

# Configuration Manual

MSc Research Project  
Cloud Computing

Siarhei Staravoitau  
Student ID: x18162070

School of Computing  
National College of Ireland

Supervisor: Horacio Gonzalez-Velez

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Siarhei Staravoitau
<b>Student ID:</b>	x18162070
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2022
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Horacio Gonzalez-Velez
<b>Submission Due Date:</b>	19/09/2022
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	1590
<b>Page Count:</b>	12

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.

<b>Signature:</b>	
<b>Date:</b>	18th September 2022

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

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

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

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Siarhei Staravoitau  
x18162070

## 1 Introduction

This configuration manual is designed to walk through the research project hardware and software requirements, and how to replicate testing steps and install and deploy code to the AWS cloud. Research Question:

To what extent Machine Learning Cloud-Based Services can enhance the efficiency of stock price prediction and replace it to support investors and stock market traders?

The main idea of this research is to establish and compare which cloud computing solution, including PySpark framework and distributed model training, is more efficient when performing Data Analytics task like TensorFlow Sequential time series stock price prediction by measuring average execution time and accuracy (RMSE, MAE, MAPE, MDAPE) with different hyperparameters (batch size, epochs and number of neurons). The research is testing the efficiency of using Cloud Computing platforms (Google Colab Pro+ with GPU and TPU, Jupyter Lab), tools and features like PySpark scalability and TensorFlow distributed model training when performing ML LSTM Stock price prediction in Google Colab Pro, AWS EMR, AWS Lambda and Local PC. The accuracy and model processing time of performing ML tasks are benchmarks for the evaluation of changing cloud computing environment parameters. Changing model training hyperparameters is used to change test workload for evaluation of defined Cloud Computing environments. Project completion stages include the following stages: prepare data set including polarity index calculation; use this data set in model training, testing and prediction on Google Colab Pro, AWS EMR, AWS Lambda and Local PC.

## 2 System Configuration

### 2.1 Prerequisites set up

#### 2.1.1 Twitter Developer Academic Account

Create Twitter Developer Academic account to be able to perform searches for a long period

#### 2.1.2 Create AWS account with admin rights

Create an AWS account with Admin rights to be able to manage access to the services.

### 2.1.3 AWS S3

1. Log in as Admin user
2. In IAM Create a user with programmatic access
3. Create two AWS S3 buckets: *mscloudetop-etl* and *mscloudetop-results* with un-ticked Block all public access.
4. In IAM creates two Policies granting access rights as per Figure 1 for each bucket and attach them to the user created in step 2.



```
1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "VisualEditor0",
6-       "Effect": "Allow",
7-       "Action": [
8-         "s3:GetBucketPublicAccessBlock",
9-         "s3:GetBucketPolicyStatus",
10-        "s3:GetAccountPublicAccessBlock",
11-        "s3:ListAllMyBuckets",
12-        "s3:GetBucketAcl",
13-        "s3:GetBucketLocation"
14-      ],
15-      "Resource": "*"
16-    },
17-    {
18-      "Sid": "VisualEditor1",
19-      "Effect": "Allow",
20-      "Action": "s3:ListBucket",
21-      "Resource": "arn:aws:s3:::mscloudetop-results"
22-    },
23-    {
24-      "Sid": "VisualEditor2",
25-      "Effect": "Allow",
26-      "Action": "s3:*Object",
27-      "Resource": "arn:aws:s3:::mscloudetop-results/*"
28-    }
29-  ]
30- }
```

Figure 1: AWS S3 access policy

5. Create access keys for user created in step 2 and use them in ETL Jupyter notebook and ML job notebooks for reading and writing stock\_prices.csv and results.csv file

### 2.1.4 AWS ECR

1. Create user having full access rights to ECR
2. Open AWS ECR page and create Repository
3. Select repository from step 2 and click on View push commands
4. Copy commands from step 3 and follows them when required to upload docker image from local PC

## 2.1.5 AWS EMR

1. Create user having full access rights to EMR
2. Login to AWS Management Console and open EMR [1] service page.
3. Open Notebooks page and click Create Notebook. Provide a notebook name, and choose to Create a cluster.
4. Specify instance *c5.24xlarge*
5. Click Create Notebook (Figure 2) and wait for Cluster and notebook to be provisioned.

