

AI VIETNAM
All-in-One Course
(TA Session)

Big Data

Extra



AI VIET NAM
@aivietnam.edu.vn

Dinh-Thang Duong – TA
Truong-Binh Duong – STA

Outline

- Introduction
- Hadoop
- Spark
- Question

Getting Started

◆ Objectives



In this session, we will discuss about:

- Introduction to big data.
- Introduction to Hadoop.
- How to install virtual machine.
- How to install and use Hadoop on virtual machine.
- Introduction to spark and pyspark.

Introduction

Introduction

◆ Getting Started

3 Important Statistics About How Much Data Is Created Every Day



1 How much data is generated every minute?

41,666,667

messages shared by WhatsApp users

1,388,889

video / voice calls made by people worldwide

404,444

hours of video streamed by Netflix users

347,222

stories posted by Instagram users

150,000

messages shared by Facebook users

147,000

photos shared by Facebook users

2 Estimated Data Consumption from 2021 to 2024



3 Data Growth in 2021

2 TRILLION

searches on Google by the end of 2021

1.134 TRILLION MB

volume of data created every day

3,026,626

emails sent every second, 67% of which are spam

278,108 PETABYTES

global IP data per month by the end of 2021

230,000

new malware versions created every day

82%

share of video in total global internet traffic at the end of 2021

UNIT	VALUE
bit	1 bit
byte (B)	8 bits
kilobyte (KB)	1024 bytes
megabyte (MB)	1024 kilobytes
gigabyte (GB)	1024 megabytes
terabyte (TB)	1024 gigabytes
petabyte (PB)	1024 terabytes
exabyte (EB)	1024 petabytes
zettabyte (ZB)	1024 exabytes
yottabyte (YB)	1024 zettabytes

Introduction

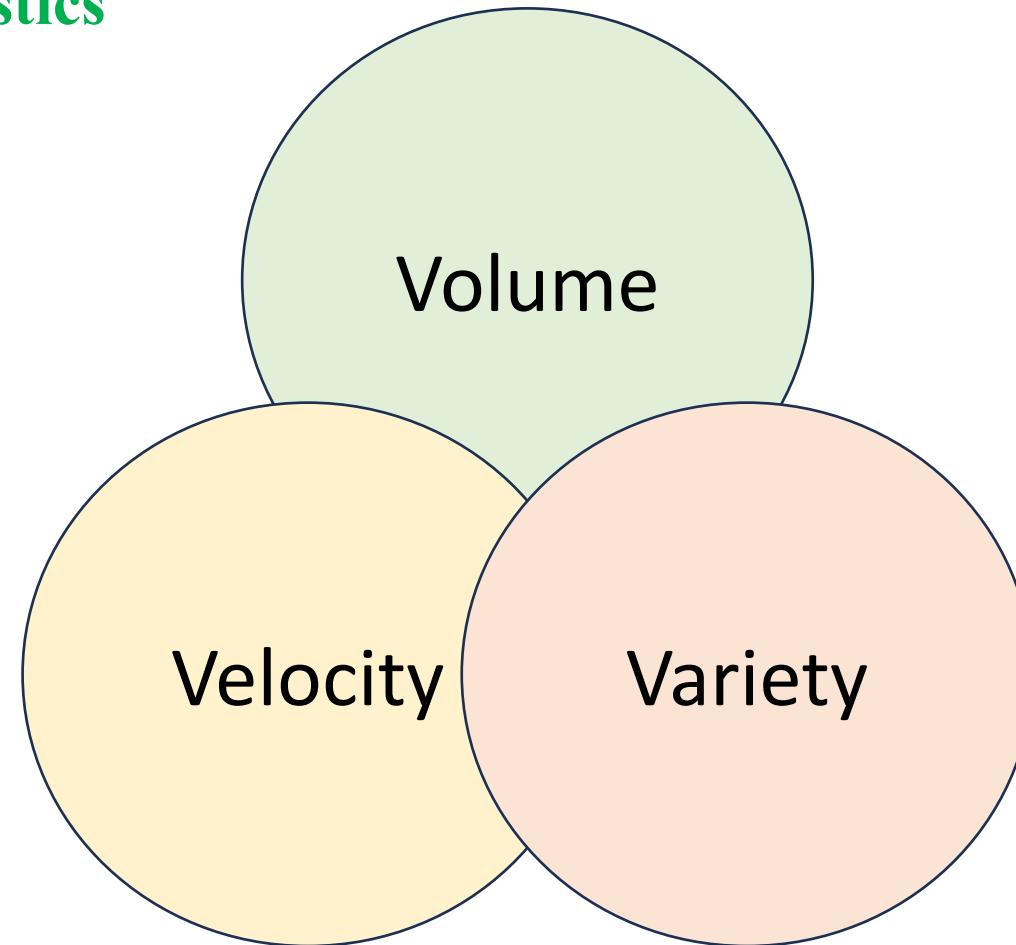
◆ Big Data Definition



Big Data: A collection of datasets that is large, complex that make it very difficult to process using traditional data processing applications.

Introduction

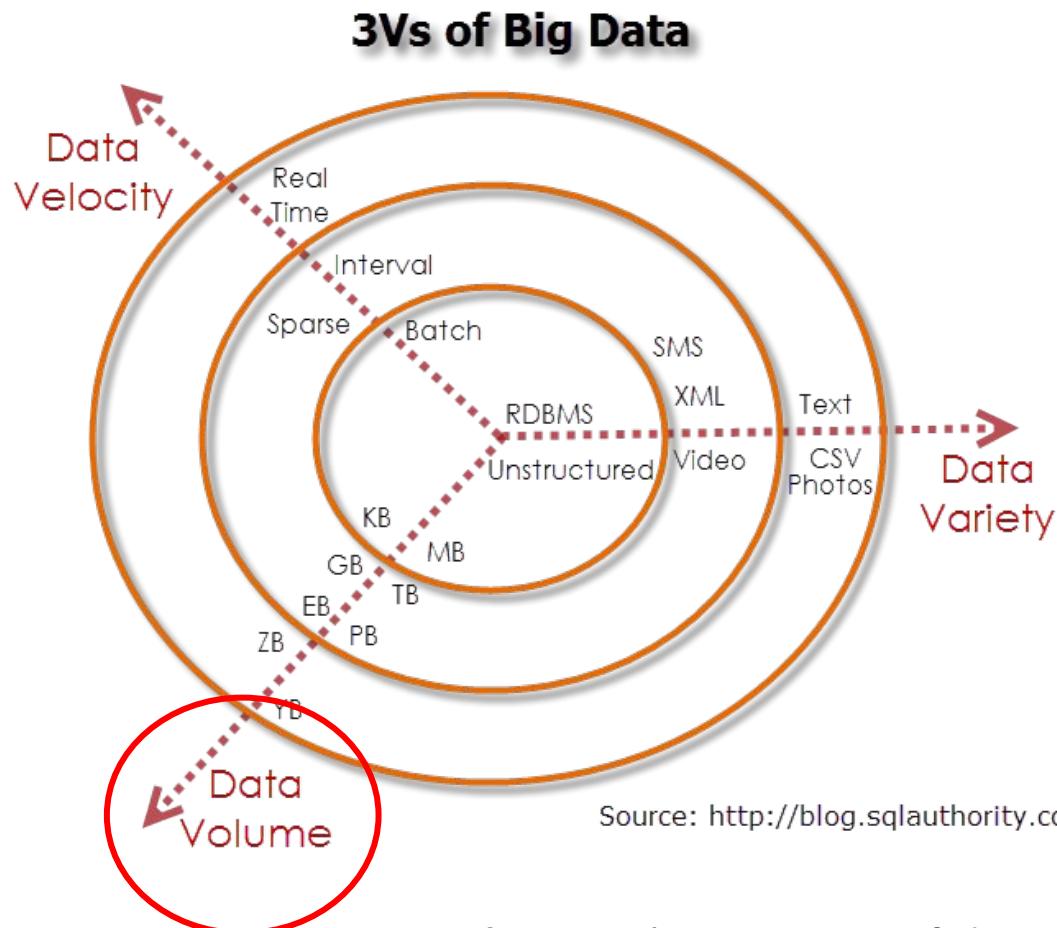
◆ Big Data Characteristics



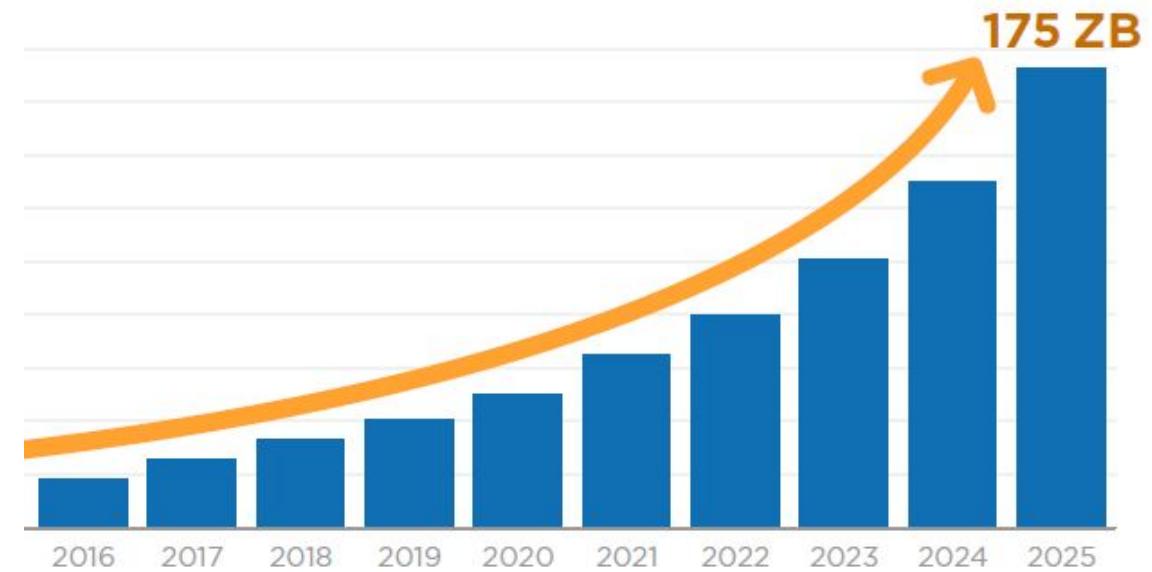
3V of Big Data

Introduction

◆ Big Data Characteristics



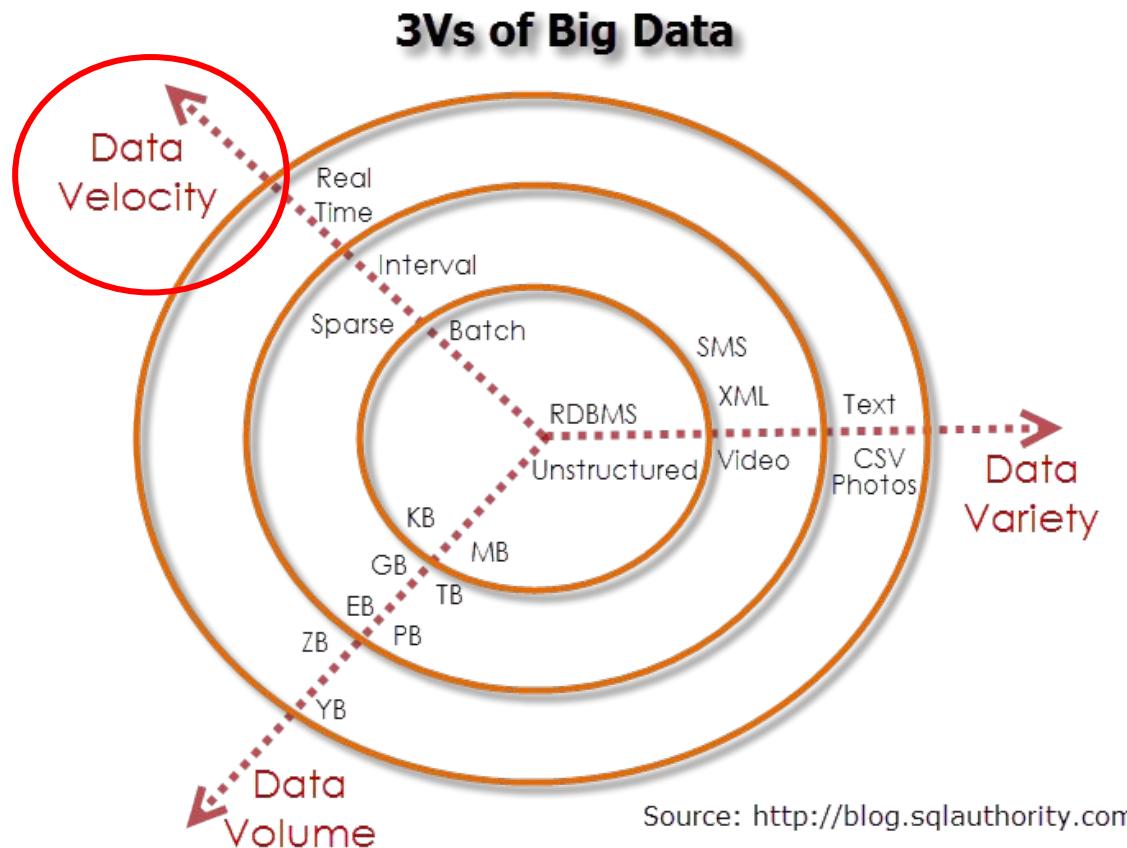
Volume: The amount of data



The amount of data generated on Internet each year

Introduction

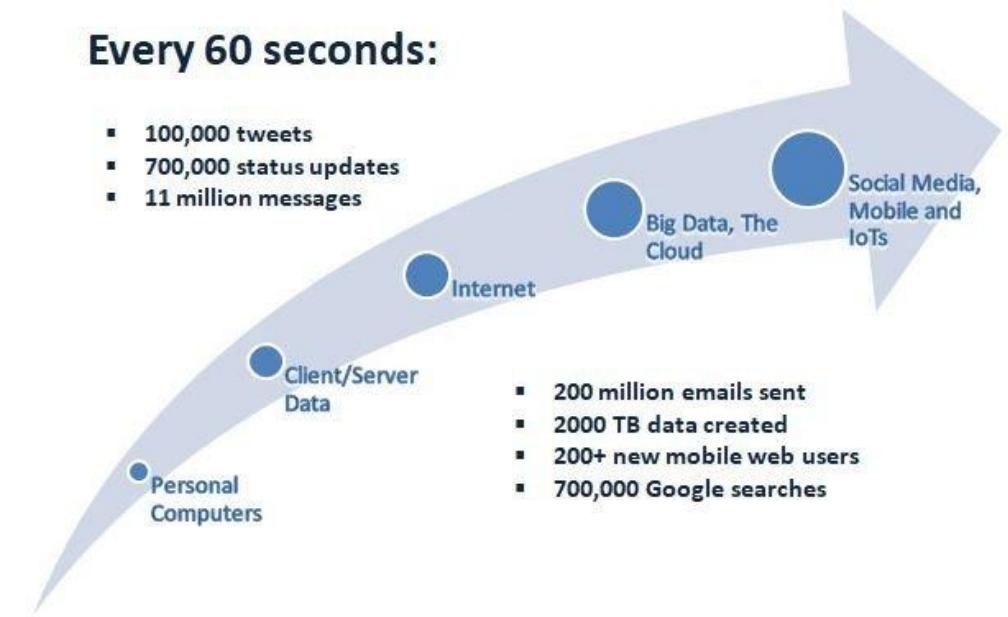
◆ Big Data Characteristics



Velocity: The growth of data

Every 60 seconds:

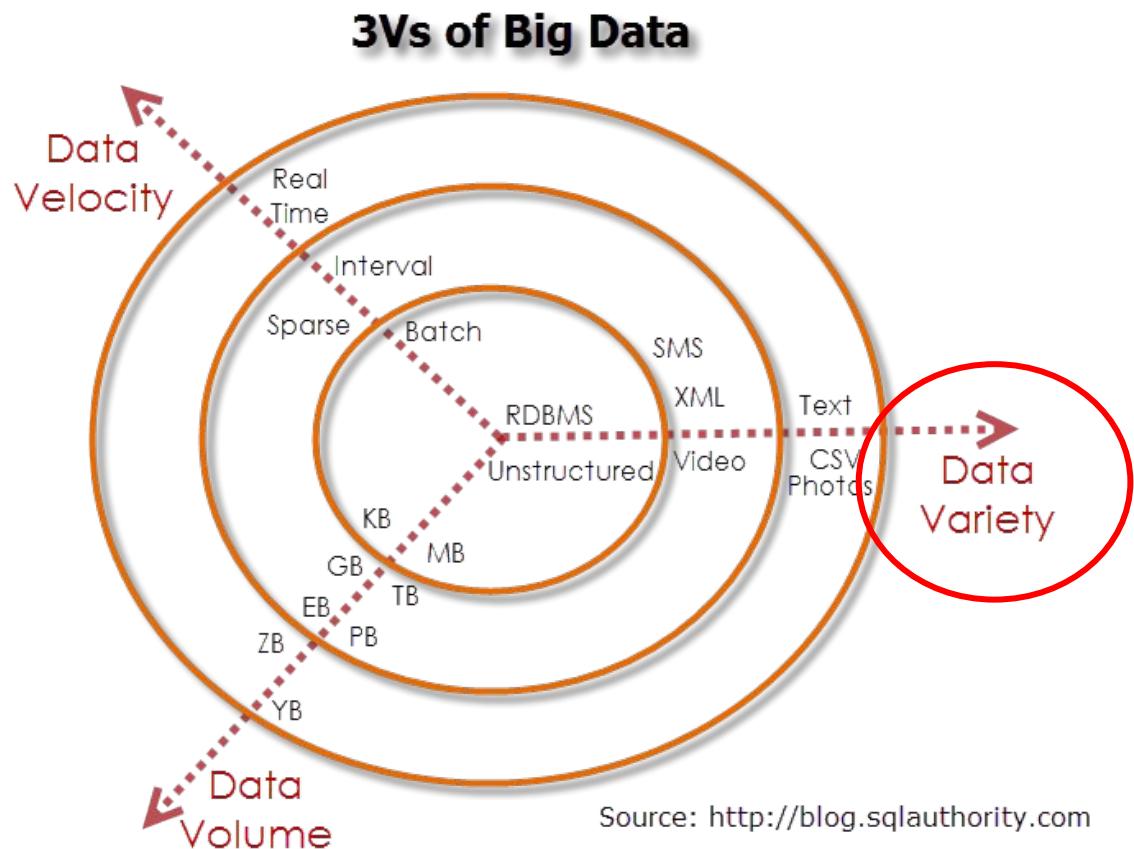
- 100,000 tweets
- 700,000 status updates
- 11 million messages



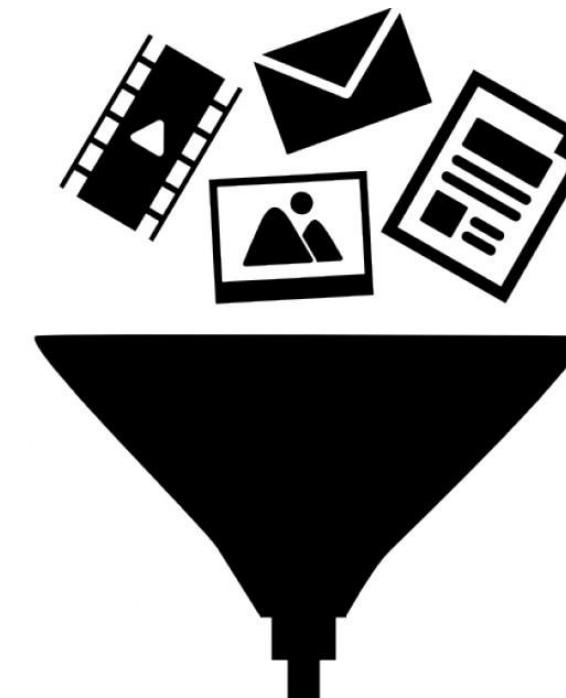
Data generated on Internet every 1 minute

Introduction

◆ Big Data Characteristics



Variety: The diversity of data

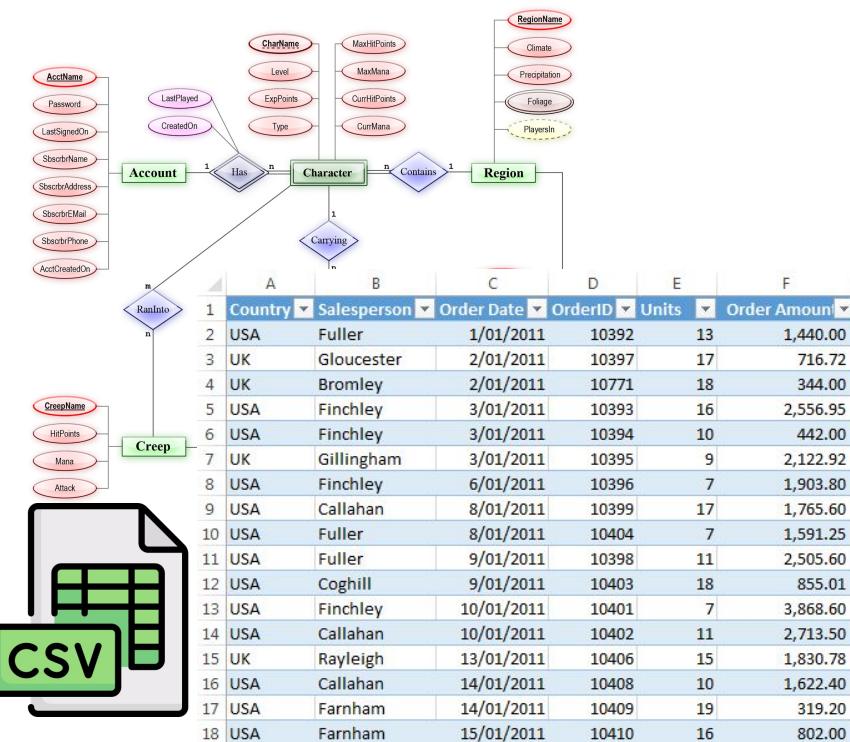


Lots of types of data to be processed

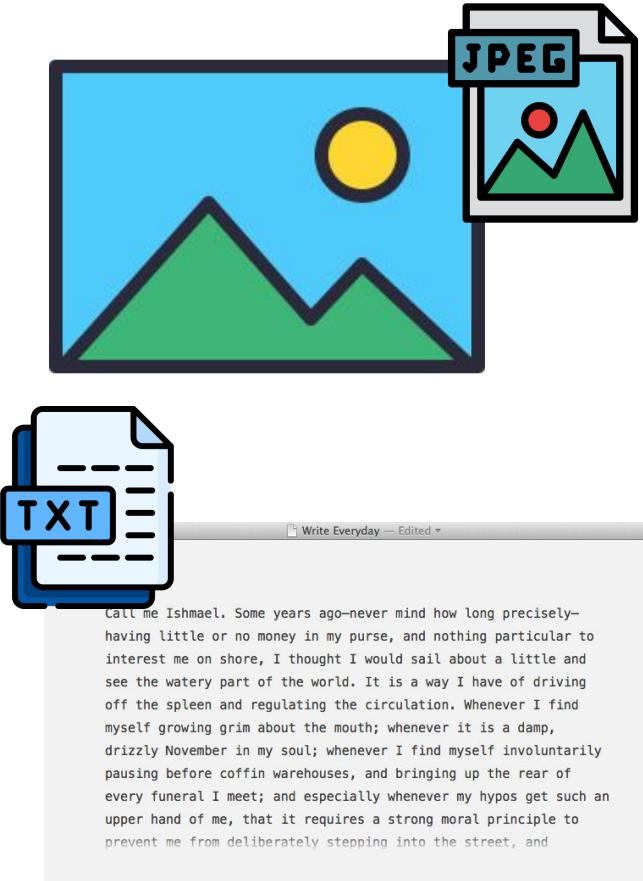
Introduction

◆ Types of big data

Structured Data



Unstructured Data



Semi-structured Data

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Numbers</title>
<link rel="stylesheet" href="css/my_file.css">
</head>
<body>
</body>
</html>
```

my_file.css

