# Big Data Analytics and Reasoning - Practice 02

Giuseppe Mazzotta

# 1. Brief recap

## Main concepts to take in mind

- **HDFS**
  Distributed file system
  One namenode and many datanodes

- **YARN**
  Resource management tool
  One resourcemanager and many nodemangers

- **Map Reduce**
  Programming paradigm for running distributed applications

# 2. Cluster Setup

Main Technology

- **VirtualBox** ✅
  Software for Virtual Machine handling
- **SSH** ✅
  Secure SHell, cryptographic network protocol to deal with network services in a secure way
- **Java**
  Programming language on which hadoop relies on
- **MySQL**
  Relational database used by some hadoop service

# How to download hadoop ?

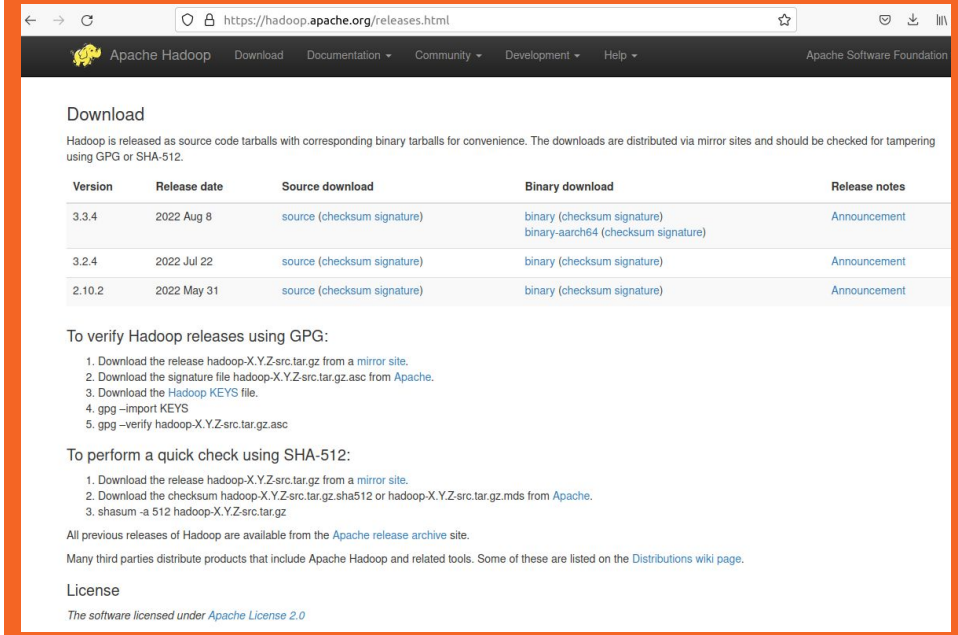
# Download Hadoop

Download from official website binary distribution

Use *wget* to download the zip from command line

Hadoop should be downloaded in each cluster machine

Used version 3.3.6



# What is inside the hadoop distribution?

- bin: contains scripts to submit request to hdfs and yarn
- sbin: contains scripts to handle hadoop daemons
- logs: store daemons' log files
- share: store libraries' jar
    - hadoop/mapreduce contains a jar with basic mapreduce examples

```
hadoop hadoop  4096 Sep 28 09:32 bin
hadoop hadoop  4096 Jul 29 12:35 etc
hadoop hadoop  4096 Jul 29 13:44 include
hadoop hadoop  4096 Jul 29 13:44 lib
hadoop hadoop  4096 Jul 29 13:44 libexec
hadoop hadoop 24707 Jul 28 20:30 LICENSE-binar
hadoop hadoop  4096 Jul 29 13:44 licenses-bina
hadoop hadoop 15217 Jul 16 18:20 LICENSE.txt
hadoop hadoop  4096 Oct 10 06:15 logs
hadoop hadoop 29473 Jul 16 18:20 NOTICE-binary
hadoop hadoop  1541 Apr 22 14:58 NOTICE.txt
hadoop hadoop   175 Apr 22 14:58 README.txt
hadoop hadoop  4096 Jul 29 12:35 sbin
hadoop hadoop  4096 Jul 29 14:21 share
```

