SUBMITTED BY-SOMI POOREY

https://github.com/somipoorey/Final_Exam

INTRODUCTION:

A bank wants to increase revenue by getting more customers to subscribe to long-term deposits. Based on data from past telemarketing campaigns, we want to predict which clients are more likely to subscribe, so that the bank can target these customers and improve their conversion rate. This includes building and deploying a machine learning model as a web service which can be demonstrated as a client-side application.

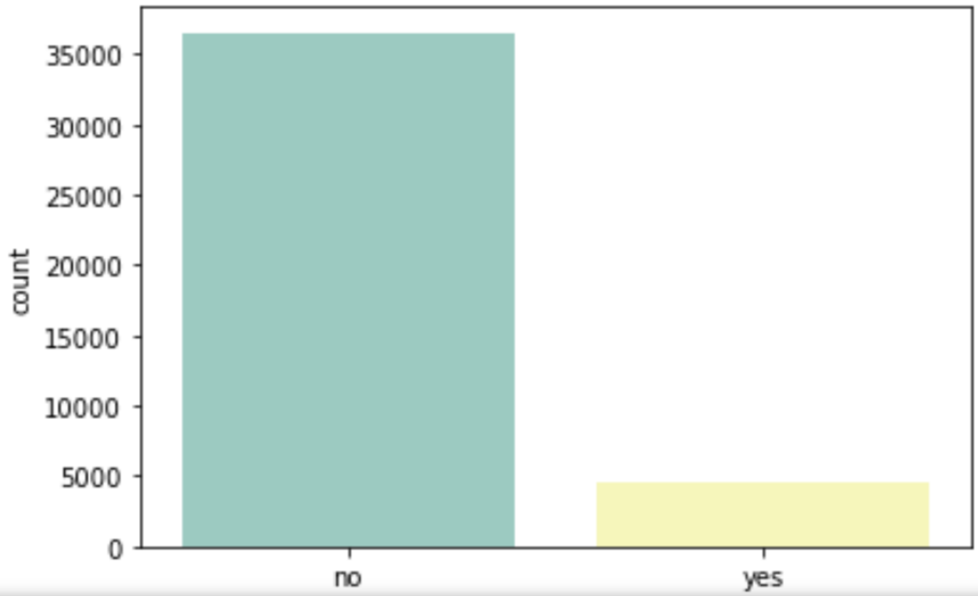
GOAL: The classification goal is to predict if the client will subscribe a term deposit (target variable y).

ABOUT THE DATASET:

The selected dataset was provided to us and it consists of direct marketing campaigns data of a banking institution. The dataset was picked from UCI Machine Learning Repository which is an amazing source for publicly available datasets. The dataset consists of 36,537 data points with 20 independent variables out of which 10 are numeric features and 10 are categorical features.

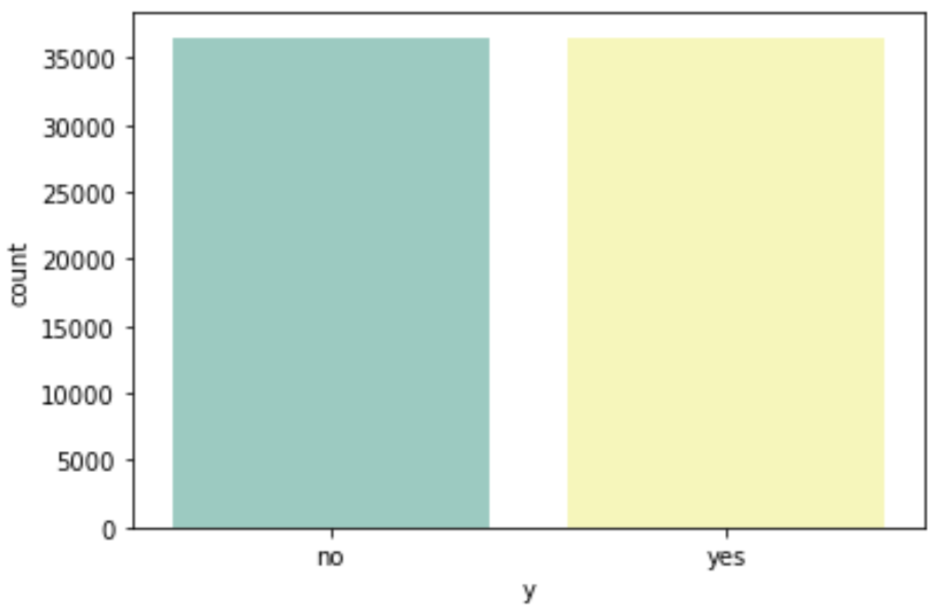
HIGHLIGHTS:

The dataset is imbalanced, with only 11% of the customers called subscribing to long-term deposits. This means a majority classifier would easily get around 89% classification accuracy,



We can see from the above plot that the dataset in imbalanced, where the number of negative classes is close to 8 times the number of positive classes.

