National College of Ireland

# Configuration Manual

MSc Research Project
Cloud Computing


Priyesh Shah
Student ID: x18207731

School of Computing
National College of Ireland


Supervisor: Sean Heeney

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Priyesh Shah |
| **Student ID:** | x18207731 |
| **Programme:** | Cloud Computing |
| **Year:** | 2020 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Sean Heeney |
| **Submission Due Date:** | 17/08/2020 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 1782 |
| **Page Count:** | 7 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 16th August 2020 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Priyesh Shah
x18207731

## 1 Introduction

The model developed as part of this project requires certain technologies and libraries to be installed. The project is developed using Python programming language. Apache Spark is used for analysis and MongoDB is used to store the data. Java is also required for the execution of the source code. Some of the important libraries used are PyMongo and PySpark. The visualization of the results is done using Microsoft PowerBI. The project was developed on local setup and evaluated on the local setup as well as AWS. A detailed guide on the system specification, installation of various components, and executing the source code are mentioned in this document.

This paper is divided into multiple sections. Section 2 highlights the system specification of the local setup and AWS both, section 3 discusses the detailed installation steps for various components, and section 4 talks about the artefact execution process.

## 2 System Specification

This section highlights the specification of the system used to develop and evaluate the project. It talks about the various tools, technologies, and libraries used for development and testing the model. The project was initially developed on a local setup using a virtual machine (VM). Once the development and evaluation was done locally, it was deployed on AWS and tested.

The local setup was done on a laptop running Windows 10 OS and having Intel i5 8th Gen Processor (4 cores and 8 virtual CPUs), 16GB RAM, 8GB Intel UHD Graphics Card, 128GB SSD, and 1TB HDD. A VM was deployed on top of the Host OS using Oracle VirtualBox. On AWS, the setup was done using two m5.xlarge instances. Java, Python, Spark, and PyMongo, PySpark, and other required libraries were installed on one instance and MongoDB was installed on the other instance. All these were installed on the VM as well. The specifications are mentioned in the table 1.

## 3 Installation

This section talks about the steps to be followed to install the various components required for setting up the system. This project is developed and tested on Ubuntu 18.04 OS but any Linux based OS such as Debian, Kali Linux, etc. can also be used. Also, the host OS on the local setup was Windows 10 but any other host platforms can be used. The

| | Local Setup | AWS |
|---|---|---|
| OS | Ubuntu 18.04 | Ubuntu 18.04 |
| RAM | 8GB | 16GB |
| Storage | 100GB | 16GB Elastic Block Storage |
| Processors | 4 CPUs | 4 virtual CPUs |
| Java | OpenJDK Version 1.8.0_265 | OpenJDK Version 1.8.0_265 |
| Python | Version 3.6.9 | Version 3.6.9 |
| Spark | Version 2.4.5 | Version 2.4.5 |
| MongoDB | Version 4.2.8 | Version 4.4.0 |
| PySpark Library | Version 2.4.6 | Version 2.4.5 |
| PyMongo Library | Version 3.10.1 | Version 3.11.1 |

Table 1: System Specification

installation steps remain the same for all Linux-based OS. The model was developed on local setup and tested on the local setup as well as AWS. The initial configuration for local setup and AWS differs a bit which is mentioned in section 3.1. Once that is done, the configuration for local setup and AWS is common which is mentioned in section 3.2.

## 3.1 Initial Configuration

### 3.1.1 Local Setup

To set up the system locally on a laptop, a hypervisor is required on top of which a VM will be hosted. Below are the steps to be followed:

1. Download the latest version of Oracle VirtualBox Windows Installer from here and install the same. Installers for other platforms are also available.

2. Download the desktop image for Ubuntu 18.04 LTS (Bionic Beaver) from here.

3. Follow the steps mentioned here to configure the VM.

### 3.1.2 AWS

Follow the below steps to configure and launch two instances on AWS running Ubuntu 18.04 AMI. The same configuration is required for both instances.

1. Open AWS console and go to EC2 launch wizard.

2. Select 64-bit (x86) Ubuntu Server 18.04 LTS (HVM), SSD Volume Type AMI.

3. Select m5.xlarge (4 vCPUs and 16GB RAM) instance or any other instance having a better configuration.

4. Add at least 16GB of storage.

5. Select an existing security group or create a new one and ensure the security group allows SSH (Port 22) and TCP (Port 27017) connection from your IP Address.