# What is inside the hadoop distribution?

etc/hadoop: configuration files

- -env.sh files: site-specific customization of the Hadoop daemons' process environment
- core-site.xml: configuration settings for Hadoop Core
- hdfs-site.xml: configuration settings for HDFS daemons
- mapred-site.xml: configuration settings for MapReduce daemons
- yarn-site.xml: configuration settings for Yarn daemons
- workers: tells hadoop who are slave machines

```
hadoop hadoop  9213 Jul 29 13:19 capacity-scheduler.xml
hadoop hadoop  1335 Jul 29 13:22 configuration.xsl
hadoop hadoop  2567 Jul 29 13:19 container-executor.cfg
hadoop hadoop   857 Sep 27 10:46 core-site.xml
hadoop hadoop  3999 Jul 29 12:34 hadoop-env.cmd
hadoop hadoop 16689 Sep 27 10:59 hadoop-env.sh
hadoop hadoop  3321 Jul 29 12:34 hadoop-metrics2.properties
hadoop hadoop 11765 Jul 29 12:34 hadoop-policy.xml
hadoop hadoop  3414 Jul 29 12:34 hadoop-user-functions.sh.example
hadoop hadoop   683 Jul 29 12:49 hdfs-rbf-site.xml
hadoop hadoop  1116 Sep 27 10:28 hdfs-site.xml
hadoop hadoop  1484 Jul 29 12:47 httpfs-env.sh
hadoop hadoop  1657 Jul 29 12:47 httpfs-log4j.properties
hadoop hadoop   620 Jul 29 12:47 httpfs-site.xml
hadoop hadoop  3518 Jul 29 12:35 kms-acls.xml
hadoop hadoop  1351 Jul 29 12:35 kms-env.sh
hadoop hadoop  1860 Jul 29 12:35 kms-log4j.properties
hadoop hadoop   682 Jul 29 12:35 kms-site.xml
hadoop hadoop 13700 Jul 29 12:34 log4j.properties
hadoop hadoop   951 Jul 29 13:22 mapred-env.cmd
hadoop hadoop  1764 Jul 29 13:22 mapred-env.sh
hadoop hadoop  4113 Jul 29 13:22 mapred-queues.xml.template
hadoop hadoop   961 Oct 10 07:54 mapred-site.xml
hadoop hadoop  4096 Jul 29 12:34 shellprofile.d
hadoop hadoop  2316 Jul 29 12:34 ssl-client.xml.example
hadoop hadoop  2697 Jul 29 12:34 ssl-server.xml.example
hadoop hadoop  2681 Jul 29 12:41 user_ec_policies.xml.template
hadoop hadoop    14 Sep 27 10:46 workers
hadoop hadoop  2250 Jul 29 13:19 yarn-env.cmd
hadoop hadoop  6329 Jul 29 13:19 yarn-env.sh
hadoop hadoop  2591 Jul 29 13:19 yarnservice-log4j.properties
hadoop hadoop   877 Oct 10 07:55 yarn-site.xml
```

# How to configure an hadoop cluster ?

# Install Hadoop

Unfold the downloaded archive

Configure HADOOP_HOME variable

Add $HADOOP_HOME/bin and $HADOOP_HOME/sbin to PATH variable

https://hadoop.apache.org/releases.html

Apache Hadoop | Download | Documentation | Community | Development | Help | Apache Software Foundation

## Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

| Version | Release date | Source download | Binary download | Release notes |
|---|---|---|---|---|
| 3.3.4 | 2022 Aug 8 | source (checksum signature) | binary (checksum signature)<br>binary-aarch64 (checksum signature) | Announcement |
| 3.2.4 | 2022 Jul 22 | source (checksum signature) | binary (checksum signature) | Announcement |
| 2.10.2 | 2022 May 31 | source (checksum signature) | binary (checksum signature) | Announcement |

### To verify Hadoop releases using GPG:

1. Download the release hadoop-X.Y.Z-src.tar.gz from a mirror site.
2. Download the signature file hadoop-X.Y.Z-src.tar.gz.asc from Apache.
3. Download the Hadoop KEYS file.
4. gpg –import KEYS
5. gpg –verify hadoop-X.Y.Z-src.tar.gz.asc

### To perform a quick check using SHA-512:

1. Download the release hadoop-X.Y.Z-src.tar.gz from a mirror site.
2. Download the checksum hadoop-X.Y.Z-src.tar.gz.sha512 or hadoop-X.Y.Z-src.tar.gz.mds from Apache.
3. shasum -a 512 hadoop-X.Y.Z-src.tar.gz

All previous releases of Hadoop are available from the Apache release archive site.

