Published in CodeX

Andrew Zhu Follow

May 11, 2021 · 3 min read ★ · Listen

Set up a local Spark cluster step by step in 10 minutes

Set up a local Spark cluster with one master node and one worker node in Ubuntu from scratch completely, and for free.

Tortle with 4 legs by Charles Zhu, my 6 yo son

This is an action list to install the open-sourced Spark master(or driver) and worker in local Ubuntu completely for free. (in contrast to Databricks for $$$)

The following setup runs in a home intranet. On one Linux(Ubuntu) physical machine(Jetson Nano) and one WSL2(Ubuntu) inside of Windows 10.

Step 1. Prepare environment

Make sure you have Java installed

```
sudo apt install openjdk-8-jdk
```

Check if you get Java installed

If you are going to use PySpark, go get Python installed

```
sudo apt install python3
```

Check if you get Python installed

```
python3 --version
```

Step 2. Download and install Spark in the Driver machine

From the Spark download page, select your version, I select the newest. (in any directory)

```
curl -O https://apache.claz.org/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
```

Unpack it.

```
tar xvf spark-3.1.1-bin-hadoop3.2.tgz
```

Now, you should see a new folder `spark-3.1.1-bin-hadoop3.2` . Move this folder to `/opt/spark` .

```
sudo mv spark-3.1.1-bin-hadoop3.2/ /opt/spark
```

To run Spark related bash script from anywhere, add related PATH to `~/.bashrc`




Press `i` to shift to write mode, and append the following to the end of the file.

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

Press `esc` and `:`, then type `wq` to save and quit vim.

Run the following command to activate the changes.

```
source ~/.bashrc
```

Step 3. Configure the master node, give IP address instead of the computer name

Navigate to the folder that holds configuration files.

```
cd /opt/spark/conf
```

Add a new file named `spark-env.sh`

```
sudo vim spark-env.sh
```

Add one variable

```
SPARK_MASTER_HOST=192.168.0.1
```



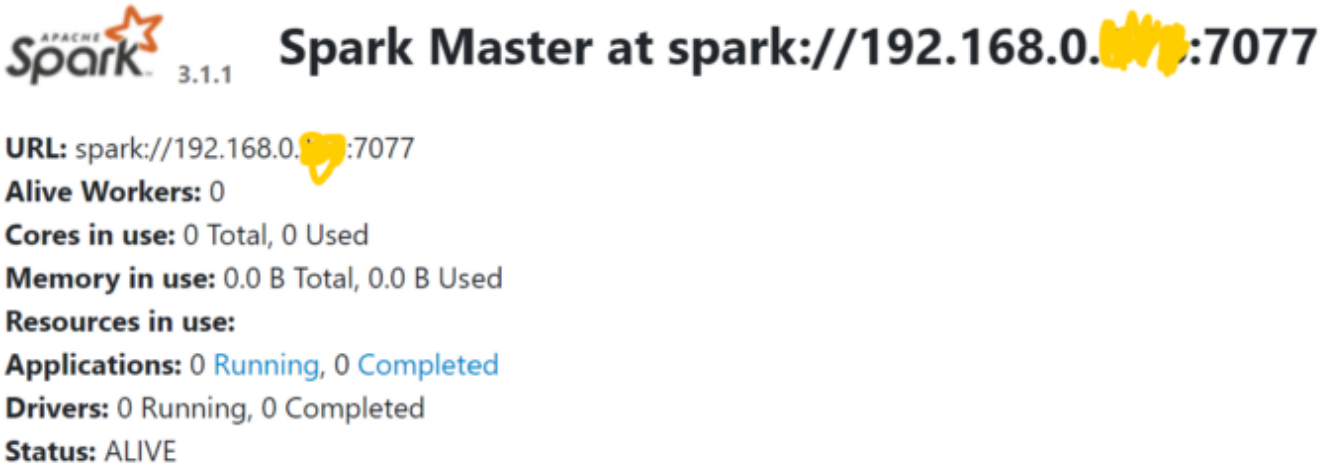

```
start-master.sh
```

To stop it, run the following script. This will be useful for troubleshooting if you don't get the worker node connected, you will need to stop and start the master node.

```
stop-master.sh
```

The `start-master.sh` command will also start a spark node web page to show basic information include both and later connected worker nodes. do remember use 8080 port by default.

```
http://<master_ip_address>:8080
```



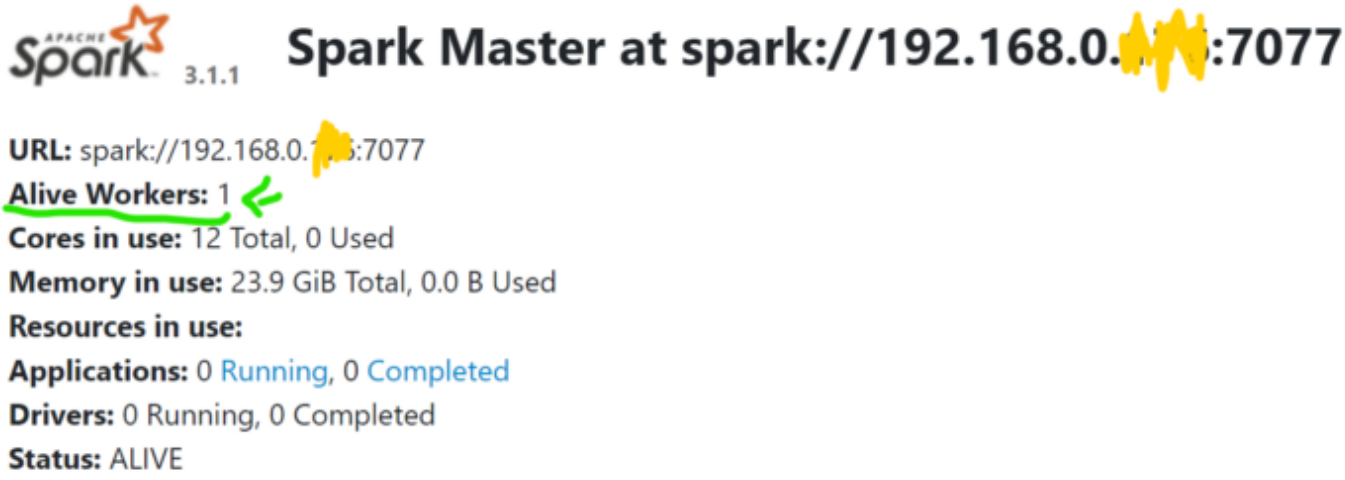
Step 4. Setup Spark worker node in another Linux(Ubuntu) machine

Go open another Linux(Ubuntu) machine and repeat **step 2**. No need to take **Step 3** in the worker node.

```
start-worker.sh spark://192.168.0.123:7077
```

Replace IP address `192.168.0.123` with yours.

Now, refresh the master UI web, you should see Alive Workers:1



If you see this, you have a good Spark cluster full under your control.

Connect Spark from remote PySpark

You can also verify the Spark cluster by starting a connection from PySpark.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.\
        master("spark://192.168.0.123:7077").getOrCreate()
print("spark session created")
```

Links that are helpful

- How to SSH into WSL2 on Windows 10 from an external machine So that I don't "really" need to shift machines, I sit in one chair and get everything done.

- spark worker not connecting to master
- Install Apache Spark on Ubuntu 20.04/18.04 & Debian 10/9