So, I balanced the dataset by upsampling the minority class



The features are a mix of numerical and categorical data. The features can be divided into 3 broad categories: customer data (age, marital status, job, etc.), campaign data (number of calls to the customer during this campaign, number of days since last contact, etc.) and economic data (employment variation rate, consumer price index, etc.).

One of the features under campaign data is duration, which gives the call duration in seconds when the customer was last contacted.

DATA TYPES:

There are multiple types of data types available in the data set. some of them are numerical type and some of categorical type. You are required to get the idea about the data types after reading the data frame. Following are the some of the types of variables:

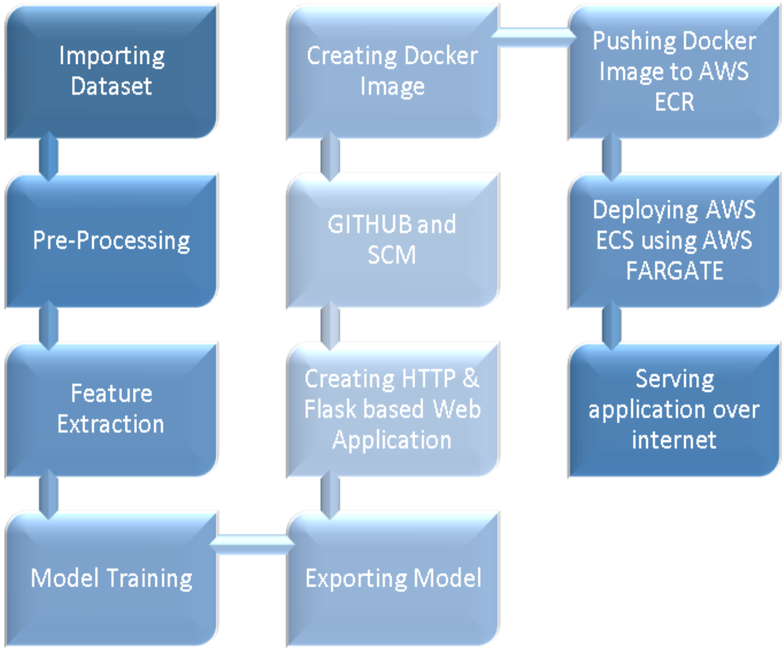
Numeric data type: banking dataset: salary, balance, duration and age.

Categorical data type: banking dataset: education, job, marital, poutcome and month etc.

Ordinal data type: banking dataset: Age group.

Time and date type

PROJECT WORKFLOW



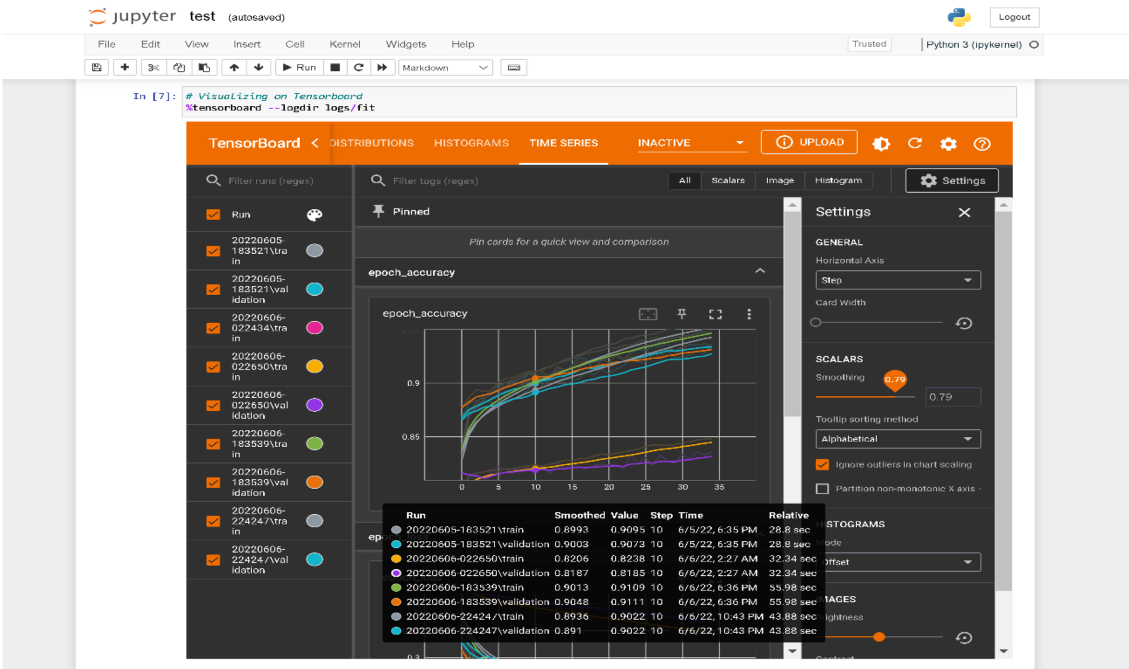
DEPENDENCIES:

