



## Module 1 Lab - Creating Single Node Spark / PySpark Cluster

Poonam Dighe

MPS in Analytics, Northeastern University

ALY 6110: Data Management and Big Data

Dr. Mohammad Shafiqul Islam

November 10, 2022

## Table of Contents

1.0 Introduction:.....	3
2.0 Cluster:.....	3
2.1 WSL 2: .....	3
3.0 Creating Spark Cluster:.....	4
3.1 Install Java:.....	4
3.2 Install Python3: .....	5
3.3 Install Scala:.....	7
3.4 Install Spark with Hadoop.....	8
3.5 Activate Spark .....	9
3.6 Test pySpark.....	9
3.7 Stop You Cluster.....	10

## 1.0.Introduction

The primary goal of Lab 0 is to create an environment in which one can practice big data knowledge in a real-world setting. This includes installation of Java, python3 , Scala, Spark with Hadoop. Also, activating, testing spark and stopping the cluster. Thus, it will create the big data environment ready for the hands on practice.

## 2.0. Install WSL2

create a single node cluster in an Ubuntu Virtual Machine (VM) to run PySpark. One can configure an Ubuntu VM as per preference and Operating System.

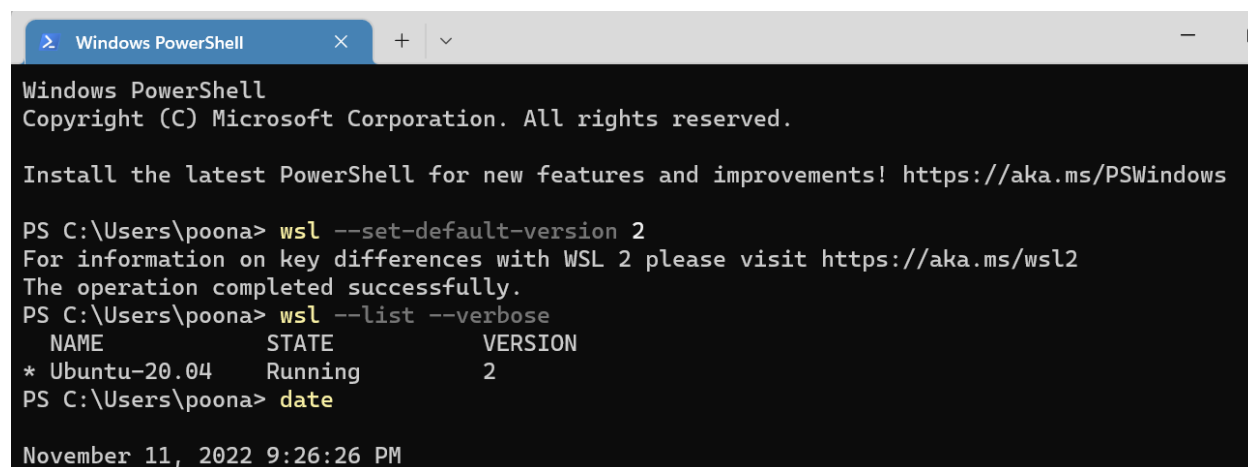
To install Windows subsystem for Linux check Windows Specifications as mentioned below -

1.Edition - Windows 11 Home

2.version 22H2

3.OS build 22621.819

Go to **Turn windows features on or off** and check in **Windows subsystem for Linux** and **Virtual machine platform**. Then click Ok and windows will install the required files. Then restart the system. Once the system is restarted open the link to install WSL. Once WSL is installed then install Ubuntu 20.04 Linux distribution.



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\poona> wsl --set-default-version 2
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
The operation completed successfully.
PS C:\Users\poona> wsl --list --verbose
  NAME                STATE      VERSION
* Ubuntu-20.04        Running    2
PS C:\Users\poona> date
November 11, 2022 9:26:26 PM

```

## 3.0. Creating Spark Cluster

Once Ubuntu 20.04 installation and configuration is done, we need to perform following steps to form Spark Cluster. Once WSL2 is installed on Windows 11 we can create single node spark/PySpark Cluster.

```
[04:44:37] poonamdighe say "Welcome $USER! It's now $(date '+%A %B %d %Y %r')"
```

```
-----
/ Welcome poonamdighe! It's now Friday \
\ November 11 2022 04:44:54 AM          /
-----

  ^  ^
  (oo)\_____
  (__) \       )\/\
        ||----w |
        ||     ||

poonamdighe@POOH:~$
```

### 3.1. Install Java

The following commands will be executed to install JAVA jdk which will run Spark Cluster. Openjdk version 11.0.17 is installed successfully.

