**Step 1 — Installing Java**

To get started, you’ll update our package list and install OpenJDK, the default Java Development Kit on Ubuntu 20.04:

1. sudo apt update
2. sudo apt install default-jdk

Copy

Once the installation is complete, let’s check the version.

1. java -version

Copy

Output

openjdk version "11.0.13" 2021-10-19

OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)

OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)

This output verifies that OpenJDK has been successfully installed.

**Step 2 — Installing Hadoop**

With Java in place, you’ll visit the [Apache Hadoop Releases page](http://hadoop.apache.org/releases.html) to find the most recent stable release.

Navigate to **binary** for the release you’d like to install. In this guide you’ll install Hadoop 3.3.1, but you can substitute the version numbers in this guide with one of your choice.

On the next page, right-click and copy the link to the release binary.

On the server, you’ll use wget to fetch it:

1. wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz

Copy

**Note:** The Apache website will direct you to the best mirror dynamically, so your URL may not match the URL above.

In order to make sure that the file you downloaded hasn’t been altered, you’ll do a quick check using SHA-512, or the Secure Hash Algorithm 512. Return to the [releases page](http://hadoop.apache.org/releases.html), then right-click and copy the link to the checksum file for the release binary you downloaded:

Again, you’ll use wget on our server to download the file:

1. wget https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz.sha512

Copy

Then run the verification:

1. shasum -a 512 hadoop-3.3.1.tar.gz

Copy

Output

2fd0bf74852c797dc864f373ec82ffaa1e98706b309b30d1effa91ac399b477e1accc1ee74d4ccbb1db7da1c5c541b72e4a834f131a99f2814b030fbd043df66 hadoop-3.3.1.tar.gz

Compare this value with the SHA-512 value in the .sha512 file:

1. cat hadoop-3.3.1.tar.gz.sha512

Copy

~/hadoop-3.3.1.tar.gz.sha512

...

SHA512 (hadoop-3.3.1.tar.gz) = 2fd0bf74852c797dc864f373ec82ffaa1e98706b309b30d1effa91ac399b477e1accc1ee74d4ccbb1db7da1c5c541b72e4a834f131a99f2814b030fbd043df66

...

The output of the command you ran against the file you downloaded from the mirror should match the value in the file you downloaded from [apache.org](http://apache.org/).

Now that you’ve verified that the file wasn’t corrupted or changed, you can extract it:

1. tar -xzvf hadoop-3.3.1.tar.gz

Copy

Use the tar command with the -x flag to extract, -z to uncompress, -v for verbose output, and -f to specify that you’re extracting from a file.

Finally, you’ll move the extracted files into /usr/local, the appropriate place for locally installed software:

1. sudo mv hadoop-3.3.1 /usr/local/hadoop

Copy

With the software in place, you’re ready to configure its environment.

**Step 3 — Configuring Hadoop’s Java Home**

Hadoop requires that you set the path to Java, either as an environment variable or in the Hadoop configuration file.

The path to Java, /usr/bin/java is a symlink to /etc/alternatives/java, which is in turn a symlink to default Java binary. You will use readlink with the -f flag to follow every symlink in every part of the path, recursively. Then, you’ll use sed to trim bin/java from the output to give us the correct value for JAVA\_HOME.

To find the default Java path

1. readlink -f /usr/bin/java | sed "s:bin/java::"

Copy

Output

/usr/lib/jvm/java-11-openjdk-amd64/

You can copy this output to set Hadoop’s Java home to this specific version, which ensures that if the default Java changes, this value will not. Alternatively, you can use the readlink command dynamically in the file so that Hadoop will automatically use whatever Java version is set as the system default.

To begin, open hadoop-env.sh:

1. sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Copy

Then, modify the file by choosing one of the following options:

**Option 1: Set a Static Value**

/usr/local/hadoop/etc/hadoop/hadoop-env.sh

. . .

#export JAVA\_HOME=

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64/

. . .

**Option 2: Use Readlink to Set the Value Dynamically**

/usr/local/hadoop/etc/hadoop/hadoop-env.sh

. . .

#export JAVA\_HOME=

export JAVA\_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")

. . .

If you have trouble finding these lines, use CTRL+W to quickly search through the text. Once you’re done, exit with CTRL+X and save your file.

**Note:** With respect to Hadoop, the value of JAVA\_HOME in hadoop-env.sh overrides any values that are set in the environment by /etc/profile or in a user’s profile.

**Step 4 — Running Hadoop**

Now you should be able to run Hadoop:

1. /usr/local/hadoop/bin/hadoop

Copy

Output

Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]

or hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]

where CLASSNAME is a user-provided Java class

OPTIONS is none or any of:

--config dir Hadoop config directory

--debug turn on shell script debug mode

--help usage information

buildpaths attempt to add class files from build tree

hostnames list[,of,host,names] hosts to use in slave mode

hosts filename list of hosts to use in slave mode

loglevel level set the log4j level for this command

workers turn on worker mode

SUBCOMMAND is one of:

. . .

This output means you’ve successfully configured Hadoop to run in stand-alone mode.

You’ll ensure that Hadoop is functioning properly by running the example MapReduce program it ships with. To do so, create a directory called input in our home directory and copy Hadoop’s configuration files into it to use those files as our data.

1. mkdir ~/input
2. cp /usr/local/hadoop/etc/hadoop/\*.xml ~/input

Copy

Next, you can use the following command to run the MapReduce hadoop-mapreduce-examples program, a Java archive with several options:

1. /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar grep ~/input ~/grep\_example 'allowed[.]\*'

Copy

This invokes the grep program, one of the many examples included in hadoop-mapreduce-examples, followed by the input directory, input and the output directory grep\_example. The MapReduce grep program will count the matches of a literal word or regular expression. Finally, the regular expression allowed[.]\* is given to find occurrences of the word allowed within or at the end of a declarative sentence. The expression is case-sensitive, so you wouldn’t find the word if it were capitalized at the beginning of a sentence.

When the task completes, it provides a summary of what has been processed and errors it has encountered, but this doesn’t contain the actual results.

Output

. . .

File System Counters

FILE: Number of bytes read=1200956

FILE: Number of bytes written=3656025

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

Map-Reduce Framework

Map input records=2

Map output records=2

Map output bytes=33

Map output materialized bytes=43

Input split bytes=114

Combine input records=0

Combine output records=0

Reduce input groups=2

Reduce shuffle bytes=43

Reduce input records=2

Reduce output records=2

Spilled Records=4

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=41

Total committed heap usage (bytes)=403800064

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=147

File Output Format Counters

Bytes Written=34

**Note:** If the output directory already exists, the program will fail, and rather than seeing the summary, the output will look something like:

Output

. . .

at java.base/java.lang.reflect.Method.invoke(Method.java:564)

at org.apache.hadoop.util.RunJar.run(RunJar.java:244)

at org.apache.hadoop.util.RunJar.main(RunJar.java:158)

Results are stored in the output directory and can be checked by running cat on the output directory:

1. cat ~/grep\_example/\*

Copy

Output

22 allowed.

1 allowed

The MapReduce task found 19 occurrences of the word allowed followed by a period and one occurrence where it was not. Running the example program has verified that our stand-alone installation is working properly and that non-privileged users on the system can run Hadoop for exploration or debugging.