### Create notebook

#### Name and configure your notebook

Name your notebook, choose a cluster or create one, and customize configuration options if desired. [Learn more](#)

**Notebook name\***   
Names may only contain alphanumeric characters, hyphens (-), or underscores (\_).

**Description**   
256 characters max.

**Cluster\***  Choose an existing cluster  
 Create a cluster **i**

**Cluster name:**

**Release:** emr-5.36.0

**Applications:** Hadoop, Spark, Livy, Hive, JupyterEnterpriseGateway

**Instance:**

**EMR role** [EMR\\_DefaultRole](#)  Use EMR\_DefaultRole\_V2 **i**

**EC2 instance profile** [EMR\\_EC2\\_DefaultRole](#) **i**

**EC2 key pair:**  **i**

**Auto-termination**  Enable auto-termination  
Terminate cluster when it is idle after  hours  minutes

**Security groups**  Use default security groups **i**  
 Choose security groups (vpc-0c735787e36a3c094)

**AWS service role\***  **i**

**Notebook location\*** Choose an S3 location where files for this notebook are saved.  
 Use the default S3 location  
s3://aws-emr-resources-250738637992-eu-west-1/notebooks/  
 Choose an existing S3 location in eu-west-1

▶ **Git repository** Link to a Git repository

▶ **Tags** **i**

\* Required

[Cancel](#) [Create notebook](#)

Figure 2: EMR Notebook creation

## 2.1.6 AWS Lambda

1. Create user having full access rights to Lambda

## 2.1.7 Create Google account and log in to Google Colab Pro+

## 2.2 Google Colab Pro +

For using Google Colab Pro+ it is required

1. Create a Google account and log in on the Google Colab website: <https://colab.research.google.com/> and connect to a hosted runtime[2]
2. Open [https://colab.research.google.com/signup/pricing?utm\\_source=resource\\_tab&utm\\_medium=link&utm\\_campaign=want\\_more\\_resources](https://colab.research.google.com/signup/pricing?utm_source=resource_tab&utm_medium=link&utm_campaign=want_more_resources) and enable Colab Pro or Colab Pro+ subscription to use high power GPU[3]

In order to connect Google Colab to local runtime, it is required to follow Google instructions (Figure 3) <https://research.google.com/colaboratory/local-runtimes.html>[4]

### Setup instructions

In order to allow Colaboratory to connect to your locally running Jupyter server, you'll need to perform the following steps.

#### Step 1: Install Jupyter

Install [Jupyter](#) on your local machine.

#### Step 2: Install and enable the `jupyter_http_over_ws` jupyter extension (one-time)

The `jupyter_http_over_ws` extension is authored by the Colaboratory team and available on [GitHub](#).

```
pip install jupyter_http_over_ws
jupyter serverextension enable --py jupyter_http_over_ws
```

#### Step 3: Start server and authenticate

New notebook servers are started normally, though you will need to set a flag to explicitly trust WebSocket connections from the Colaboratory frontend.

```
jupyter notebook \
--NotebookApp.allow_origin='https://colab.research.google.com' \
--port=8888 \
--NotebookApp.port_retries=0
```

Once the server has started, it will print a message with the initial backend URL used for authentication. Make a copy of this URL as you'll need to provide this in the next step.

#### Step 4: Connect to the local runtime

In Colaboratory, click the "Connect" button and select "Connect to local runtime...". Enter the URL from the previous step in the dialog that appears and click the "Connect" button. After this, you should now be connected to your local runtime.

Figure 3: Google Colab Local Runtime Connection

- Install Jupyter lab on local PC by running in Terminal or Command Prompt (<https://jupyter.org/install>)[5]: `pip install jupyterlab`
- Run in Terminal or Command Prompt to start Jupyter lab: `jupyter-lab` or follow Google's instructions to connect Google Colab to local runtime:
  1. `pip install jupyter\_http\_over\_ws`

2. `jupyter serverextension enable --py jupyter\_http\_over\_ws`
  
3. `jupyter notebook \`  
`--NotebookApp.allow\_origin='https://colab.research.google.com' \`  
`--port=8888 \`  
`--NotebookApp.port\_retries=0`
  