Many third parties distribute products that include Apache Hadoop and related tools. Some of these are listed on the Distributions wiki page.

### License

*The software licensed under Apache License 2.0*

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-amd64"
export PATH=${PATH}:${JAVA_HOME}/bin
export HADOOP_HOME="/home/hadoop/hadoop"
export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin
```

**Remember**

Hadoop distribution has to be download in all nodes of our cluster

# Configuring Hadoop

Hadoop configuration files are stored in etc/hadoop

- core-site.xml
- hdfs-site.xml
- mapred-site.xml
- yarn-site.xml
- hadoop-env.sh
  - export JAVA_HOME
- workers

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
core-site.xml (END)
```

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>/home/hadoop/hdfs_nn-storage</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hadoop/hdfs_dn_storaga</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.blocksize</name>
<value>10m</value>
</property>
</configuration>
hdfs-site.xml (END)
```

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.application.classpath</name>
<value>${HADOOP_HOME}/share/hadoop/mapreduce/*</value>
</property>
</configuration>
mapred-site.xml (END)
```

```
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.resourcemanager.hostname</name>
<value>master</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

</configuration>
yarn-site.xml (END)
```

```
slave1
slave2
workers (END)
```

# Run hadoop cluster

For each configuration file many different properties could be set in order to customize the cluster based on hardware and software needs.

Initial step: format hdfs

*hdfs namenode -format <clustername>*

Start/Stop hdfs: *start-dfs.sh/stop-dfs.sh*

Start/Stop yarn: *start-yarn.sh/stop-yarn.sh*

Start/Stop both: *start-all.sh/stop-all.sh*

## Tip

More details about configuration property:

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/core-default.xml

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-common/yarn-default.xml

https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml

# Let’s start hadoop configuration on your clusters !!!

# 3. HDFS In Action

Basic hdfs command line commands

- **Create a directory into hdfs**
  *hdfs dfs -mkdir path/to/new/hdfs/folder*
- **Store file into hdfs**
  *hdfs dfs -put local/file/to/store hdfs/location*
- **Download file from hdfs**
  *hdfs dfs -get hdfs/file/location local/path*
- **Show folder content**
  *hdfs dfs -ls path/hdfs/folder*

**Remark**:
relative path starts from /user/username folder of hdfs

# 4.1 Hadoop FileSystem API

- **Path**
  *A file in the hadoop filesystem is represented by path object*
- **FileSystem**
  *General file system API to retrieve an instance of used filesystem (HDFS)*
- **FSDataInputStream**
  *Encapsulate an input stream for a file*
  *Supports random access (implements Seekable)*
- **FSDataOutputStream**
  *Encapsulate an output stream for a file*