6. Review and launch the instance.

7. To access AWS instances from a laptop, download, install, and configure the AWS CLI as mentioned here on the laptop.

8. Access the AWS instance using the command *ssh -i "D:\AWS\thesis-project.pem" ubuntu@ec2-54-197-151-139.compute-1.amazonaws.com* from CLI

## 3.2 Common Configuration

The steps to install the other required components are the same for local setup as well as on AWS. The steps shown below are performed from CLI for the AWS instances. Java, Python, Spark, and PyMongo, PySpark, and other required libraries were installed on one instance and MongoDB was installed on the other instance. However, on the local setup, all these should be installed on the same VM. The same steps can be followed to configure the local setup. Execute the command *sudo apt-get update* before proceeding further:

### 3.2.1 Install Python

Ubuntu comes pre-installed with Python. Verify the existing Python version and install if required using the below steps:

1. Check if Python 3.6.9 or higher exists using *python3 –version.*

2. If it does not exist, install using *sudo apt-get install python3.6.*

### 3.2.2 Install Java 8

Java is required to execute Python scripts using Spark runtime. Install Java using the below steps:

1. Install JDK using *sudo apt-get install openjdk-8-jdk*

2. Install JRE using *sudo apt-get install openjdk-8-jre*

### 3.2.3 Install and Configure Apache Spark

Apache Spark is used to analyze large data hence it is required for this project. Install Spark using the below steps:

1. Change the directory using *cd /usr/local*

2. Download Spark 2.4.5 using
*sudo wget https://archive.apache.org/dist/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz*