4. Copy localhost link from Terminal or Command Prompt and Paste it into Google Colab URL prompt

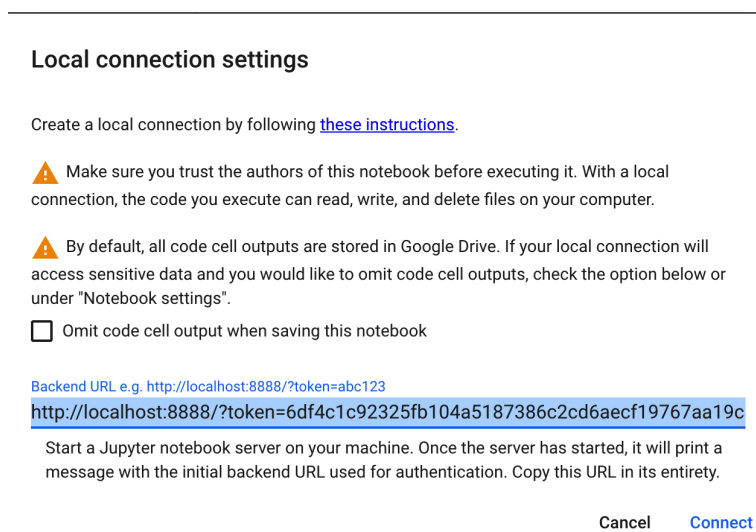


Figure 4: Google Colab Connect to Local Runtime

## 2.3 AWS EMR

1. Open AWS EMR Jupyter lab Notebook
2. Import Test Jupyter notebooks *Pandas.ipynb* and *pyspark.ipynb*, open them and connect to Kernel Python 3

## 2.4 AWS Lambda

Required:

- PC with Intel or AMD processor
- MS Visual Studio Code installed from <https://code.visualstudio.com/download>[6]
- Docker Desktop installed installed from <https://www.docker.com/products/docker-desktop/>[7]
- Install AWS CLI: `pip install awscli` from <https://pypi.org/project/awscli/>[8]
- Login to AWS console, open ECR[9] page
- Create repository mljob, select repository with radio button and click View Push Commands

- Retrieve an authentication token and authenticate your Docker client to your registry:  
`aws ecr get-login-password --region us-east-1` — `docker login` and other parameters to log in to AWS ECR from the terminal.

For deploying AWS Lambda code required:

1. unarchive file `Lambda.zip`
2. In MS Visual Studio Code open folder `Lambda`. it should be three files: `app.py`; `Dockerfile`; `requirements.txt`.
3. Open file `app.py` and update AWS S3 bucket access key and secret and name of S3 buckets: `s3://mscloudetop-etl` and `s3://msccloudetop-results` and save it. Or request access key and secret if required to test on existing buckets.
4. Build your Docker image using the following command

```
docker build -t mljob .
```

5. Tag docker image

```
docker tag mljob:latest useracc.dkr.ecr.us-east-1.amazonaws.com/mljob:latest
```

6. Push docker image to mljob AWS repository:

```
docker push useracc.dkr.ecr.us-east-1.amazonaws.com/mljob:latest
```

7. Open AWS Lambda [10] page and click Create function
8. Select Container image, provide function name and choose container image

```
mljob:latest
```

9. Architecture option should be x86 64
10. Click Create function
11. When Function is created, select function Configuration page and configure time out 15 mins, Memory 3008MB, Ephemeral storage 10240MB
12. Add AWS S3 [11] trigger using buckets from step 3 to invoke Lambda function when the file is saved in a bucket `mscloudetop-etl`. NB: it is required to configure the AWS IAM role to be able to read and write files into the AWS S3 bucket.