# 4.1 Hadoop FileSystem API

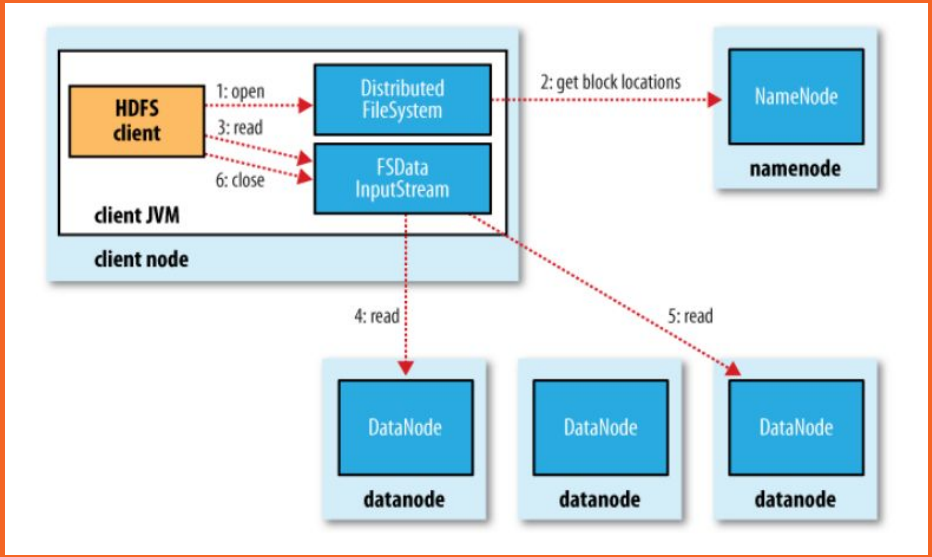
**Querying FileSystem**

- **FileStatus**
  *Encapsulate filesystem metadata for file and directories in the filesystem*
- **FileSystem::getFileStatus()**
  *Return a FileStatus object for a single file/dir*
- **FileSystem::listStatus()**
  *Return an array of FileStatus for files/dirs in directory*
- **PathFilter**
  *Allows you specifying some filtering condition on files*
  *It can be given to listStatus method*

# How input/output streams work under the hood?

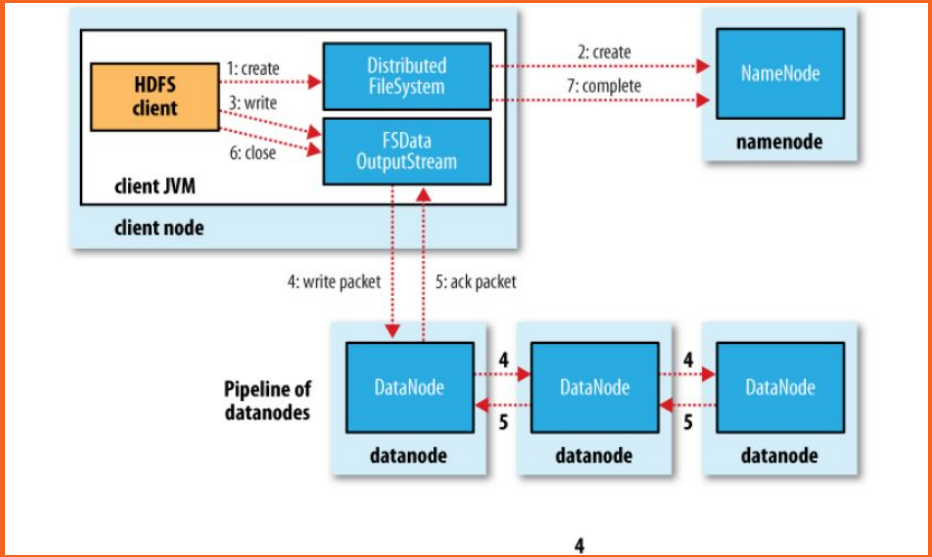
# Reading from HDFS

- Client call *open* method on FileSystem
- FileSystem query the namenode (NN) for blocks' location
- NN returns the list of datanodes (DN) storing blocks' copy
- DNs are sorted with proximity metrics
- An FSDataInputStream (IS) is created
- While client calls *read*, IS connects to the nearest DN holding current block and return data to the client
- When listed blocks are consumed IS ask NN for next blocks



# Writing to HDFS

- Client call *create* method on FileSystem
- FileSystem ask Namenode (NN) to create a file with no blocks associated
- NN checks that the file doesn't exists, user permission and more
- An FSDataOutputStream (OS) is created
- While client calls write data on OS, it splits data into packets stored into a DataQueue (DQ)
- The DQ is consumed by a DataStreamer (DS)
- DS asks NN to allocate new blocks
- NN return a pipeline of datanode where blocks' replica have to be stored
- DS writes packets on the pipeline and wait acks
- Client closes OS
- DS flush remaining packets and send a complete signal to NN
- NN wait until minimum replication requirements are satisfied and return success



# Let’s see some java examples