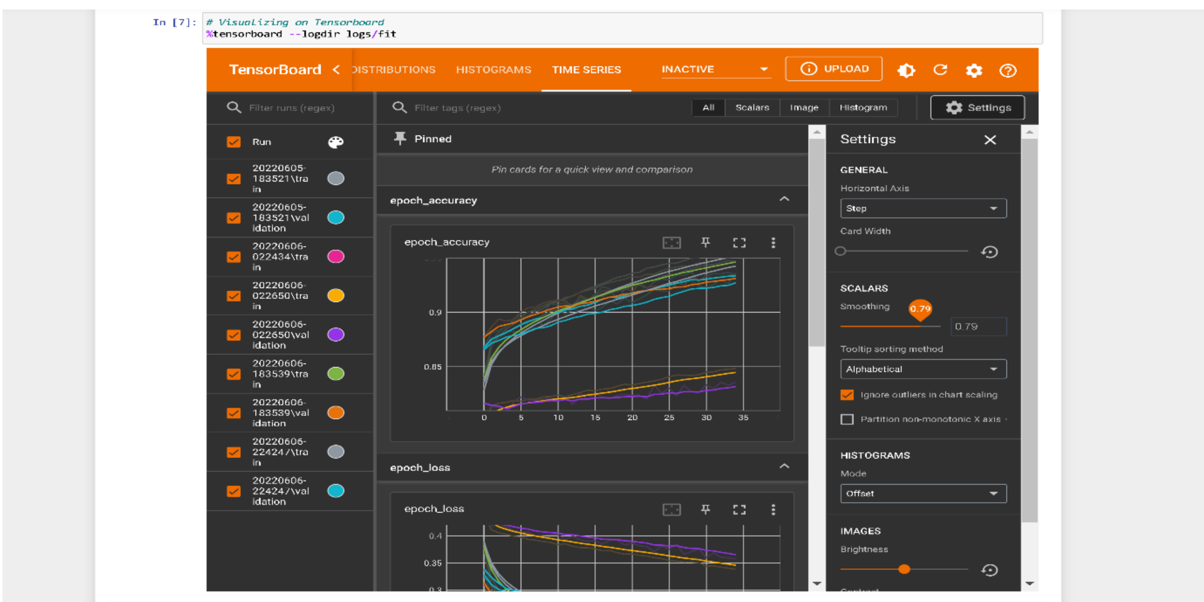
- Python 3
- Scikit-Learn
- Pandas/Numpy
- Seaborn/Matplotlib
- Current Web Browser
- Jupyter Notebook
- MLflow
- Tensorflow
- Pickle
- Amazon AWS
- Flask, Flasgger APIs
- Docker/CLI

TRAINING MODEL

The Python programming language was used to create the model. In addition, the packages Matplotlib, Keras, and NumPy were utilized for system implementation. Keras provides built-in functions such as activation functions, optimizers, layers, etc. TensorFlow was also used as the system's back-end.

Tensor board can be seen tracking:





EXPORTING MODEL

You must first export your trained models in TensorFlow SavedModel format before deploying them to AI Platform Prediction and using them to deliver predictions. TensorFlow's recommended format for exporting models is a

SavedModel, and it is required for deploying trained TensorFlow models on AI Platform Prediction. When you export your trained model as a SavedModel, you save your training graph, including its assets, variables, and metadata, in a format that AI Platform Prediction can ingest and restore for predictions.