## 2.5 Local PC

In case if Local PC is Apple Macbook Pro M1 Max. It has an ARM-based CPU and 24-core GPU. In order to execute TensorFlow ML job training it is required to install following software (<https://developer.apple.com/metal/tensorflow-plugin/>) [12]:

1. Install Conda environment

```
chmod +x ~/Downloads/Miniforge3-MacOSX-arm64.sh
sh ~/Downloads/Miniforge3-MacOSX-arm64.sh
source ~/miniforge3/bin/activate
```

2. Install TensorFlow dependencies

```
conda install -c apple TensorFlow-deps
```

3. Install base TensorFlow

```
python -m pip install tensorflow-macos
```

4. Install TensorFlow metal plugin

```
python -m pip install tensorflow-metal
```

## 3 Running Tests

### 3.1 ETL Jupyter Notebook

1. Open Google Colab and import ETLNotebook.ipynb
2. Enter AWS access key, secret, Twitter bearerToken and AWS S3 bucket name.
3. ETL Jupyter notebook requires from an end user only entry of a number of years, stock ticker and hashtag.
4. Connect to remote host and

```
Runtime\Run all
```

It will automatically retrieve data, clean tweets, calculate polarity indexes, retrieve data from Yahoo Finance, merge daily polarity indexes with stock prices and save CSV file in AWS S3 bucket *mscloudetop-etl* or other bucket specified by the user.

## 3.2 Running ML Job notebooks in Google Colab Pro+

1. Open Google Colab and import Pandas.ipynb and PySpark.ipynb notebooks
2. Enter AWS access key, secret and AWS S3 bucket names for *mscloude-top-etl* and *msccloude-top-results* if using different ones.
3. Uncomment required test Variant hyperparameters and relevant model compiling cell. Please check Model Training Cell to be commented out when running Elephas and Keras Tuner tests to avoid crashes.
4. When running TPU-based tests - make sure that connected to TPU.
5. Connect to remote host using GPU or TPU with extended memory and

```
Runtime\Run all
```

### 1. Configure test hyperparameters for test execution

```
[ ] 1  epochs = 30 # Specify number of epochs
    2  sequence_length = 30 # = Sliding window
    3  n_neurons = sequence_length * 7 # Number of neurons calculation
    4  batch_size = 16 # Specify Batch size
    5  Number_Of_Workers = 1 # Specify number of workers
    6  DataFrame = 'PySpark' # Specify Data Frame used: PySpark or Pandas
    7  GPU = 'GPU' # processor type: GPU, TPU, EMR, Lambda
    8  Platform = 'Google Colab' # Specify platform: Google Colab, EMR, Lambda, Local PC
    9  Experiment_Name = 'ML_Job_PYSPARK_Colab' # Provide experiment name
   10 # b = 's3://mscloude-top-etl/stock_price_NDX.csv' # specify data set file for test model training
   11 b = 's3://mscloude-top-etl/stock_price.csv'
   12 # b = 's3://mscloude-top-etl/stock_price.csv'
```

Figure 5: Hyperparameters set up

Figure 5 demonstrates how Hyperparameters are to be set up first in Google Colab Jupyter Lab Notebook when connected to a remote host.

#### 3.2.1 AWS EMR tests execution

1. Open AWS EMR Jupyter Lab Notebook and import Pandas.ipynb and PySpark.ipynb notebooks
2. Enter AWS access key, secret and AWS S3 bucket names for *mscloude-top-etl* and *msccloude-top-results* if using different ones.
3. Uncomment required test Variant hyperparameters and relevant model compiling cell. Please check Model Training Cell to be commented out when running Elephas and Keras Tuner tests to avoid crashes.
4. Connect to Python 3 kernel and click

```
Runtime\Run all
```

### 3.2.2 AWS Lambda Tests execution

1. Open Google Colab and import hyperparameters.ipynb notebook
2. Enter AWS access key, secret and AWS S3 bucket names for *mscloudeetop-etl* and *mscccloudeetop-results* if using different ones.
3. Run cells one by one to retrieve current hyperparameters, review and update them by pushing csv to AWS bucket
4. Run cells to retrieve test results from the bucket
5. check AWS CloudWatch for related Lambda function logs

