

How to Install Spark on Ubuntu

Prerequisites

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

Install Packages Required for Spark

Before downloading and setting up Spark, you need to install necessary dependencies. This step includes installing the following packages:

- Scala
- Git

Open a terminal window and run the following command to install all two packages at once (If you install Hadoop first, you don't need to install JDK this time:

```
$ sudo apt install scala git -y
```

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

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

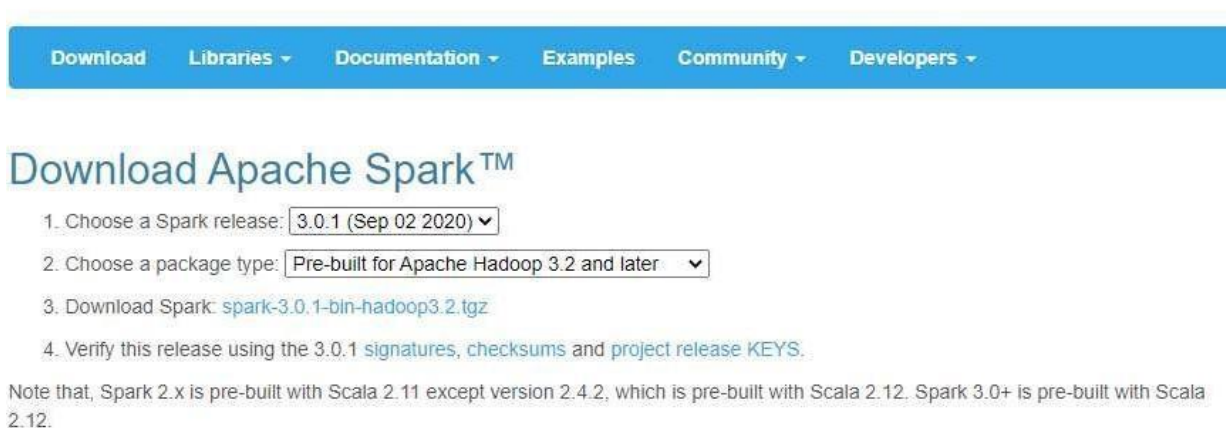
```
hadoop@yixi-VirtualBox:~$ java -version; javac -version; scala -version; git --version
openjdk version "1.8.0_272"
OpenJDK Runtime Environment (build 1.8.0_272-8u272-b10-0ubuntu1~20.10-b10)
OpenJDK 64-Bit Server VM (build 25.272-b10, mixed mode)
javac 1.8.0_272
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
git version 2.27.0
```

The output prints the versions if the installation completed successfully for all packages.

Download and Set Up Spark on Ubuntu

Now, **you need to download the version of Spark you want** from their website. We will go for *Spark 3.0.1* with *Hadoop 3.2* as it is the latest version at the time of writing this article.

Or, you can go to <https://spark.apache.org/downloads.html> to choose the version you want.



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

```
$ wget https://ftp.wayne.edu/apache/spark/spark-3.0.1/spark-3.0.1-bin-hadoop3.2.tgz
```

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

```
hadoop@yixi-VirtualBox: ~$ wget https://ftp.wayne.edu/apache/spark/spark-3.0.1/spark-3.0.1-bin-hadoop3.2.tgz
--2020-11-09 20:57:37-- https://ftp.wayne.edu/apache/spark/spark-3.0.1/spark-3.0.1-bin-hadoop3.2.tgz
Resolving ftp.wayne.edu (ftp.wayne.edu)... 141.217.0.199
Connecting to ftp.wayne.edu (ftp.wayne.edu)[141.217.0.199]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 224062525 (214M) [application/x-gzip]
Saving to: 'spark-3.0.1-bin-hadoop3.2.tgz'

spark-3.0.1-bin-had 100%[=====>] 213.68M  16.9MB/s   in 13s

2020-11-09 20:57:50 (16.8 MB/s) - 'spark-3.0.1-bin-hadoop3.2.tgz' saved [224062525/224062525]
```

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.
```

Configure Spark Environment

Before starting a master server, you need to configure environment variables.

You can 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.

```
# include .bashrc if it exists
if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```



^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos
^X Exit	^R Read File	^_\ Replace	^U Uncut Text	^T To Spell	^_ Go To Line

Then, add these three lines:

```
export SPARK_HOME=/opt/spark

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

export PYSARK_PYTHON=/usr/bin/python3
```

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.