CREATING HTTP & FLASK BASED WEB APPLICATION

In order to create a Gender Detection web application, I have created three different files:

Flask for the back-end engine - app.py

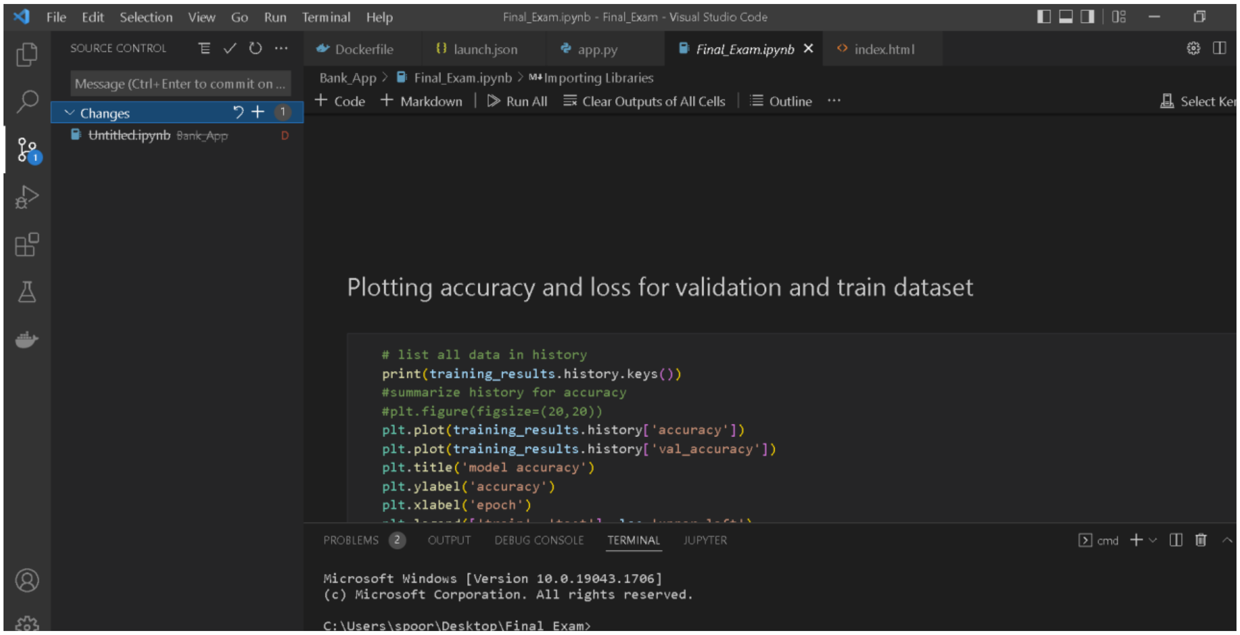
HTML for the front-end - /templates/index.html and output.html

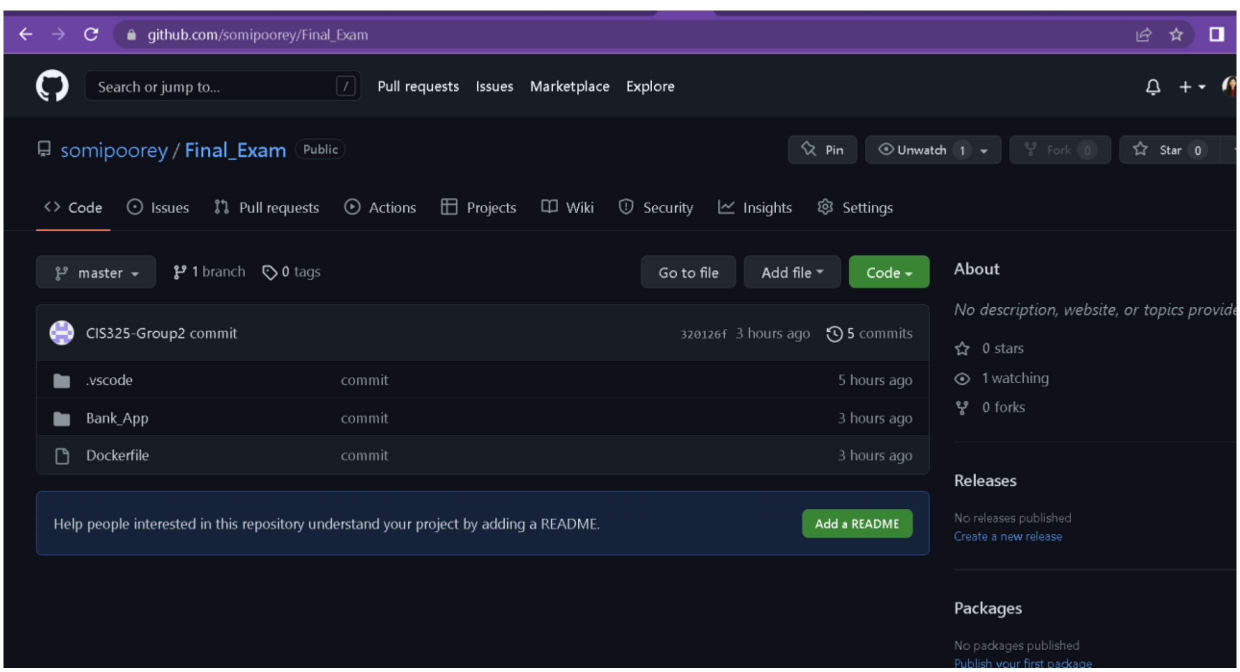
The front-end HTML acts as a medium to interact with and accept input from the user, who then receives predictions from the model. First, the POST request is received from the HTML. Then, when a request is received, the ML model is loaded, and the input file is pre-processed according to the steps described in the Flask back-end engine. Finally, the model generates the prediction, which is subsequently returned to the user via the *render_template()* function.