2. Set up hyperparameters settings as required. Processing file name could be `stock_price.csv` as it generated by ETL Jupyter Notebook. But for testing purposes it was selected to run all tests on the same dataset `stock_price_GOOG.csv`

```

1 hyperparameters['epochs'].iloc[0] = 50 # Number of epochs
2 hyperparameters['sequence_length'].iloc[0] = 30 # Days lag
3 hyperparameters['batch_size'].iloc[0] = 32 # Batch size
4 hyperparameters['DataFrame'].iloc[0] = 'Pandas' # Pandas or PySpark
5 hyperparameters['GPU'].iloc[0] = 'Lambda' # Processor type: LAMBDA
6 hyperparameters['Platform'].iloc[0] = 'AWS Lambda' # Platform: AWS Lambda
7 hyperparameters['Experiment_Name'].iloc[0] = 'AWS Lambda Testing hyperparameters' # Experiment name
8 hyperparameters['Processing_File'].iloc[0] = 's3://mscloudeetop-etl/stock_price.csv' # Data set file location. For testing purposes was used
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
9 |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |                                     |
10
11 Platform = hyperparameters['Platform'].iloc[0]
12 DataFrame = hyperparameters['DataFrame'].iloc[0]
13 Experiment_Name = hyperparameters['Experiment_Name'].iloc[0]
14 epochs = hyperparameters['epochs'].iloc[0]
15 Number_of_Neurons = int(hyperparameters['sequence_length'].iloc[0]) * 7
  
```

Figure 6: AWS Lambda Hyperparameters set up

Jupyter Notebook could be executed in Google Colab when connected to remote or local after specifying hyperparameters (Figure 6) and saving hyperparameters into AWS S3 bucket. To view test run results user could execute code in the relevant cell.

### 3. Review and save hyperparameters to the mscloudetop-etl bucket.

```
[ ] 1 hyperparameters
```

epochs	sequence_length	batch_size	DataFrame	GPU	Platform	Experiment_Name	Processing_File
0	20	10	64	Pandas	Lambda	AWS Lambda	AWS Lambda Testing hyperparameters s3://mscloudetop-etl/stock_price.csv

```
[ ] 1 hyperparameters.to_csv(hyper,
2 # mode='a',
3 index=False,
4 header=True,
5 storage_options={'key': AWSKey, 'secret': AWSSecret})
```

### 4. Retrieve data from results.csv file in AWS S3 bucket

```
1 results_path = 's3://mscloudetop-results/results.csv'
2 results_test = pd.read_csv(results_path, storage_options={'key': AWSKey, 'secret': AWSSecret})
3
4 results_test.loc[
5 (results_test['Platform'] == Platform) &
6 (results_test['Symbol'] == 'GOOG')
7 & (results_test['DataFrame'] == DataFrame)
8 & (results_test['Experiment_Name'] == Experiment_Name)
9 & (results_test['Number_of_Epochs'] == epochs)
10 & (results_test['Batch_Size'] == batch)
11 & (results_test['Number_of_Neurons'] == Number_of_Neurons)
12 ]
```

Figure 7: Save AWS Lambda Hyperparameters in AWS S3 bucket

### 3.2.3 Keras Tuner Test execution

There are two options considered to execute the Keras tuner:

1. Google Colab remote host
2. Local PC

It is required to enter the directory path parameter where Keras Tuner will save trial files (Figure 8). When running tests on a local PC - Please make sure to enter the local path. Otherwise, the model training job will fail.

#### 1. Configure test hyperparameters for test execution

```
[1] 1 directory = "/content/elephas/keras_tuner", # Specify path to folder where Keras tuner will save trials.
2 # directory = "/Users/sergei/Projects/keras_tuner" # Specify local path to folder where Keras tuner will save trials.
3 project = "MSCCLOUDETOP"
4
5 epochs = 30 # Specify number of epochs
6 sequence_length = 30 # = Sliding window
7 n_neurons = sequence_length * 7 # Number of neurons calculation
8 batch_size = 8 # Specify Batch size
9 Number_of_Workers = 1 # Specify number of workers
10 DataFrame = 'Pandas' # Specify Data Frame used: PySpark or Pandas
11 GPU = 'GPU' # processor type: GPU, TPU, EMR, Lambda
12 Platform = 'Local_PC' # Specify platform: Google Colab, EMR, Lambda, Local PC
13 Experiment_Name = 'Keras_Tuner_ML_Job_No_Spark_Local_PC' # Provide experiment name
14 # b = 's3://apple-etl/stock_price_NDX.csv' # specify data set file for test model training
15 b = 's3://mscloudetop-etl/stock_price.csv'
16 # b = 's3://apple-etl/stock_price.csv'
```

