CHICAGO TAXI FARE PREDICTION

- Deployment STEPS

AUTHOR: **KARTHIKEYAN NG**

VERSION: **1.0**

CREATION DATE: November 6, 2020

Table of ContentS

[System Requirements: 3](#_Toc55562992)

[Model Training: 3](#_Toc55562993)

[Docker Container 4](#_Toc55562994)

[Deploy in ECS: 5](#_Toc55562995)

This document details the steps for deploying the training module and API developed for Chicago Taxi Fare prediction Project.

# System Requirements:

1. Operating System – macOS Catalina
2. Software:
   1. AWS CLI 2.0.22
   2. GIT 2.26.2
   3. Docker Engine 19.03.8
3. AWS:
   1. Account – Access to read and write from S3, Create and execute permissions for EC2 and EMR
   2. An S3 bucket
4. Keys:
   1. AWS Account Number
   2. Google Geocoding API key

# Model Training:

1. In MacBook, open terminal
2. Navigate to the working folder
3. Enter the below command to download the master.py file

curl -O <https://raw.githubusercontent.com/karthigkk/chicago_taxi_ss/master/packages/regression_model/regression_model/master.py>

1. Enter the below command to download the zip file

curl -O <https://raw.githubusercontent.com/karthigkk/chicago_taxi_ss/master/packages/regression_model/regression_model.zip>

1. Download spark-bigquery jar from the below path

https://mvnrepository.com/artifact/com.google.cloud.spark/spark-bigquery\_2.12/0.17.3

1. Use the below command to push the downloaded zip file to AWS S3

aws s3 sync . s3://<s3 bucket name>

1. Log into AWS account
2. Search for EMR
3. Click on Create Cluster
4. Click on “Go to Advanced Options”
5. Select Release as emr-5.30.1. In checkboxes below leave the default ones and check “Spark 2.4.5”
6. Enter the below configuration

[{"configurations":[{"classification":"export","properties":{"PYSPARK\_PYTHON":"/usr/bin/python3"}}],"classification":"spark-env","properties":{}}]

1. Select m5.2xlarge instances and select # of instances as 4
2. Update root device volume to 100GB
3. Leave all other options default and create cluster
4. Once the cluster is moved to “Waiting” status, navigate to Steps tab
5. Add the below step
6. spark-submit --deploy-mode cluster --py-files <s3 bucket path to zip file> --files <google key file> --packages com.microsoft.ml.spark:mmlspark\_2.11:1.0.0-rc2 --repositories https://mmlspark.azureedge.net/maven --jars <path to spark-bigquery-latest.jar file> <s3 bucket path to master.py file> <s3 path to save the trained model>
7. Wait for the status to show as complete
8. Move the final trained model into project’s /regression\_model/regression\_model/trained\_models folder

# Docker Container

1. In MacBook, open terminal
2. Navigate to the working folder
3. Enter the below command to download the code from git

git clone <https://github.com/karthigkk/chicago_taxi_ss.git>

1. Navigate into the folder Chicago\_taxi\_ss
2. Enter the below command to create docker image

docker build --build-arg GEO\_API\_KEY=%GEO\_API\_KEY% -t regression\_model:latest .

1. Enter the below command to push the docker image to AWS container registry
2. Tag the docker image by running the below command

docker tag regression\_model:latest <account\_id>.dkr.ecr.us-east-1.amazonaws.com/regression\_model:latest

1. Get ECR login password my entering the below command

aws ecr get-login-password --region us-east-1

1. Copy the output of the above command
2. Use in the output in the below command to log in

docker login -u AWS -p <output from step 8> https://<account\_id>.dkr.ecr.us-east-1.amazonaws.com

1. On executing above command, we will get message as “Login Succeeded”
2. Enter the below command to push the image to ECR
3. docker push <account\_id>.dkr.ecr.us-east-1.amazonaws.com/regression\_model:latest
4. Create ECR repository using the below command

aws ecr create-repository --repository-name regression\_model

1. Push the docker image into the newly created repository by using the below command

docker push <account\_id>.dkr.ecr.us-east-1.amazonaws.com/regression\_model:latest

1. Login to AWS EMR and validate if the new repository is available with the image pushed in

# Deploy in ECS:

For this project, we are using Elastic Container Services for deploying our API. ECS is highly scalable Docker container running service and is integrated with Elastic Load Balancing, which will help is distribute the workload across multiple instances, based on workload

1. Login into AWS ECS
2. On the left pane click on “Task Definition”
3. Click on “Create new Task Definition”
4. Select EC2 option and click Next button
5. Enter the below values
   1. Task Definition Name - Chicago\_Taxi\_ML\_Task
   2. Task Role – ecsTaskExecutionRole
   3. Network Mode – Bridge
   4. Task Execution Role – ecsTaskExecutionRole
   5. Task Memory – 512
   6. Task CPU – 1024
   7. Click “Add Container” button
   8. Container Name - chicago\_taxi\_container
   9. Image - <Enter image URI from EMR>
   10. Enter Memory Limit – 512
   11. Port Mapping –
       1. Host Port – 80
       2. Container Port – 5000
   12. Under Environment Variables
       1. Key – GEO\_API\_KEY
       2. Value - <Enter the geocoding api key>
   13. Click Add button
   14. Click Create Button
6. In left Pane – Click on Clusters
7. Click on Create Cluster
8. Select EC2 Linux + Networking and click on Next button
9. Enter cluster name as “ChicagoTaxiProject”
10. Select EC2 Instance type as t2.large
11. Enter Number of instances as 2 (This can be updated as many based on the workload)
12. Once cluster is created, click on “View Cluster” button
13. Click on Tasks tab
14. Click Run new task
15. Select Launch Type as EC2
16. Task Definition as “Chicago\_Task\_ML\_Task”
17. Leave all other options as default and click on Run task
18. Wait for the last status to be updated to “RUNNING”
19. Click on task name
20. Scroll down and expand the Container row
21. Under Network bindings section, click on the link under “External Link” column. This is the application url. This can be mapped to the corresponding domain name as required