```
/* CSS styles */
```

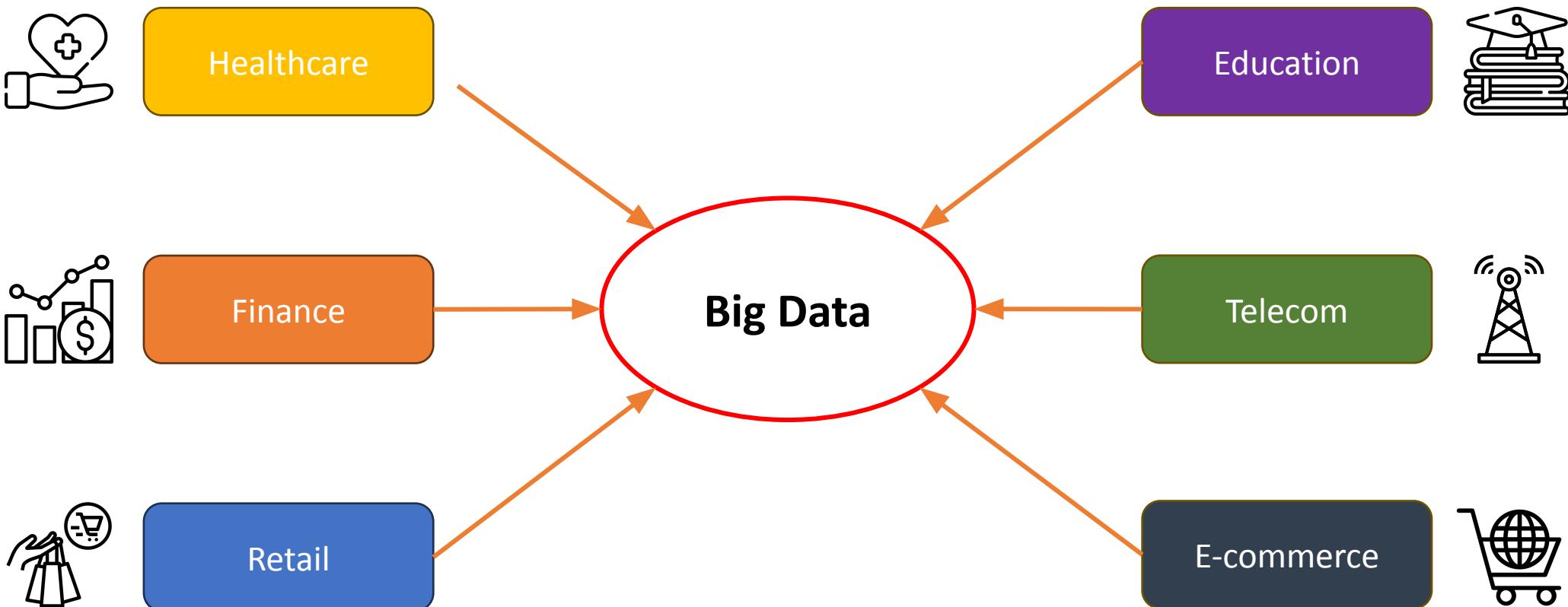
HTML

```
<?xml version="1.0" encoding="UTF-8"?>
- <EmployeeData>
  - <employee id="34594">
    <firstName>Heather</firstName>
    <lastName>Banks</lastName>
    <hireDate>1/19/1998</hireDate>
    <deptCode>BB001</deptCode>
    <salary>72000</salary>
  </employee>
  - <employee id="34593">
    <firstName>Tina</firstName>
    <lastName>Young</lastName>
    <hireDate>4/1/2010</hireDate>
    <deptCode>BB001</deptCode>
    <salary>65000</salary>
  </employee>
</EmployeeData>
```

XML

Introduction

◆ Big Data Applications



Introduction

❖ Tools for Big Data



Hadoop

Hadoop

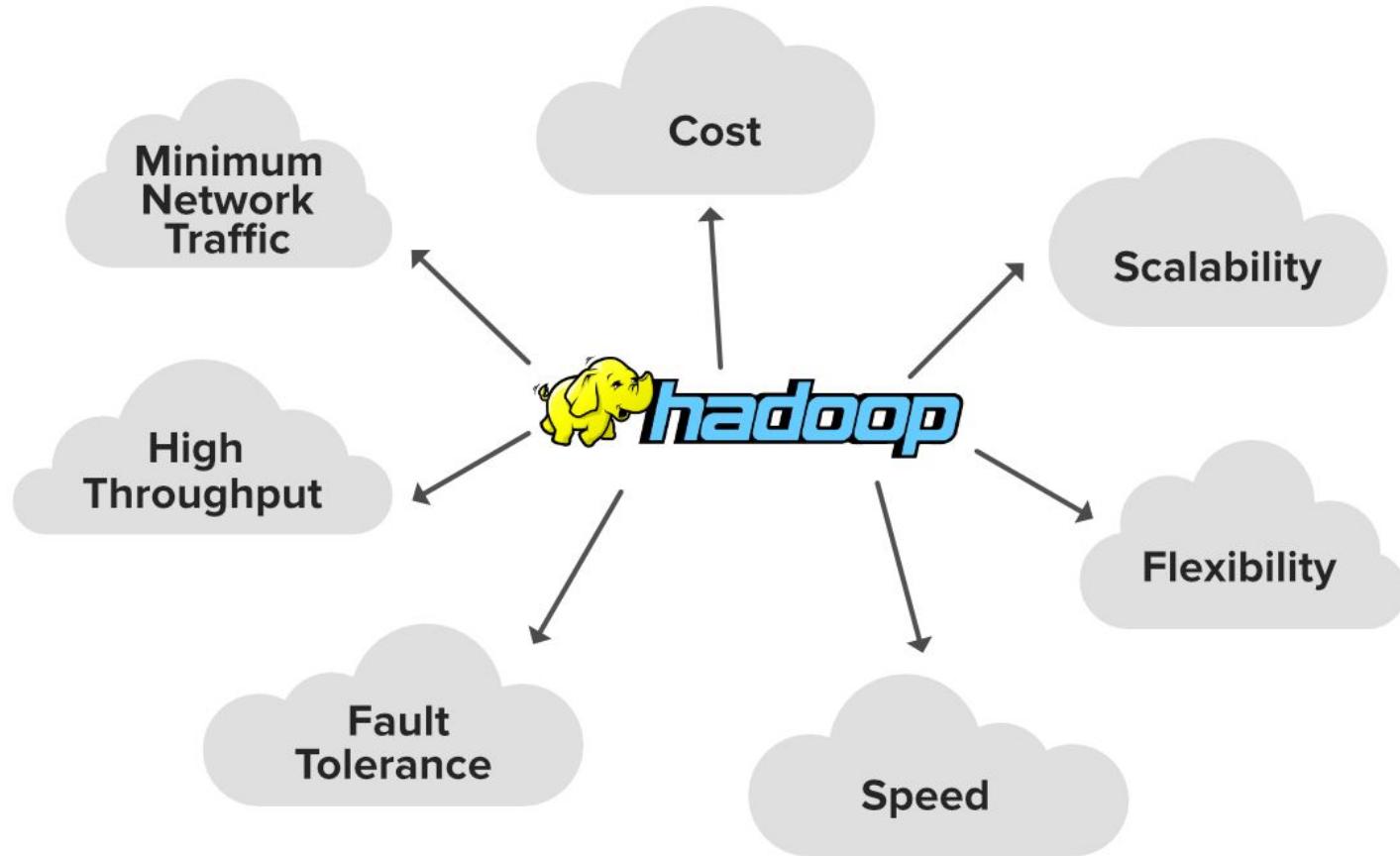
❖ Introduction



Apache Hadoop: an open-source framework that is used to **efficiently store and process large datasets** ranging in size from gigabytes to petabytes of data.

Hadoop

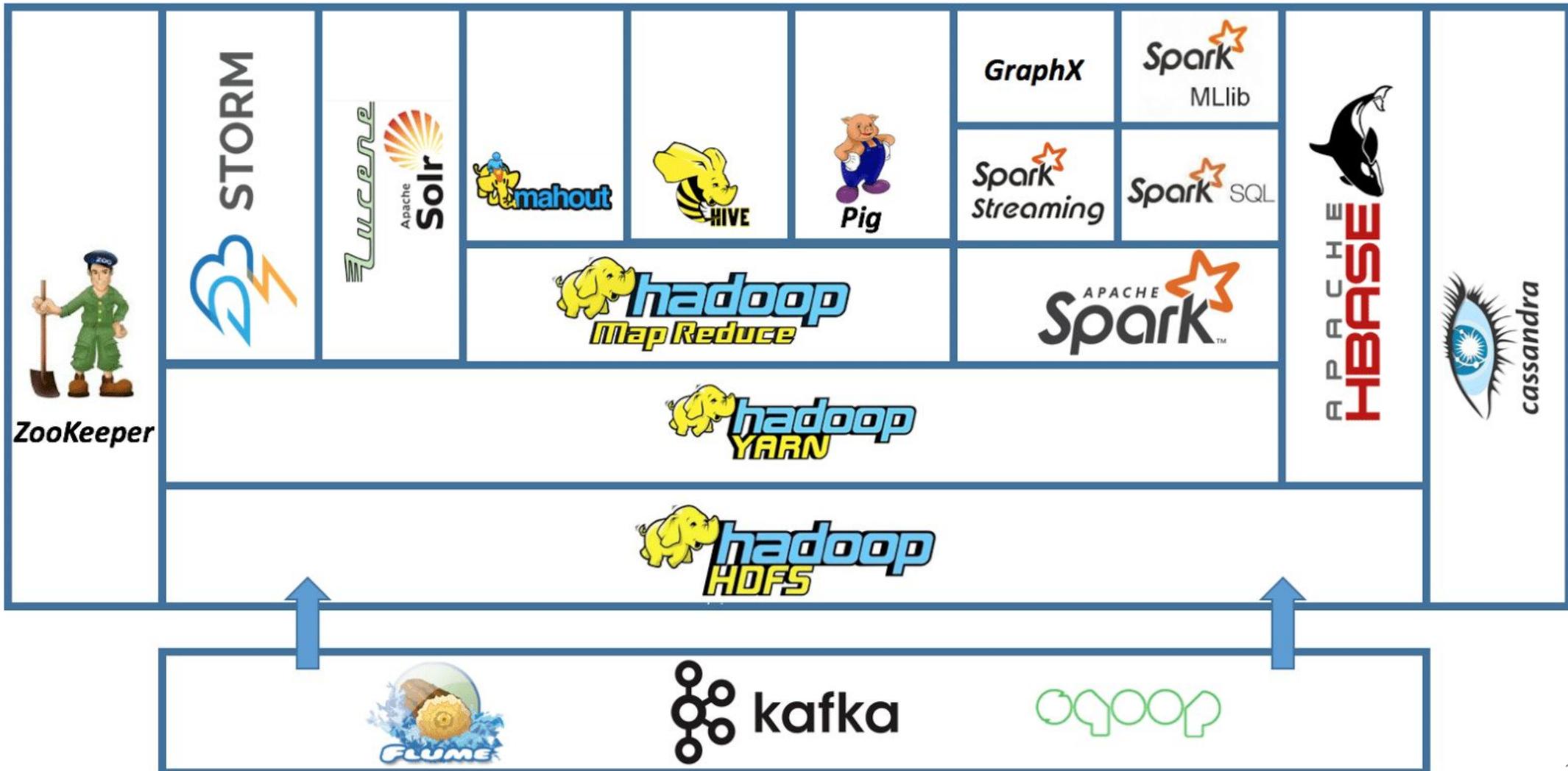
❖ Introduction



Hadoop Advantages

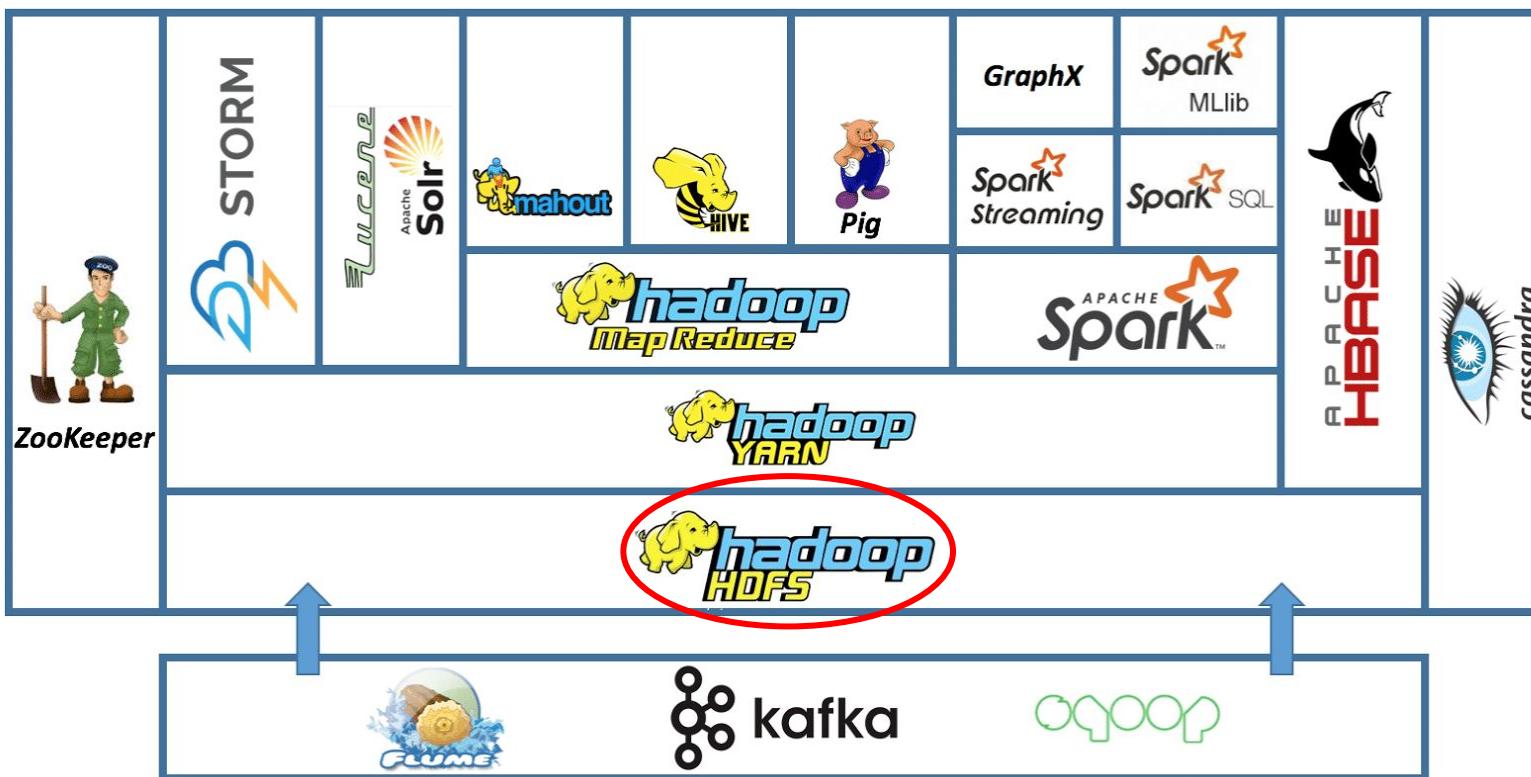
Hadoop

◆ Hadoop Ecosystem



Hadoop

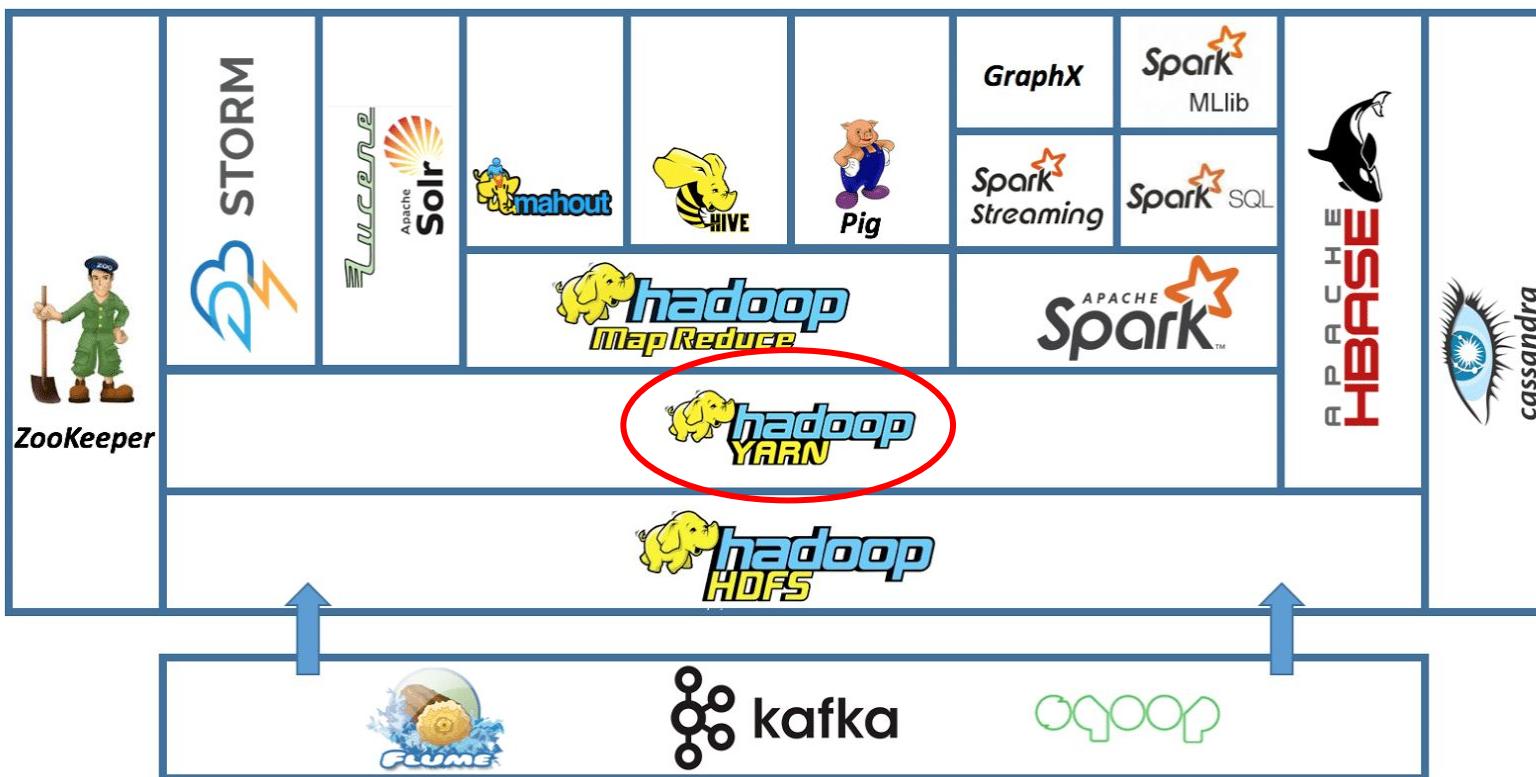
◆ Hadoop Ecosystem: Hadoop HDFS



Hadoop HDFS (Hadoop Distributed File System): A distributed file system that handles large data sets running on commodity hardware.

Hadoop

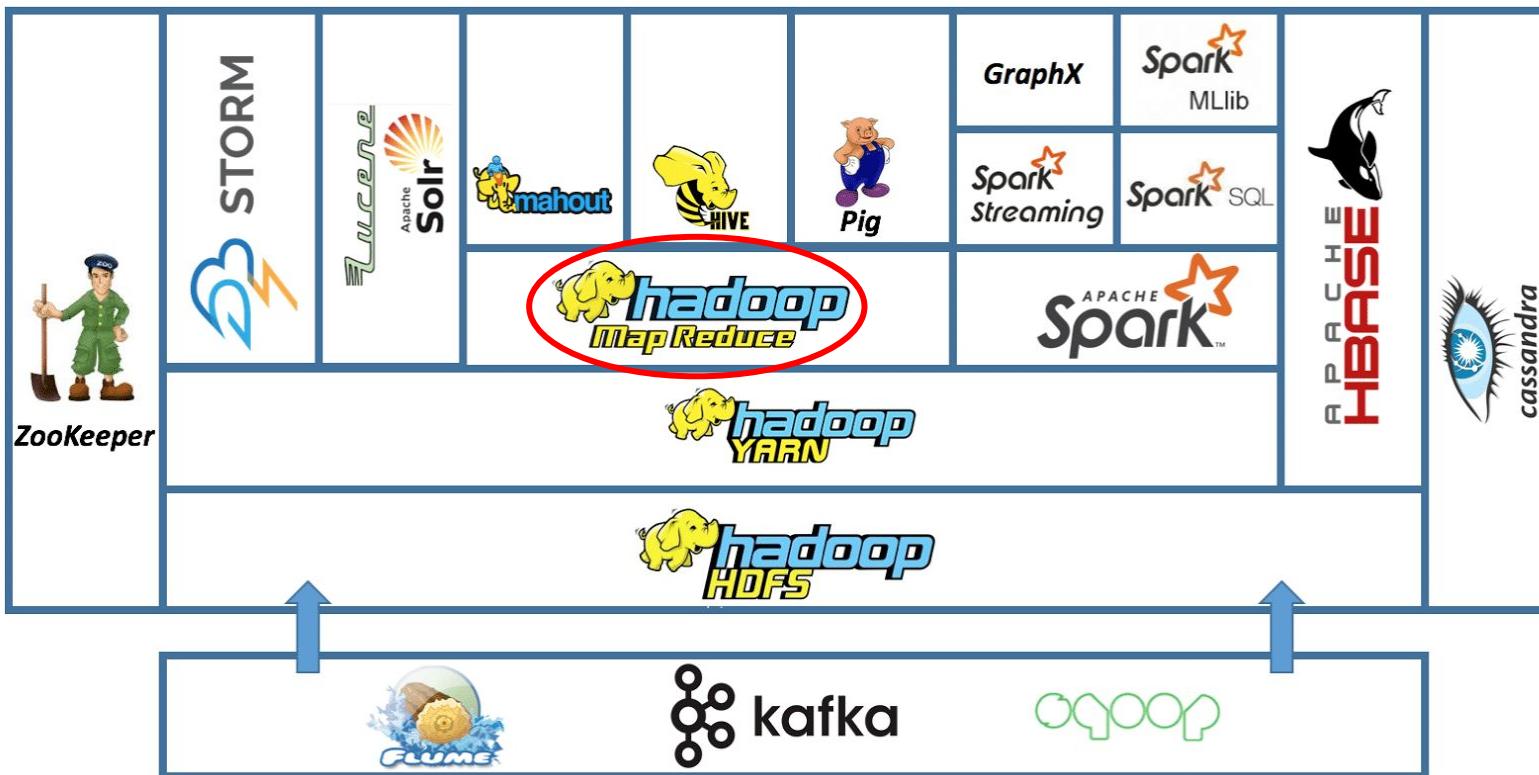
◆ Hadoop Ecosystem: Hadoop YARN



Hadoop YARN (Yet Another Resource Negotiator): the resources management and job scheduling technology.

Hadoop

❖ Hadoop Ecosystem: Hadoop MapReduce



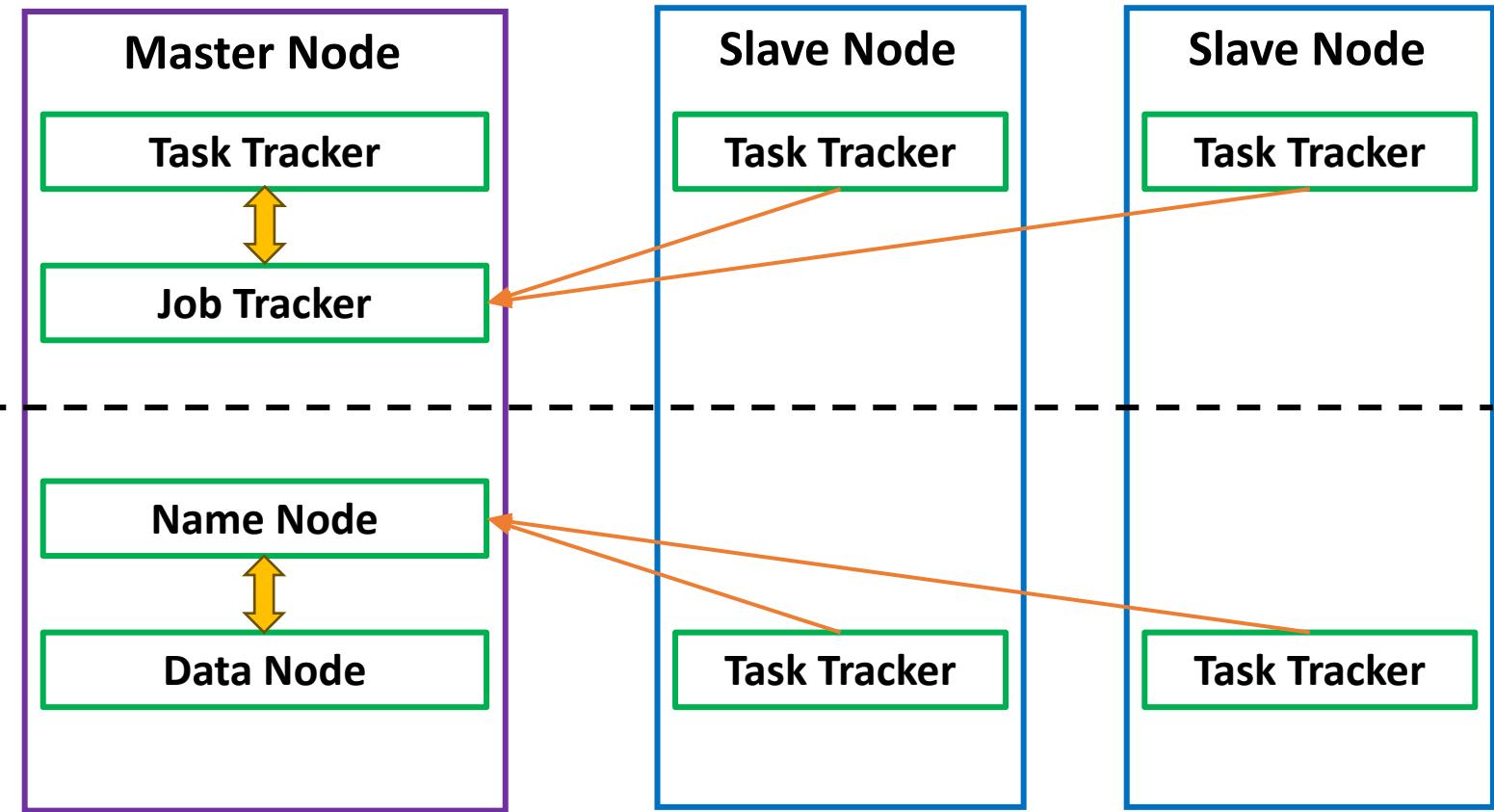
Hadoop MapReduce: A highly sufficient methodology for parallel processing of huge volumes of data.

Hadoop

❖ Hadoop Architecture

MapReduce Layer

HDFS Layer



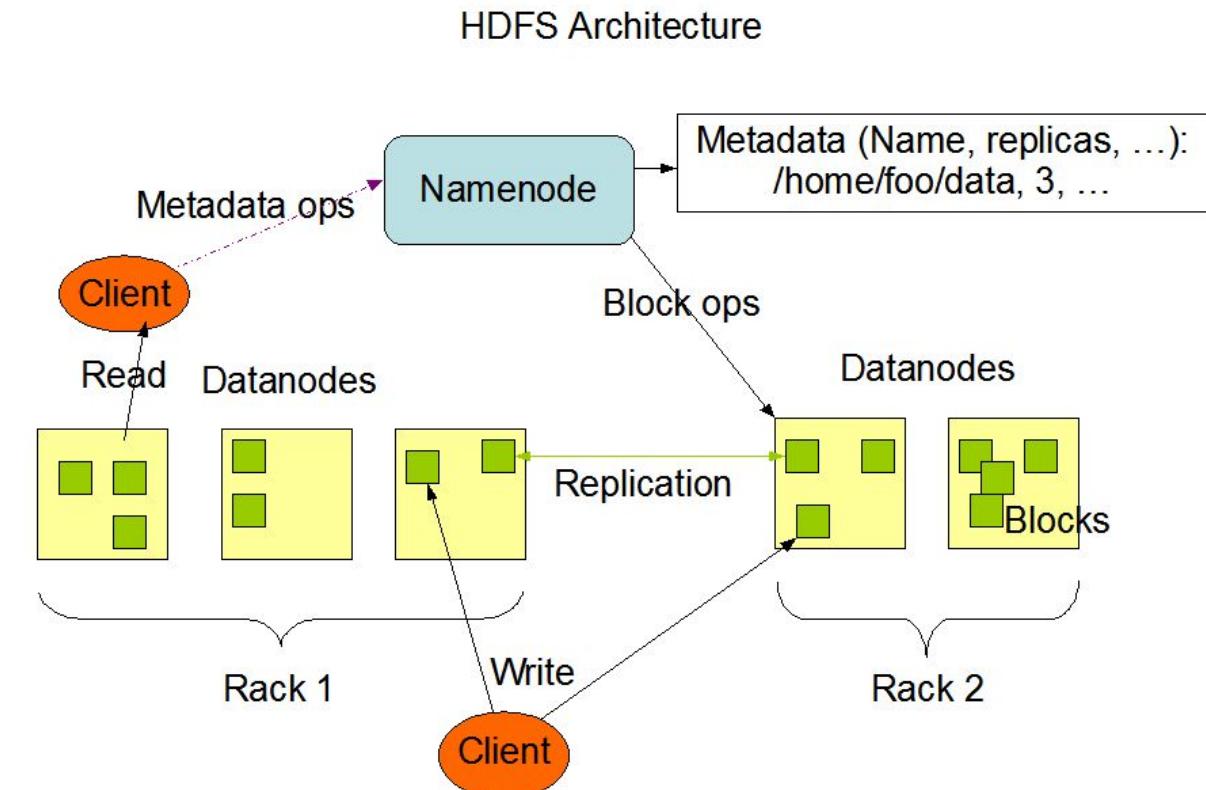
High-level Architecture of Hadoop

Hadoop

❖ Hadoop HDFS

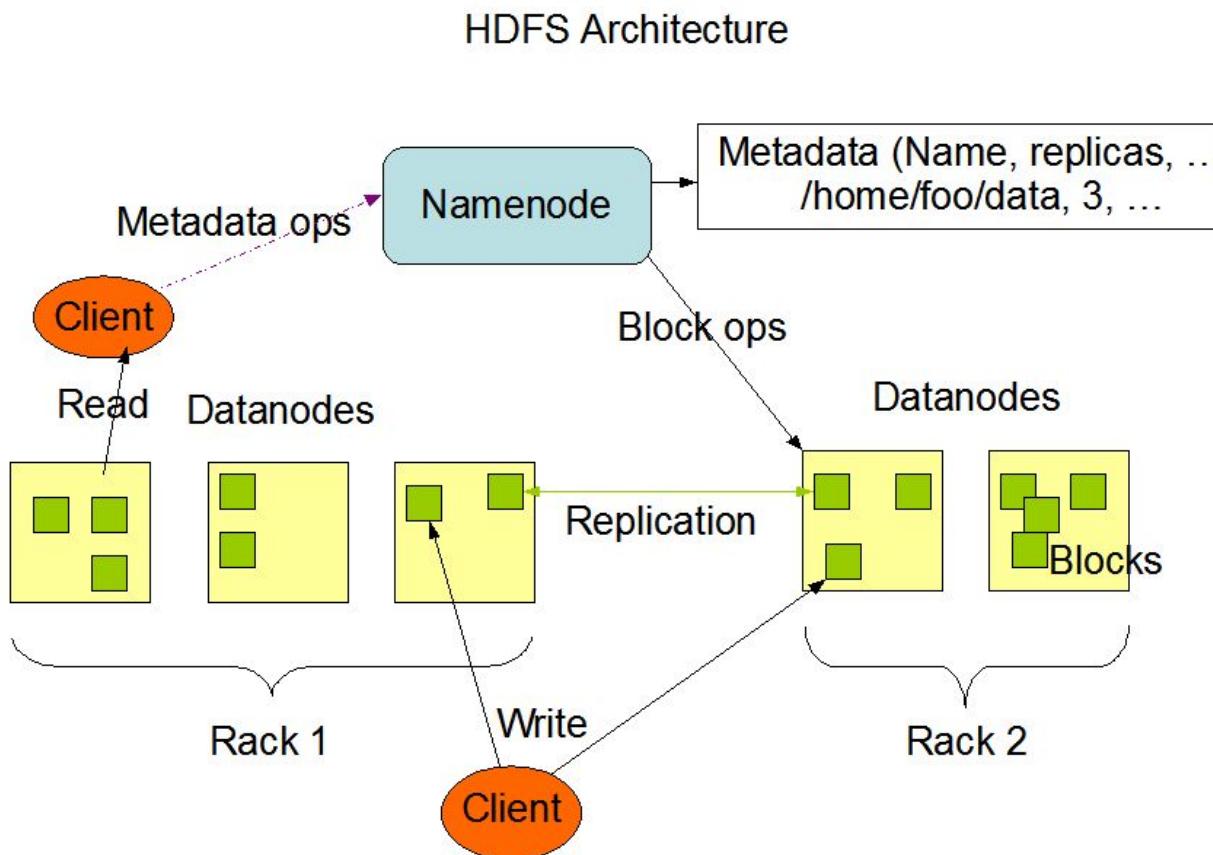


HDFS (Hadoop Distributed File System): A distributed file system designed to store and process large amounts of data across multiple machines in a Hadoop cluster, providing high fault-tolerance and scalability.



Hadoop

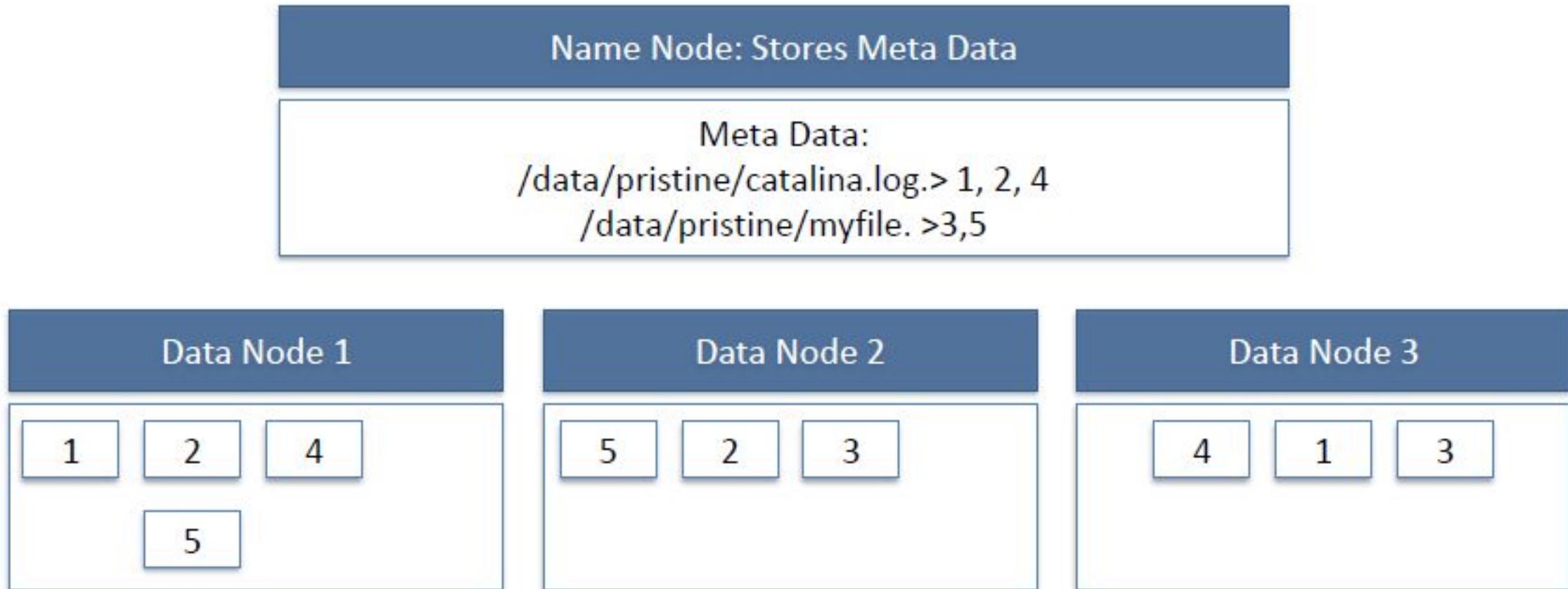
❖ Hadoop HDFS



- **Namenode:** Master node in a Hadoop cluster, managing file system's metadata.
- **Datanode:** Worker node in a Hadoop cluster, storing data blocks.
