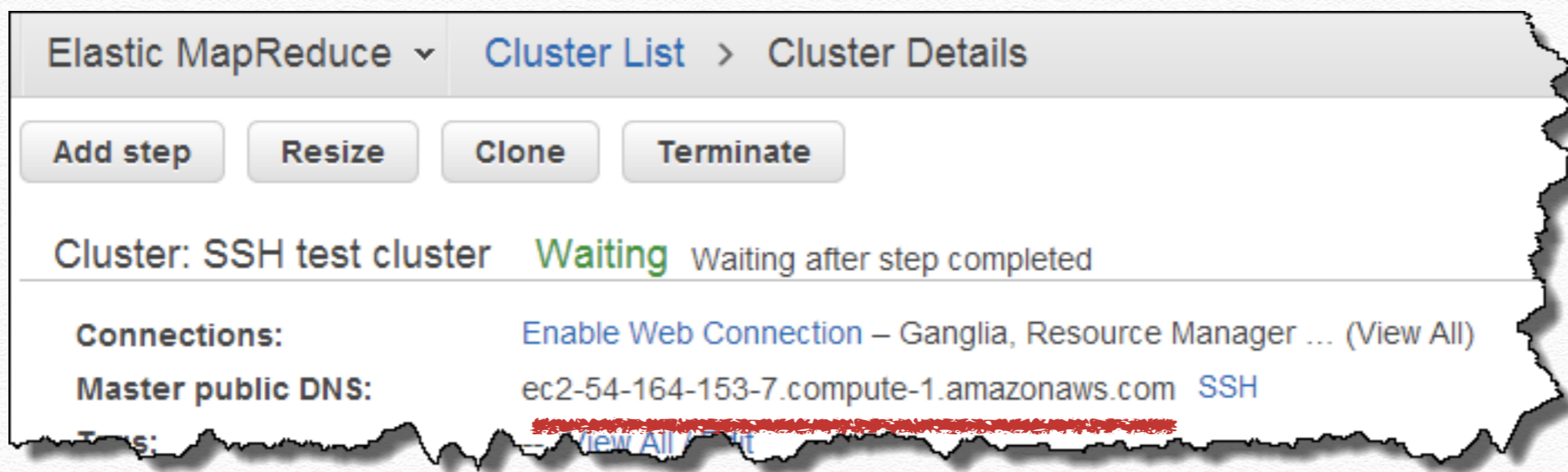
Hadoop 3 Running on AWS

- ❖ **Authorization** of the .pem file

```
$:~ chmod 400 roberto_bigdata.pem
```

- ❖ **Upload files (data and your personal jars)** on hadoop of EMR cluster:

```
$:~ scp -i <file .pem> <file> hadoop@<DNS_EMR_CLUSTER>:~
```



The screenshot shows the AWS EMR console interface. At the top, there are navigation tabs: 'Elastic MapReduce' (with a dropdown arrow), 'Cluster List', and 'Cluster Details'. Below the tabs are four buttons: 'Add step', 'Resize', 'Clone', and 'Terminate'. The main content area displays the cluster name 'SSH test cluster' followed by the state 'Waiting' in green text and the message 'Waiting after step completed'. Underneath, there is a 'Connections:' section with a link 'Enable Web Connection - Ganglia, Resource Manager ... (View All)'. Below that is the 'Master public DNS:' field with the value 'ec2-54-164-153-7.compute-1.amazonaws.com' and the protocol 'SSH'. At the bottom left, there is a 'Tags:' section with a link '(View All)'. A red scribble is present over the bottom part of the screenshot.

```
$:~ scp -i roberto_bigdata.pem ./example_data/words.txt  
hadoop@ec2-54-164-153-7.compute-1.amazonaws.com:~
```

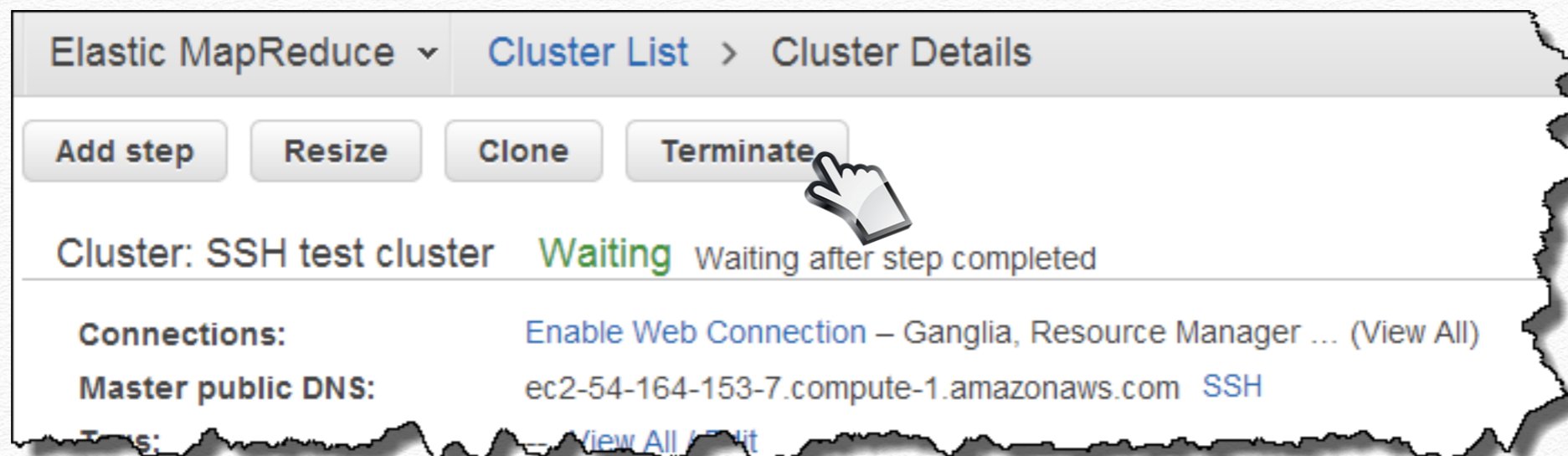

Hadoop 3 Running on AWS

- ❖ **Connection to** hadoop of EMR cluster:

```
$:~ ssh hadoop@<DNS_EMR_CLUSTER> -i <file .pem>
```

```
$:~ ssh hadoop@ec2-54-164-153-7.compute-1.amazonaws.com  
-i roberto_bigdata.pem
```

- ❖ Then you can execute MR jobs as in your local machine
- ❖ Finally **TERMINATE** your cluster





Hadoop 3 Configuration and First Examples

Big Data - 25/03/2020