Figure 8: Keras Tuner Hyperparameters Configuration

1. Open Google Colab and import Pandas.ipynb and PySpark.ipynb notebooks

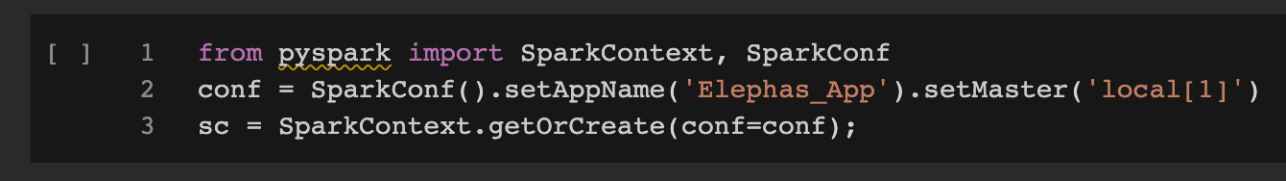
2. Enter AWS access key, secret and AWS S3 bucket names for *mscloudetop-etl* and *mscloudetop-results* if using different ones.
3. Uncomment required test Variant hyperparameters and relevant model compiling cell - Keras Tuner. Please check Model Training Cell to be commented out when running Elephas and Keras Tuner tests to avoid crashes.
4. When running TPU-based tests - make sure that connected to TPU.
5. Connect to remote host and

```
Runtime\Run all
```

### 3.2.4 Elephas Tests execution

1. Open Google Colab and import Pandas.ipynb notebooks. It will crash in PySpark.ipynb due to PySpark application ports clashing.
2. Enter AWS access key, secret and AWS S3 bucket names for *mscloudetop-etl* and *mscloudetop-results* if using different ones.
3. Uncomment required test Variant hyperparameters and relevant model compiling cell - Elephas. Please check Model Training Cell to be commented out when running Elephas and Keras Tuner tests to avoid crashes.
4. When running TPU-based tests - make sure that connected to TPU.
5. Connect to remote host and

```
Runtime\Run all
```



```
[ ] 1 from pyspark import SparkContext, SparkConf
    2 conf = SparkConf().setAppName('Elephas_App').setMaster('local[1]')
    3 sc = SparkContext.getOrCreate(conf=conf);
```

Figure 9: Configure Elephas Number of Workers

Similar to the tests above, it is required to set up hyperparameters. Also, it is required to configure a number of workers (Figure 9)

## References

- [1] Big Data Platform – Amazon EMR – Amazon Web Services. <https://aws.amazon.com/emr/>, last accessed 2022-08-10.
- [2] Colab - Colaboratory. <https://colab.research.google.com/>, last accessed 2022-08-10.

- [3] Google Colab. [https://colab.research.google.com/signup/pricing?utm\\_source=resource\\_tab&utm\\_medium=link&utm\\_campaign=want\\_more\\_resources](https://colab.research.google.com/signup/pricing?utm_source=resource_tab&utm_medium=link&utm_campaign=want_more_resources), last accessed 2022-08-10.
- [4] Google Colab. <https://research.google.com/colaboratory/local-runtimes.html>, last accessed 2022-08-10.
- [5] Project Jupyter | Installing Jupyter. <https://jupyter.org/install>, last accessed 2022-08-10.
- [6] Download Visual Studio Code - Mac, Linux, Windows. <https://code.visualstudio.com/download>, last accessed 2022-08-10.
- [7] Docker Desktop - Docker. <https://www.docker.com/products/docker-desktop/>, last accessed 2022-08-10.
- [8] awscli · PyPI. <https://pypi.org/project/awscli/>, last accessed 2022-08-10.
- [9] Fully Managed Container Registry – Amazon Elastic Container Registry – Amazon Web Services. [https://aws.amazon.com/ecr/?nc1=h\\_ls](https://aws.amazon.com/ecr/?nc1=h_ls), last accessed 2022-08-10.
- [10] Serverless Computing - AWS Lambda - Amazon Web Services. <https://aws.amazon.com/lambda/>, last accessed 2022-08-10.
- [11] Cloud Object Storage – Amazon S3 – Amazon Web Services. <https://aws.amazon.com/s3/>, last accessed 2022-08-10.
- [12] Tensorflow Plugin - Metal - Apple Developer. <https://developer.apple.com/metal/tensorflow-plugin/>, last accessed 2022-08-10.