- **Blocks:** A fixed-size chunk of data. Default is 128MB (or 64MB) size.

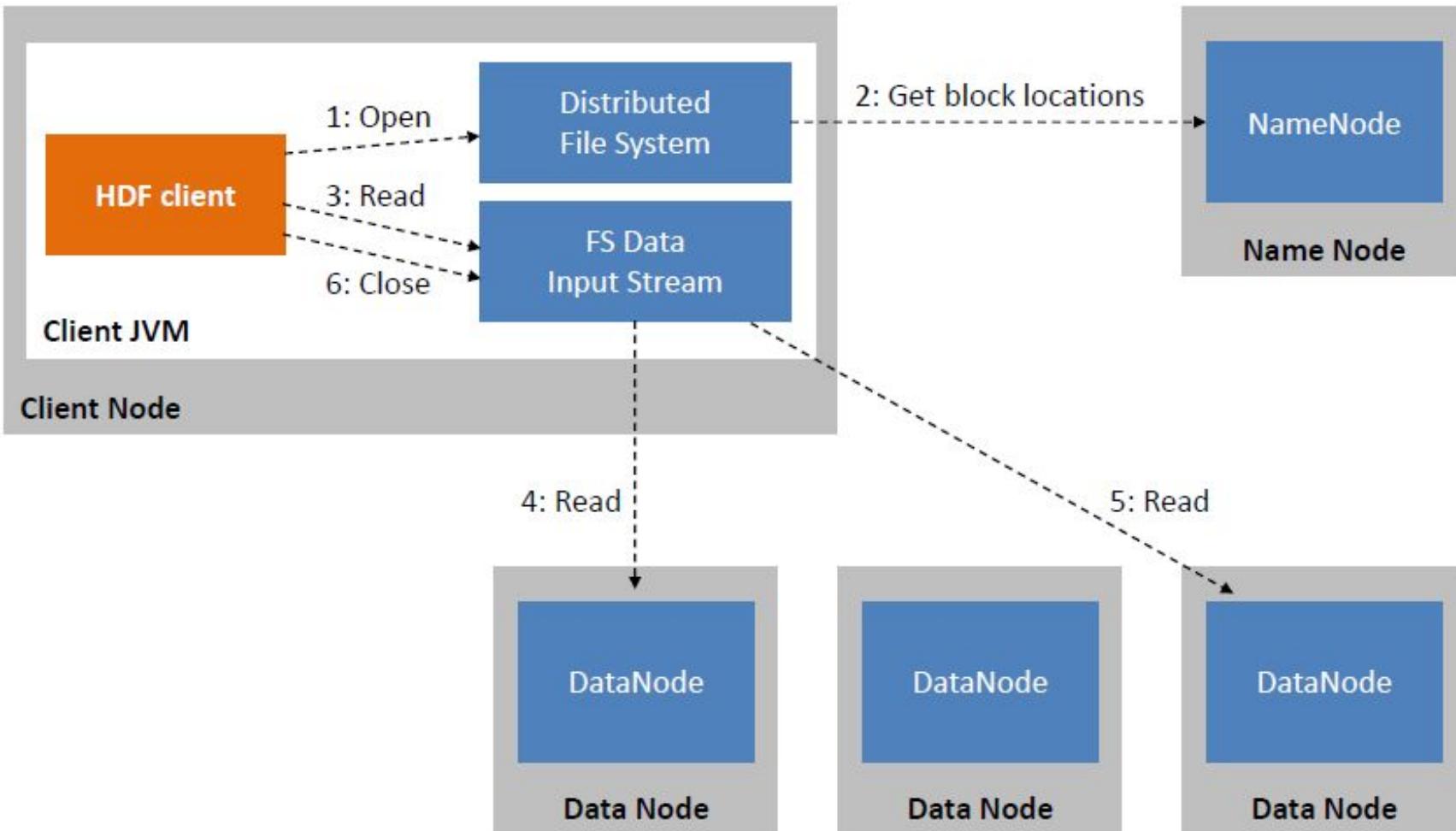
Hadoop

◆ Hadoop HDFS



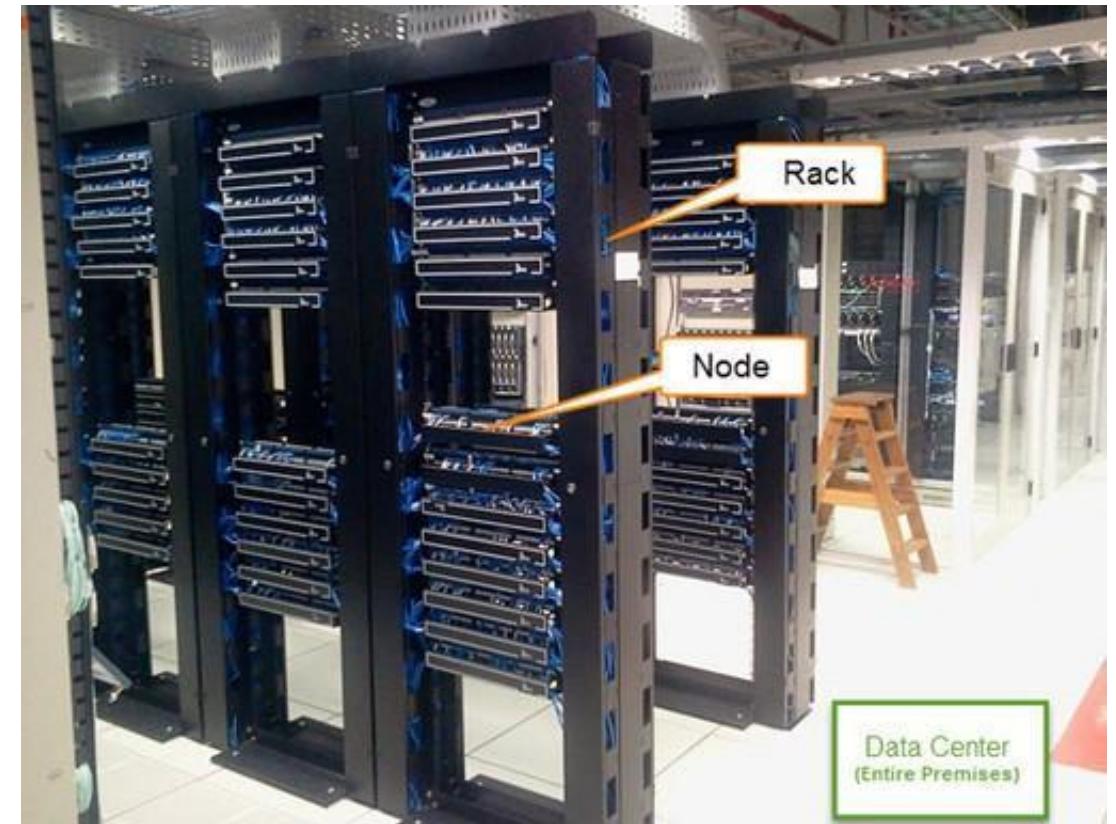
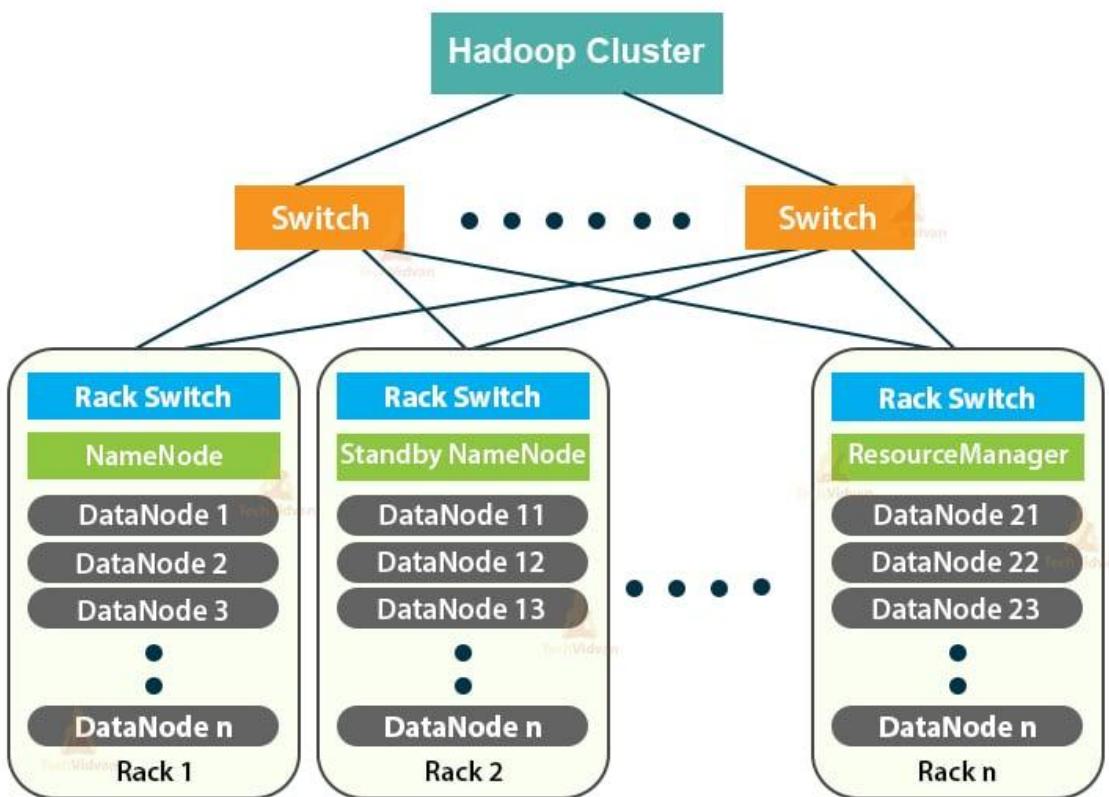
Hadoop

❖ Hadoop HDFS



Hadoop

❖ Hadoop HDFS

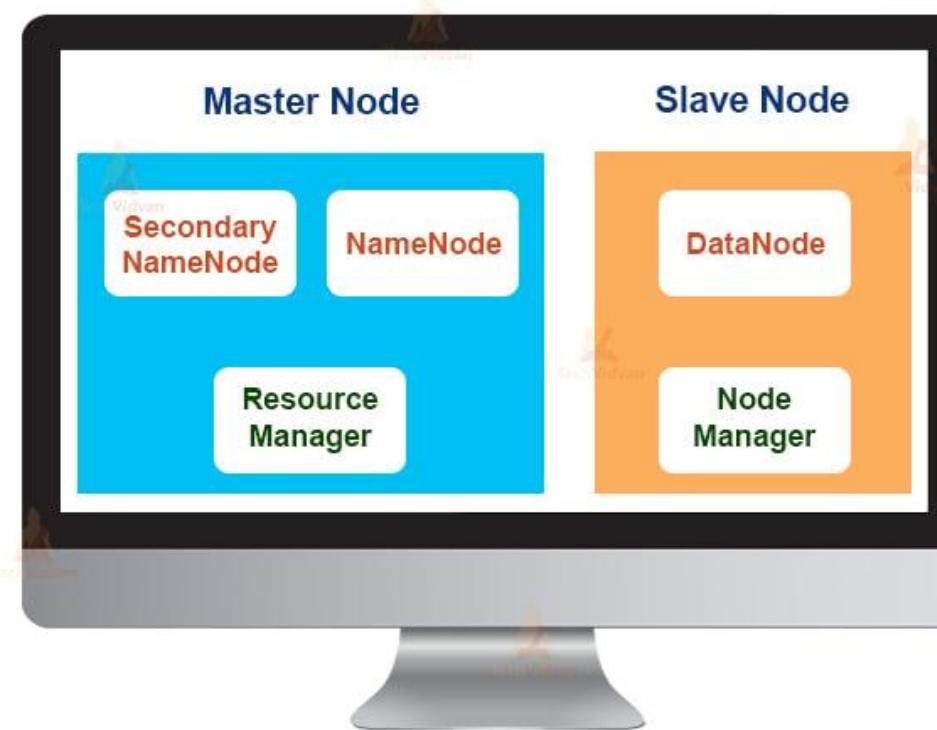


Hadoop Cluster Architecture

Hadoop

❖ Hadoop HDFS

Single Node Hadoop Cluster



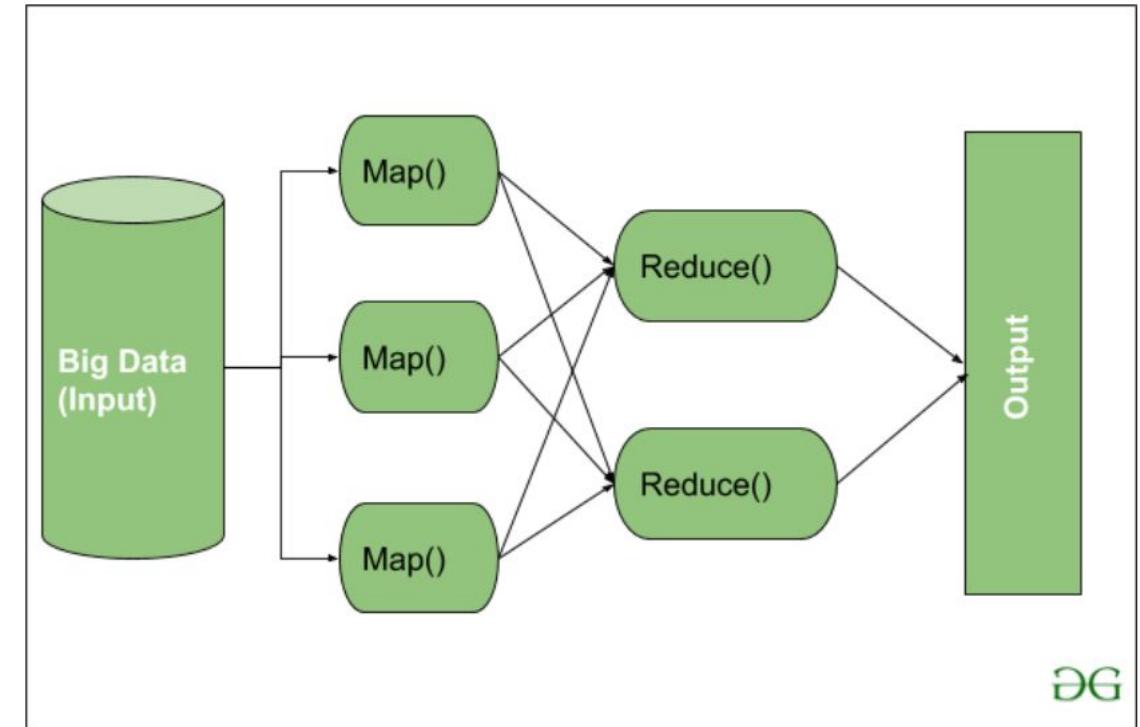
Hadoop Single Node Cluster Architecture

Hadoop

❖ Hadoop MapReduce



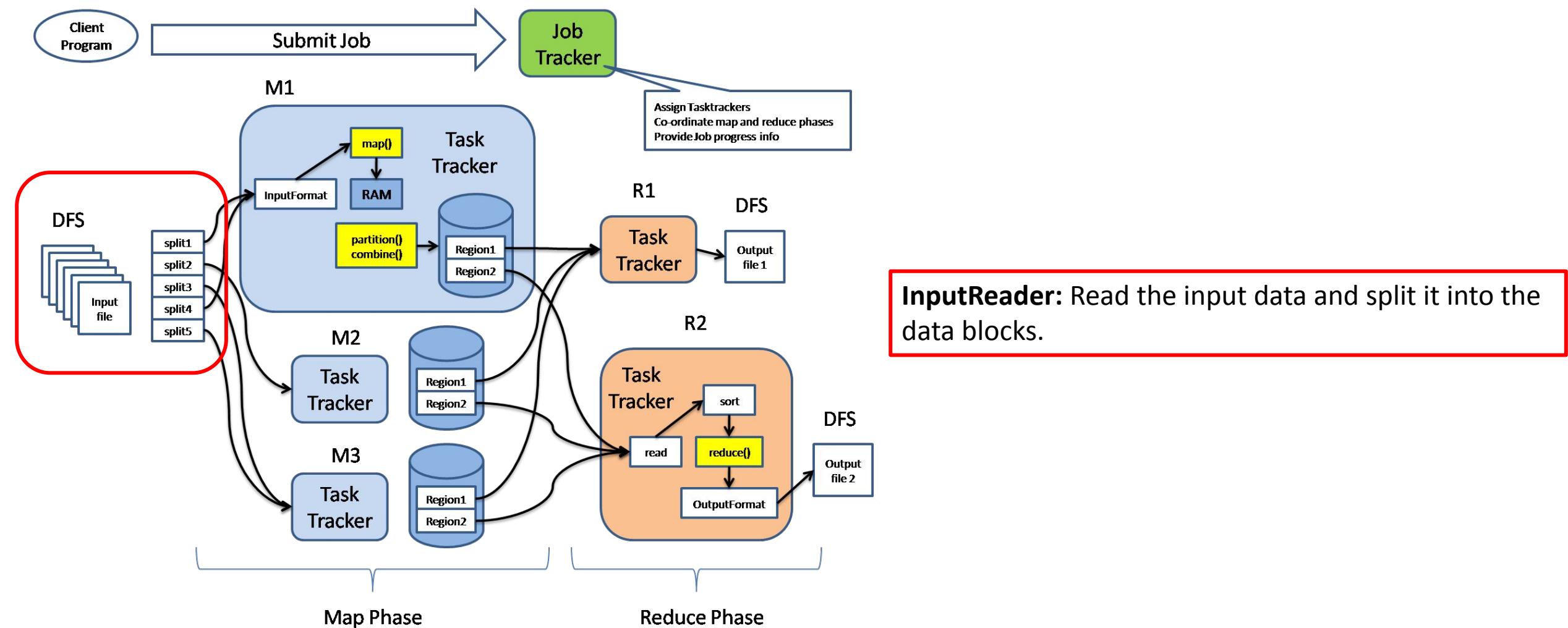
Hadoop MapReduce: A distributed programming model and framework that breaks down large-scale data processing tasks into smaller, parallelizable operations (map and reduce), allowing for efficient processing and analysis of big data across a cluster of machines.



<https://www.geeksforgeeks.org/hadoop-architecture/>

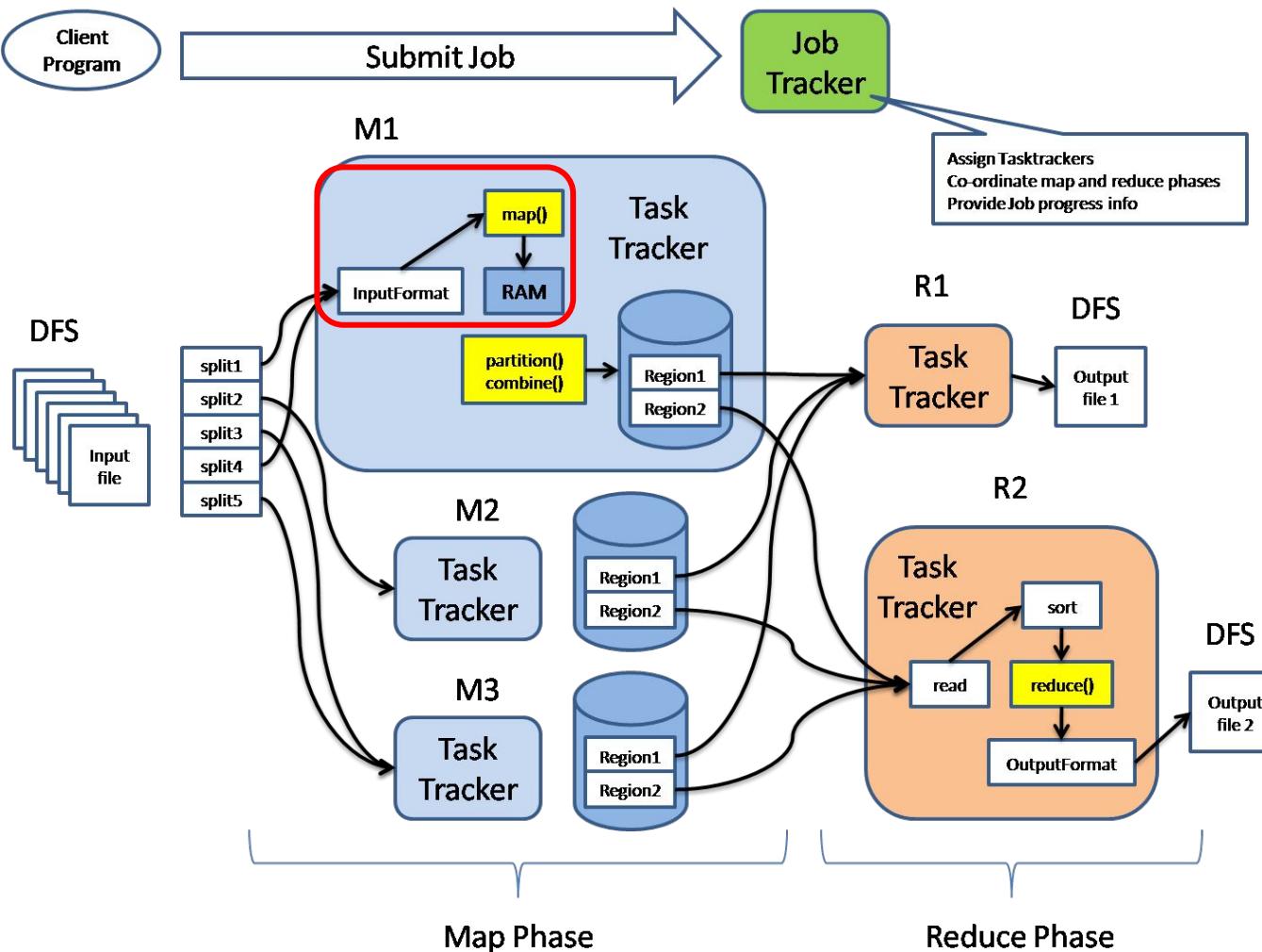
Hadoop

❖ Hadoop MapReduce



Hadoop

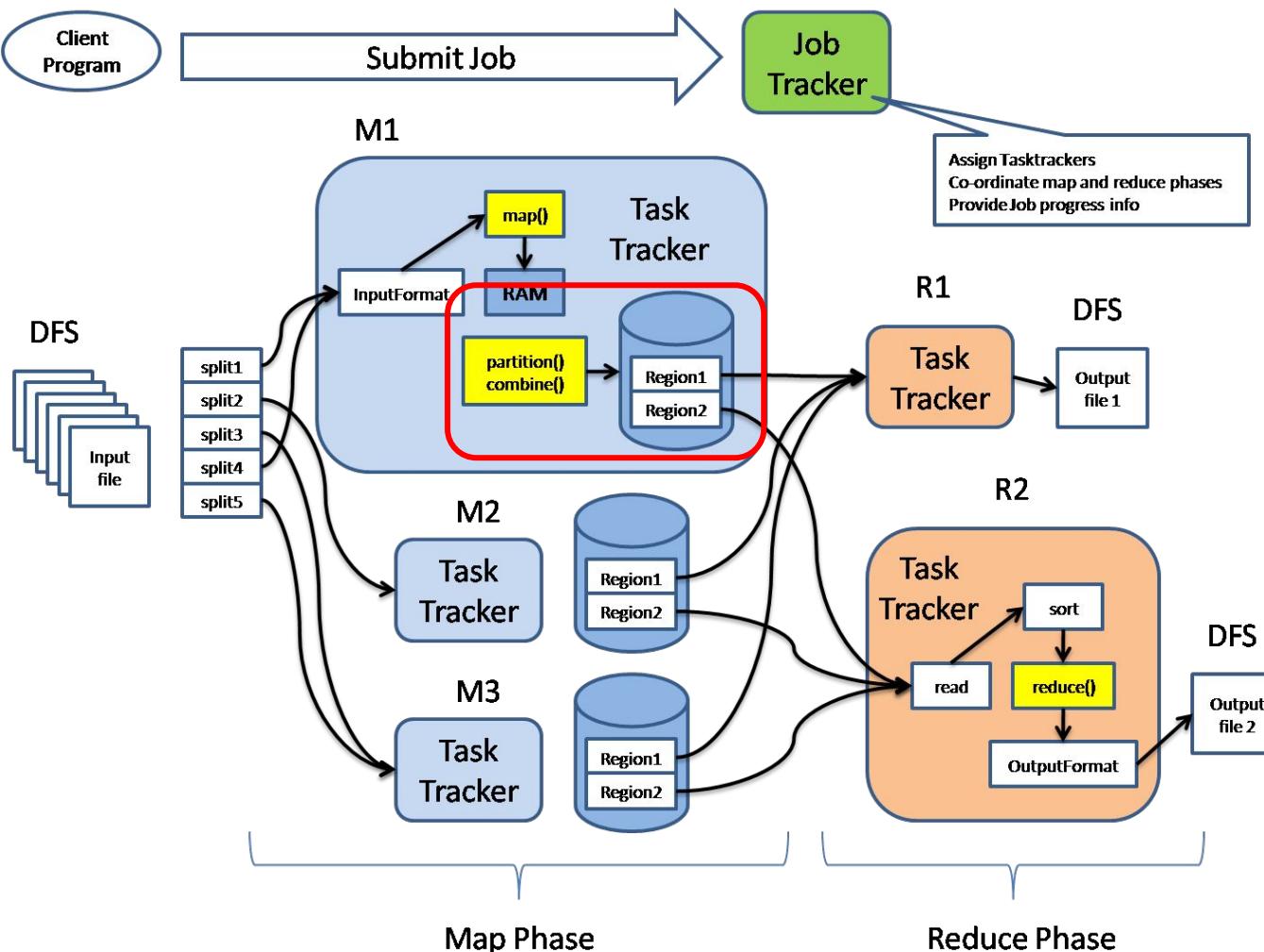
❖ Hadoop MapReduce



MapFunction: Process the key-value pairs and generate the corresponding output key-value pairs.

Hadoop

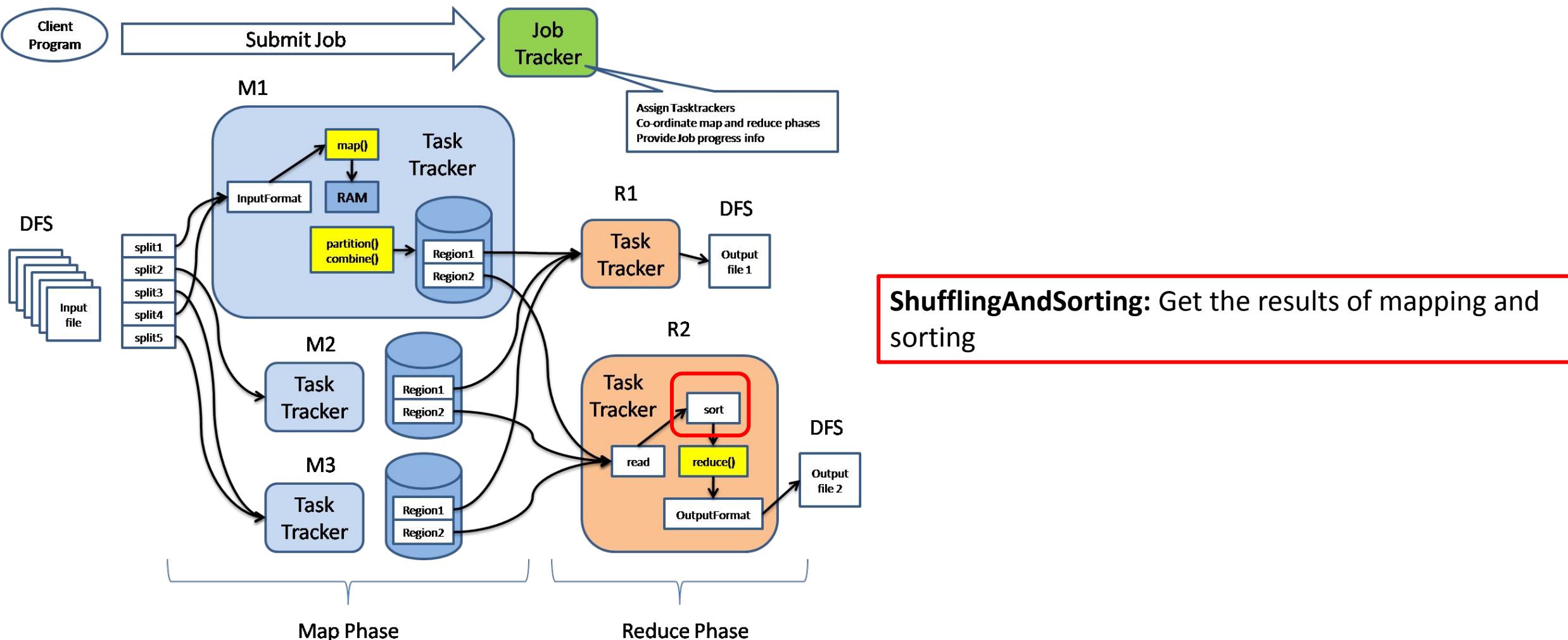
❖ Hadoop MapReduce



PartitionFunction: Assign the output of Mapping to the appropriate Reducer.

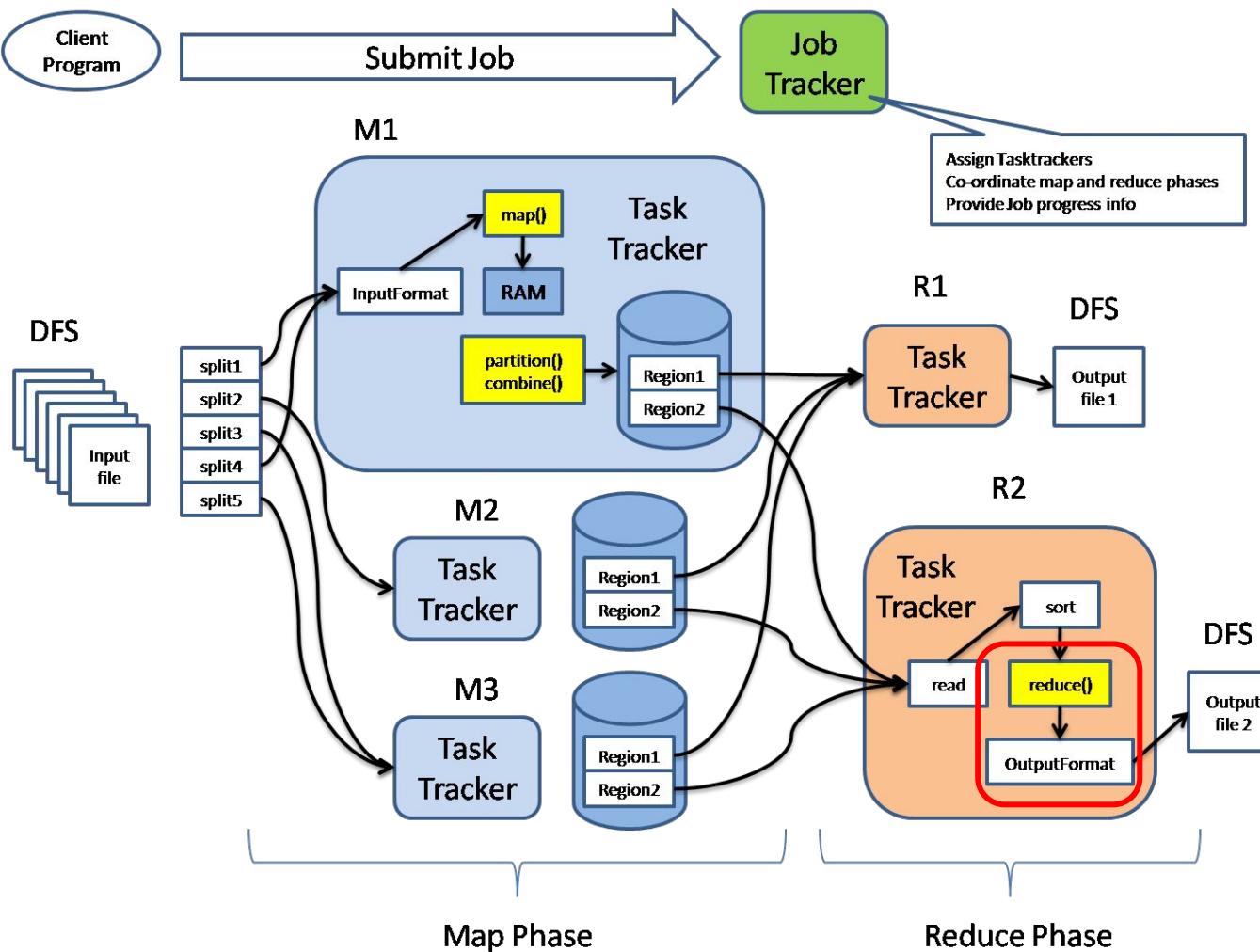
Hadoop

❖ Hadoop MapReduce



Hadoop

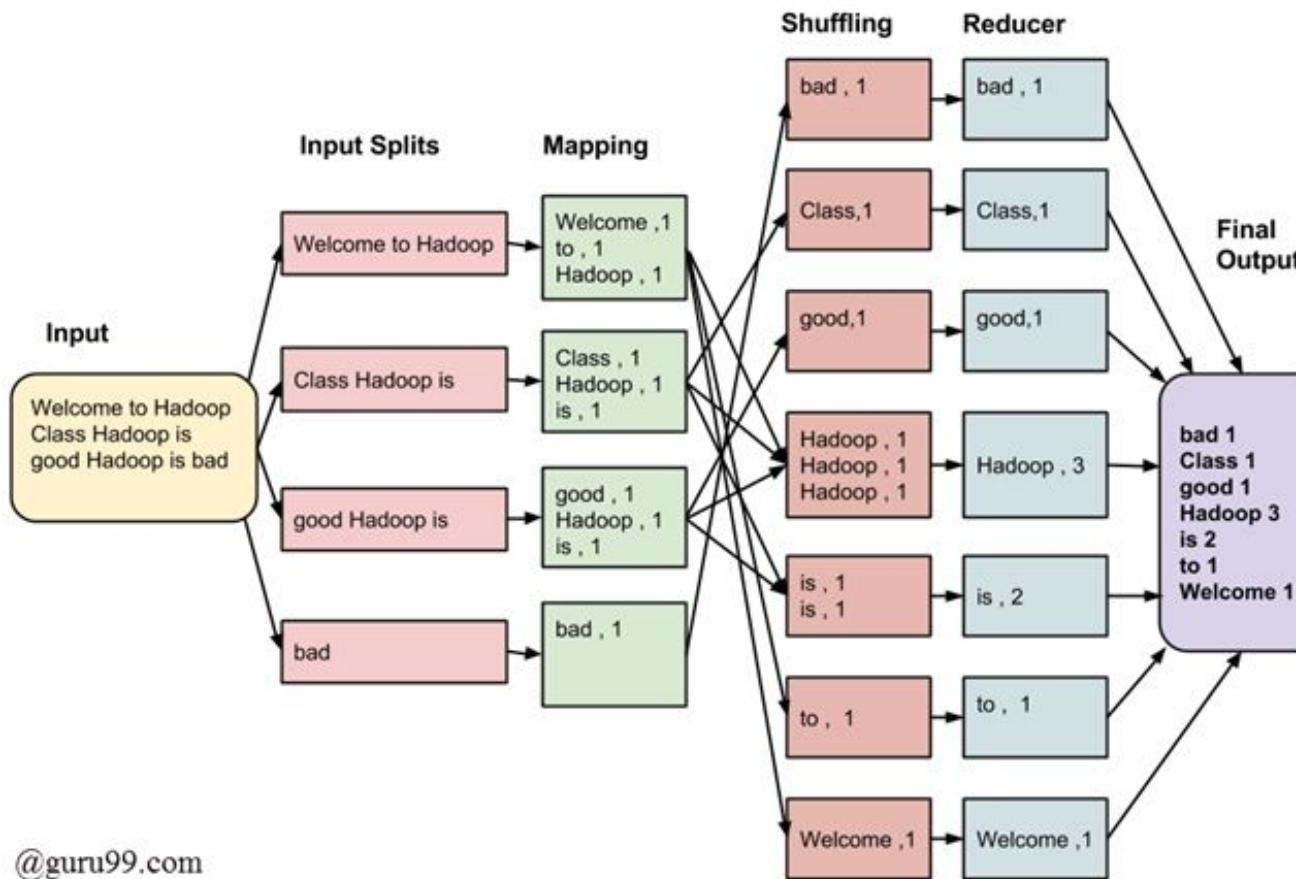
❖ Hadoop MapReduce



ReduceFunction: Grouping and processing the intermediate key-value pairs produced by the map phase.

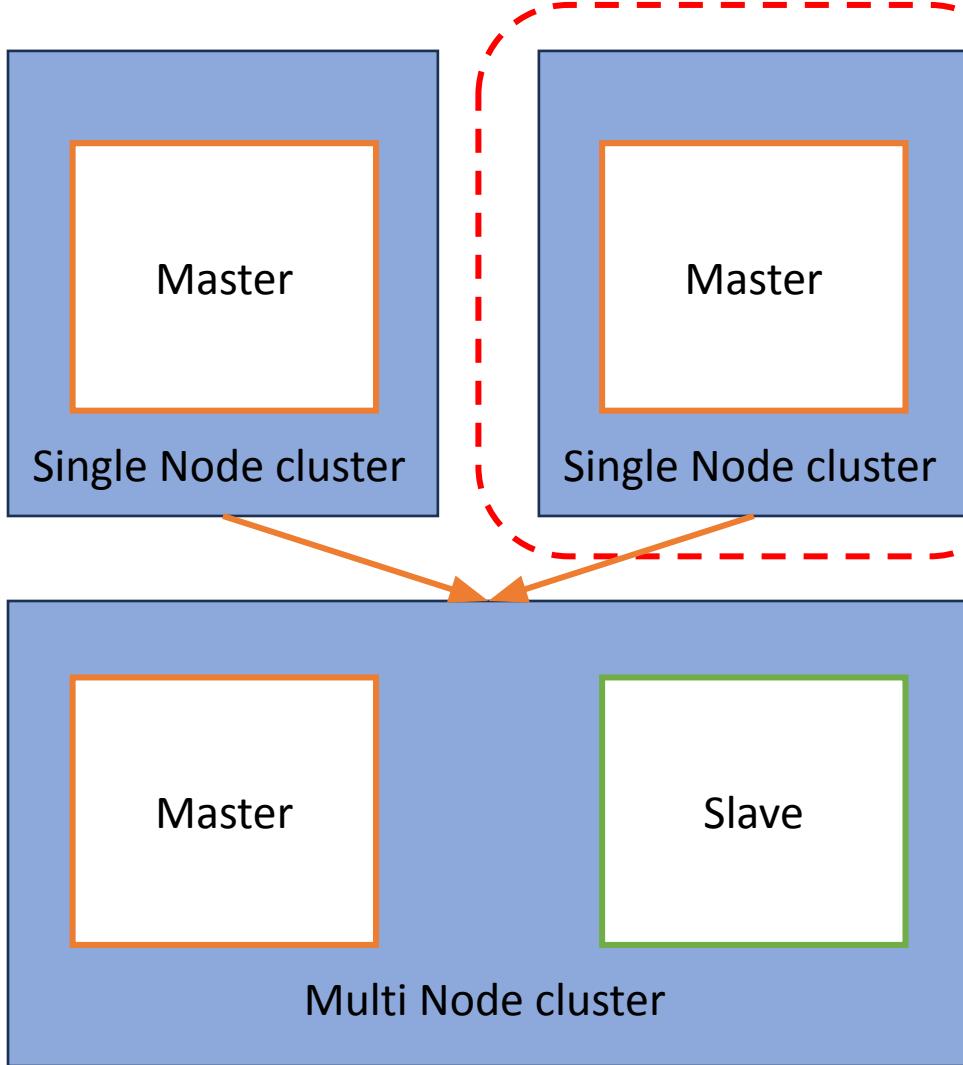
Hadoop

❖ Hadoop MapReduce: WordCount Example

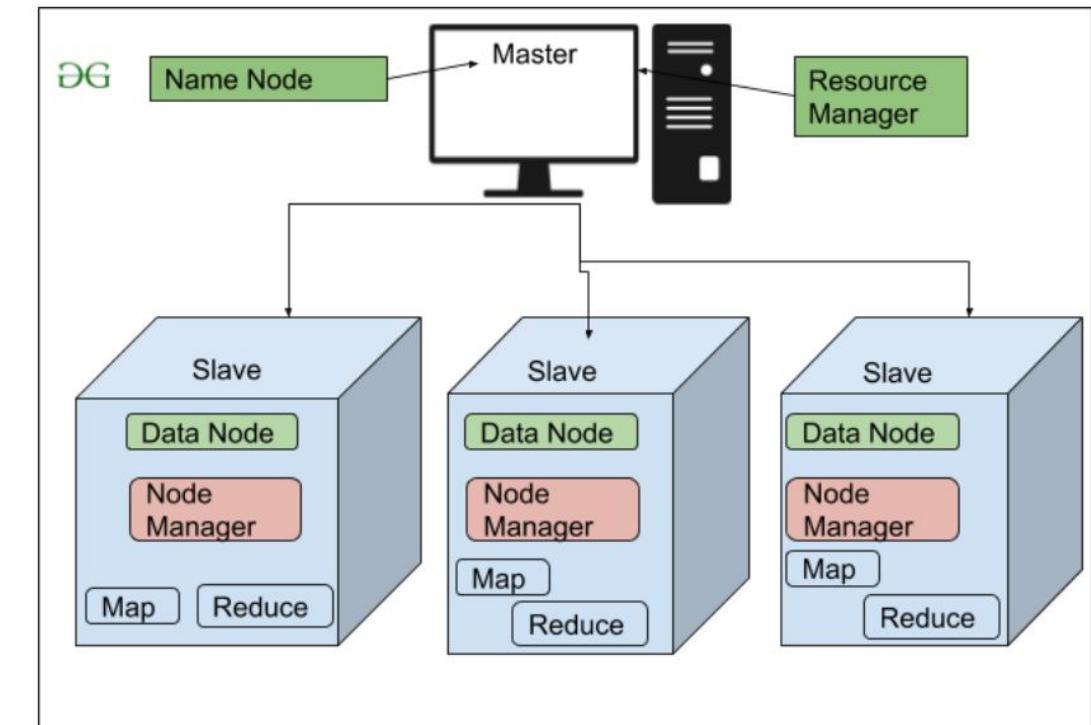


Hadoop

❖ Installation



Hadoop Single Node (Hadoop pseudo-distributed mode):
A configuration option in Hadoop that allows us to run Hadoop on a single machine.



<https://www.geeksforgeeks.org/hadoop-architecture/>

Hadoop

❖ Installation: Requirements

1. Virtual Machine



VirtualBox



VMWare



Cloudera

3. Hadoop



2. CentOS

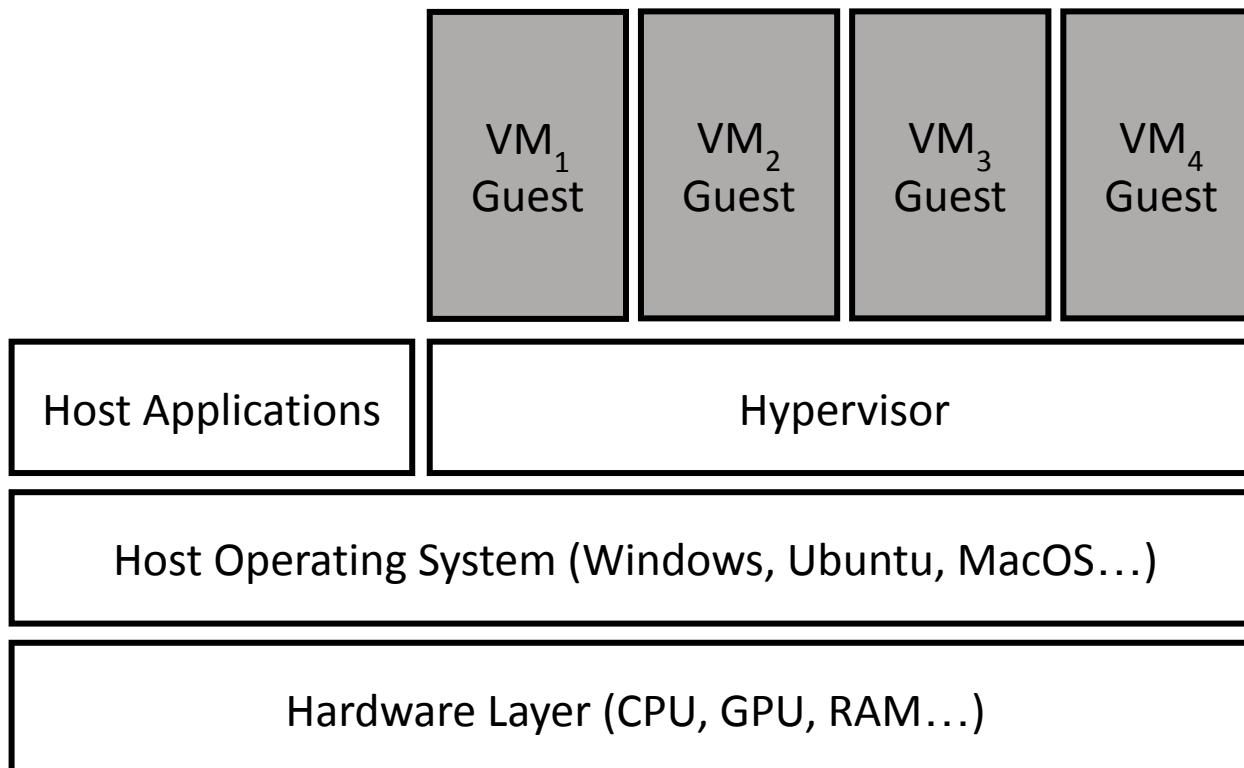


CentOS

Note: The following installation demo is on Mac M1. You need to choose proper version for your computer.

Hadoop

❖ Virtual Machine



