How To Install Spark On Ubuntu

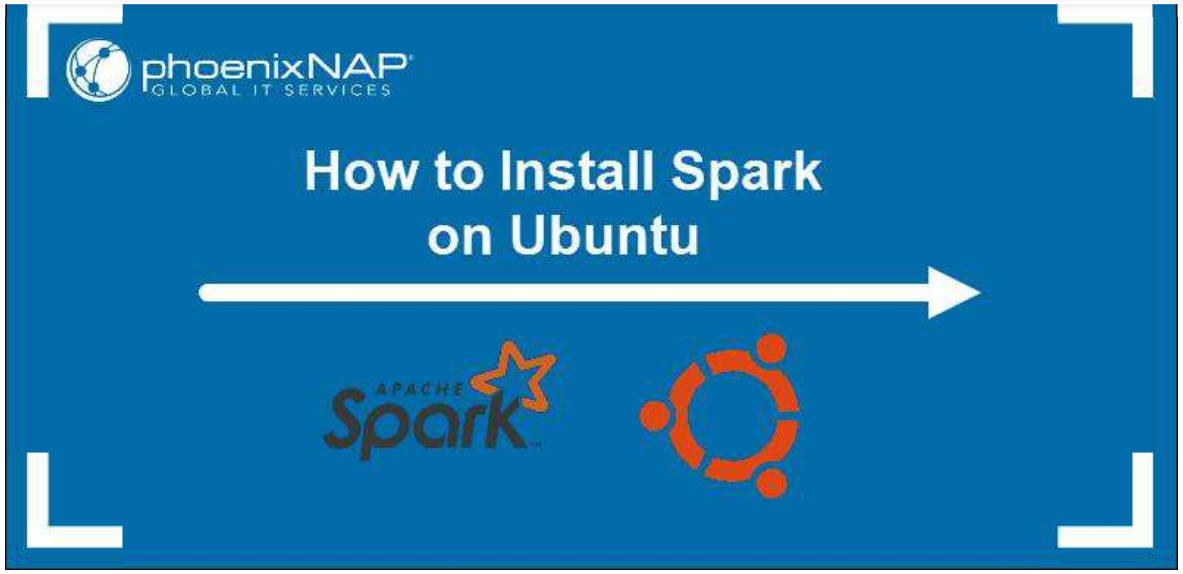
Posted April 13, 2020

Introduction

Apache Spark is a framework used in cluster computing environments for **analyzing big data**. This platform became widely popular due to its ease of use and the improved data processing speeds over Hadoop.

Apache Spark is able to distribute a workload across a group of computers in a cluster to more effectively process large sets of data. This **open-source engine** supports a wide array of programming languages. This includes Java, Scala, Python, and R.

In this tutorial, you will learn **how to install Spark on an Ubuntu machine**. The guide will show you how to start a master and slave server and how to load Scala and Python shells. It also provides the most important Spark



Contents

Prerequisites

- An Ubuntu system.
- Access to a terminal or command line.
- A user with sudo or **root** permissions.

