**BCG CaseStudy Project RunBook**

**DESCRIPTION:**

The purpose of this document is to setup whole code and environment easily and verify the result. As part of this process, we will setup single node Spark cluster on a VM. Then we will link the spark so that we can use spark in Jupyter notebook. Then we will write the application code and run the spark-submit to get the required results. We will get the final results in /Output folder(or the output path we specify in the spark-submit). Finally, we will push the code to a Git Repository.

**PRE REQUISITE**:

* VM setup
* Anaconda
* GIT

**SETUP:**

**Source Data and CaseStudy:**

Download the data from below path.

[https://bcg01.egnyte.com/dl/](https://bcg01.egnyte.com/dl/RG40r0uyNQ)

**Spark Cluster:** For Fresh setup.

**Prerequists**:

An Ubuntu system.

Access to a terminal or command line.

A user with sudo or **root** permissions.

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

JDK

Scala

Git

**Downloading Spark:**

Now, **you need to download the version of Spark you want** form their website. We will go for Spark 3.1.2 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-3.1.2/spark-3.1.2-bin-hadoop2.7.tgz

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.

**Configuring Spark:**

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

Then, add these three lines:

export SPARK\_HOME=/opt/spark

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

export PYSPARK\_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

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

In the terminal, type:

start-master.sh

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.

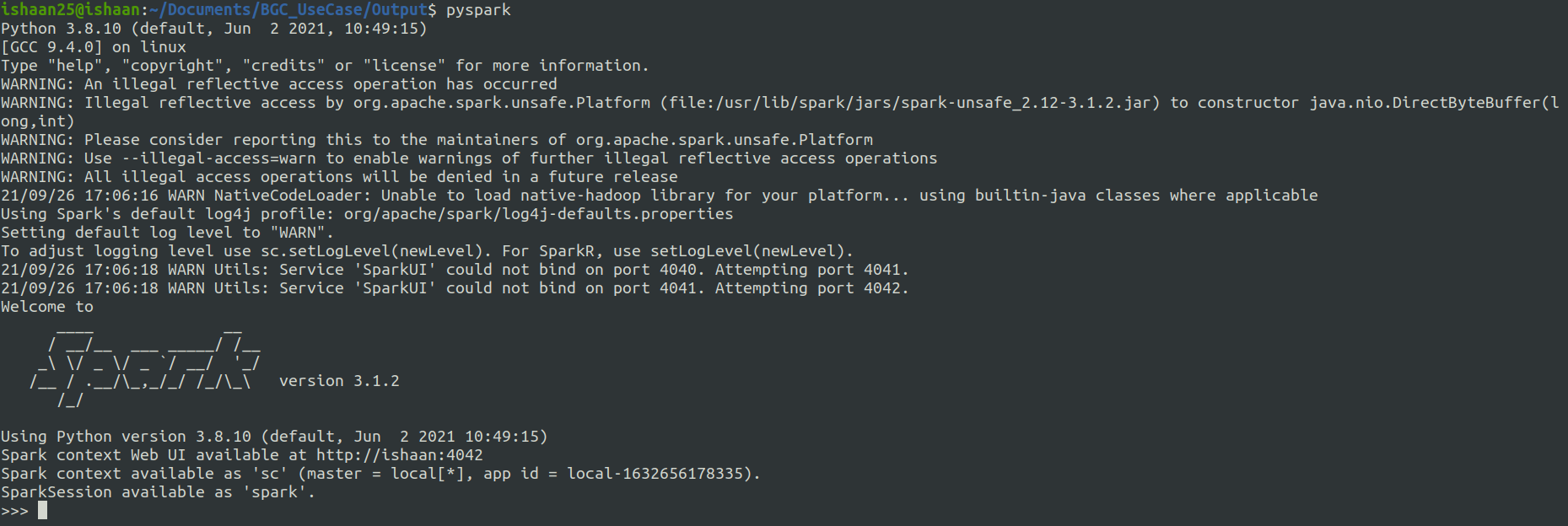
**Tesing Spark Installation:**

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

Load the shell by entering:

pyspark

You should get a screen with notifications and Spark information.

**Reference:**

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

**UseCase Unalysis:**

After setup of all prerequisites, you can follow the below steps.

**Step by Step:**

1. Pull the data from Git Repository given below:

<https://github.com/IshaanBhatnagar25/BCG_CrashAnalysis_CaseStudy>

1. From CLI:
   1. Open the CLI and go to the path where the CaseStudy is present.
   2. Run the below command:

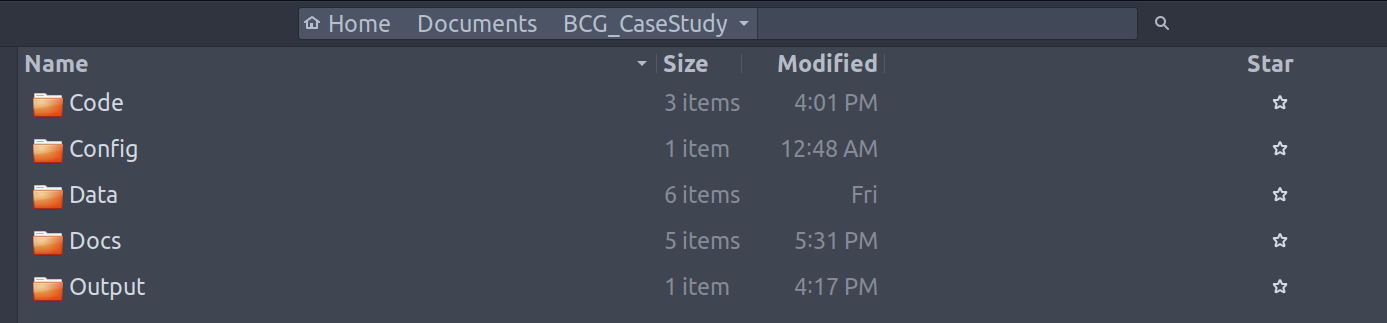
**spark-submit --master local[8] --deploy-mode client --properties-file <<UseCaseConfigPath>>input\_path.prop <<UseCaseCodePath>>/US\_Crashes\_Analysis.py > <<UseCaseOutputPath>>/analysis\_report.txt**

1. From Jupiter Notebook:
   1. Open the .ipynb file in JupiterNootbook
   2. Run each cell to see the output.

**TROUBLESHOOTING:**

* If you get issue to connect to Spark after proper installation, make sure to update the environment variables properly and start the deamons.

**Project Folder Structure:**

****

**/Code:**

Contains the .py file and Jupyter Notebooks

**/Config:**

Contains a config file where we can specify the path of all 6 input csv files.

Thus, the project can be configured on different machines easily without making changes to the code.

**/Data:**

Contains the Input CSV files

**/Docs:**

It contains the requirements, data dictionaries, artifacts, system information and project run book.

**/Output:**

Contains the output of .py file. The final output can be seen at this folder.