```
poonamdighe@POOH:~$ java -version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu220.04, mixed mode, sharing)
```

```
poonamdighe@POOH: ~
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jaotc to provide /usr/bin/jaotc (jaotc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
Setting up libpthread-stubs0-dev:amd64 (0.4-1) ...
Setting up xtrans-dev (1.4.0-1) ...
Setting up openjdk-11-jdk:amd64 (11.0.17+8-1ubuntu2~20.04) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Setting up xorg-sgml-doctools (1:1.11-1) ...
Setting up x11proto-dev (2019.2-1ubuntu1) ...
Setting up libxau-dev:amd64 (1:1.0.9-0ubuntu1) ...
Setting up libice-dev:amd64 (2:1.0.10-0ubuntu1) ...
Setting up libsm-dev:amd64 (2:1.2.3-1) ...
Setting up libxdmcp-dev:amd64 (1:1.1.3-0ubuntu1) ...
Setting up x11proto-core-dev (2019.2-1ubuntu1) ...
Setting up libxcb1-dev:amd64 (1.14-2) ...
Setting up libx11-dev:amd64 (2:1.6.9-2ubuntu1.2) ...
Setting up libxt-dev:amd64 (1:1.1.5-1) ...
Processing triggers for man-db (2.9.1-1) ...
poonamdighe@POOH:~$ java -version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu220.04, mixed mode, sharing)
poonamdighe@POOH:~$ sudo apt update && upgrade
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
poonamdighe@POOH:~$
Hit:3 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
58 packages can be upgraded. Run 'apt list --upgradable' to see them.
upgrade: command not found
```

When I first ran `sudo apt update && Upgrade`, the upgrade command was not found, as indicated by the yellow highlight. There was a syntax error. If we want to upgrade as well as update, the command should be `sudo apt update && upgrade`. I was using `$sudo apt update && upgrade`,

but the correct command is `sudo apt update && sudo apt upgrade`. The preceding command is made up of two commands separated by '&&'. 'apt update' only refreshes the list of available packages and their versions, whereas 'apt upgrade' installs newer versions of the packages from the refreshed list. As a result, we can run both commands separately to achieve the same result. In this case, we used && to combine multiple commands into a single command; in this case, we combined the update and upgrade commands. -y option at the end of the command skips the upgrade confirmation prompt and proceeds directly to the system upgrade.

The previously mentioned command contains two commands: update and upgrade. The update will check the apt repositories to see if any updates are required. If you've used Windows, it's like "Check for Update," which only checks to see if an update is needed. The upgrade command, on the other hand, will upgrade both existing packages and the system.

When you go to settings and click update in Windows, it only updates the Windows, but in Linux, it's a little different. Linux verifies everything. It will determine whether a system update is available, as well as whether or not any software installed in the system requires an update. and everything will be updated.

### 3.2. Install Python3

```
[05:08:21] poonamdighe@POOH:~$ python3 - V
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

### 3.3. Install Scala

```
poonamdighe@POOH:~$ scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
```

After installing Java, Scala and python3 versions are checked which is 11.0.17, 2.11.12 and 3.8.10 respectively. After installation setting alias for jupyter notebook in bashrc.sh file. Root name aitms is changed to username of student.

```

GNU nano 4.8 /home/poonamdighe/.bashrc
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# 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
alias jupyter-notebook="~/local/bin/jupyter-notebook --no-browser"

#SCALA
export SCALA_HOME="/home/poonamdighe/scala-2.13.3"
export PATH=$PATH:$SCALA_HOME/bin

#SPARK WITH HADOOP
export SPARK_HOME="/home/poonamdighe/spark-3.1.1-bin-hadoop3.2"
export PATH=$PATH:$SPARK_HOME/bin

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo        M-A Mark Text
^X Exit          ^R Read File    ^_ Replace      ^U Paste Text   ^T To Spell     ^G Go To Line   M-E Redo        M-G Copy Text

13°C Cloudy
Search
ENG US 9:21 PM 2022-11-11

```

### 3.4. Install Spark with Hadoop

To activate the spark change directory command is used cd to Spark home. Scala shell was present in spark 3.1.1-bin-hadoop3.2. Thus, spark is activated successfully.

```

poonamdighe@POOH:~$ nano ~/.bashrc
poonamdighe@POOH:~$ nano ~/.bashrc
poonamdighe@POOH:~$ source ~/.bashrc
poonamdighe@POOH:~$ cd $SPARK_HOME
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ java - version
Unrecognized option: -
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ java -version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu220.04, mixed mode, sharing)
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ python3 -V
Python 3.8.10