GITHUB & SCM

Source code management (SCM) is a technique for tracking changes to a source code repository. SCM maintains a running history of modifications to a code base and aids in dispute resolution when integrating updates from various contributors.

Visual Studio Code can be a valuable tool if you're working with a remote repository. You only need to add your remote repository to VS to be able to control the changes. Another option is to execute git commands from the terminal. Both of them can be seen in the snapshot below.



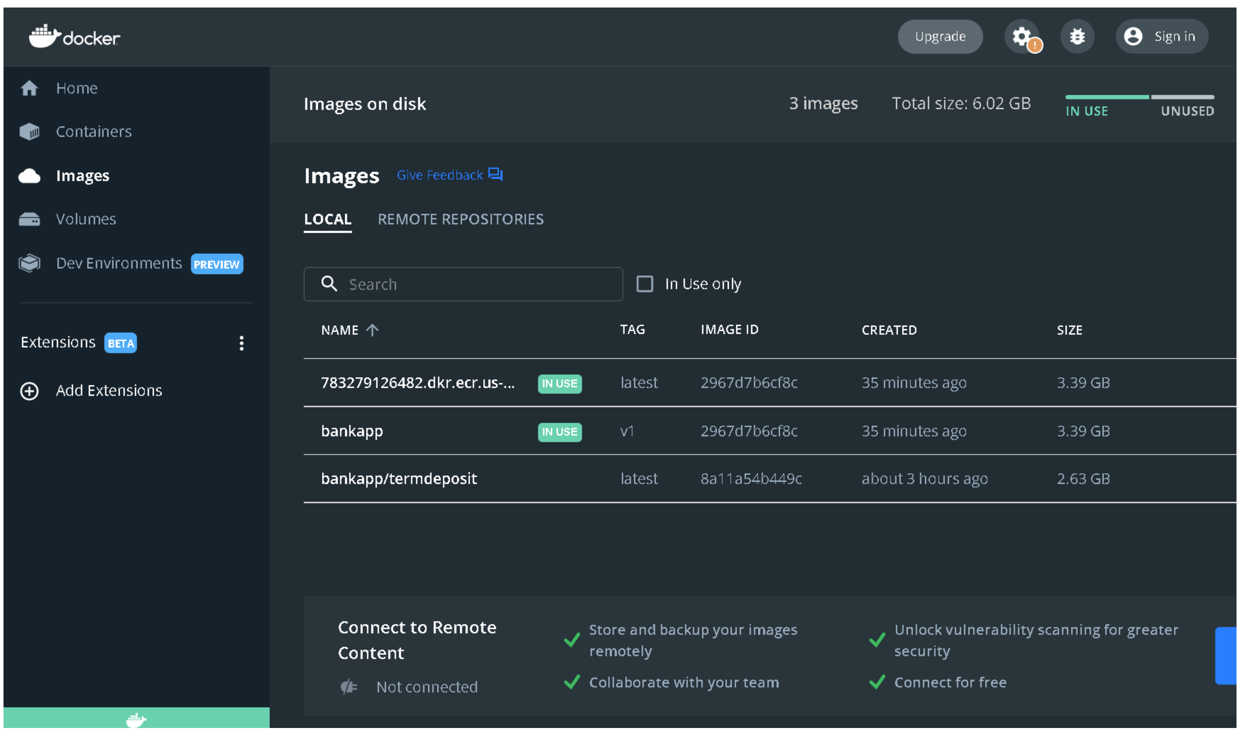


CREATING A CUSTOM IMAGE FROM DOCKER

When we want to automate and operate a custom application, the easiest option to deploy your containerized application is to create your own docker image with the desired configuration. This custom configuration may be passed using the Dockerfile, and a container can then be created using this image.