3. Extract it using *sudo tar -xvzf spark-2.4.5-bin-hadoop2.7.tgz*

4. Create a symbolic link using *sudo ln -s spark-2.4.5-bin-hadoop2.7 spark*

5. Change the ownership of the link using *sudo chown -R ubuntu:ubuntu spark*

6. Delete the tar file using *sudo rm -rf spark-2.4.5-bin-hadoop2.7.tgz*

7. Change the directory using *cd*

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
export PYSPARK_PYTHON=python3
```

Figure 1: Screenshot of .bashrc file

8. Add the lines in *.bashrc* file using *sudo nano .bashrc* as shown in figure 1.
9. Compile the file using *source .bashrc*.
10. Change the directory using *cd /usr/local/spark*.
11. Test the installation by going to the Python shell using *./bin/pyspark*.
12. Type the command *print("Hello World")* and Hello World should be printed.

#### 3.2.4 Install Required Libraries

PyMongo and PySpark are the main libraries required for the project. However, installing these would require pip which is installed first. Install the required using the below steps:

1. Install pip using *sudo apt-get install python3-pip*.
2. Install PyMongo using *pip3 install pymongo*.
3. Install PySpark using *pip3 install pyspark*.
4. Verify PyMongo and PySpark installation using *pip3 freeze | grep pymongo* and *pip3 freeze | grep pyspark*.

#### 3.2.5 Install and Configure MongoDB

MongoDB has been used to store the huge amount of data gathered from various data sources. Also, the aggregated data is also stored in MongoDB. Install MongoDB using the below steps:

1. Import MongoDB public key using *wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -*.
2. Generate the MongoDB list file using
   *echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list*.
3. Install MongoDB using *sudo apt-get install -y mongodb-org*.
4. Enable MongoDB service on startup using *sudo systemctl enable mongod.service*.
5. Start MongoDB service using *sudo systemctl start mongod*.
6. Open the MongoDB configuration file using *sudo nano /etc/mongod.conf*.
7. Remove the comment from the security section and add line as shown in figure 2.

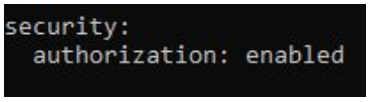



Figure 2: Screenshot of mongo.conf file

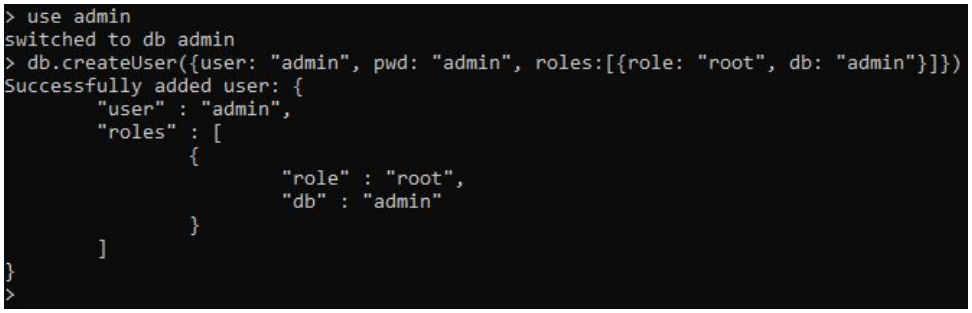



Figure 3: Screenshot of MongoDB create user

8. Restart the MongoDB service using *sudo systemctl restart mongod*.
9. Open the MongoDB shell using *mongo*.
10. Switch to the admin database using *use admin*.
11. Create a new admin user as shown in figure 3.
12. Verify the user using *mongo -u admin -p –authenticationDatabase admin* and then enter the password.

# 4 Execution

This section describes the prerequisites, the steps to execute the scripts, and the output. The details shown in this section are from the execution of scripts on AWS instance. However, the same steps can be followed for execution on the local setup as well. Download the CSV files for historical taxi booking (TLC (2020)) and weather (NOAA (2020)) data on a laptop and mention the filenames and file path in scripts. Generate an API key to access the OpenWeatherMap API (OpenWeatherMap (2020)) and mention it in the script along with the latitude and longitude of New York City. Mention the MongoDB URL, Port, Database Name, and Collection Names as required in the scripts. Create a directory to copy the scripts on the AWS instance. Copy all the scripts to the AWS instance as shown in figure 4.

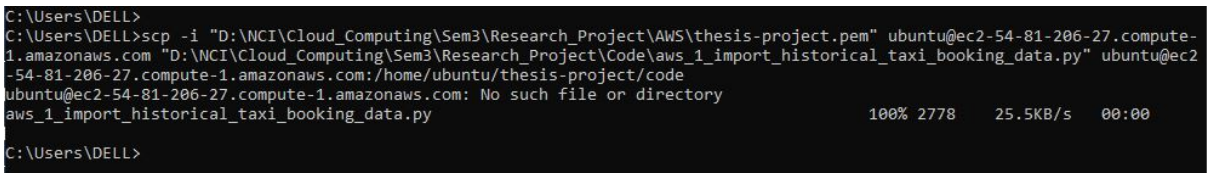



Figure 4: Copy scripts to AWS instance