```



```

poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ pip3 install traitlets==5.1.1
Requirement already satisfied: traitlets==5.1.1 in /home/poonamdighe/.local/lib/python3.8/site-packages (5.1.1)
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ pip3 install pygments==2.4.1
Requirement already satisfied: pygments==2.4.1 in /home/poonamdighe/.local/lib/python3.8/site-packages (2.4.1)
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ jupyter-notebook
Traceback (most recent call last):
  File "/home/poonamdighe/.local/bin/jupyter-notebook", line 5, in <module>
    from notebook.notebookapp import main
  File "/home/poonamdighe/.local/lib/python3.8/site-packages/notebook/__init__.py", line 25, in <module>
    from .nbextensions import install_nbextension
  File "/home/poonamdighe/.local/lib/python3.8/site-packages/notebook/nbextensions.py", line 20, in <module>
    from ipython_genutils.py3compat import string_types, cast_unicode_py2
ModuleNotFoundError: No module named 'ipython_genutils'
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ python3 -V
Python 3.8.10
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [11.2 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2197 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [16.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [976 kB]
Fetched 3536 kB in 1s (3345 kB/s)
Reading package lists... Done

```

After installing python3-ipython-genutils using sudo command which is super user DO python setup was completed.

```

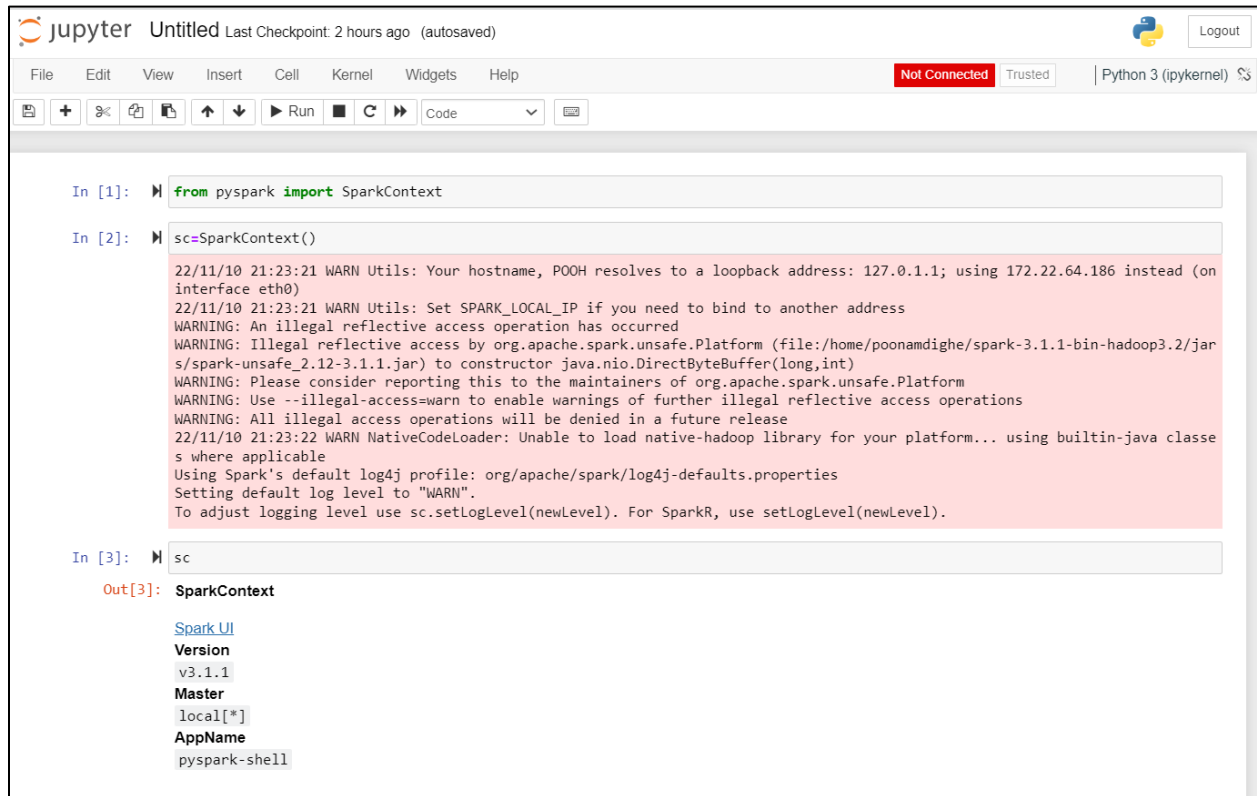
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ sudo apt-get install python3-ipython-genutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  python3-ipython-genutils
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 21.1 kB of archives.
After this operation, 86.0 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-ipython-genutils all 0.2.0-1ubuntu1 [21.1 kB]
Fetched 21.1 kB in 0s (145 kB/s)
Selecting previously unselected package python3-ipython-genutils.
(Reading database ... 40941 files and directories currently installed.)
Preparing to unpack .../python3-ipython-genutils_0.2.0-1ubuntu1_all.deb ...
Unpacking python3-ipython-genutils (0.2.0-1ubuntu1) ...
Setting up python3-ipython-genutils (0.2.0-1ubuntu1) ...
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2$ jupyter-notebook
[I 19:49:22.021 NotebookApp] Writing notebook server cookie secret to /home/poonamdighe/.local/share/jupyter/runtime/notebook_cookie_secret
[I 19:49:22.237 NotebookApp] Serving notebooks from local directory: /home/poonamdighe/spark-3.1.1-bin-hadoop3.2
[I 19:49:22.237 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 19:49:22.237 NotebookApp] http://localhost:8888/?token=7a299913de03d65b1a01514565978c1026e31f572526ad6b
[I 19:49:22.237 NotebookApp] or http://127.0.0.1:8888/?token=7a299913de03d65b1a01514565978c1026e31f572526ad6b
[I 19:49:22.237 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:49:22.243 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/poonamdighe/.local/share/jupyter/runtime/nbserver-1310-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=7a299913de03d65b1a01514565978c1026e31f572526ad6b
    or http://127.0.0.1:8888/?token=7a299913de03d65b1a01514565978c1026e31f572526ad6b