Once the image is ready, it can be used to deploy containers in Docker. In order to test our application on Docker, we have created a container using *docker run -d -p 5000:5000 --name BANK_APP <IMAGE_NAME: VERSION>* command.

The application can be seen running at *localhost:5000*



PUSHING LOCAL DOCKER IMAGE TO AWS ELASTIC CONTAINER REGISTRY ECR

To push a local docker image to AWS, we must first configure AWS CLI for the first time. This can be performed by using the instructions listed in the ReadMe file. Once the CLI is ready, we must use the terminal to create a repository and push the local docker image to ECR.

SUBMITTED BY-SOMI POOREY

https://github.com/somipoorey/Final_Exam

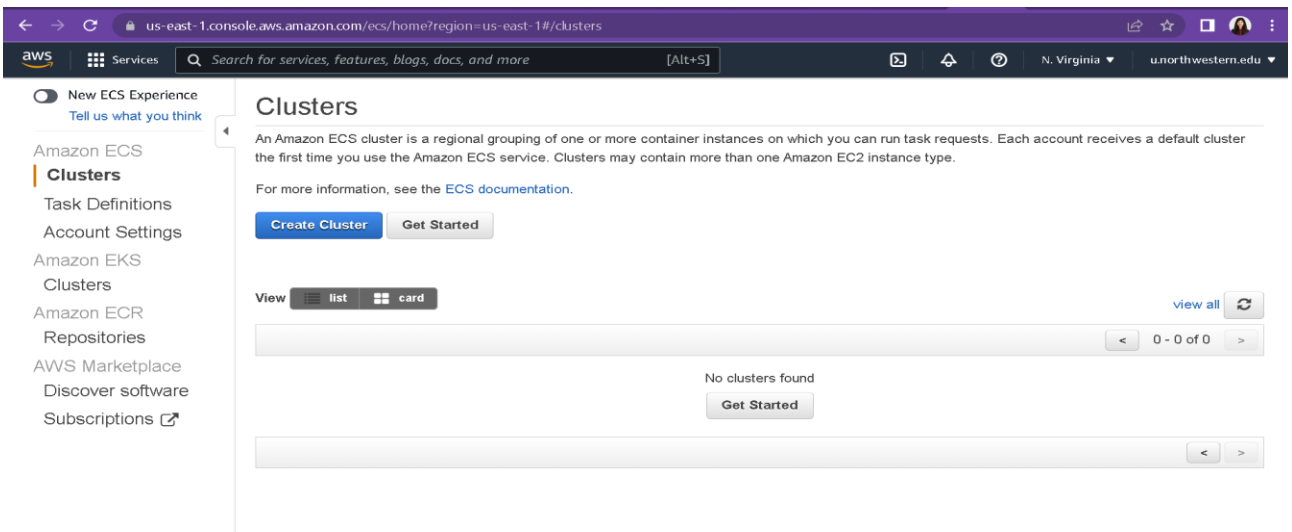
```
Command Prompt - docker push 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp
1qcXJZRkJDZS9YVVNQU2JHMGpvU2NRTERaNlE3VlRtMWk0ZkVDQ1VwdExXUlh2djNFUnpMUDM5d2dzKzVQUVp2UnR4bWpyYWFsR3B0VGJvTFZqbXBLS2trT0hZTXNIa3ZlTnFLY3d2b2lyVERTODBrOT
ZVdHp1MlhZbTB4QTllN2tiZFVFMitJQThzYnY2SExYRHVVL2JNVHBRb3NOTVg2UDg1cUNURWhaSkFXaVpjN3RrZjVVOTR0SitWWENWYnVGaU9VU24yVnhUTjdNZmNrc2lqRUZZZ1h5TTVva3VoZWxkcz
plZG11OVpZa0pUKzRLcXhuaG9IS2I0OG41RURPWmVqUHFhUnZCcDJSN2N0eVNFeTMzUzhOdGF3UzB3VGc3YnJlWEp2Y1ZVa0hJY2xrWUtnRjlySC9lb1I0a1Q2a3BONDRRcVgxL1NFRlhiaGI4emhFUk
NqZko2R3RPN1ZSc0d1bmRIdVRYdWwwSWx6Tk1HMUt1TjM4TWk1aEFubWlrcTRaVnZQZ2k3ZEZUM25iMzg3eDRvbVVSTmJWbkF1RTQ5RmdnT2h6WG53enVYdDA1OVk4SUgrckpjZmc9PSIsImRhdGFrZX
FJU0plUnRKbTVuMUc2dXFlZWtYdW9YWFBlNVVGY2U5UnE4LzE0d0FBQUg0d2ZBWUpLb1pJaHZjTkFRY0dvRzh3YlFJQkFEQm9CZ2txaGtpRzl3MEJCd0V3SGdZSllJWklBV1VEQkFFdU1CRUVESi9PWE
VJQTdQckV3N3E3ZGEwbEJOUHBMNThJMDljNkZ0L1BmZlpaT0hFVHNTUmExSG9CL2xhQUNJMnRFc21SYm5scm1WYXE2cUhuN1lYQzRPS3FML2E4PSIsInZlcnNpb24iOiIyIiwidHlwZSI6IkRBVEFfS0
I6MTY1NDYxMTM5OX0= 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

aws: error: the following arguments are required: operation

WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded

C:\Users\spoor\Desktop\Final_Exam>docker tag bankapp:v1 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp

C:\Users\spoor\Desktop\Final_Exam>docker push 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp
Using default tag: latest
The push refers to repository [783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp]
32933dc6fe61: Pushing [>                                                  ]  2.172MB/2.029GB
25830c8108ad: Pushing [>                                                  ]  2.202MB/423.1MB
7868ca7ea887: Pushing [========>                                          ]   2.87MB/17.11MB
d45f5b9da273: Pushing [==========>                                        ]  3.735MB/17.68MB
5f70bf18a086: Pushed
ca4af606d3fc: Pushing [============================================>      ]  16.85MB/18.92MB
5b02e1831b24: Waiting
b7de56037388: Preparing
e81f053ad655: Waiting
5ece9340957c: Waiting
093501b0a9e2: Preparing
b1169e57b139: Waiting
b3577d595e75: Waiting
3ee270f20d54: Waiting
4ef4adca5c3b: Waiting
```

```
Command Prompt

sage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
o see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

ws: error: the following arguments are required: operation

ARNING! Using --password via the CLI is insecure. Use --password-stdin.
ogin Succeeded

:\Users\spoor\Desktop\Final_Exam>docker tag bankapp:v1 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp

:\Users\spoor\Desktop\Final_Exam>docker push 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp
sing default tag: latest
he push refers to repository [783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp]
2933dc6fe61: Pushed
5830c8108ad: Pushed
868ca7ea887: Pushed
45f5b9da273: Pushed
f70bf18a086: Pushed
a4af606d3fc: Pushed
b02e1831b24: Pushed
7de56037388: Pushed
81f053ad655: Pushed
ece9340957c: Pushed
93501b0a9e2: Pushed
1169e57b139: Pushed
3577d595e75: Pushed
ee270f20d54: Pushed
ef4adca5c3b: Pushed
atest: digest: sha256:651332a2fb7982521b68c04211797d9e56e5e0a08785b10f1016896bdc741af5 size: 3484

:\Users\spoor\Desktop\Final_Exam>_
```