Install Packages Required for Spark

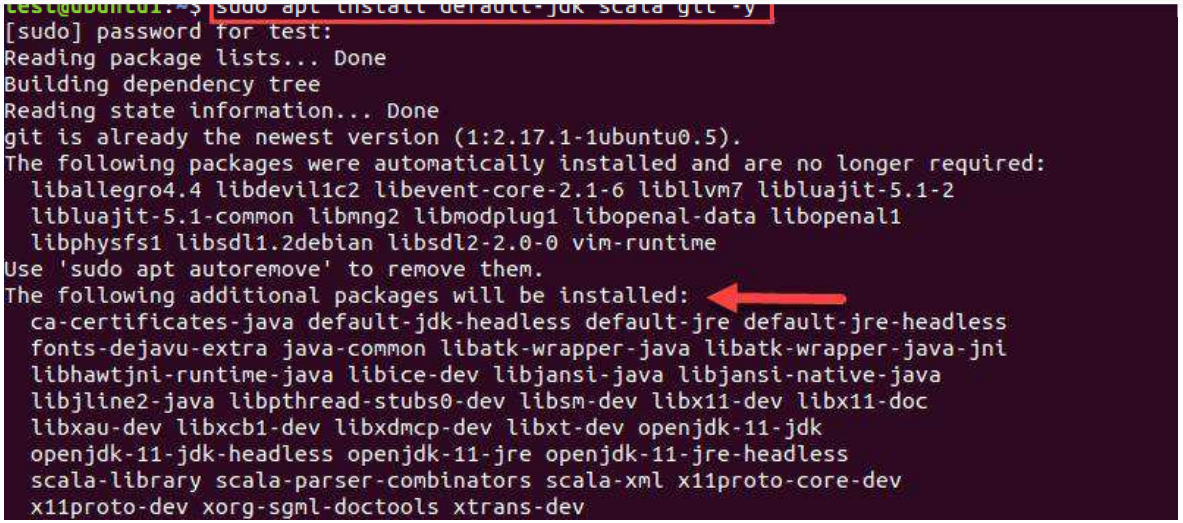
- JDK
- Scala
- Git

Open a terminal window and run the following command to install all three packages at once:

```
sudo apt install default-jdk scala git -y
```

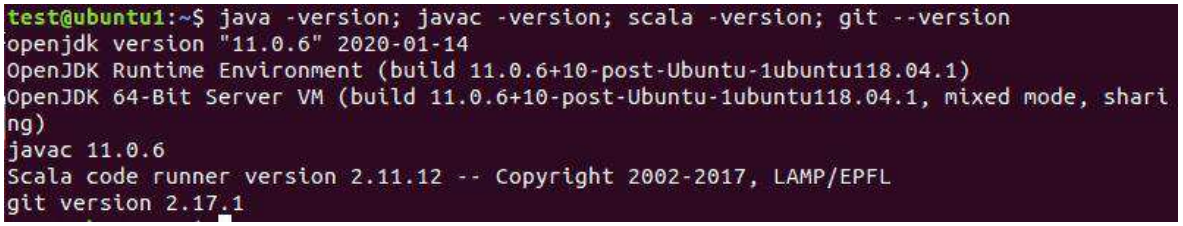
You will see which packages will be installed.

Contents



Once the process completes, **verify the installed dependencies** by running these commands:

```
java -version; javac -version; scala -version; git --version
```



Contents

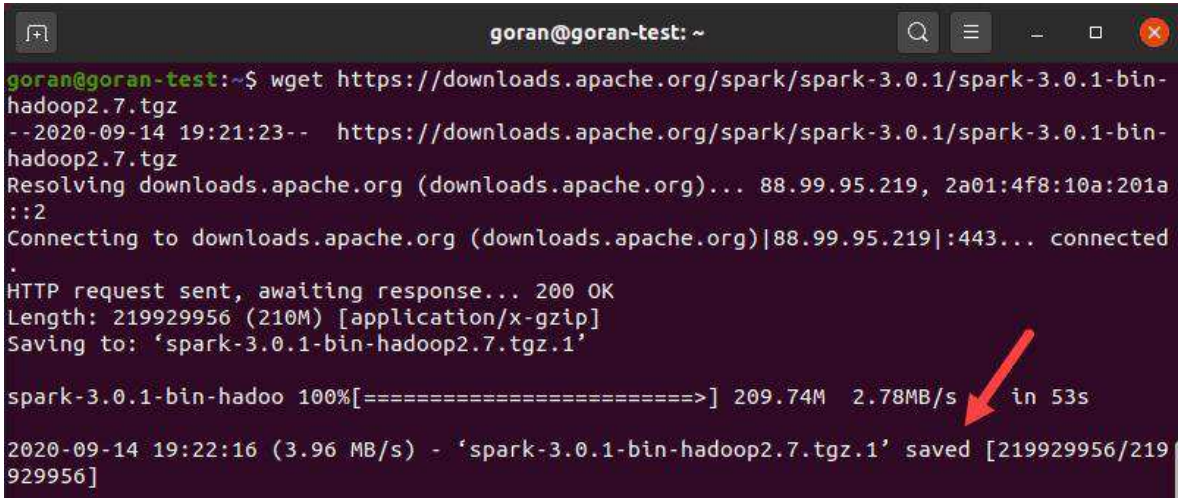
Download and Set Up Spark on Ubuntu

Now, **you need to download the version of Spark you want** form their website. We will go for *Spark 3.0.1* with *Hadoop 2.7* as it is the latest version at the time of writing this article.