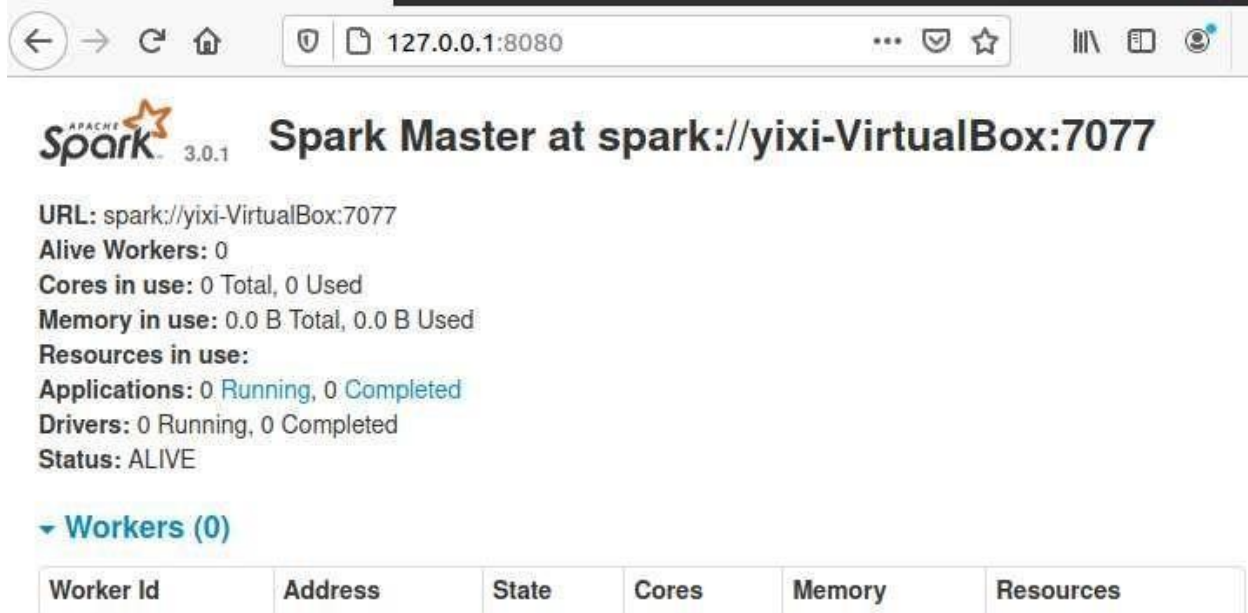
In the terminal, type:

```
$ 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.



The screenshot shows a web browser window with the address bar set to `127.0.0.1:8080`. The page title is "Spark Master at spark://yixi-VirtualBox:7077". The Apache Spark logo (3.0.1) is visible. The main content area displays the following status information:

- URL: spark://yixi-VirtualBox:7077
- Alive Workers: 0
- Cores in use: 0 Total, 0 Used
- Memory in use: 0.0 B Total, 0.0 B Used
- Resources in use:
- Applications: 0 Running, 0 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Below this information is a section titled "Workers (0)" with a dropdown arrow. Underneath is a table with the following columns: Worker Id, Address, State, Cores, Memory, and Resources. The table is currently empty.

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

1. 127.0.0.1:8080
2. localhost:8080
3. yixi-virtualbox:8080

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://yixi-virtualbox:7077
```

```
hadoop@yixi-VirtualBox:~$ start-slave.sh spark://yixi-virtualbox:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-
hadoop-org.apache.spark.deploy.worker.Worker-1-yixi-VirtualBox.out
```

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

Spark Master at spark://yixi-VirtualBox:7077

URL: spark://yixi-VirtualBox:7077

Alive Workers: 1

Cores in use: 1 Total, 0 Used

Memory in use: 2.8 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20201109221848-10.0.2.15-36101	10.0.2.15:36101	ALIVE	1 (0 Used)	2.8 GiB (0.0 B Used)	

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.

For example, to start a worker and assign only **one CPU core** to it, enter this command:

```
$ start-slave.sh -c 1 spark://yixi-virtualbox:7077
```

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

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20201109221848-10.0.2.15-36101	10.0.2.15:36101	ALIVE	1 (0 Used)	2.8 GiB (0.0 B Used)	

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://yixi-virtualbox:7077
```

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

worker-20201109222210-10.0.2.15-44119	10.0.2.15:44119	ALIVE	1 (0 Used)	512.0 MiB (0.0 B Used)	
---------------------------------------	-----------------	-------	------------	------------------------	--

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:

```
20/11/09 22:24:48 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
20/11/09 22:24:49 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
Spark context Web UI available at http://haifeng-vpc.csunet.csuohio.edu:4040
Spark context available as 'sc' (master = local[*], app id = local-1604978701909
).
Spark session available as 'spark'.
Welcome to

  ____      __
 / _  \    /  \
/_  /\  /_  /\  \
 \__\/  \__\/  \__\
   \__\         \__\

version 3.0.1

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_272)
Type in expressions to have them evaluated.
Type :help for more information.
```

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.

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
```



```
Using Python version 3.8.6 (default, Sep 25 2020 09:36:53)
SparkSession available as 'spark'.
```

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:

```
$ stop-master.sh
```

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.

Workers (2)				
Worker Id	Address	State	Cores	Memory
worker-20200331183244-10.0.2.15-45371	10.0.2.15:45371	DEAD	2 (0 Used)	1024.0 MB (0.0 B Used)
worker-20200331203427-10.0.2.15-37971	10.0.2.15:37971	ALIVE	2 (0 Used)	1024.0 MB (0.0 B Used)

Conclusion

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.

Reference

<https://phoenixnap.com/kb/install-spark-on-ubuntu>