```



### 3.6. Testing pyspark using jupyter notebook



```

In [1]: from pyspark import SparkContext

In [2]: sc=SparkContext()

22/11/10 21:23:21 WARN Utils: Your hostname, POOH resolves to a loopback address: 127.0.1.1; using 172.22.64.186 instead (on interface eth0)
22/11/10 21:23:21 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/poonamdighe/spark-3.1.1-bin-hadoop3.2/jars/spark-unsafe_2.12-3.1.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
22/11/10 21:23:22 WARN NativeCodeLoader: Unable to load native-hadoop library for 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 setLogLevel(newLevel).

In [3]: sc

Out[3]: SparkContext

Spark UI
Version
v3.1.1
Master
local[*]
AppName
pyspark-shell

```

### 3.7. Stop Cluster

To stop the cluster the actual path starts from spark-3.1.1-bin-hadoop3.2 then sbin path. In sbin there are list of files present like start-worker.sh, stop worker.sh, etc. The basic commands for starting and stopping the Apache Spark master server and workers are given in the snapshot. Because this configuration is only for one machine, the scripts running will be default to the localhost. Run the command start-master.sh to launch a master server instance on the current machine. The stop worker command is used to terminate a running worker process. The start-all command can be used to start both master and server instances. Similarly, the stop-all.sh command can be used to terminate all instances.

```

poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2/sbin$ sudo ./stop-worker.sh
no org.apache.spark.deploy.worker.Worker to stop
poonamdighe@POOH:~/spark-3.1.1-bin-hadoop3.2/sbin$ ls
decommission-slave.sh  start-history-server.sh  start-worker.sh  stop-slave.sh
decommission-worker.sh start-master.sh          start-workers.sh stop-slaves.sh
slaves.sh              start-mesos-dispatcher.sh stop-all.sh      stop-thriftserver.sh
spark-config.sh        start-mesos-shuffle-service.sh stop-history-server.sh stop-worker.sh
spark-daemon.sh        start-slave.sh           stop-master.sh   stop-workers.sh
spark-daemons.sh      start-slaves.sh          stop-mesos-dispatcher.sh workers.sh
start-all.sh          start-thriftserver.sh    stop-mesos-shuffle-service.sh

```

```
poonamdighe@P00H:~/spark-3.1.1-bin-hadoop3.2/sbin$ sudo sh stop-master.sh
no org.apache.spark.deploy.master.Master to stop
poonamdighe@P00H:~/spark-3.1.1-bin-hadoop3.2/sbin$ sudo ./stop-worker.sh
no org.apache.spark.deploy.worker.Worker to stop
poonamdighe@P00H:~/spark-3.1.1-bin-hadoop3.2/sbin$
```

## Conclusion

As a result, we have successfully installed all of the prerequisites for a big data environment, such as java, Scala, Spark, Ubuntu, and Pyspark and installed Spark on an Ubuntu machine, as well as the dependencies required. This assignment allows you to run basic tests before configuring a Spark cluster and performing advanced actions.

## Reference

*Module 1 Lab 0—Grab Azure VM/Single Node Spark Cluster*. (n.d.). Retrieved May 30, 2022, from <https://northeastern.instructure.com/courses/126673/assignments/1487827>

*Islam, S (2022), Single Node Spark/PySpark Cluster on Windows Subsystem for Linux (WSL2), MLearning.ai in Medium*. Retrieved May 30, 2022, from [Single Node Spark/PySpark Cluster on Windows Subsystem for Linux \(WSL2\) | by Shafiqul Islam | MLearning.ai | Medium](#)