Use the **`wget`** command and the direct link to download the Spark archive:

```
wget https://downloads.apache.org/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
```

When the download completes, you will see the *saved* message.



Contents

version. Remember to replace the Spark version number in the subsequent commands if you change the download URL.

Now, extract the saved archive using the **`tar`** command:

```
tar xvf spark-*
```

Let the process complete. The output shows the files that are being unpacked from the archive.

Finally, move the unpacked directory *spark-3.0.1-bin-hadoop2.7* to the ***opt/spark*** directory.

Use the **`mv`** command to do so:

```
sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark
```

The terminal returns no response if it successfully moves the directory. If you mistype the name, you will get a message similar to:

```
mv: cannot stat 'spark-3.0.1-bin-hadoop2.7': No such file or directory.
```

Contents

Before starting a master server, you need to configure environment variables. There are a few Spark home paths you need to add to the user profile.

Use the **echo** command to add these three lines to *.profile*:

```
echo "export SPARK_HOME=/opt/spark" >> ~/.profile
echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile
echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile
```

You can also add the export paths by editing the *.profile* file in the editor of your choice, such as nano or vim.

For example, to use nano, enter:

```
nano .profile
```

When the profile loads, scroll to the bottom of the file.

Contents

Then, add these three lines:

```
export SPARK_HOME=/opt/spark

export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Exit and save changes when prompted.

When you finish adding the paths, load the *.profile* file in the command line by typing:

```
source ~/.profile
```

Start Standalone Spark Master Server

Now that you have completed configuring your environment for Spark, you can start a master server.

In the terminal, type:

```
start-master.sh
```

To view the Spark Web user interface, open a web browser and enter the localhost IP address on port 8080.

```
http://127.0.0.1:8080/
```

The page shows your **Spark URL**, status information for workers, hardware resource utilization, etc.

Contents

Contents

The URL for Spark Master is the name of your device on port 8080. In our case, this is ***ubuntu1:8080***. So, there are three possible ways to load Spark Master's Web UI:

3. *deviceName*:8080

Note: Learn how to automate the deployment of Spark clusters on Ubuntu servers by reading our Automated Deployment Of Spark Cluster On Bare Metal Cloud article.

Start Spark Slave Server (Start a Worker Process)

In this single-server, standalone setup, we will start one slave server along with the master server.

To do so, run the following command in this format:

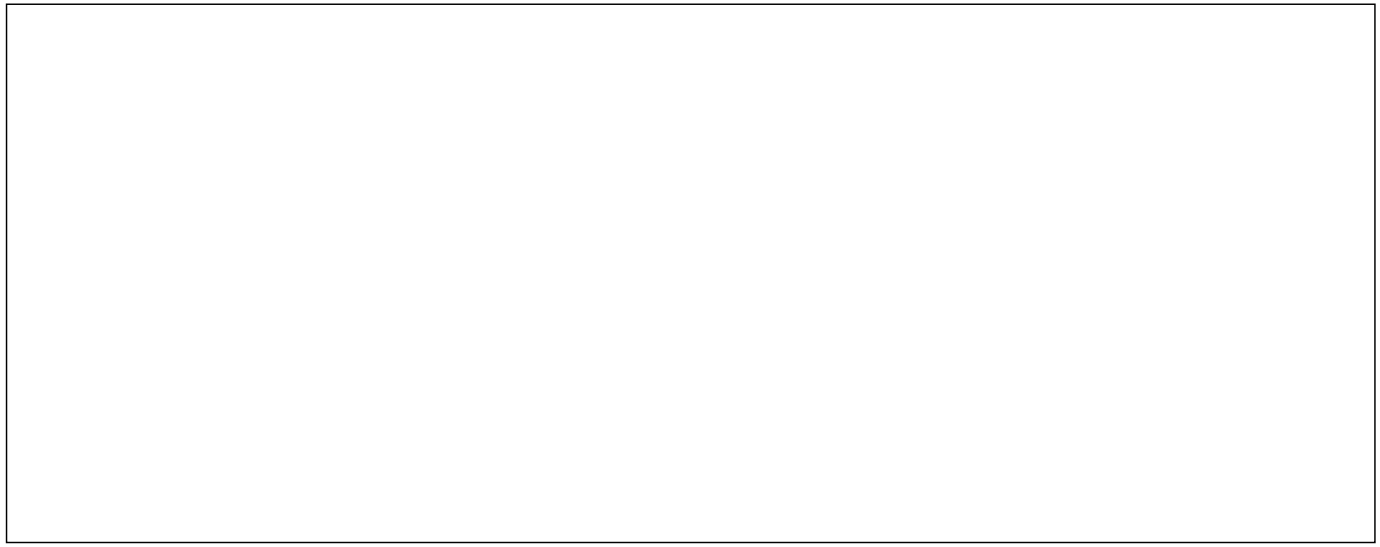
```
start-slave.sh spark://master:port
```

The **`master`** in the command can be an IP or hostname.

In our case it is **`ubuntu1`**:

```
start-slave.sh spark://ubuntu1:7077
```

Now that a worker is up and running, if you reload Spark Master's Web UI, you should see it on the list:



Specify Resource Allocation for Workers

The default setting when starting a worker on a machine is to use all available CPU cores. You can specify the number of cores by passing the `-c` flag to the `start-slave` command.

```
start-slave.sh -c 1 spark://ubuntu1:7077
```

Reload Spark Master's Web UI to confirm the worker's configuration.

Similarly, you can assign a specific amount of memory when starting a worker. The default setting is to use whatever amount of RAM your machine has, minus 1GB.

To start a worker and assign it a specific amount of memory, add the `-m` option and a number. For gigabytes, use `G` and for megabytes, use `M`.

For example, to start a worker with 512MB of memory, enter this command:

```
start-slave.sh -m 512M spark://ubuntu1:7077
```

Reload the Spark Master Web UI to view the worker's status and confirm the configuration.

Contents

Test Spark Shell

After you finish the configuration and start the master and slave server, test if the Spark shell works.

Load the shell by entering:

```
spark-shell
```

You should get a screen with notifications and Spark information. Scala is the default interface, so that shell loads when you run *spark-shell*.

The ending of the output looks like this for the version we are using at the time of writing this guide:

Contents

Type `:q` and press **Enter** to exit Scala.

Test Python in Spark

If you do not want to use the default Scala interface, you can switch to Python.

Make sure you quit Scala and then run this command:

```
pyspark
```

The resulting output looks similar to the previous one. Towards the bottom, you will see the version of Python.

Contents

To exit this shell, type `quit()` and hit **Enter**.

Basic Commands to Start and Stop Master Server and Workers

Below are the basic commands for starting and stopping the Apache Spark master server and workers. Since this setup is only for one machine, the scripts you run default to the localhost.

To start a master server instance on the current machine, run the command we used earlier in the guide:

```
start-master.sh
```

To stop the master instance started by executing the script above, run:

Contents

To stop a running worker process, enter this command:

```
stop-slave.sh
```

The Spark Master page, in this case, shows the worker status as DEAD.

You can **start both master and server** instances by using the start-all command:

```
start-all.sh
```

Similarly, you **can stop all instances** by using the following command:

```
stop-all.sh
```

Contents

This tutorial showed you **how to install Spark on an Ubuntu machine**, as well as the necessary dependencies.

The setup in this guide enables you to perform basic tests before you start configuring a Spark cluster and performing advanced actions.

Contents

Next you should also read

DevOps And Development, Databases

How to Install Elasticsearch on Ubuntu 18.04

April 23, 2020

Elasticsearch is an open-source engine that enhances searching,

Virtualization, Databases, MySQL

MySQL Docker Container Tutorial: How to Set Up & Configure

February 10, 2020

Deploying MySQL in a container is a fast and efficient solution for small

Web Servers, Databases

How to Deploy PostgreSQL on Docker Container

January 16, 2020

PostgreSQL is the third most popular Docker image used for deploying

SysAdmin, Databases,

How to Improve Performance Wi Tuning

January 15, 2020

The performance of MySQL databases is an essential factor in the

READ MORE READ MORE READ MORE READ MORE

Author

Goran Jevtic

Goran combines his passions for research, writing and technology as a technical writer at phoenixNAP. Workin multiple departments and on a variety of projects, he has developed extraordinary understanding of cloud and virtualization technology trends and best practices.

Contents

Contents