Once the above steps are completed, access the AWS instance using SSH from a local terminal and go to the path where the scripts are saved on the AWS instance. Execute the scripts in the same sequence as shown in figure 5. Use the command *python3 aws_1_import_historical_taxi_booking_data.py* to execute the first script. Similarly, the other scripts can be executed. Scripts 1, 2, and 4 should be executed only when new data is to be uploaded. Scripts 3, 5, and 6 should be executed every time to fetch the latest weather forecast, predict the taxi bookings, and calculate the number of resources required.

| Sr. No. | Script Name | Description |
|---|---|---|
| 1 | aws_1_import_historical_taxi_booking_data.py | Script to import the historical taxi booking data from CSV and store in MongoDB |
| 2 | aws_2_import_historical_weather_data.py | Script to import the historical weather data from CSV and store in MongoDB |
| 3 | aws_3_fetch_weather_forecast.py | Script to fetch the weather forecast from OpenWeatherMap API and store in MongDB |
| 4 | aws_4_historical_taxi_booking_aggregator.py | Script to aggregate the taxi bookings for each historical date and store in MongoDB |
| 5 | aws_5_user_demand_predictor.py | Script to predict the user demand for each forecast day |
| 6 | aws_6_resource_calculator.py | Script to calculate the resources required for each forecast day |

Figure 5: List of Script

Script 1 takes a long time for execution as a huge amount of data is transferred over the internet and stored in MongoDB. While testing the developed model, over 90 million records for historical taxi bookings were stored in the database. Script 4 also takes some time for execution as all 91 million records are aggregated to count the number of taxi bookings per day and it is stored in the database. Script 5 predicts the number of taxi bookings for each forecast day and writes it in a CSV file whereas script 6 calculates the number of resources required to fulfill the demand and writes it in a CSV file. Figures 6 and 7 show the execution of script 5 and figures 8 shows the execution of script 6. Figure 9 shows the content of CSV files generated by scripts 5 and 6. In figure 9a, the first column shows the date, the second column is rain in inches and the third column is predicted taxi bookings. In figure 9b, the first column is the date, the second column is rain in inches, the third column is predicted taxi bookings, and the fourth column is the calculated resources required.

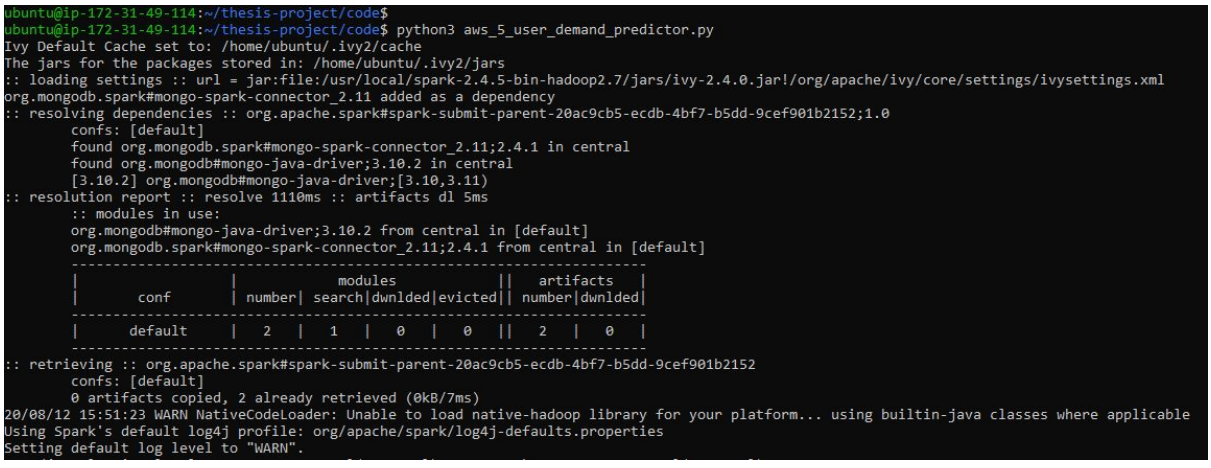



Figure 6: Script 5 Execution Start

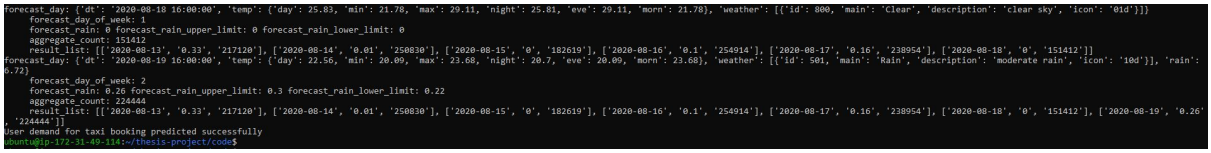



Figure 7: Script 5 Execution Result

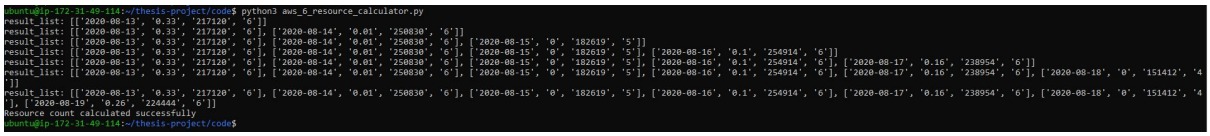
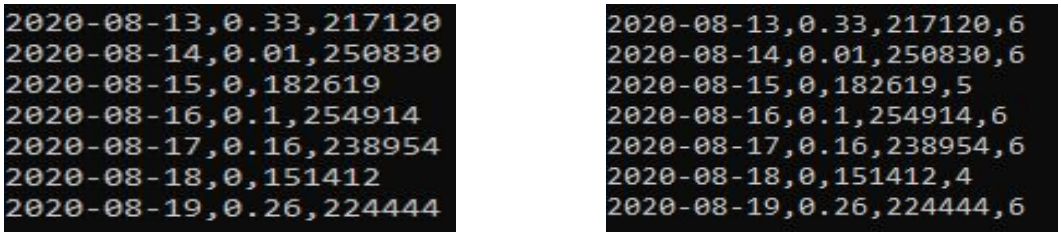



Figure 8: Script 6 Execution




(a) Script 5 (b) Script 6

Figure 9: Output CSV Files

The CSV file generated by script 6 is the final output containing all the information such as the forecast date, rain, predicted taxi bookings, and the calculated resources required. This file is then used to generate graphs using Microsoft PowerBI.

# References

NOAA, N. (2020). Local climatological data (lcd) | data tools | climate data online (cdo) | national climatic data center (ncdc).
**URL:** *https://www.ncdc.noaa.gov/cdo-web/datatools/lcd*

OpenWeatherMap (2020). One call api: Weather data for any geographical coordinate - openweathermap.
**URL:** *https://openweathermap.org/api/one-call-api*

TLC, N. (2020). About tlc - tlc.
**URL:** *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*