Virtual Machines (VMs): An emulation of a computer system, where these machines use computer architectures to provide the functionality of a physical computer.



Example: A Mac OS computer running two Windows VMs

Hadoop

❖ Installation: VMware



VMware: A software providing machine virtualization

Download VMware Workstation Pro

[VMware Workstation Pro](#) is the industry standard desktop hypervisor for running virtual machines on Linux or Windows PCs. Start your free, fully functional 30-day trial today.

[BUY ONLINE](#)

For Windows/Linux: Use [VMware Workstation](#)

Run Windows and More on Mac

VMware Fusion

Harness the full power of your Mac when you use VMware Fusion to run Windows, Linux, containers, Kubernetes and more in virtual machines (VMs) without rebooting.

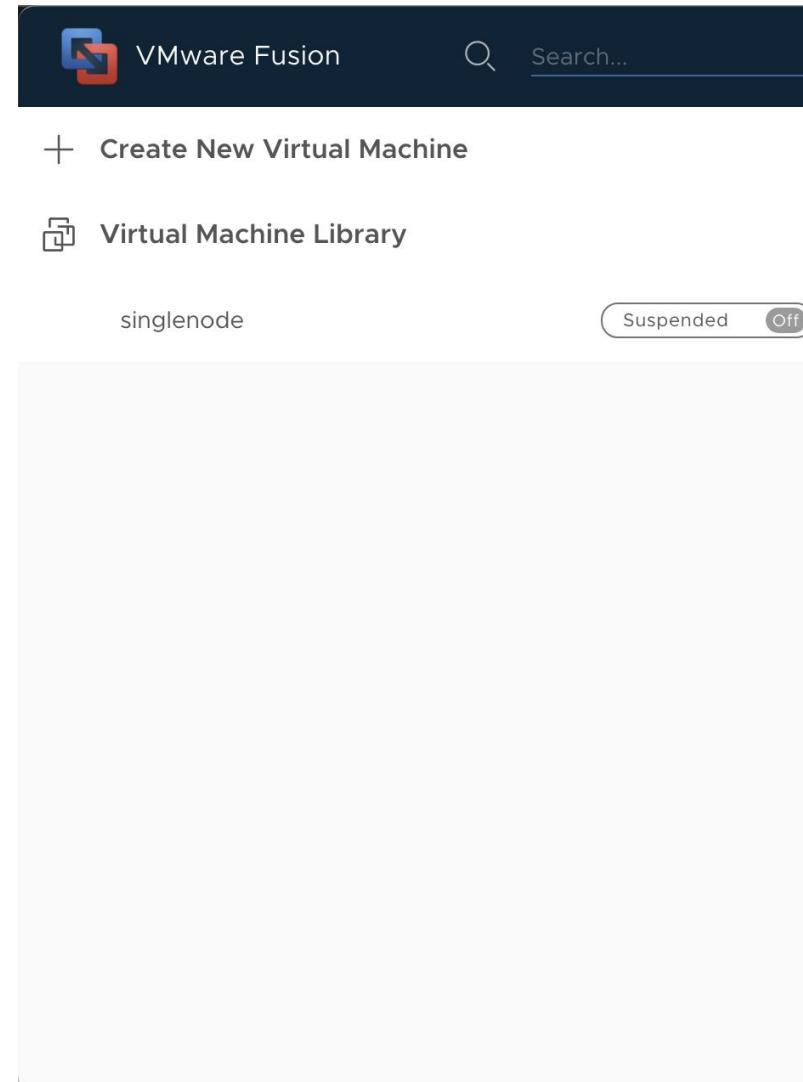
[BUY ONLINE](#)

[TRY FOR FREE](#)

For MacOS: Use [Vmware Fusion](#)

Hadoop

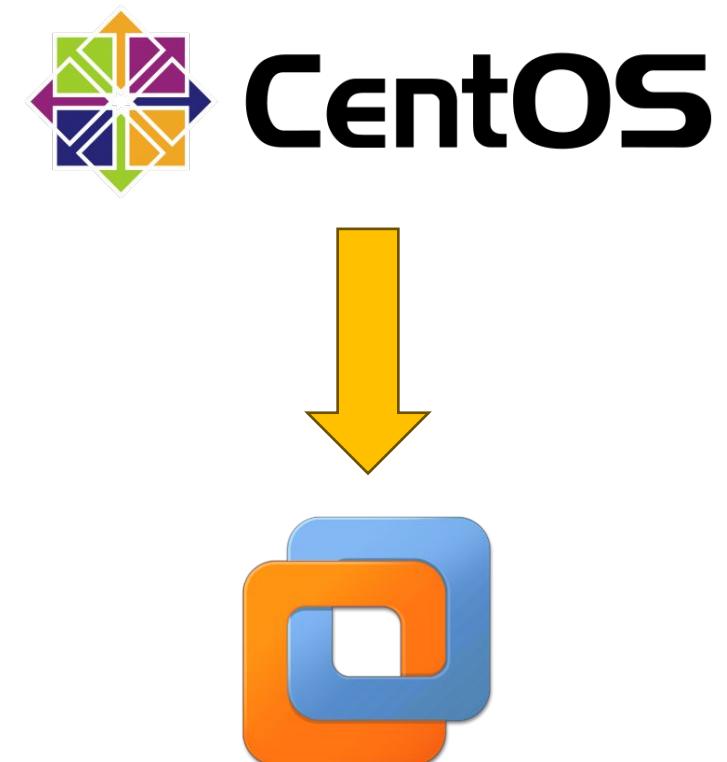
❖ Installation: VMware



VM Fusion Interface

Hadoop

❖ Installation: CentOS



CentOS: Community-driven free software effort focused around the goal of providing a rich base platform for open source communities to build upon.

Hadoop

❖ Installation: CentOS

Download

The screenshot shows the CentOS Stream download page. At the top, there are two purple buttons for "CentOS Stream 9" and "CentOS Stream 8". Below them is a navigation bar with "Architectures", "Packages", and "Others". The "Architectures" tab is selected. A red box highlights the "x86_64" and "ARM64 (aarch64)" rows. Both rows show "RPMs" as the package type and links to "Cloud | Containers | Vagrant". Other architectures listed are "IBM Power (ppc64le)", "IBM Z (s390x)", and "Documentation", "Release Notes", "Release Email", and "Website". The "End-of-life" section at the bottom states "End of RHEL9 full support phase".

Choose version:

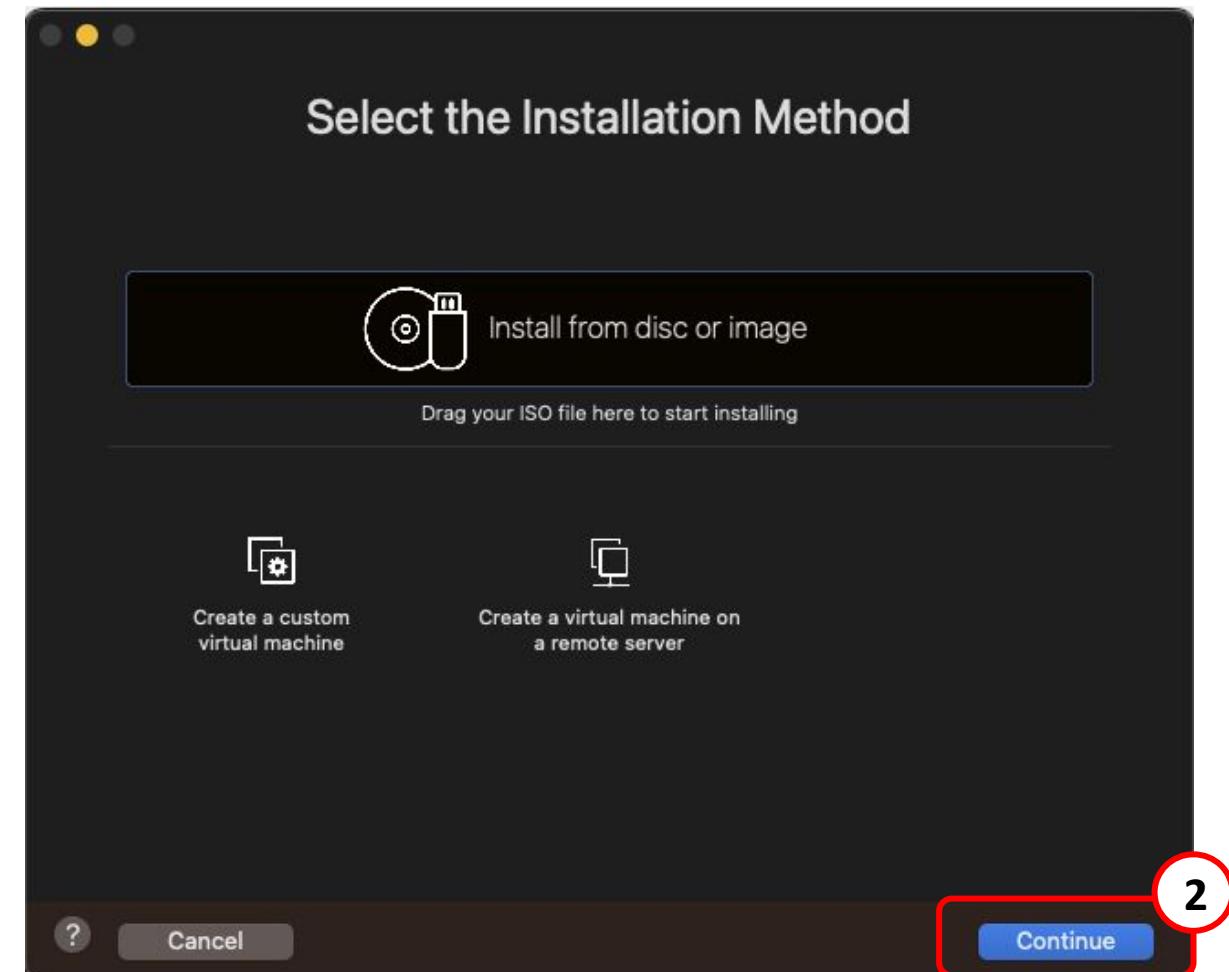
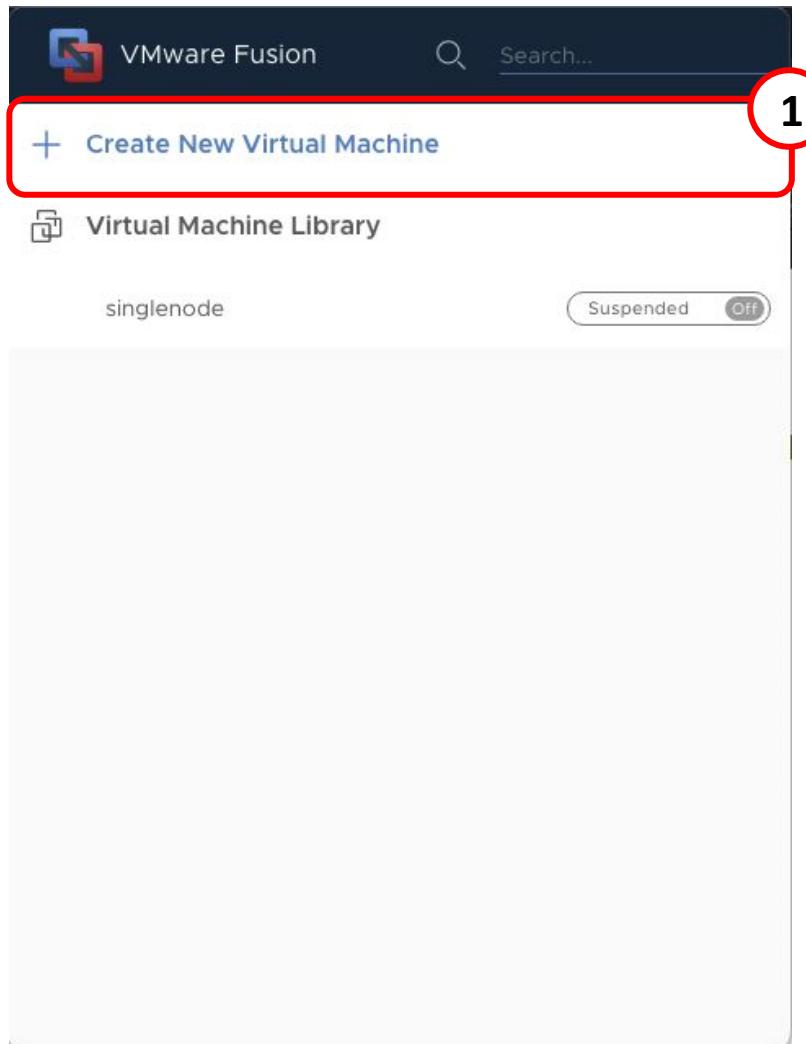
- **x86_64**: For Windows/Ubuntu/Mac Intel
- **ARM64 (aarch64)**: For Mac M1/M2



Download file .iso of CentOS: [Link](#)

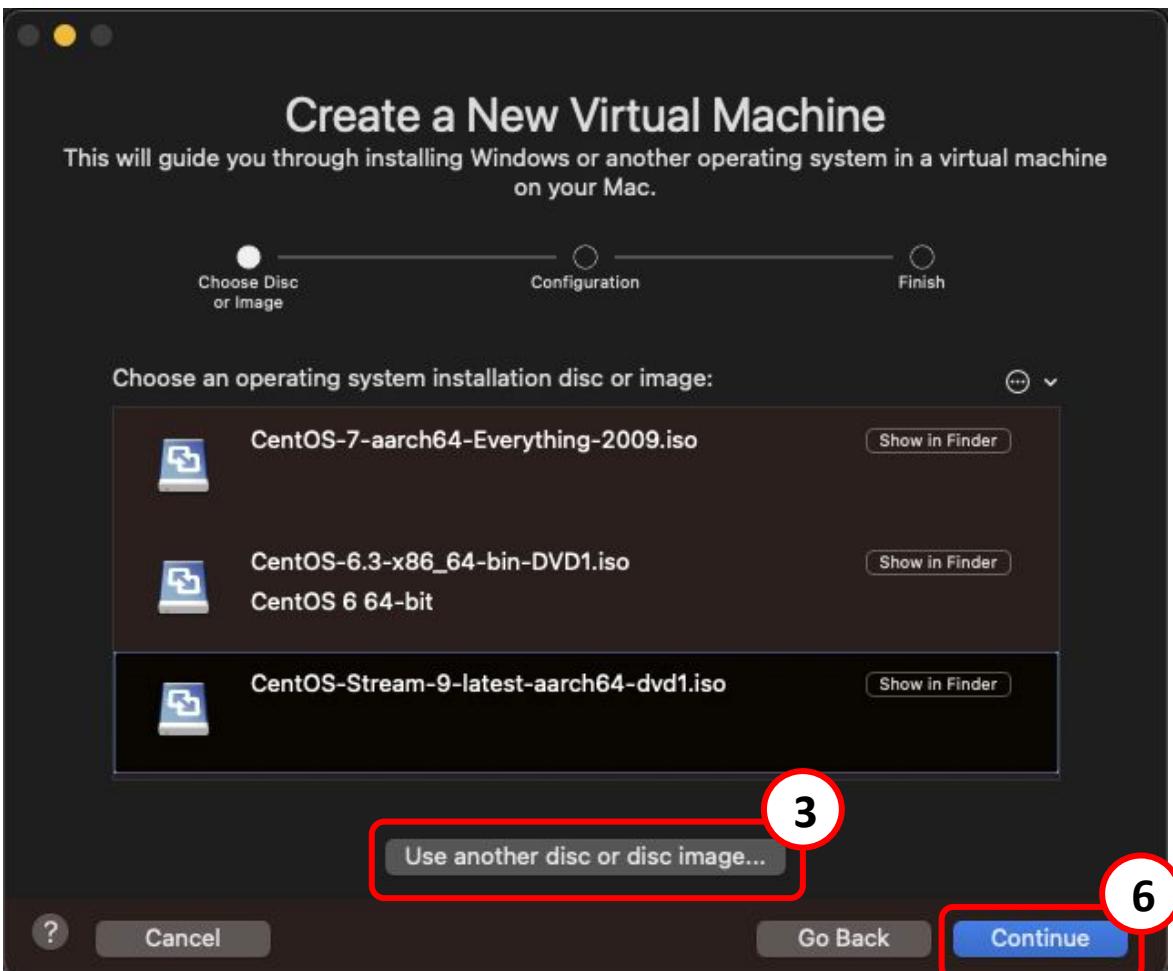
Hadoop

❖ Installation: Run CentOS in VMware



Hadoop

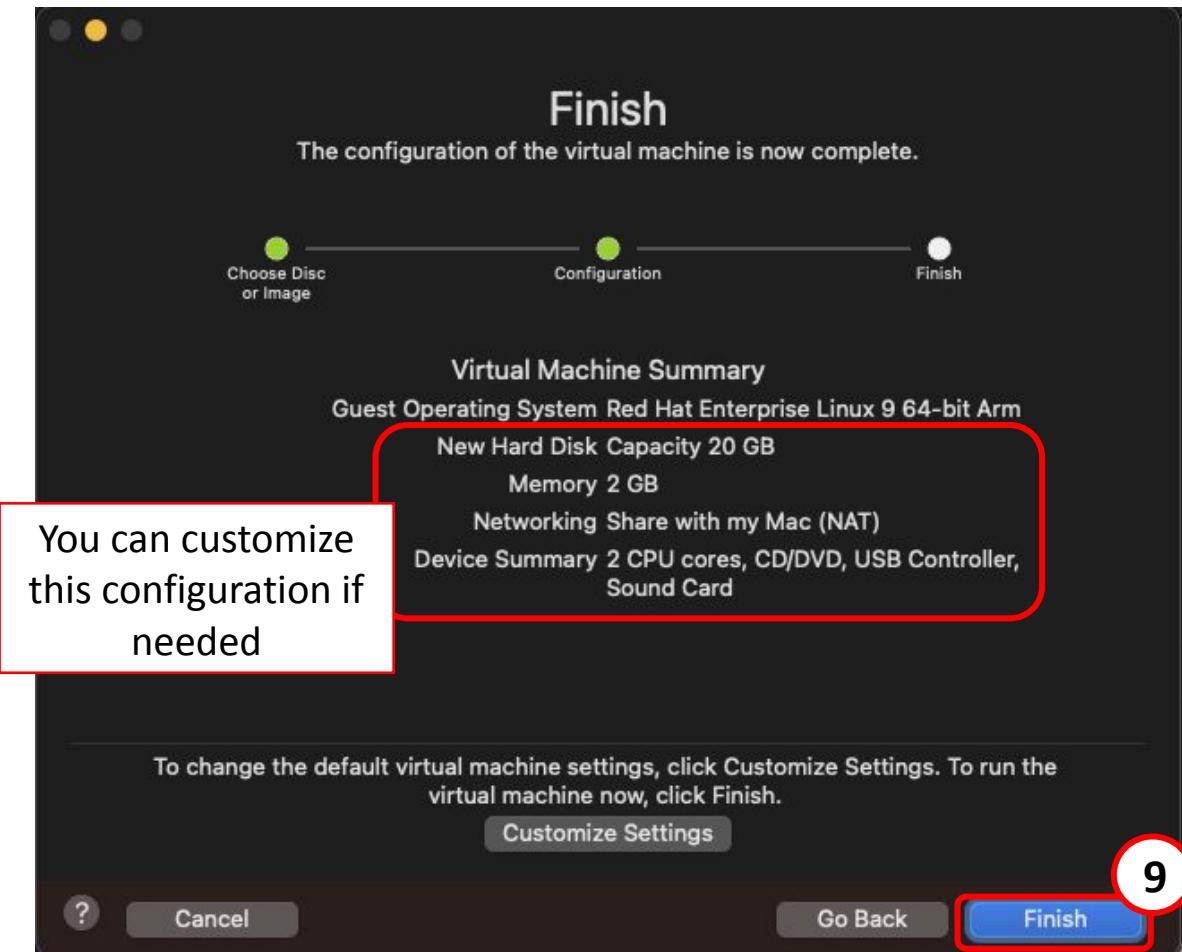
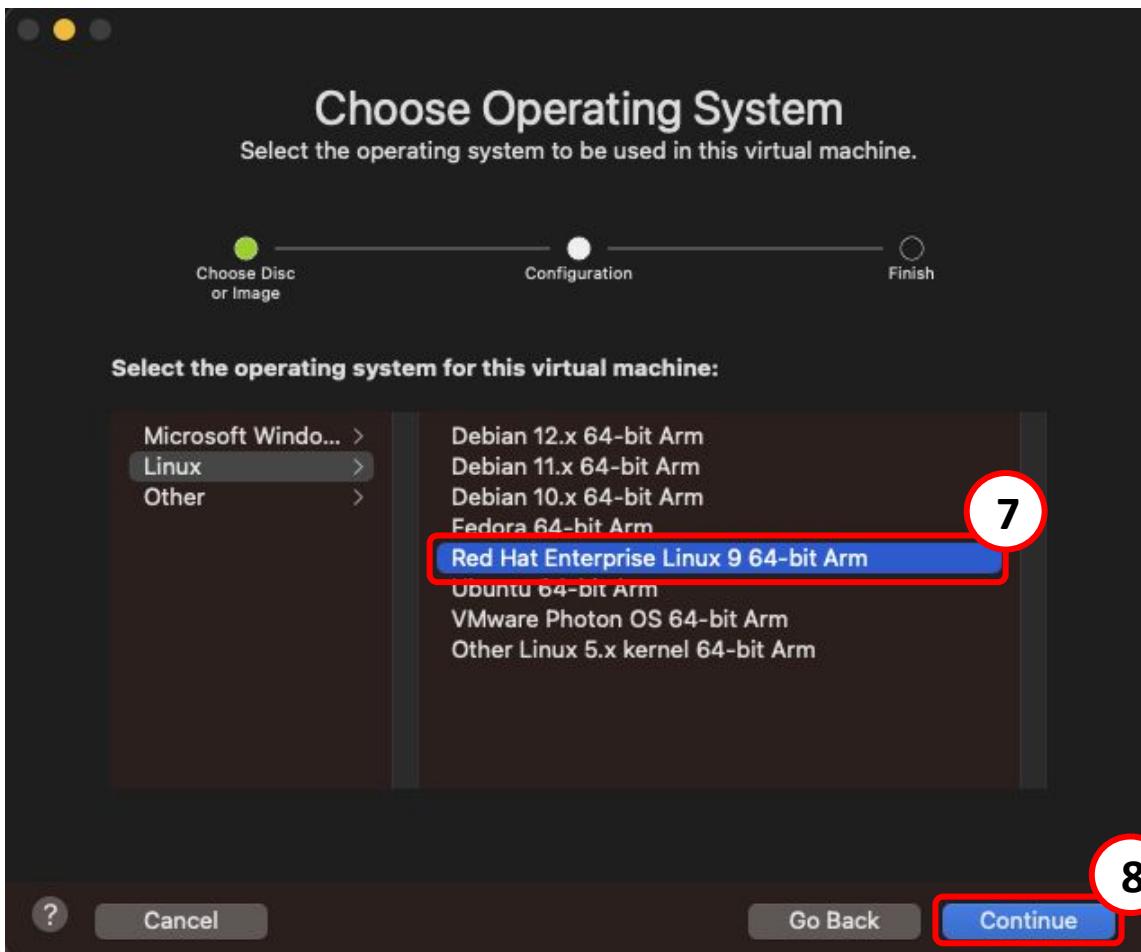
❖ Installation: Run CentOS in VMware



Choose CentOS .iso file

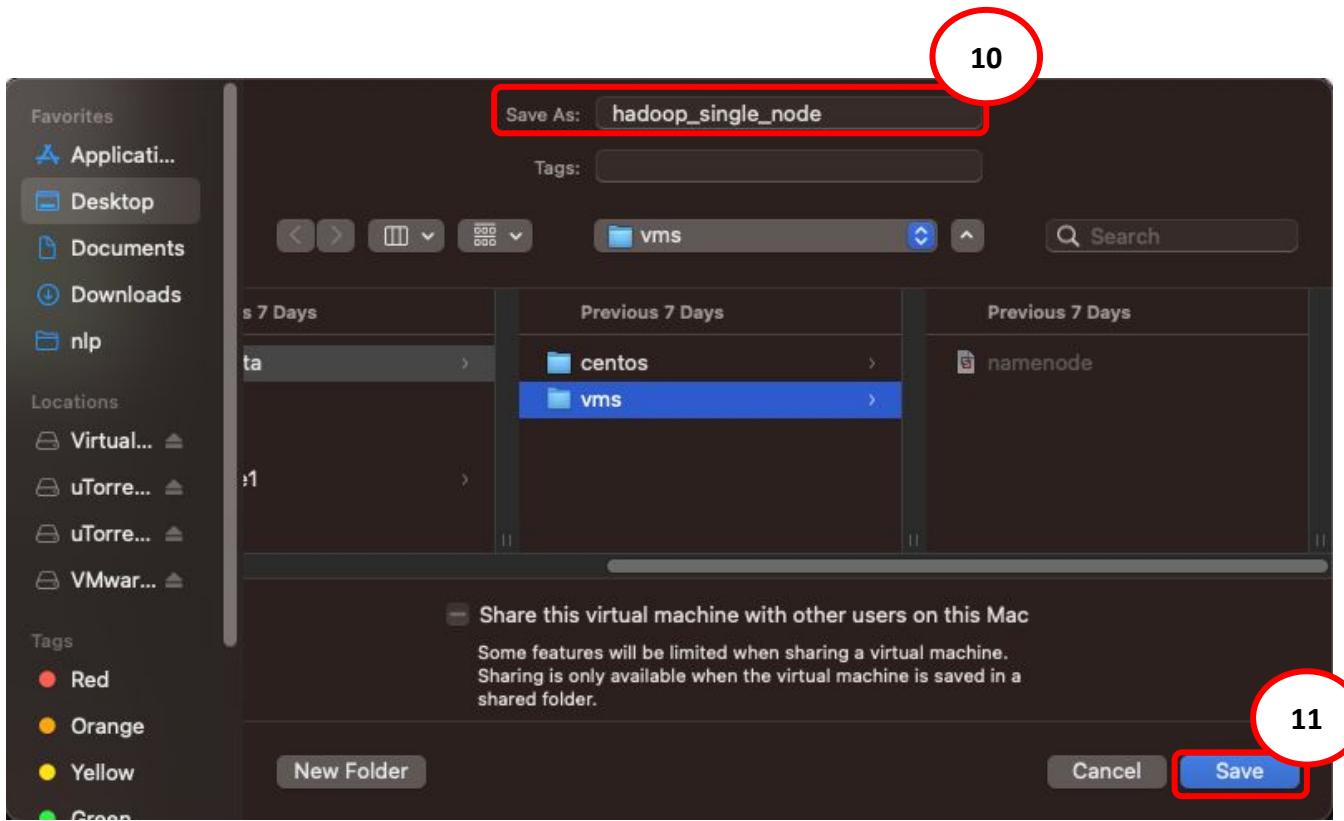
Hadoop

❖ Installation: Run CentOS in VMware

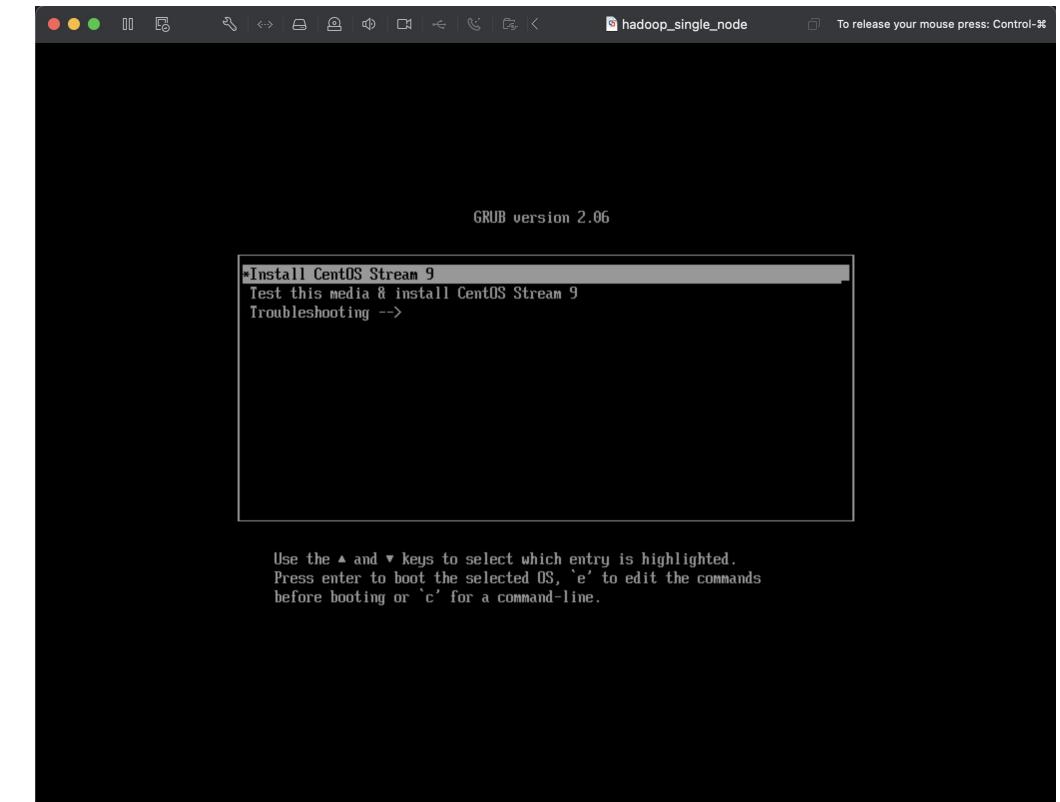


Hadoop

❖ Installation: Run CentOS in VMWare

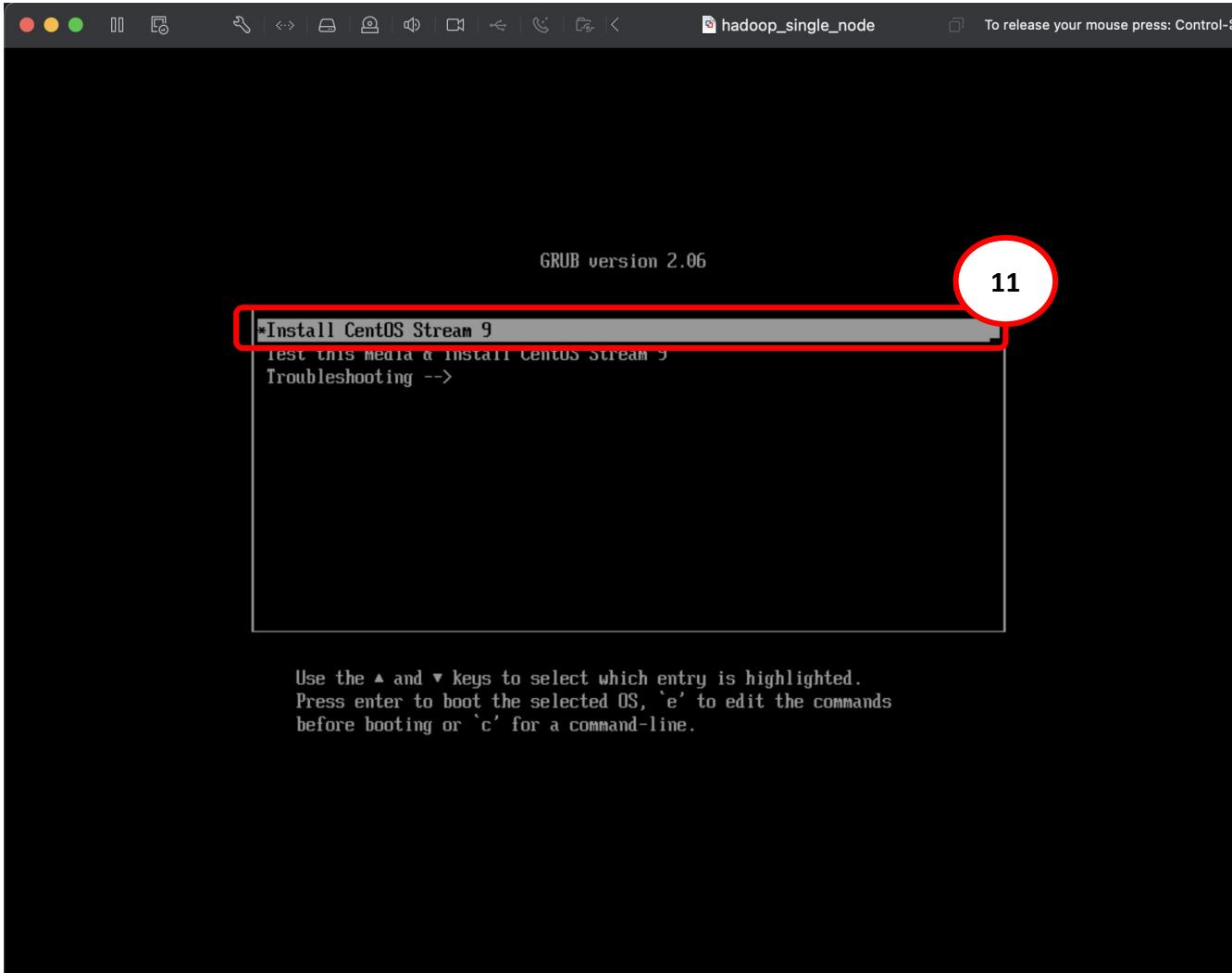


Choose where to save the new virtual machine



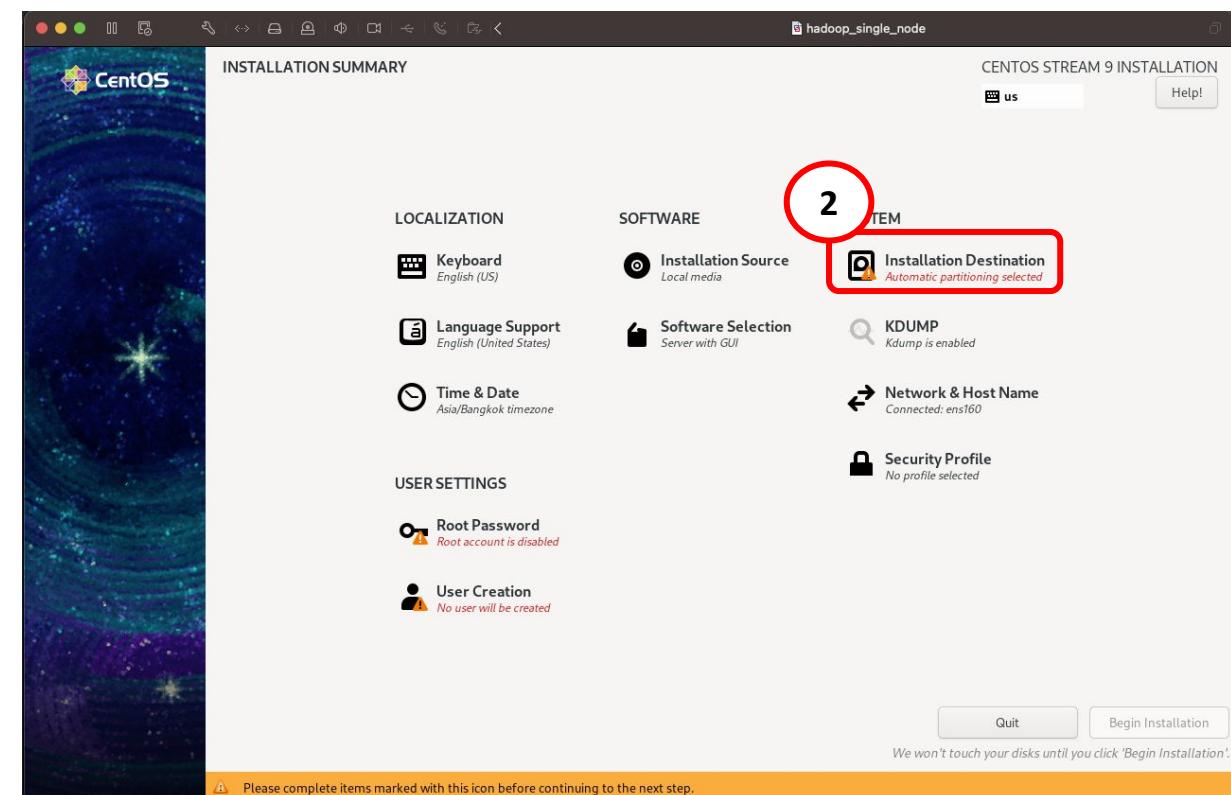
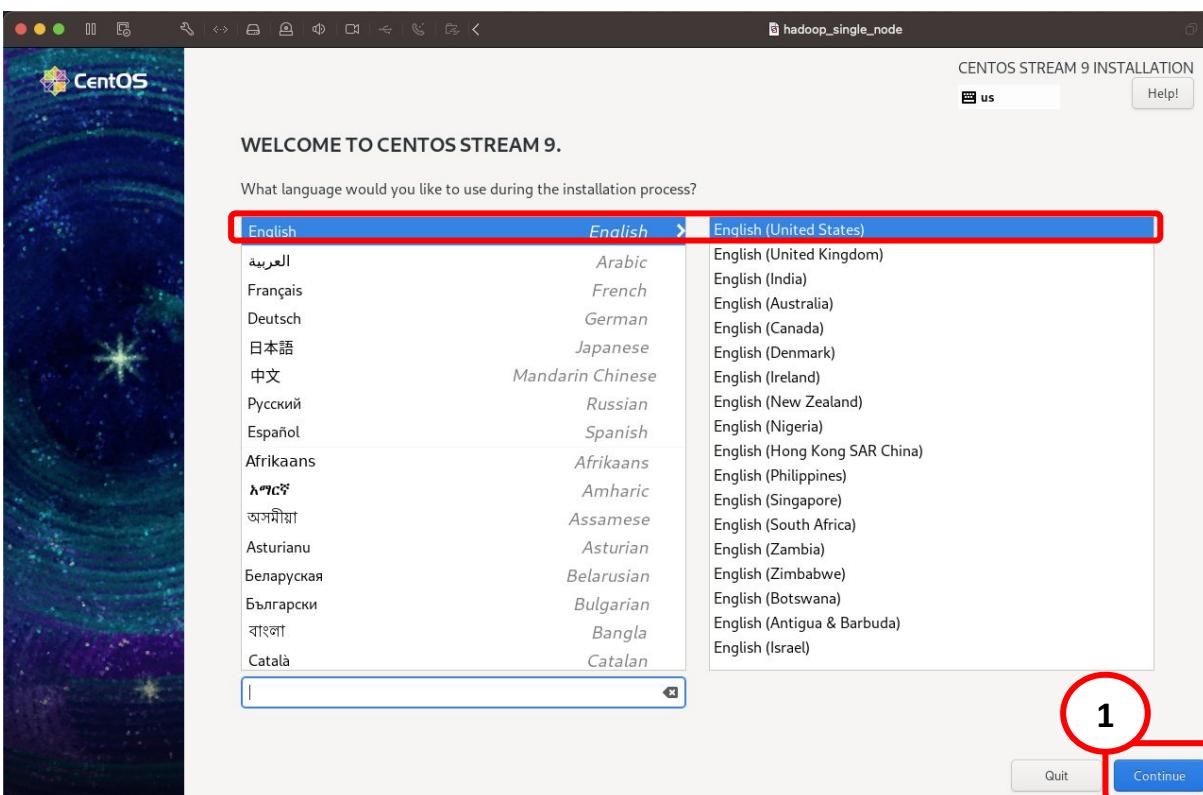
Hadoop

❖ Installation: Run CentOS in VMware



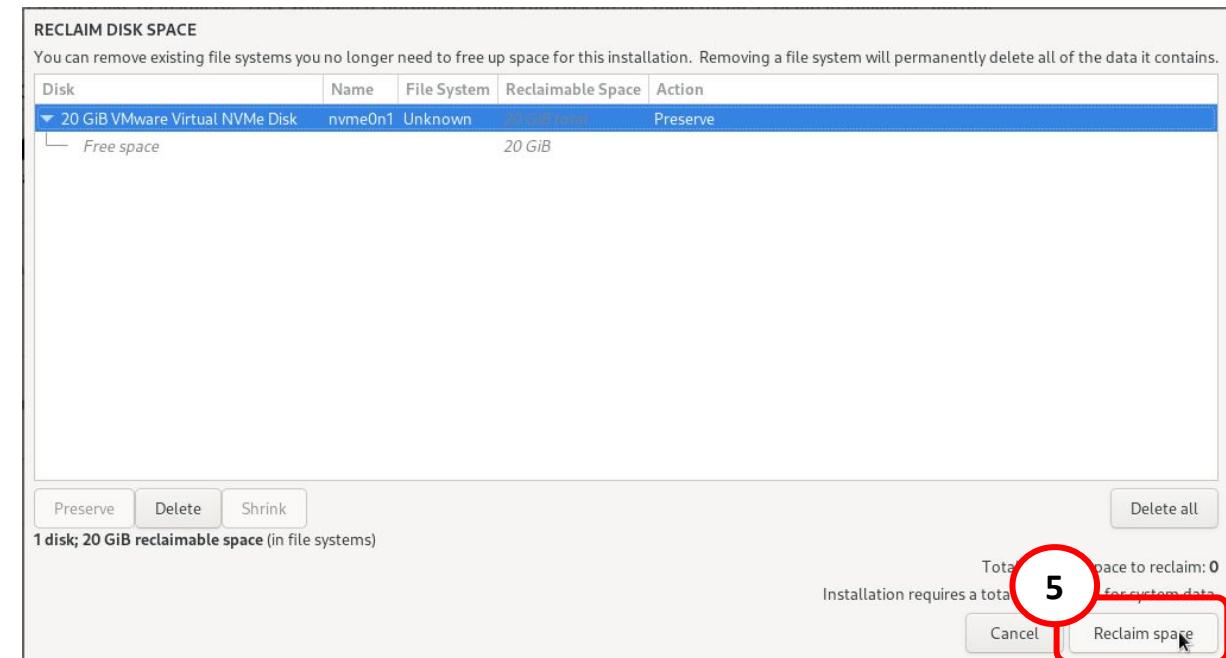
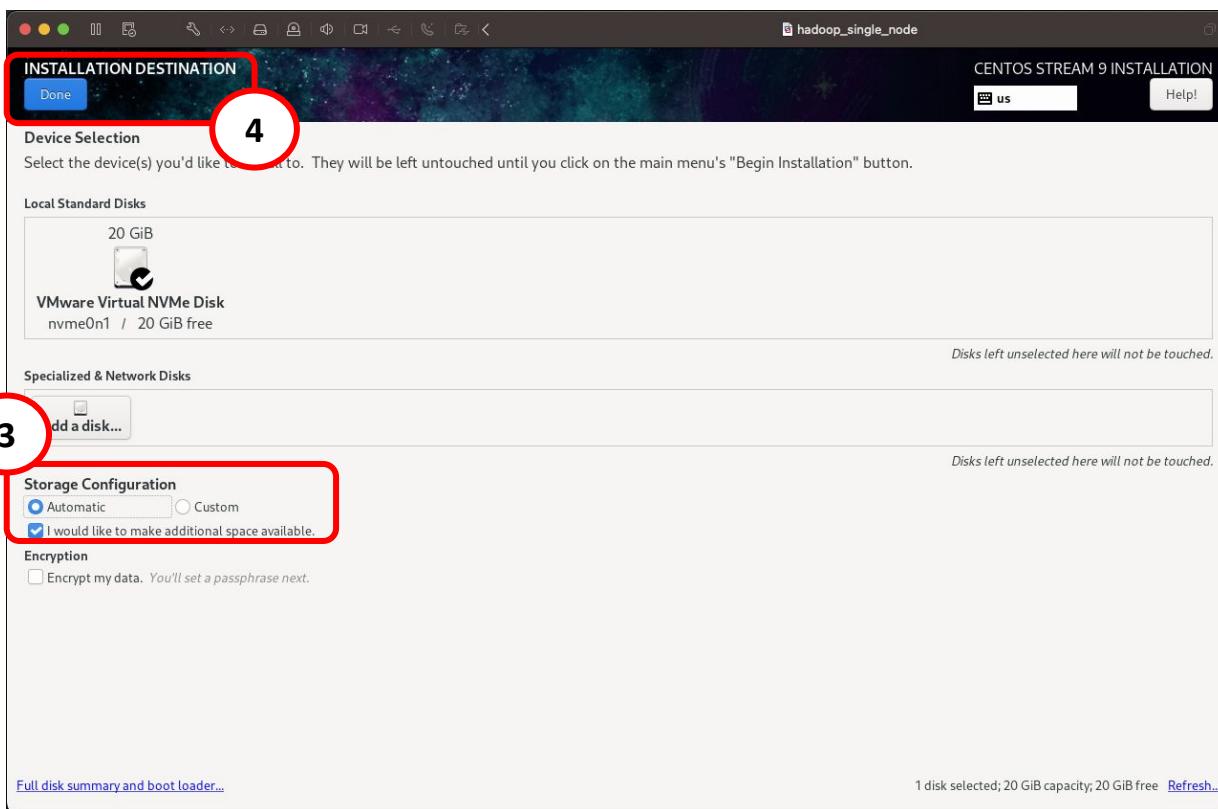
Hadoop

❖ Installation: Setup CentOS



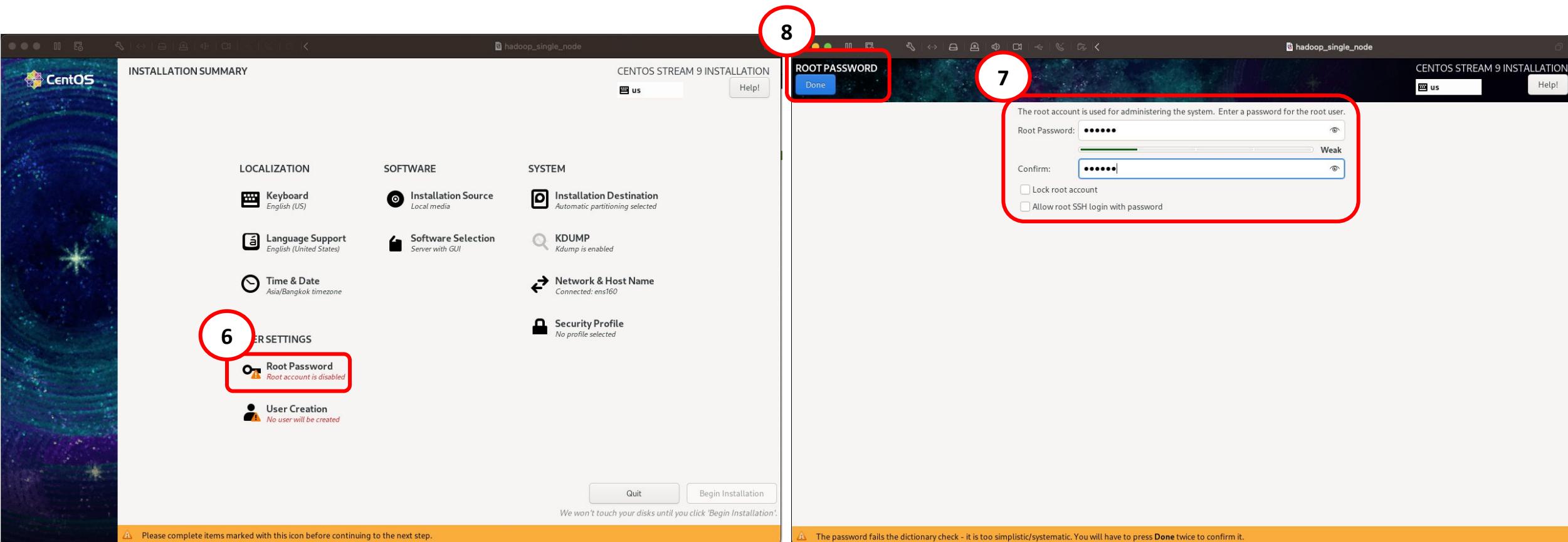
Hadoop

❖ Installation: Setup CentOS



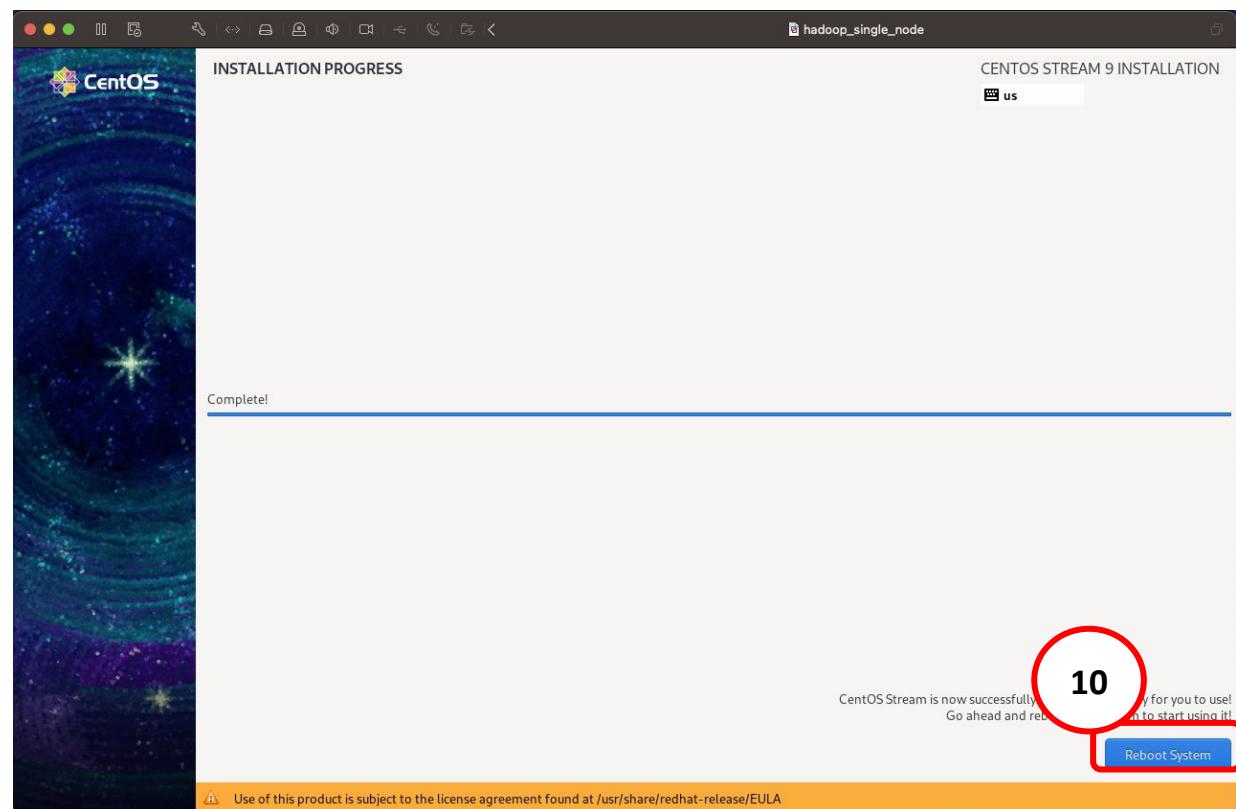
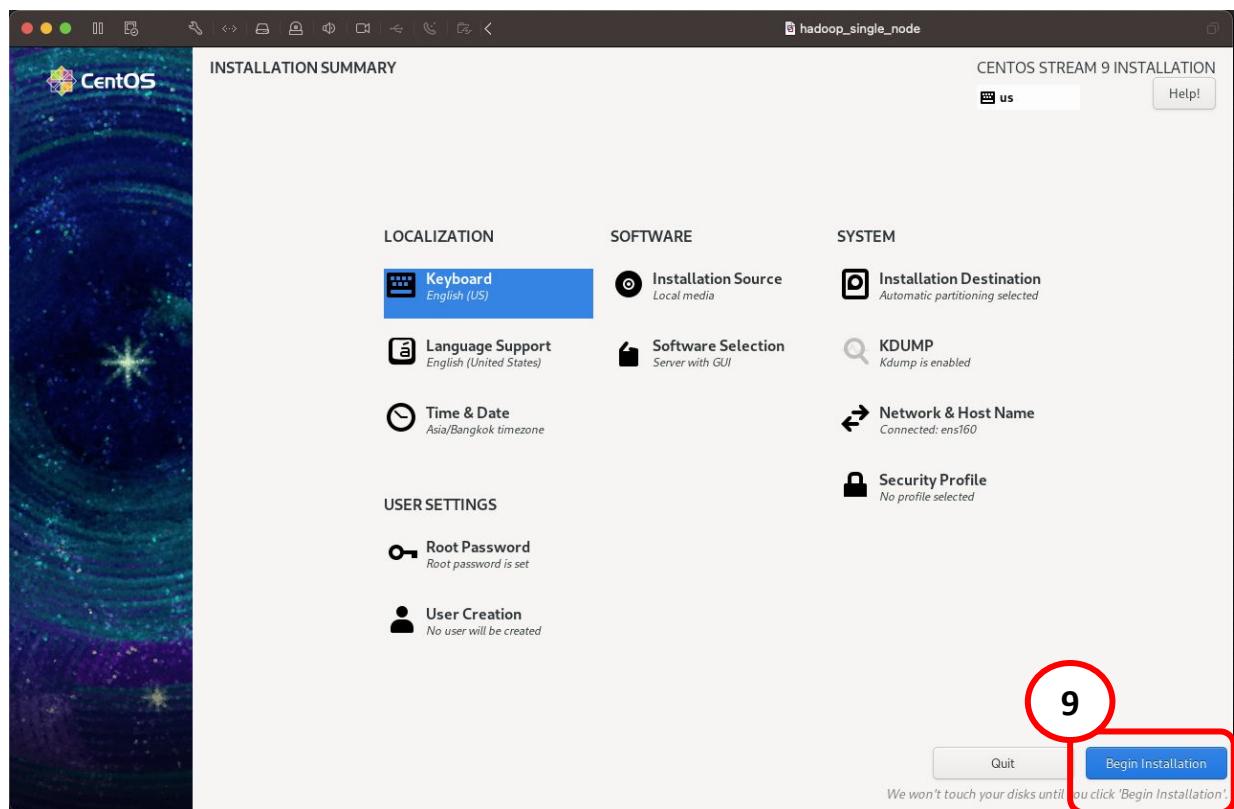
Hadoop

❖ Installation: Setup CentOS



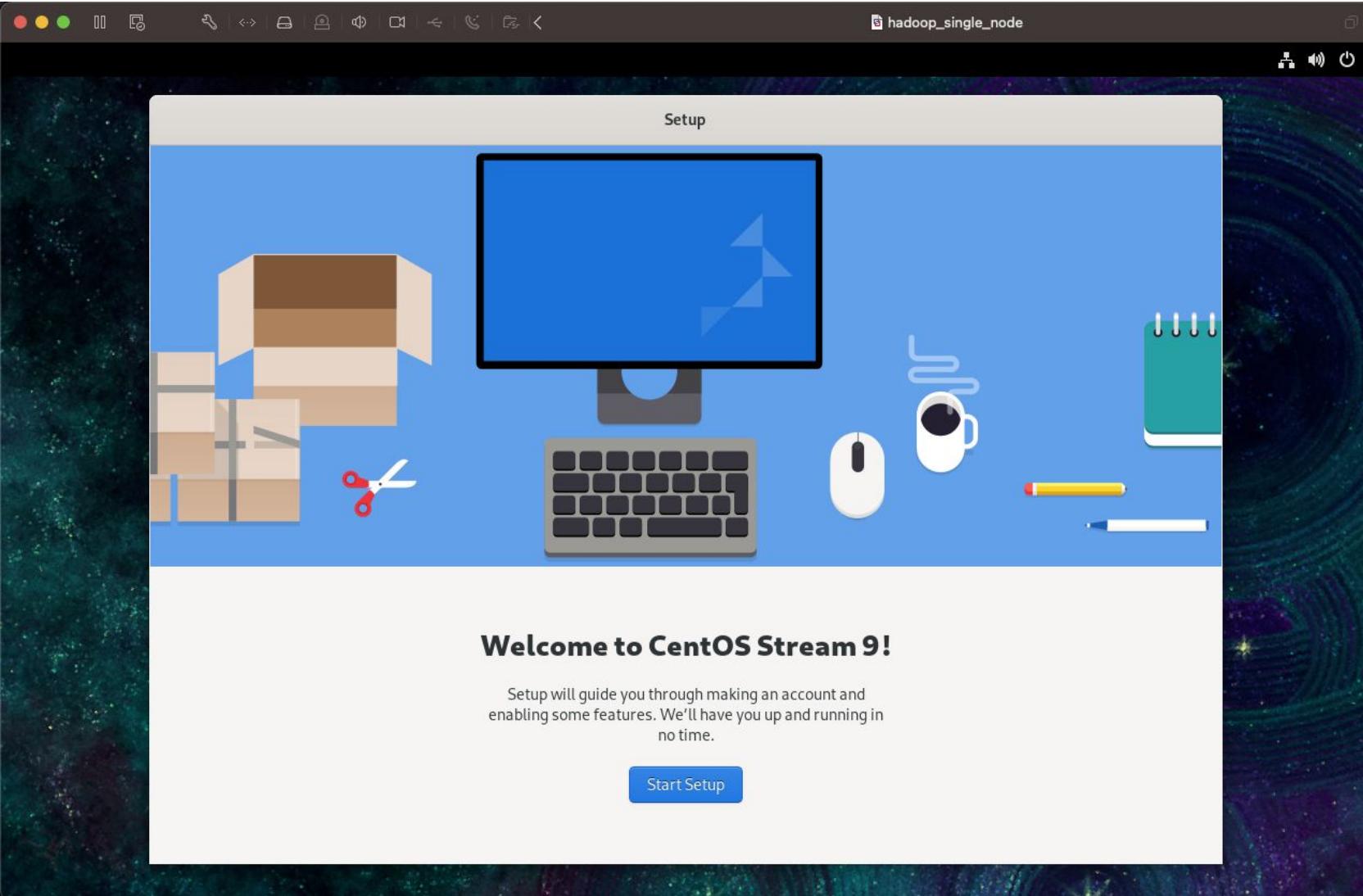
Hadoop

❖ Installation: Setup CentOS



Hadoop

❖ Installation: Setup CentOS



Hadoop

❖ Installation: Hadoop Requirements

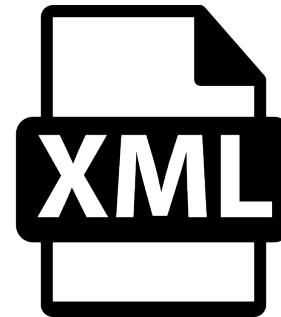
1. Install Java



2. Install Hadoop

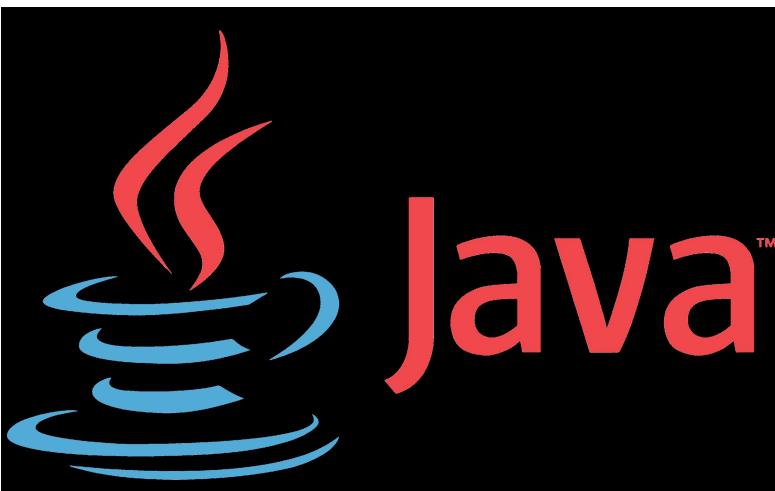


3. Set up Hadoop



Hadoop

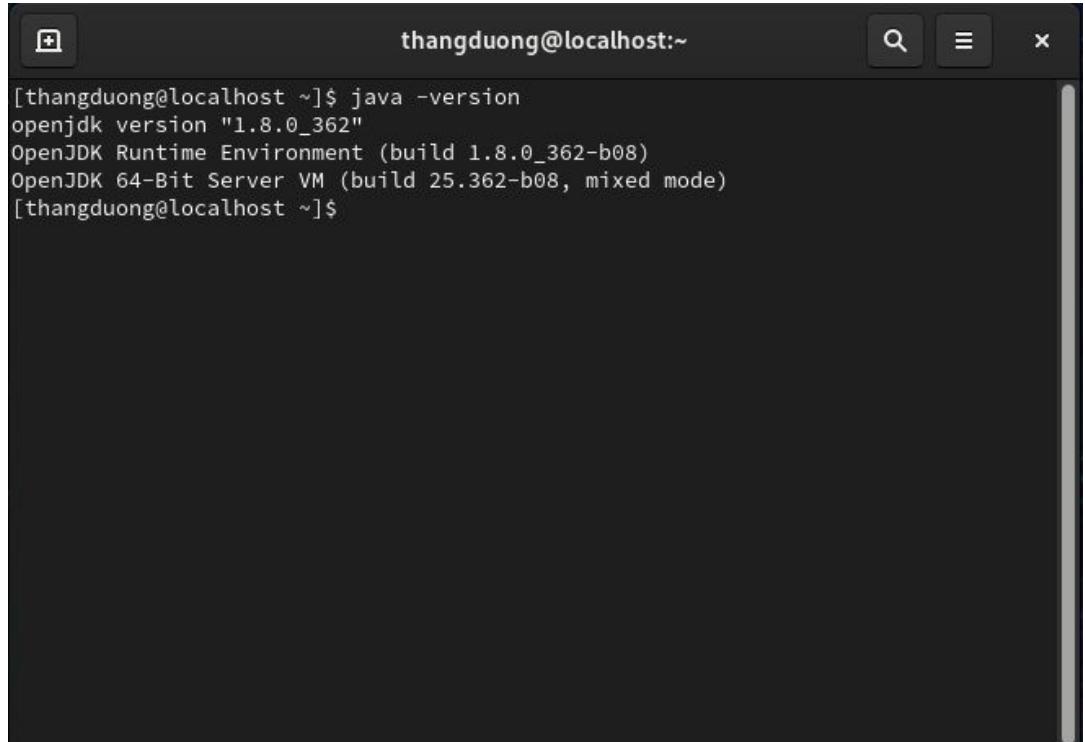
❖ Installation: Java



Java: A high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

Install Java 8 on CentOS:

1. `sudo yum -y update`
2. `sudo yum install -y java-1.8.0-openjdk`



A screenshot of a terminal window titled "thangduong@localhost:~". The window shows the command `[thangduong@localhost ~]$ java -version` being run, followed by the output: "openjdk version "1.8.0_362"\nOpenJDK Runtime Environment (build 1.8.0_362-b08)\nOpenJDK 64-Bit Server VM (build 25.362-b08, mixed mode)".

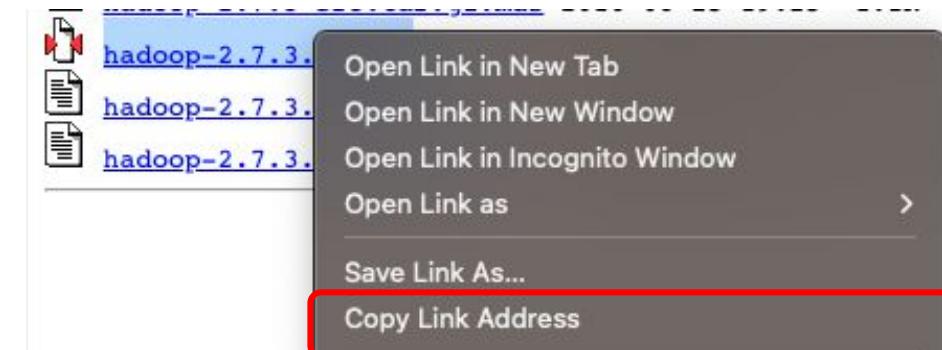
Hadoop

❖ Installation: Hadoop

Index of /dist/hadoop/core/hadoop-2.7.3

Name	Last modified	Size	Description
Parent Directory		-	
hadoop-2.7.3-src.tar.gz	2016-08-25 19:25	17M	
hadoop-2.7.3-src.tar.gz.asc	2016-08-25 19:25	521	
hadoop-2.7.3-src.tar.gz.mds	2016-08-25 19:25	1.1K	
hadoop-2.7.3.tar.gz	2016-08-25 19:25	204M	
hadoop-2.7.3.tar.gz.asc	2016-08-25 19:25	521	
hadoop-2.7.3.tar.gz.mds	2016-08-25 19:25	958	

Hadoop 2.7.3 download [page](#)



A screenshot of a terminal window titled 'thangduong@localhost:~'. The terminal shows the following command being run:

```
[thangduong@localhost ~]$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

The terminal output shows the progress of the download:

```
--2023-07-08 11:55:47-- https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz  
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2  
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 214092195 (204M) [application/x-gzip]  
Saving to: 'hadoop-2.7.3.tar.gz'
```

After the download completes, the terminal shows:

```
hadoop-2.7.3.tar.gz 100%[=====] 204.17M 2.92MB/s in 73s  
2023-07-08 11:57:02 (2.80 MB/s) - 'hadoop-2.7.3.tar.gz' saved [214092195/214092195]
```

The terminal prompt '[thangduo' is visible at the bottom.

Install Hadoop via wget command

Hadoop

❖ Installation: Hadoop

A screenshot of a terminal window titled "thangduong@localhost:~". The terminal shows the following command sequence:

```
[thangduong@localhost ~]$ ls
Desktop  Downloads      Music   Public   Videos
Documents  hadoop-2.7.3.tar.gz  Pictures  Templates
[thangduong@localhost ~]$ tar -xf hadoop-2.7.3.tar.gz
[thangduong@localhost ~]$ ls
Desktop  Downloads  hadoop-2.7.3.tar.gz  Pictures  Templates
Documents  hadoop-2.7.3  Music      Public   Videos
[thangduong@localhost ~]$ mv hadoop-2.7.3 hadoop
[thangduong@localhost ~]$ ls
Desktop  Downloads  hadoop  Pictures  Templates
Documents  hadoop-2.7.3  Music      Public   Videos
[thangduong@localhost ~]$
```

- Extract downloaded file via tar command:

```
tar -xf hadoop-2.7.3.tar.gz
```

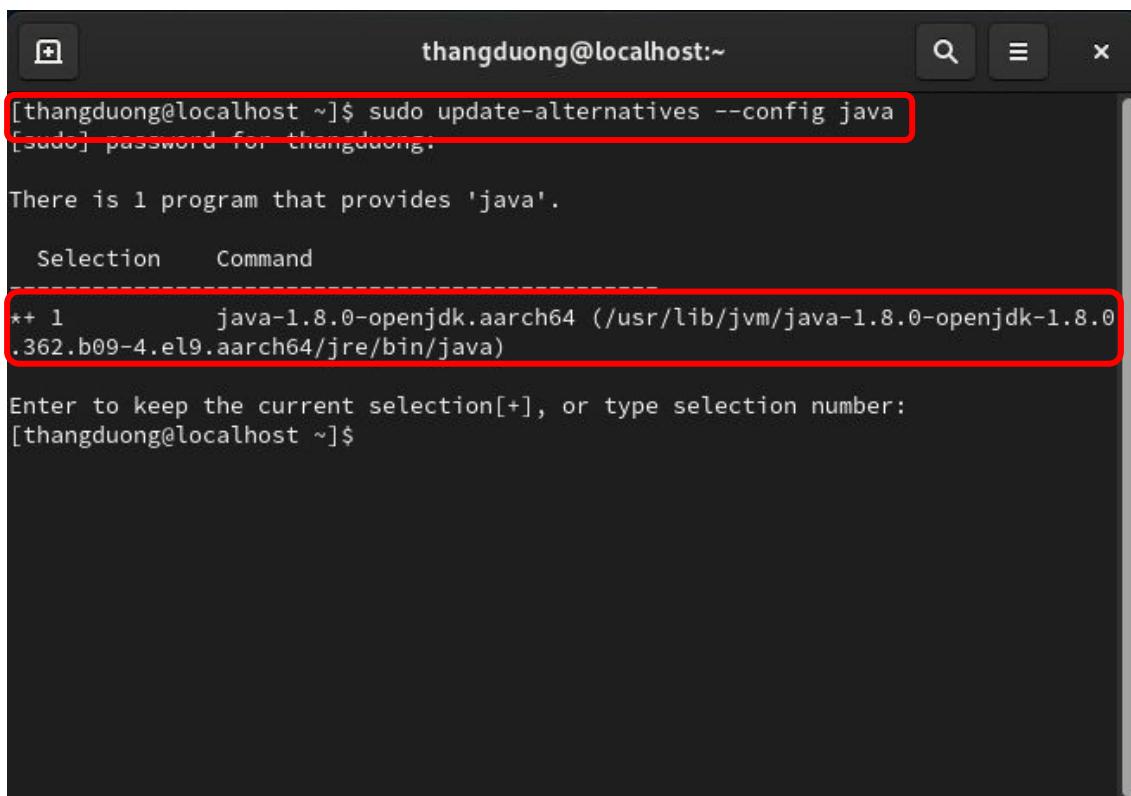
- Rename Hadoop folder:

```
mv hadoop-2.7.3 hadoop
```

Hadoop

❖ Installation: Config Hadoop

1. Check Java configuration



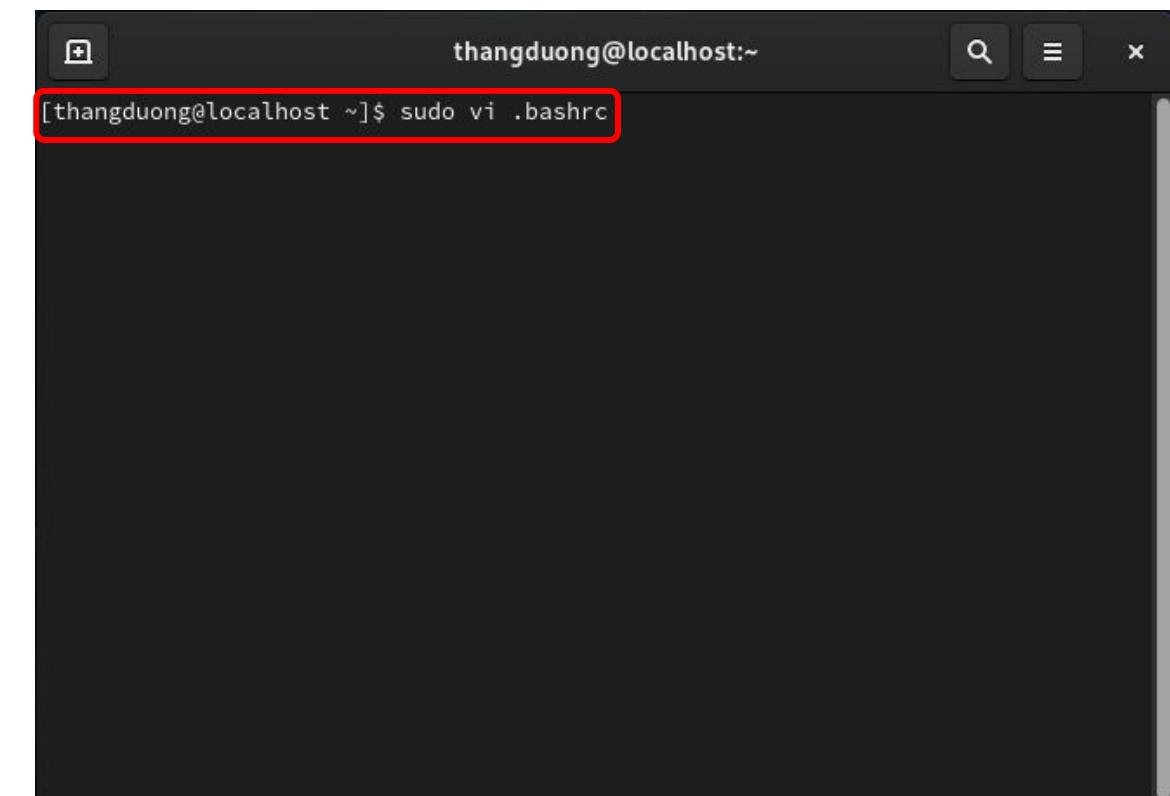
A terminal window titled "thangduong@localhost:~". The user runs the command `sudo update-alternatives --config java`. A password is entered. The output shows there is 1 program providing 'java'. The selection table lists one option: "1 java-1.8.0-openjdk.aarch64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0_362.b09-4.el9.aarch64/jre/bin/java)". The user is prompted to enter a selection number or keep the current one.

```
[thangduong@localhost ~]$ sudo update-alternatives --config java
[sudo] password for thangduong:
There is 1 program that provides 'java'.

Selection    Command
*+ 1          java-1.8.0-openjdk.aarch64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0_362.b09-4.el9.aarch64/jre/bin/java)

Enter to keep the current selection[+], or type selection number:
[thangduong@localhost ~]$
```

2. Enter ~/.bashrc script



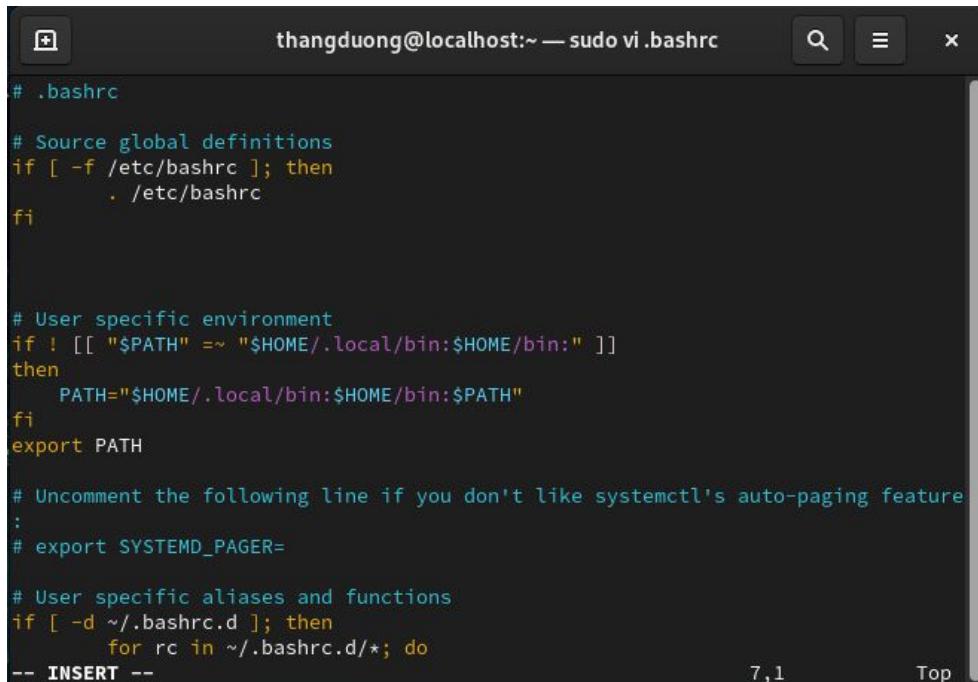
A terminal window titled "thangduong@localhost:~". The user runs the command `sudo vi .bashrc`. The screen is mostly blank, indicating the file is being edited.

```
[thangduong@localhost ~]$ sudo vi .bashrc
```

Hadoop

❖ Installation: Hadoop

3. Set up .bashrc script



A screenshot of a terminal window titled "thangduong@localhost:~ — sudo vi .bashrc". The window shows the contents of the .bashrc file. The code is as follows:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature
# export SYSTEMD_PAGER=
# User specific aliases and functions
if [ -d ~/.bashrc.d ]; then
    for rc in ~/.bashrc.d/*; do
-- INSERT --
```

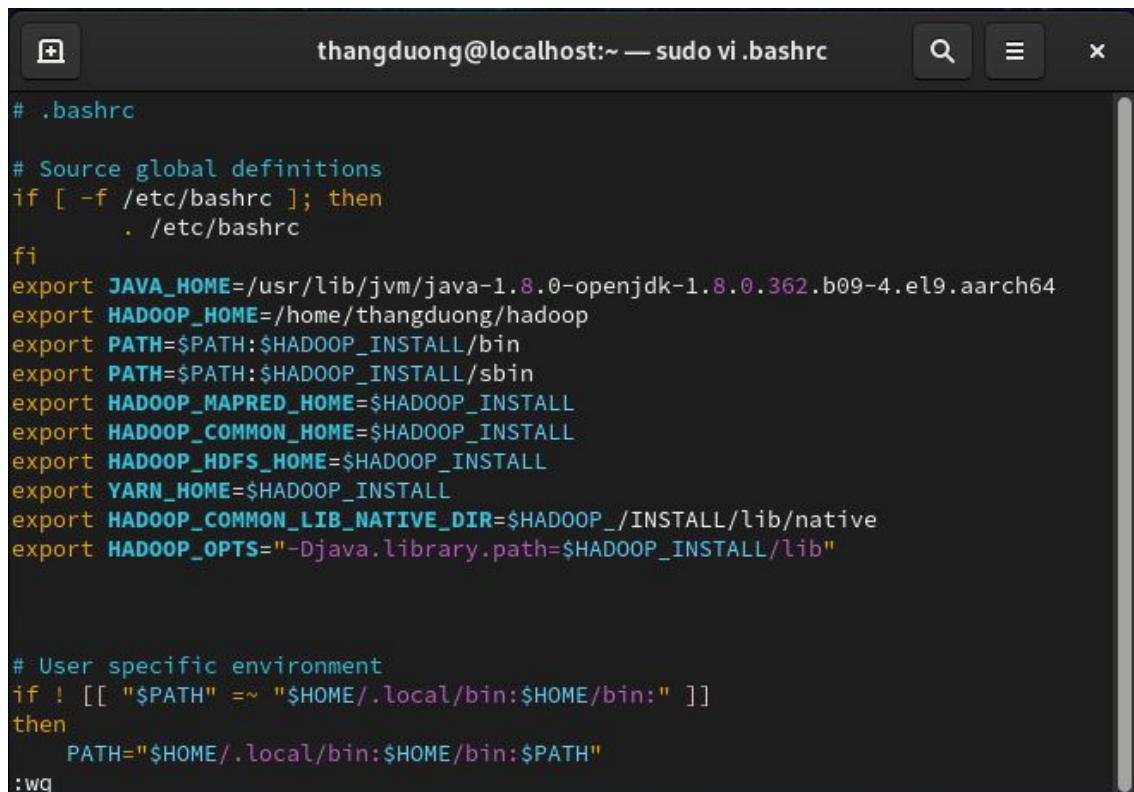
The status bar at the bottom of the terminal window shows "7,1" and "Top".

Paste the code below:

```
export
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.
0.362.b09-4.el9.aarch64
export HADOOP_HOME=/home/thangduong/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/
lib/native
export
HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTA
LL/lib"
```

Hadoop

❖ Installation: Hadoop

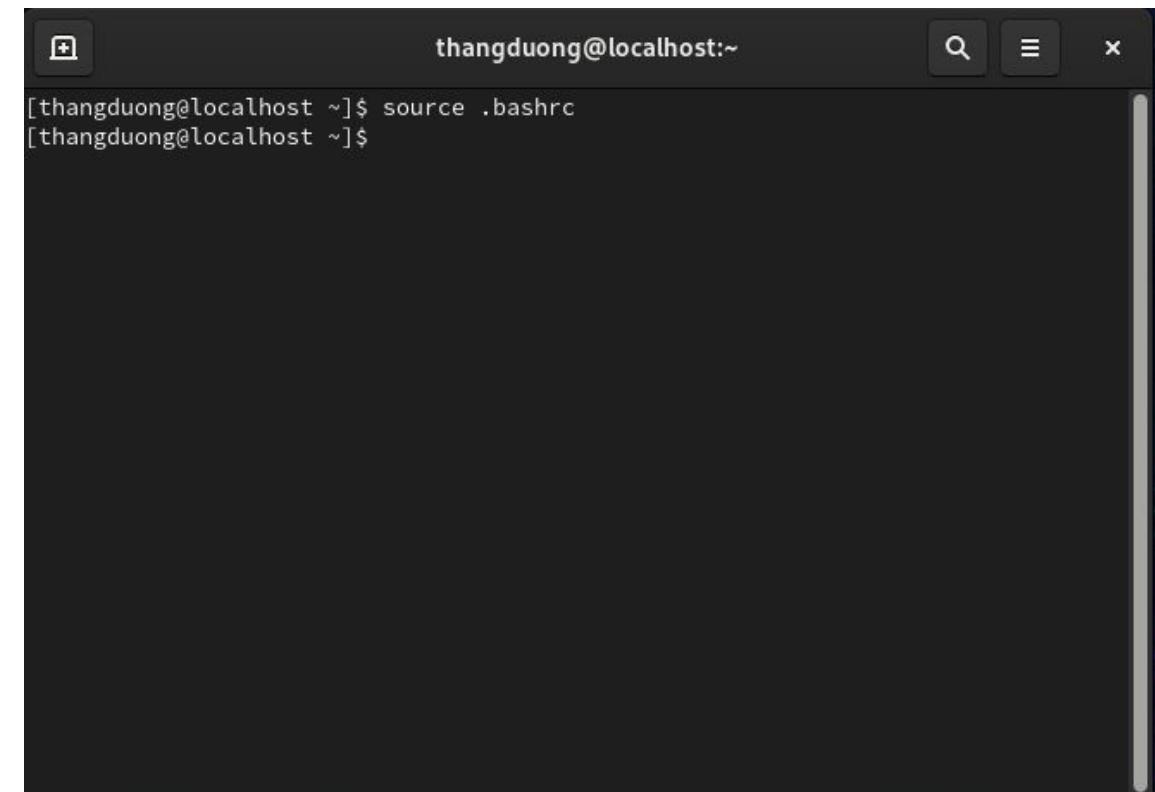


A screenshot of a terminal window titled "thangduong@localhost:~ — sudo vi .bashrc". The window shows the contents of the .bashrc file, which contains configuration for Java and Hadoop environments. The code is color-coded for syntax highlighting.

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.362.b09-4.el9.aarch64
export HADOOP_HOME=/home/thangduong/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
:wq
```



A screenshot of a terminal window titled "thangduong@localhost:~". It shows the command "source .bashrc" being run, followed by a prompt for further input.

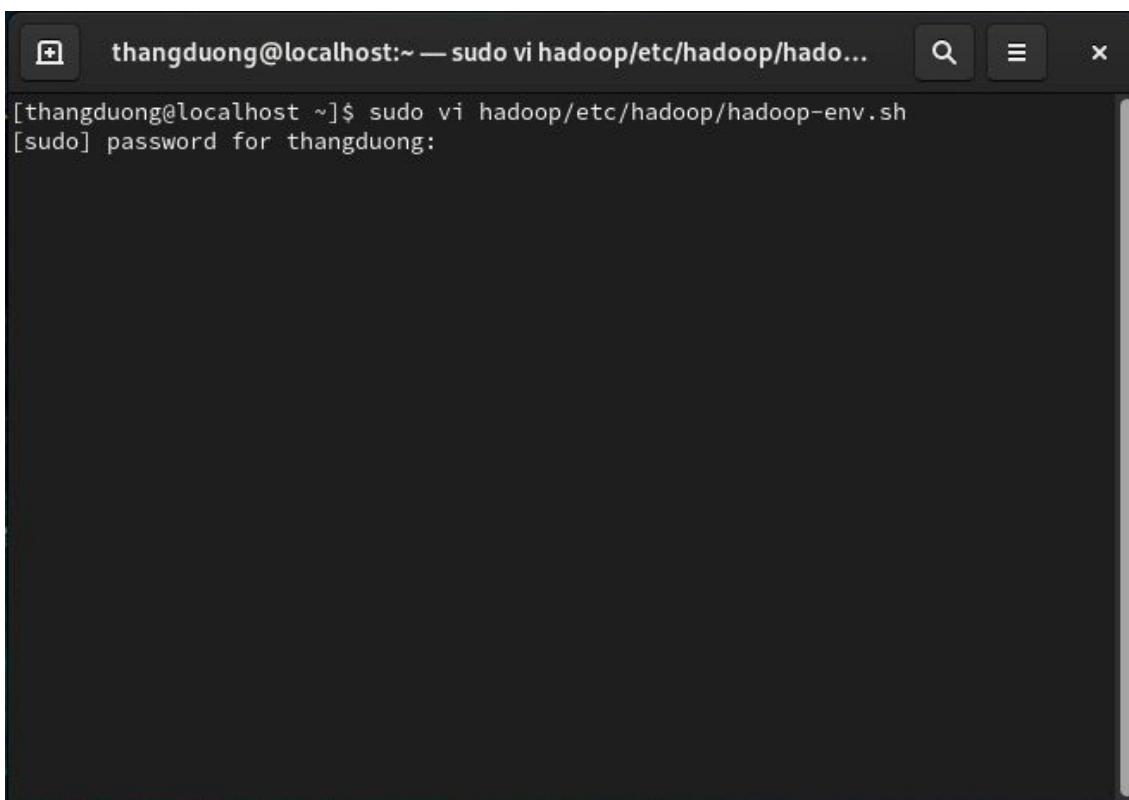
```
[thangduong@localhost ~]$ source .bashrc
[thangduong@localhost ~]$
```

3. Set up .bashrc script then run it

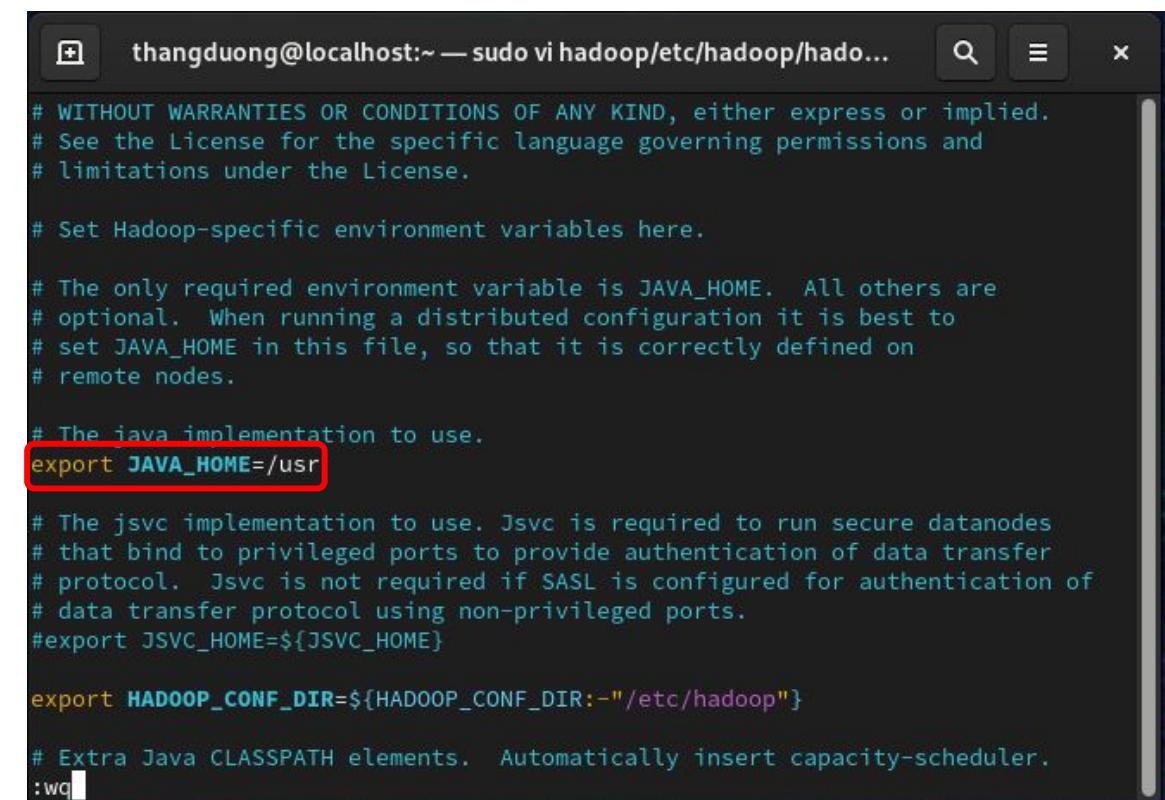
Hadoop

❖ Installation: Hadoop

4. Config hadoop-env.sh



```
[thangduong@localhost ~]$ sudo vi hadoop/etc/hadoop/hadoop-env.sh
[sudo] password for thangduong:
```



```
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol. Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}

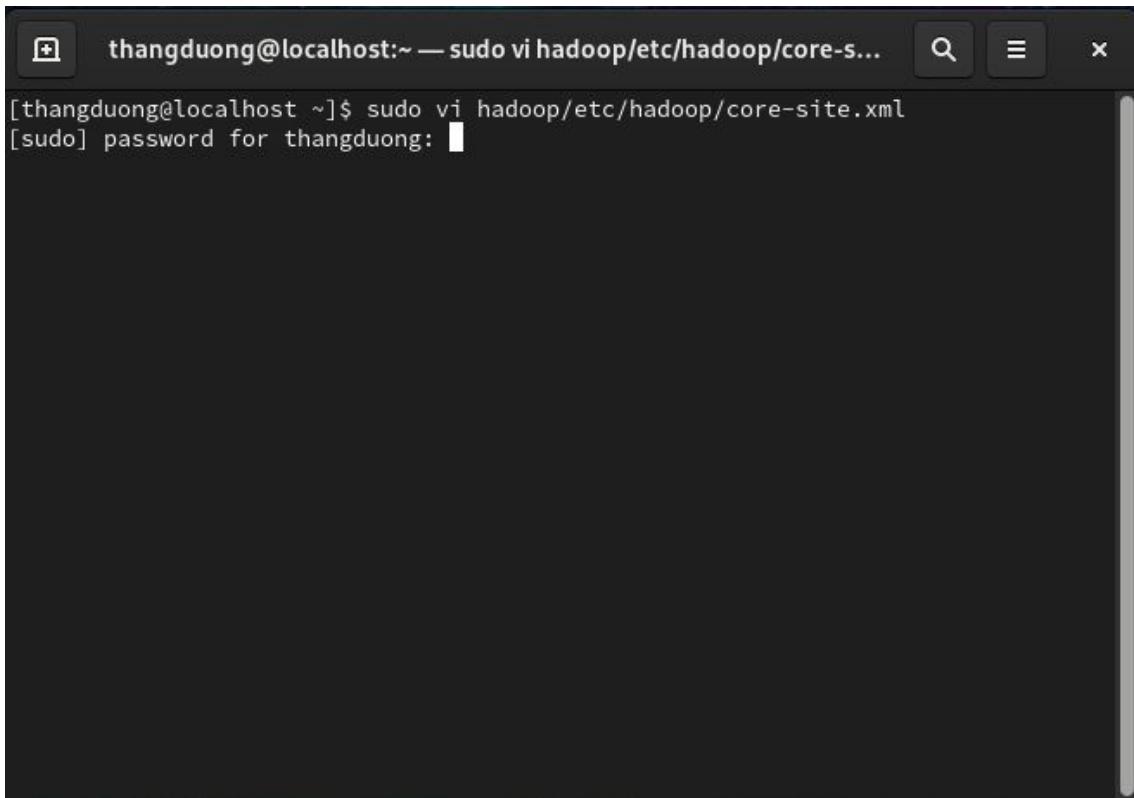
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.
:wq
```

Hadoop

❖ Installation: Hadoop

5. Config core-site.xml



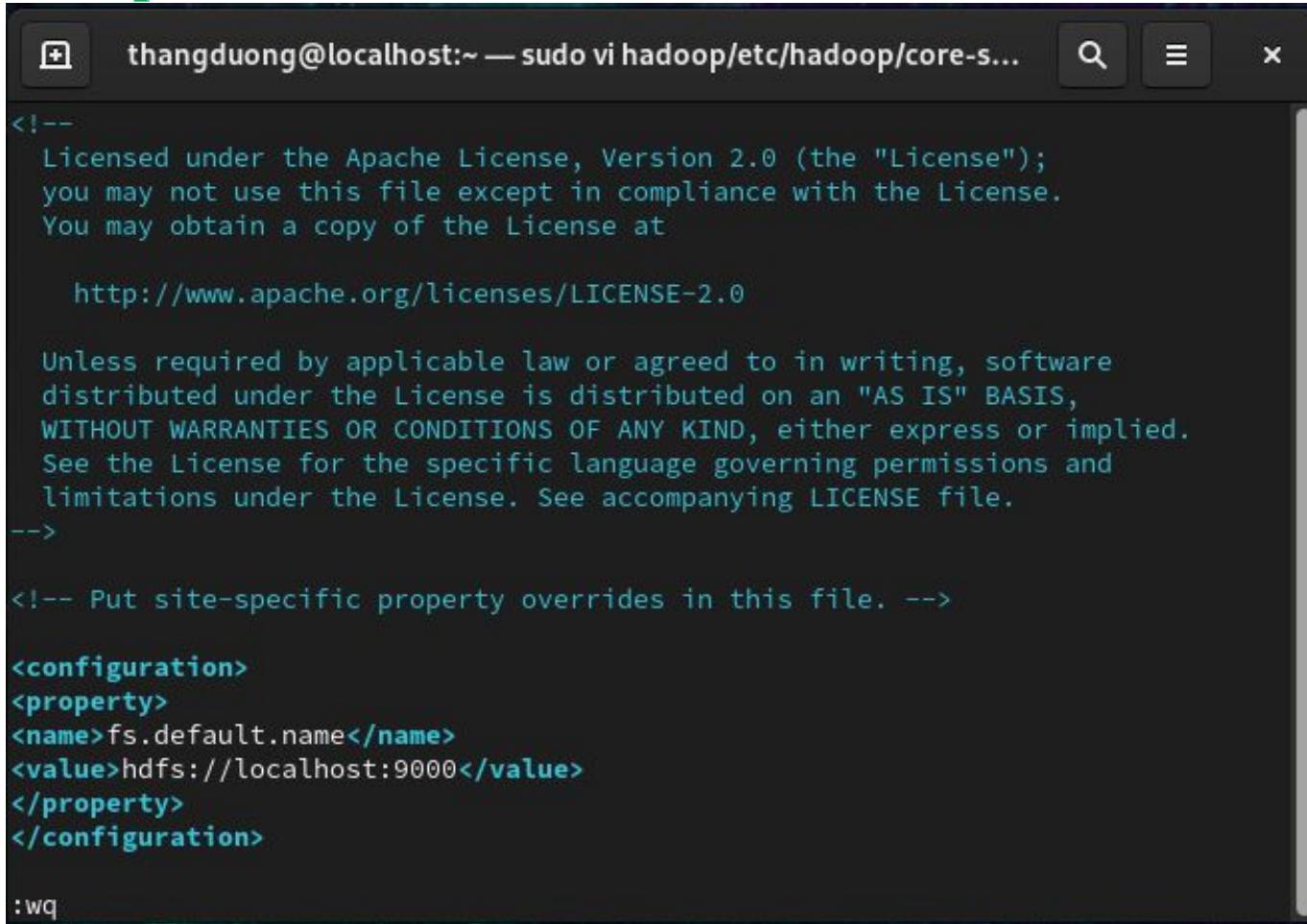
A terminal window titled "thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/core-s...". The command "sudo vi hadoop/etc/hadoop/core-site.xml" is entered, followed by a password prompt "[sudo] password for thangduong: [REDACTED]".

Paste the code below:

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

Hadoop

❖ Installation: Hadoop



A screenshot of a terminal window titled "thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/core-s...". The window displays the contents of the core-site.xml configuration file. The file includes the Apache License 2.0 header, followed by XML code defining a property for the default file system name and its value as "hdfs://localhost:9000". The command ":wq" is visible at the bottom, indicating the file has been saved and quit.

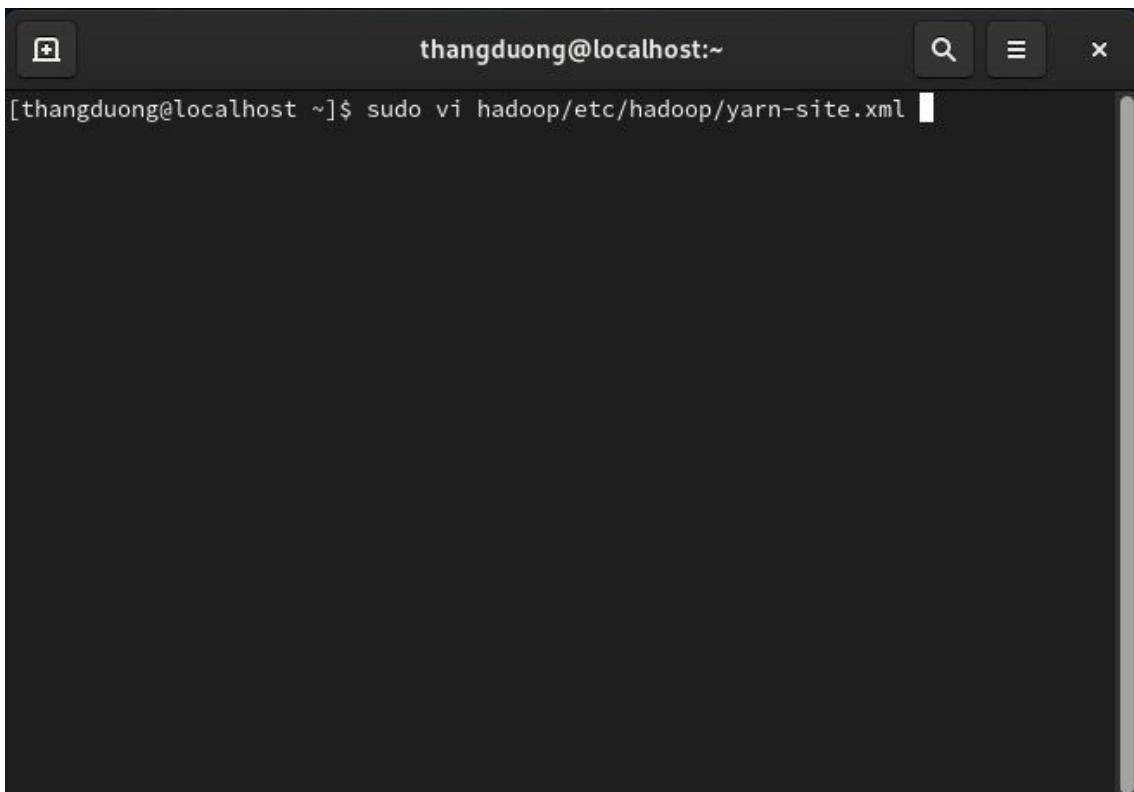
```
<!--  
 Licensed under the Apache License, Version 2.0 (the "License");  
 you may not use this file except in compliance with the License.  
 You may obtain a copy of the License at  
  
 http://www.apache.org/licenses/LICENSE-2.0  
  
 Unless required by applicable law or agreed to in writing, software  
 distributed under the License is distributed on an "AS IS" BASIS,  
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 See the License for the specific language governing permissions and  
 limitations under the License. See accompanying LICENSE file.  
-->  
  
<!-- Put site-specific property overrides in this file. -->  
  
<configuration>  
<property>  
<name>fs.default.name</name>  
<value>hdfs://localhost:9000</value>  
</property>  
</configuration>  
  
:wq
```

5. Config core-site.xml

Hadoop

❖ Installation: Hadoop

6. Config yarn-site.xml



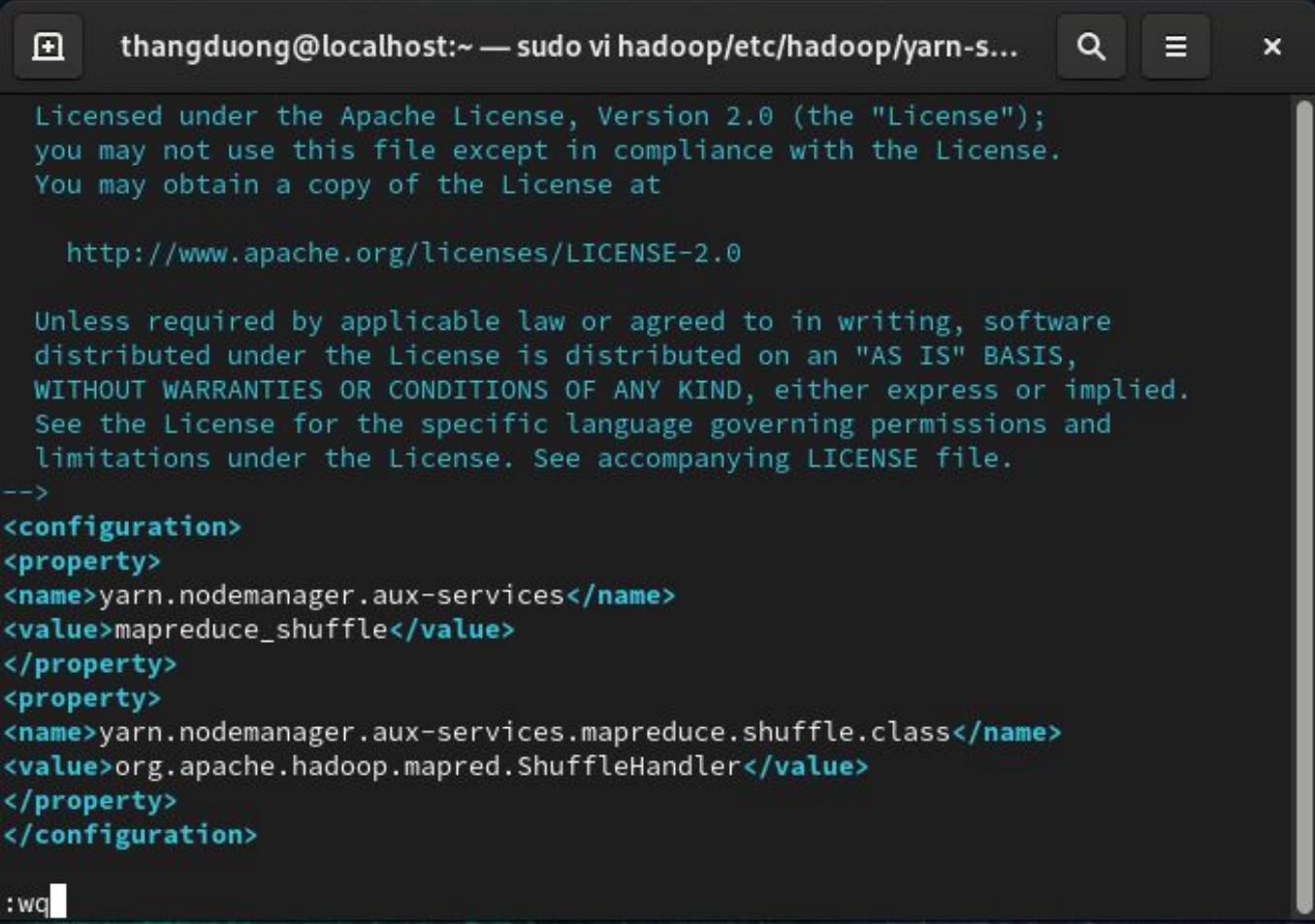
A terminal window titled "thangduong@localhost:~". The command "sudo vi hadoop/etc/hadoop/yarn-site.xml" is entered in the terminal.

Paste the code below:

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

Hadoop

❖ Installation: Hadoop



```
thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/yarn-s...

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.

-->
<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>
</configuration>

:wq
```

6. Config yarn-site.xml

Hadoop

❖ Installation: Hadoop

7. Config mapred-site.xml

```
thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/mapr...
[thangduong@localhost ~]$ cp hadoop/etc/hadoop/mapred-
mapred-env.cmd          mapred-queues.xml.template
mapred-env.sh             mapred-site.xml.template
[thangduong@localhost ~]$ cp hadoop/etc/hadoop/mapred-site.xml.template hadoop/e
tc/hadoop/mapred-site.xml
[thangduong@localhost ~]$ sudo vi hadoop/etc/hadoop/mapred-site.xml
[sudo] password for thangduong: ■
```

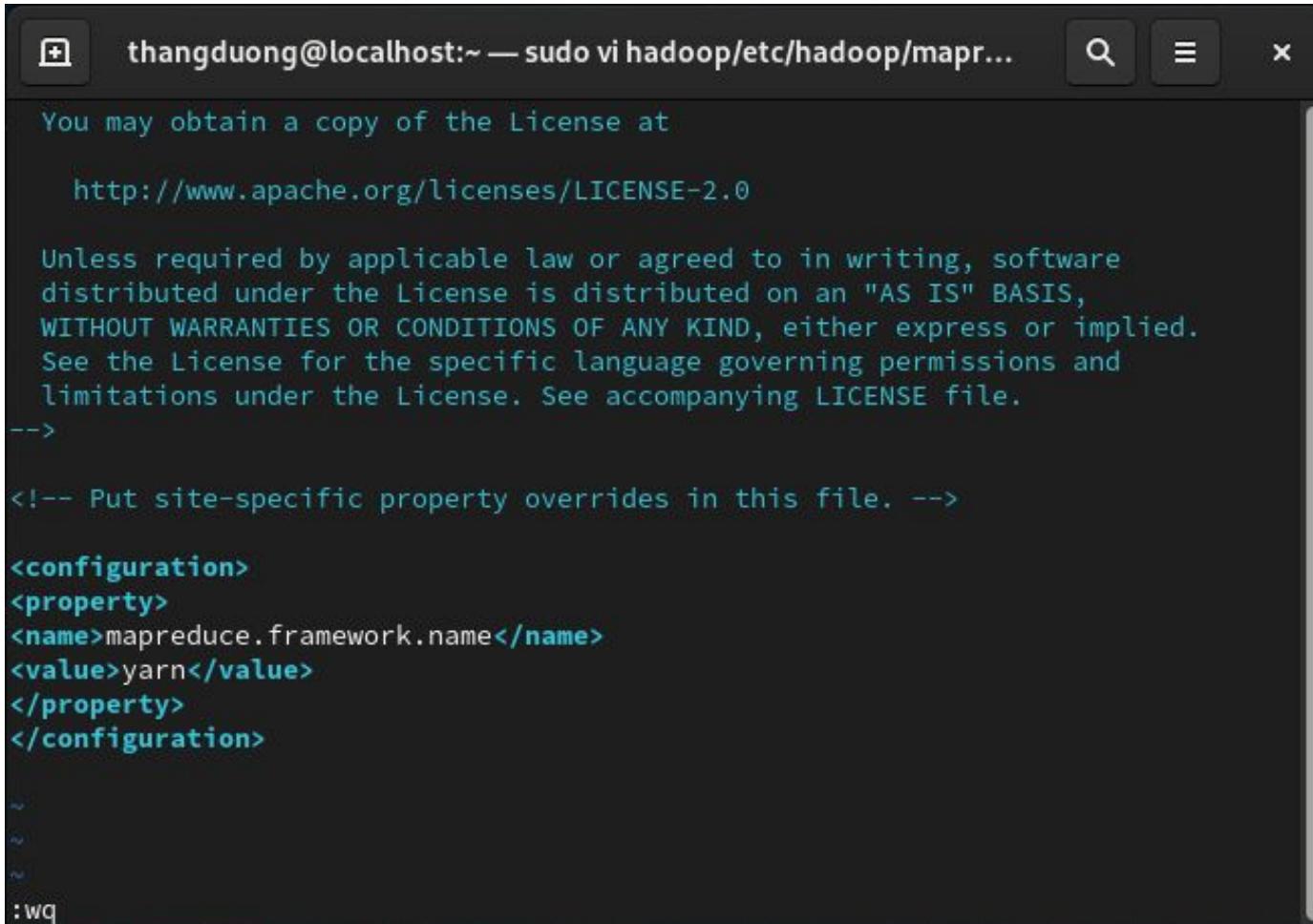
Copy and edit on new mapred-site.xml

Paste the code below:

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

Hadoop

❖ Installation: Hadoop



A screenshot of a terminal window titled "thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/mapr...". The terminal displays the contents of the mapred-site.xml configuration file. It shows the Apache License 2.0 header, followed by XML code defining the mapreduce.framework.name property to 'yarn'. The terminal prompt at the bottom is ':wq'.

```
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. -->

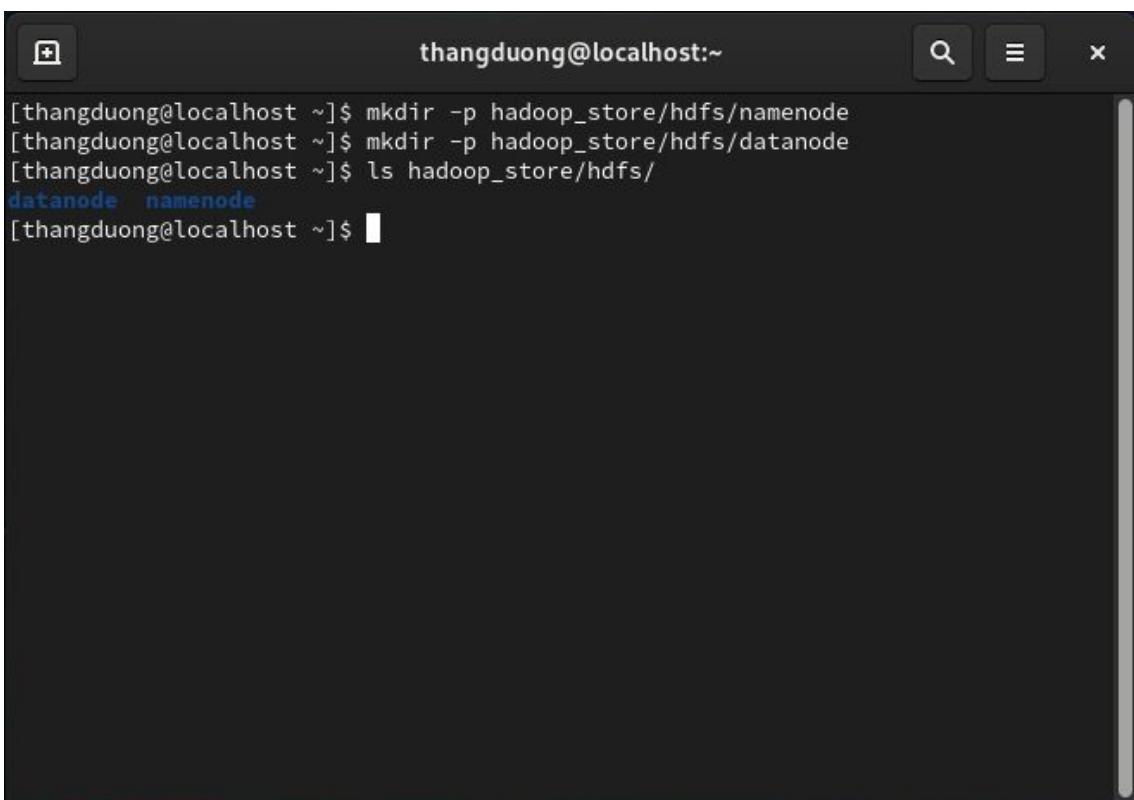
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

:q
:q
:q
:wq
```

Hadoop

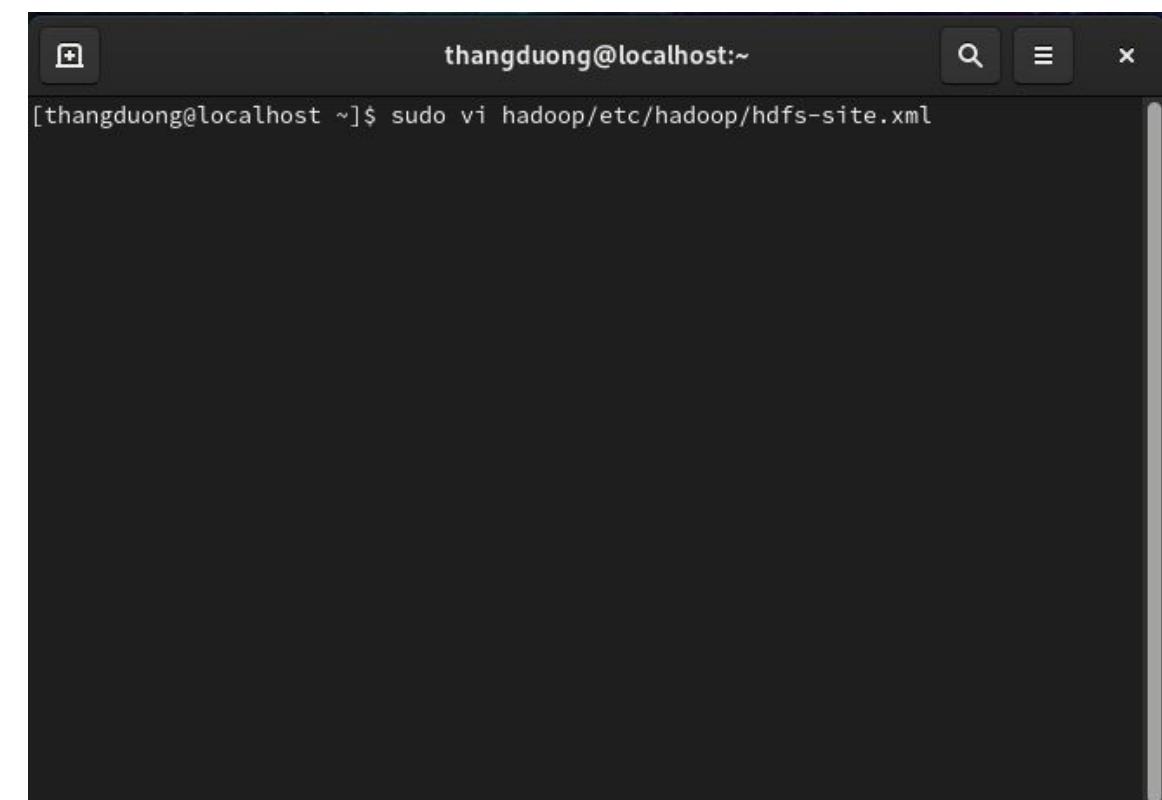
❖ Installation: Hadoop

8.1. Create namenode and datanode folder



```
thangduong@localhost:~$ mkdir -p hadoop_store/hdfs/namenode
[thangduong@localhost ~]$ mkdir -p hadoop_store/hdfs/datanode
[thangduong@localhost ~]$ ls hadoop_store/hdfs/
datanode  namenode
[thangduong@localhost ~]$
```

8.2. Config hdfs-site.xml



```
thangduong@localhost:~$ sudo vi hadoop/etc/hadoop/hdfs-site.xml
```

Hadoop

❖ Installation: Hadoop

```
thangduong@localhost:~ — sudo vi hadoop/etc/hadoop/hdfs-s...
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property><name>dfs.namenode.name.dir</name>
<value>file:/home/thangduong/hadoop_store/hdfs/namenode</value>
</property>
<property> <name>dfs.datanode.data.dir</name>
<value>file:/home/thangduong/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

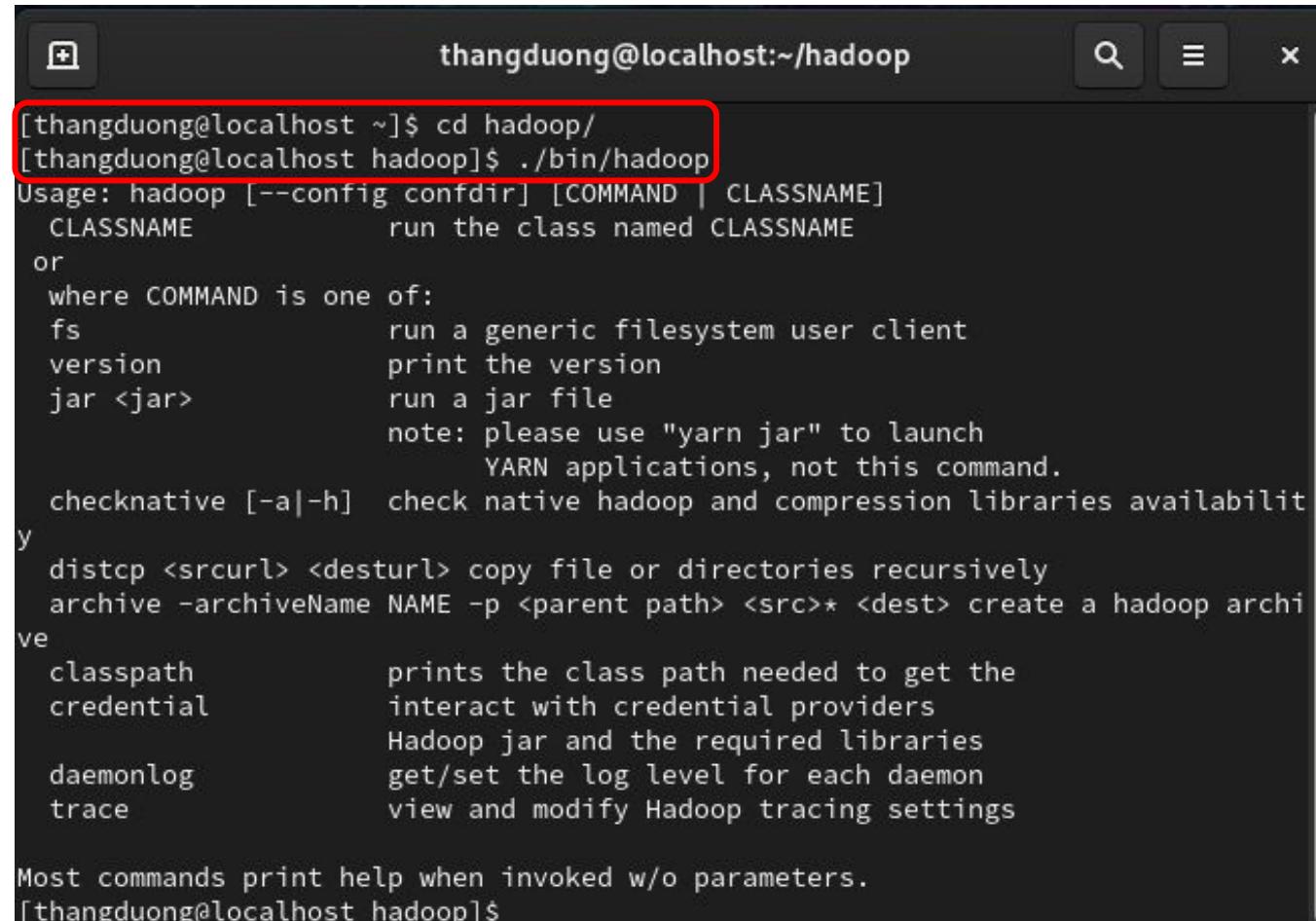
:wq
```

Paste the code below:

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/home/thangduong/hadoop_store/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/home/thangduong/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

Hadoop

❖ Installation: Check Hadoop



A screenshot of a terminal window titled "thangduong@localhost:~/hadoop". The window shows the output of the command `./bin/hadoop`. The output is a usage help message for the Hadoop command-line interface. The first two lines of the help message, which include the command name, are highlighted with a red rectangle.

```
[thangduong@localhost ~]$ cd hadoop/  
[thangduong@localhost hadoop]$ ./bin/hadoop  
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]  
CLASSNAME           run the class named CLASSNAME  
or  
where COMMAND is one of:  
fs                  run a generic filesystem user client  
version             print the version  
jar <jar>          run a jar file  
                   note: please use "yarn jar" to launch  
                   YARN applications, not this command.  
checknative [-a|-h]  check native hadoop and compression libraries availability  
distcp <srcurl> <desturl> copy file or directories recursively  
archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive  
classpath           prints the class path needed to get the  
credential          interact with credential providers  
Hadoop jar and the required libraries  
daemonlog           get/set the log level for each daemon  
trace               view and modify Hadoop tracing settings  
  
Most commands print help when invoked w/o parameters.  
[thangduong@localhost hadoop]$
```

Check Hadoop Installation

❖ Example: Search text

grep Command

Last Updated: 2023-03-24

Purpose

Searches for a pattern in a file.

Syntax

Pattern to search: “dfs[a-z.]”

```
grep [-E | -F] [-i] [-h] [-H] [-L] [-r | -R] [-s] [-u] [-v] [-w] [-x] [-y] [([-b] [-n]) | (-c | -l | -q)] [-p [Separator]] { [-e PatternList ...] | -f PatternFile ... } | PatternList ... } | File ... ]
```

Description

The **grep** command searches for the pattern specified by the *Pattern* parameter and writes each matching line to standard output. The patterns are limited regular expressions in the style of the **ed** or **egrep** command. The **grep** command uses a compact non-deterministic algorithm.

Hadoop

◆ Example: Search text

1. Initialize namenode

```
[thangduong@localhost hadoop]$ ./bin/hdfs namenode -format
23/07/08 14:26:34 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.3
STARTUP_MSG:   classpath = /home/thangduong/hadoop/etc/hadoop:/home/thangduong/hadoop/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jaxb-api-2.2.2.jar:/home/thangduong/hadoop/share/hadoop/common/lib/stax-api-1.0-2.jar:/home/thangduong/hadoop/share/hadoop/common/lib/activation-1.1.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jackson-core-asl-1.9.13.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jackson-mapper-asl-1.9.13.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jackson-xc-1.9.13.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jersey-server-1.9.jar:/home/thangduong/hadoop/share/hadoop/common/lib/asm-3.2.jar:/home/thangduong/hadoop/share/hadoop/common/lib/log4j-1.2.17.jar:/home/thangduong/hadoop/share/hadoop/common/lib/jets3t-0.9.0.jar:/home/thangduong/hadoop/share/hadoop/common/lib/httpclient-4.2.5.jar:/home/thangduong/hadoop/share/hadoop/common/lib/httpcore-4.2.5.jar:/home/thangduong/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.4.jar:/home/thangduong/hadoop
```

2. Start namenode and datanode daemon

```
[thangduong@localhost hadoop]$ ./sbin/start-dfs.sh
23/07/08 14:26:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
thangduong@localhost's password:
localhost: starting namenode, logging to /home/thangduong/hadoop/logs/hadoop-thangduong-namenode-localhost.localdomain.out
thangduong@localhost's password:
localhost: datanode running as process 39009. Stop it first.
Starting secondary namenodes [0.0.0.0]
thangduong@0.0.0.0's password:
0.0.0.0: secondarynamenode running as process 39216. Stop it first.
23/07/08 14:26:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

Hadoop

◆ Example: Search text

3. Create input folder and move all .xml file into input

```
[thangduong@localhost hadoop]$ ./bin/hdfs dfs -mkdir /input
23/07/08 14:27:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[thangduong@localhost hadoop]$ ./bin/hdfs dfs -put etc/hadoop/*.xml /input
23/07/08 14:27:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

4. Perform grep with string pattern “dfs[a-z.]⁺”

```
[thangduong@localhost hadoop]$ ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar grep /input /output 'dfs[a-z.]+'
23/07/08 14:29:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
23/07/08 14:29:21 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/07/08 14:29:22 INFO input.FileInputFormat: Total input paths to process : 9
23/07/08 14:29:22 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1257)
        at java.lang.Thread.join(Thread.java:1331)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:609)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:370)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:294)
```

Hadoop

❖ Example: Search text

```
[thangduong@localhost hadoop]$ ./bin/hdfs dfs -get /output output
23/07/08 14:30:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[thangduong@localhost hadoop]$ cat output/*
1      dfsadmin
1      dfs.replication
1      dfs.namenode.name.dir
1      dfs.datanode.data.dir
```

5. Get output and print results

Hadoop

❖ Example: Search text

```
[thangduong@localhost hadoop]$ ./bin/hdfs dfs -get /output output
23/07/08 14:30:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[thangduong@localhost hadoop]$ cat output/*
1      dfsadmin
1      dfs.replication
1      dfs.namenode.name.dir
1      dfs.datanode.data.dir
```

5. Get output and print results

Hadoop

❖ Example: Other default applications

Valid program names are:

aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.

aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.

bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi

dbcount: An example job that count the pageview counts from a database.

distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.

grep: A map/reduce program that counts the matches of a regex in the input.

join: A job that effects a join over sorted, equally partitioned datasets

multifilewc: A job that counts words from several files.

pentomino: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reduce.

sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort

terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.

wordmean: A map/reduce program that counts the average length of the words in the input files.

wordmedian: A map/reduce program that counts the median length of the words in the input files.

wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

QUIZ

Spark

Spark

❖ Introduction



Apache Spark: an open-source unified analytics engine for **large-scale data processing**. Spark is an alternative replacement for Hadoop MapReduce.

Spark

◆ Why Spark?



Fast

Batch Processing

Stores Data on Disk

Written in Java

Low Cost



100x Faster than MapReduce

Real-time Processing

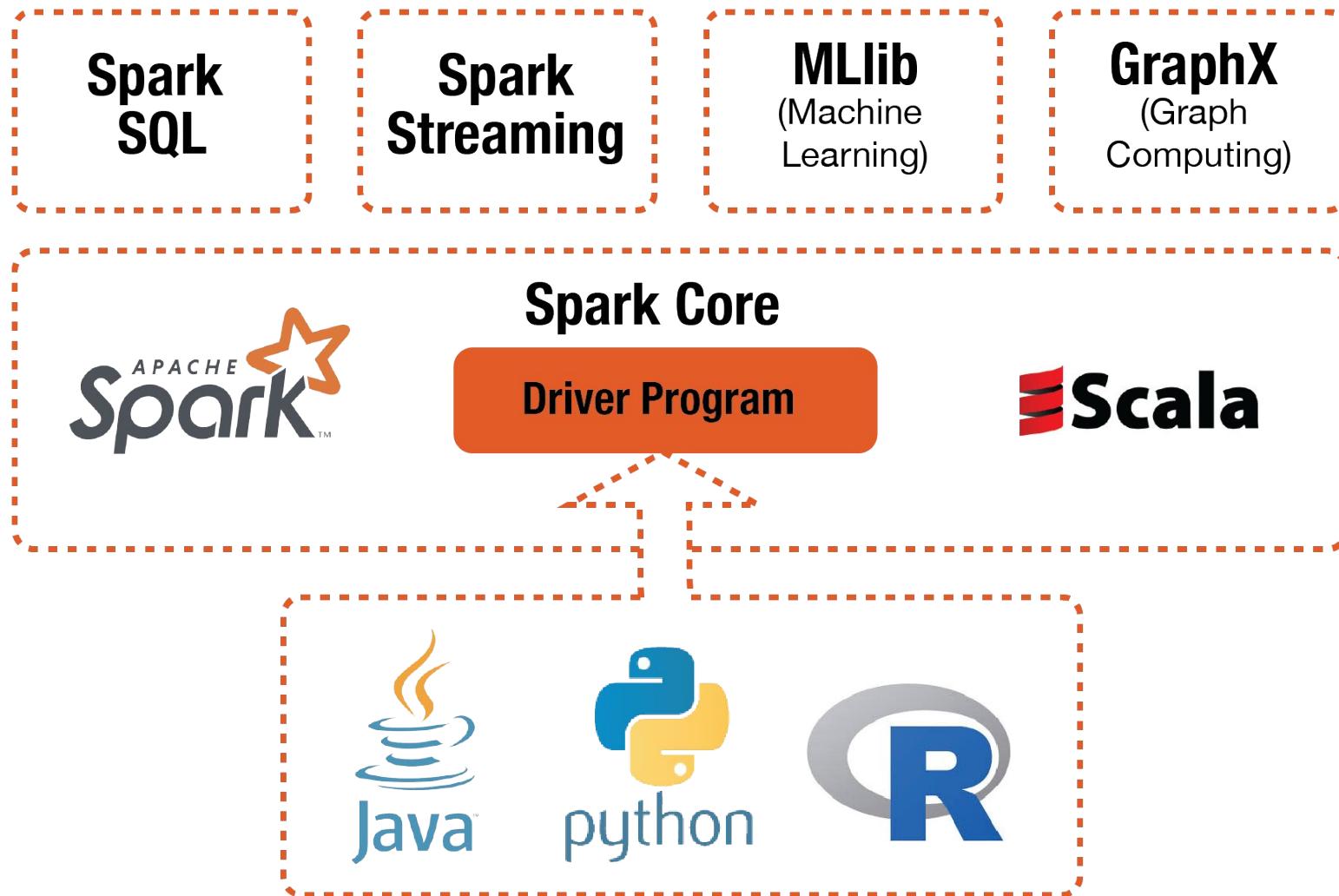
Stores Data Memory

Written in Scala

High Cost

Spark

◆ Spark Modules



Spark

❖ PySpark Introduction



PySpark: The Python API for Apache Spark. It enables to perform real-time, large-scale data processing in a distributed environment using Python.

Spark

❖ PySpark Installation

▼ Install pyspark

```
[ ] 1 !pip install pyspark==3.4.1
```

```
Requirement already satisfied: pyspark==3.4.1 in /usr/local/lib/python3.10/dist-packages (3.4.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark==3.4.1) (0.10.9.7)
```

Spark

◆ PySpark Usage

▼ Init pyspark session

```
[ ] 1 from pyspark.sql import SparkSession  
  
[ ] 1 spark = SparkSession.builder.appName('Example').getOrCreate()  
  
[ ] 1 print(spark)  
  
<pyspark.sql.session.SparkSession object at 0x7efffefc7190>
```

SparkSession: Represents the connection to a Spark cluster and provides a way to interact with various Spark APIs. It allows to create DataFrames, execute SQL queries, and perform distributed data processing tasks.

Spark

◆ PySpark Usage: Create DataFrame

```
1 data = [
2     {'name': 'Rin', 'age': 20, 'gender': 'F'},
3     {'name': 'Thang', 'age': 22, 'gender': 'M'},
4     {'name': 'Saitama', 'age': 25, 'gender': 'M'},
5     {'name': 'Edward', 'age': 31, 'gender': 'M'},
6     {'name': 'Violet', 'age': 17, 'gender': 'F'}
7 ]
```

```
1 df = spark.createDataFrame(
2     data=data
3 )
```

```
1 df.printSchema()
```

root
|-- age: long (nullable = true)
|-- gender: string (nullable = true)
|-- name: string (nullable = true)

```
1 df.show()
```

age	gender	name
20	F	Rin
22	M	Thang
25	M	Saitama
31	M	Edward
17	F	Violet

◆ PySpark Usage: Operations

```
| 1 # Select  
2 df.select('age').show()
```

```
+---+  
| age|  
+---+  
| 20|  
| 22|  
| 25|  
| 31|  
| 17|  
+---+
```

```
| 1 # Filter  
2 df.filter(df.gender == 'F').show()
```

```
+---+-----+-----+  
| age|gender| name|  
+---+-----+-----+  
| 20|      F|   Rin|  
| 17|      F|Violet|  
+---+-----+-----+
```

select(): Choose columns to show in table

filter(): Extract rows satisfied a criteria

Spark

◆ PySpark Example: Working with csv file

Description: Perform some basic operation of Spark with Flights.csv dataset. Download link: [here](#).

year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
2014	12	8	658	-7	935	-5	VX	N846VA	1780	SEA	LAX	132	954	6	58
2014	1	22	1040	5	1505	5	AS	N559AS	851	SEA	HNL	360	2677	10	40
2014	3	9	1443	-2	1652	2	VX	N847VA	755	SEA	SFO	111	679	14	43
2014	4	9	1705	45	1839	34	WN	N360SW	344	PDX	SJC	83	569	17	5
2014	3	9	754	-1	1015	1	AS	N612AS	522	SEA	BUR	127	937	7	54

Spark

◆ PySpark Example: Working with csv file

```
1 from pyspark.sql import SparkSession

1 spark = SparkSession.builder.appName('Example').getOrCreate()

1 flights_filepath = './flights_small.csv'
2 flights_df = spark.read.csv(flights_filepath, header=True)
```

```
1 print(flights_df.printSchema())
2 print(flights_df.show(5))

root
|-- year: string (nullable = true)
|-- month: string (nullable = true)
|-- day: string (nullable = true)
|-- dep_time: string (nullable = true)
|-- dep_delay: string (nullable = true)
|-- arr_time: string (nullable = true)
|-- arr_delay: string (nullable = true)
|-- carrier: string (nullable = true)
|-- tailnum: string (nullable = true)
|-- flight: string (nullable = true)
|-- origin: string (nullable = true)
|-- dest: string (nullable = true)
|-- air_time: string (nullable = true)
|-- distance: string (nullable = true)
|-- hour: string (nullable = true)
|-- minute: string (nullable = true)

None
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|year|month|day|dep_time|dep_delay|arr_time|arr_delay|carrier|tailnum|flight|origin|dest|air_time|distance|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2014| 12 | 8 | 658 | -7 | 935 | -5 | VX | N846VA | 1780 | SEA | LAX | 132 | 954 | 6 | 58 |
|2014| 1 | 22 | 1040 | 5 | 1505 | 5 | AS | N559AS | 851 | SEA | HNL | 360 | 2677 | 10 | 40 |
|2014| 3 | 9 | 1443 | -2 | 1652 | 2 | VX | N847VA | 755 | SEA | SFO | 111 | 679 | 14 | 43 |
|2014| 4 | 9 | 1705 | 45 | 1839 | 34 | WN | N360SW | 344 | PDX | SJC | 83 | 569 | 17 | 5 |
|2014| 3 | 9 | 754 | -1 | 1015 | 1 | AS | N612AS | 522 | SEA | BUR | 127 | 937 | 7 | 54 |
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Spark

◆ PySpark Example: Working with csv file

- ▼ Create new column: Flight Duration

```
[ ] 1 flights_df = flights_df.withColumn('duration_hours', flights_df.air_time / 60)
2 print(flights_df.show())
```

year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute	duration_hours
2014	12	8	658	-7	935	-5	VX	N846VA	1780	SEA	LAX	132	954	6	58	2.2
2014	1	22	1040	5	1505	5	AS	N559AS	851	SEA	HNL	360	2677	10	40	6.0
2014	3	9	1443	-2	1652	2	VX	N847VA	755	SEA	SFO	111	679	14	43	1.85
2014	4	9	1705	45	1839	34	WN	N360SW	344	PDX	SJC	83	569	17	5 1.3833333333333333	
2014	3	9	754	-1	1015	1	AS	N612AS	522	SEA	BUR	127	937	7	54 2.1166666666666667	
2014	1	15	1037	7	1352	2	WN	N646SW	48	PDX	DEN	121	991	10	37 2.0166666666666666	
2014	7	2	847	42	1041	51	WN	N422WN	1520	PDX	OAK	90	543	8	47	1.5
2014	5	12	1655	-5	1842	-18	VX	N361VA	755	SEA	SFO	98	679	16	55 1.6333333333333333	
2014	4	19	1236	-4	1508	-7	AS	N309AS	490	SEA	SAN	135	1050	12	36	2.25
2014	11	19	1812	-3	2352	-4	AS	N564AS	26	SEA	ORD	198	1721	18	12	3.3
2014	11	8	1653	-2	1924	-1	AS	N323AS	448	SEA	LAX	130	954	16	53 2.1666666666666665	
2014	8	3	1120	0	1415	2	AS	N305AS	656	SEA	PHX	154	1107	11	20 2.5666666666666667	
2014	10	30	811	21	1038	29	AS	N433AS	608	SEA	LAS	127	867	8	11 2.1166666666666667	
2014	11	12	2346	-4	217	-28	AS	N765AS	121	SEA	ANC	183	1448	23	46	3.05
2014	10	31	1314	89	1544	111	AS	N713AS	306	SEA	SFO	129	679	13	14	2.15
2014	1	29	2009	3	2159	9	UA	N27205	1458	PDX	SFO	90	550	20	9	1.5
2014	12	17	2015	50	2150	41	AS	N626AS	368	SEA	SMF	76	605	20	15 1.2666666666666666	
2014	8	11	1017	-3	1613	-7	WN	N8634A	827	SEA	MDW	216	1733	10	17	3.6
2014	1	13	2156	-9	607	-15	AS	N597AS	24	SEA	BOS	290	2496	21	56 4.833333333333333	
2014	6	5	1733	-12	1945	-10	OO	N215AG	3488	PDX	BUR	111	817	17	33	1.85

only showing top 20 rows

◆ PySpark Example: Working with csv file

▼ Assign tables

```
[ ] 1 flights_df.createOrReplaceTempView('flights')

[ ] 1 spark.catalog.listDatabases()

[Database(name='default', catalog='spark_catalog', description='default database', locationUri='file:/content/spark-warehouse')]

[ ] 1 spark.catalog.listTables()

[Table(name='airports', catalog=None, namespace=[], description=None, tableType='TEMPORARY', isTemporary=True),
 Table(name='flights', catalog=None, namespace=[], description=None, tableType='TEMPORARY', isTemporary=True)]
```

Assign DataFrame to Database

Spark

◆ PySpark Example: Working with csv file

```
1 flights_table = spark.table('flights')
2 print(flights_table.show())
```

year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute	duration_hours
2014	12	8	658	-7	935	-5	VX	N846VA	1780	SEA	LAX	132	954	6	58	2.2
2014	1	22	1040	5	1505	5	AS	N559AS	851	SEA	HNL	360	2677	10	40	6.0
2014	3	9	1443	-2	1652	2	VX	N847VA	755	SEA	SFO	111	679	14	43	1.85
2014	4	9	1705	45	1839	34	WN	N360SW	344	PDX	SJC	83	569	17	5	1.3833333333333333
2014	3	9	754	-1	1015	1	AS	N612AS	522	SEA	BUR	127	937	7	54	2.1166666666666667
2014	1	15	1037	7	1352	2	WN	N646SW	48	PDX	DEN	121	991	10	37	2.0166666666666666
2014	7	2	847	42	1041	51	WN	N422WN	1520	PDX	OAK	90	543	8	47	1.5
2014	5	12	1655	-5	1842	-18	VX	N361VA	755	SEA	SFO	98	679	16	55	1.6333333333333333
2014	4	19	1236	-4	1508	-7	AS	N309AS	490	SEA	SAN	135	1050	12	36	2.25
2014	11	19	1812	-3	2352	-4	AS	N564AS	26	SEA	ORD	198	1721	18	12	3.3
2014	11	8	1653	-2	1924	-1	AS	N323AS	448	SEA	LAX	130	954	16	53	2.1666666666666665
2014	8	3	1120	0	1415	2	AS	N305AS	656	SEA	PHX	154	1107	11	20	2.5666666666666667
2014	10	30	811	21	1038	29	AS	N433AS	608	SEA	LAS	127	867	8	11	2.1166666666666667
2014	11	12	2346	-4	217	-28	AS	N765AS	121	SEA	ANC	183	1448	23	46	3.05
2014	10	31	1314	89	1544	111	AS	N713AS	306	SEA	SFO	129	679	13	14	2.15
2014	1	29	2009	3	2159	9	UA	N27205	1458	PDX	SFO	90	550	20	9	1.5
2014	12	17	2015	50	2150	41	AS	N626AS	368	SEA	SMF	76	605	20	15	1.2666666666666666
2014	8	11	1017	-3	1613	-7	WN	N8634A	827	SEA	MDW	216	1733	10	17	3.6
2014	1	13	2156	-9	607	-15	AS	N597AS	24	SEA	BOS	290	2496	21	56	4.833333333333333
2014	6	5	1733	-12	1945	-10	OO	N215AG	3488	PDX	BUR	111	817	17	33	1.85

only showing top 20 rows

Call the datatable

Spark

◆ PySpark Example: Working with csv file

```
1 flights_table.filter(flights_table.duration_hours > 5).show()
```

year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute	duration_hours
2014	1	22	1040	5	1505	5	AS	N559AS	851	SEA	HNL	360	2677	10	40	6.0
2014	12	4	954	-6	1348	-17	HA	N395HA	29	SEA	OGG	333	2640	9	54	5.55
2014	6	7	1823	-7	2112	-28	AS	N512AS	815	SEA	LIH	335	2701	18	23	5.583333333333333
2014	4	30	801	1	1757	90	AS	N407AS	18	SEA	MCO	342	2554	8	1	5.7
2014	4	1	1010	-5	1258	-17	HA	N381HA	25	PDX	HNL	328	2603	10	10	5.466666666666667
2014	7	27	925	25	1232	27	AS	N513AS	875	SEA	LIH	343	2701	9	25	5.716666666666667
2014	8	18	2116	-4	541	18	B6	N536JB	264	SEA	JFK	304	2422	21	16	5.06666666666666
2014	11	3	840	0	1703	-17	AS	N462AS	38	SEA	FLL	307	2717	8	40	5.11666666666666
2014	9	2	705	-5	950	-15	AS	N442AS	833	PDX	HNL	326	2603	7	5	5.433333333333334
2014	12	28	1705	-10	2046	-42	AS	N517AS	863	PDX	OGG	324	2562	17	5	5.4
2014	4	10	829	-1	1128	-7	AS	N589AS	861	SEA	OGG	332	2640	8	29	5.53333333333333
2014	9	19	1039	-1	1335	-6	AS	N431AS	843	SEA	KOA	332	2688	10	39	5.53333333333333
2014	9	28	657	-3	1557	34	DL	N704X	2588	SEA	JFK	311	2422	6	57	5.18333333333334
2014	10	13	853	-2	1217	16	AS	N517AS	879	SEA	LIH	358	2701	8	53	5.966666666666667
2014	5	2	1022	-3	1408	38	AS	N508AS	843	SEA	KOA	378	2688	10	22	6.3
2014	3	30	1646	6	1938	-17	AS	N537AS	867	SEA	OGG	332	2640	16	46	5.53333333333333
2014	5	29	705	-3	1547	15	DL	N3750D	400	PDX	JFK	311	2454	7	5	5.18333333333334
2014	3	22	1007	-8	1305	-20	HA	N389HA	25	PDX	HNL	338	2603	10	7	5.63333333333334
2014	8	16	1032	-8	1315	-20	HA	N395HA	25	PDX	HNL	304	2603	10	32	5.06666666666666
2014	4	4	1008	-7	1248	-27	HA	N385HA	25	PDX	HNL	320	2603	10	8	5.33333333333333

Perform filtering: Using function or SQL Query

Spark

◆ PySpark Example: Working with csv file

```
1 spark.sql('SELECT * FROM flights WHERE duration_hours > 5').show()
```

year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute	duration_hours
2014	1	22	1040	5	1505	5	AS	N559AS	851	SEA	HNL	360	2677	10	40	6.0
2014	12	4	954	-6	1348	-17	HA	N395HA	29	SEA	OGG	333	2640	9	54	5.55
2014	6	7	1823	-7	2112	-28	AS	N512AS	815	SEA	LIH	335	2701	18	23	5.58333333333333
2014	4	30	801	1	1757	90	AS	N407AS	18	SEA	MCO	342	2554	8	1	5.7
2014	4	1	1010	-5	1258	-17	HA	N381HA	25	PDX	HNL	328	2603	10	10	5.46666666666667
2014	7	27	925	25	1232	27	AS	N513AS	875	SEA	LIH	343	2701	9	25	5.71666666666667
2014	8	18	2116	-4	541	18	B6	N536JB	264	SEA	JFK	304	2422	21	16	5.06666666666666
2014	11	3	840	0	1703	-17	AS	N462AS	38	SEA	FLL	307	2717	8	40	5.11666666666666
2014	9	2	705	-5	950	-15	AS	N442AS	833	PDX	HNL	326	2603	7	5	5.43333333333334
2014	12	28	1705	-10	2046	-42	AS	N517AS	863	PDX	OGG	324	2562	17	5	5.4
2014	4	10	829	-1	1128	-7	AS	N589AS	861	SEA	OGG	332	2640	8	29	5.53333333333333
2014	9	19	1039	-1	1335	-6	AS	N431AS	843	SEA	KOA	332	2688	10	39	5.53333333333333
2014	9	28	657	-3	1557	34	DL	N704X	2588	SEA	JFK	311	2422	6	57	5.18333333333334
2014	10	13	853	-2	1217	16	AS	N517AS	879	SEA	LIH	358	2701	8	53	5.96666666666667
2014	5	2	1022	-3	1408	38	AS	N508AS	843	SEA	KOA	378	2688	10	22	6.3
2014	3	30	1646	6	1938	-17	AS	N537AS	867	SEA	OGG	332	2640	16	46	5.53333333333333
2014	5	29	705	-3	1547	15	DL	N3750D	400	PDX	JFK	311	2454	7	5	5.18333333333334
2014	3	22	1007	-8	1305	-20	HA	N389HA	25	PDX	HNL	338	2603	10	7	5.63333333333334
2014	8	16	1032	-8	1315	-20	HA	N395HA	25	PDX	HNL	304	2603	10	32	5.06666666666666
2014	4	4	1008	-7	1248	-27	HA	N385HA	25	PDX	HNL	320	2603	10	8	5.33333333333333

only showing top 20 rows

Perform filtering: Using function or SQL Query

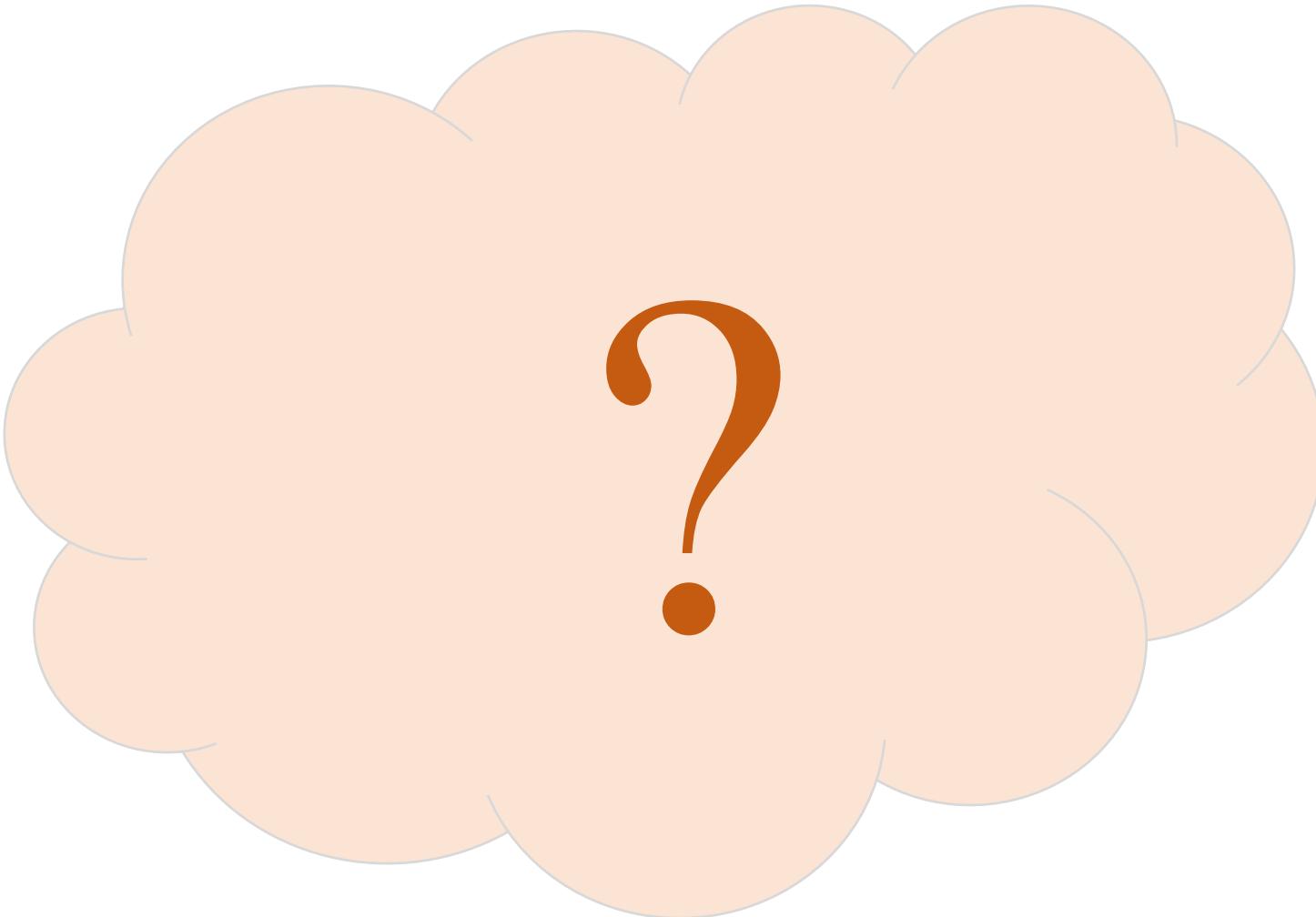
Summarization



In this session, we will discuss about:

- Introduction to big data.
- Introduction to Hadoop.
- How to install virtual machine.
- How to install and use Hadoop on virtual machine.
- Introduction to spark and pyspark.

Question



Thank you!