DEPLOYING ELASTIC CONTAINER SERVICE ECS USING AWS FARGATE

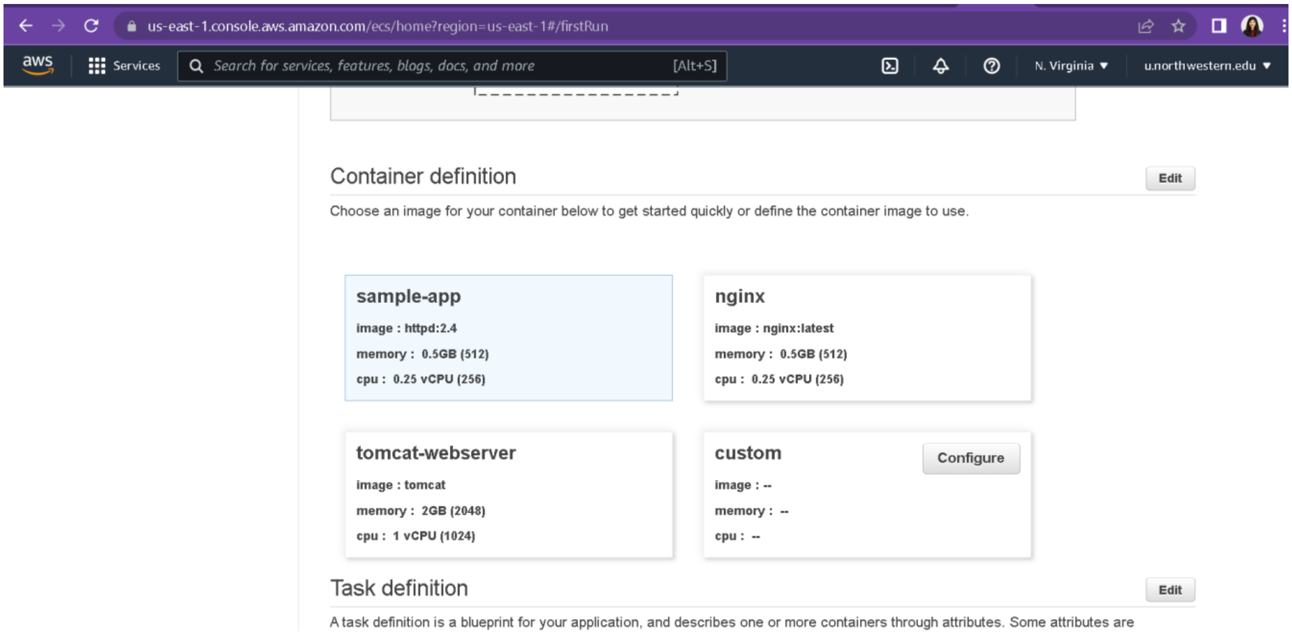
AWS FARGATE is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances. With AWS FARGATE, you no longer have to provision, configure, or scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters or optimize cluster packing.

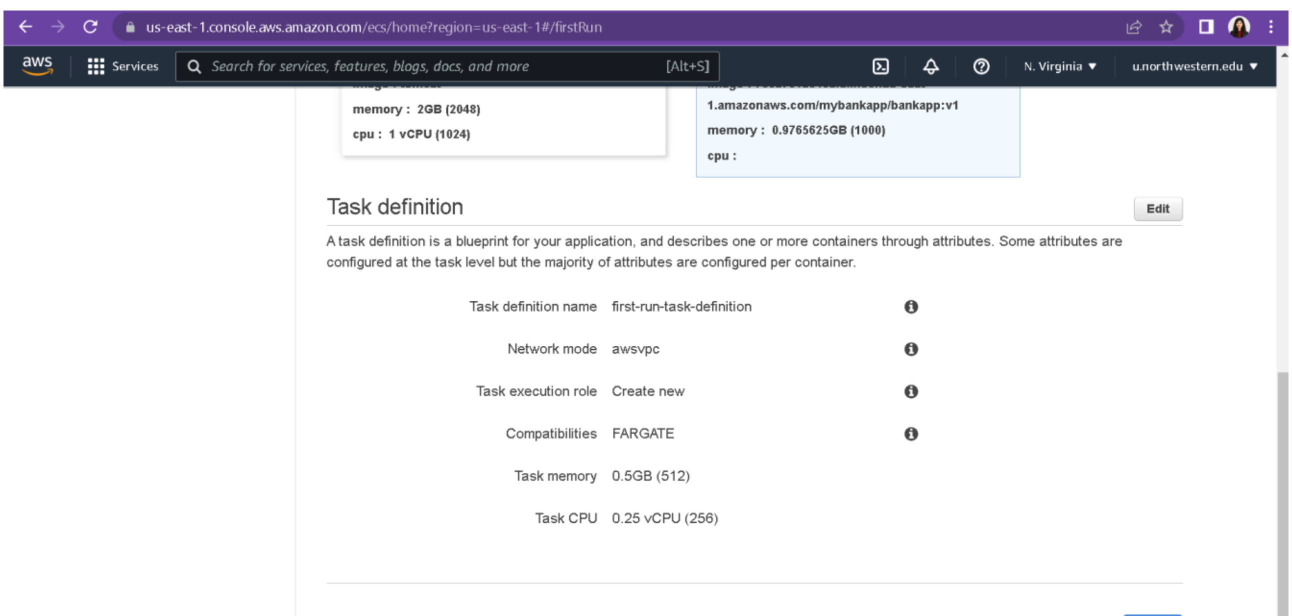
When running your tasks and services with the FARGATE launch type, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. Each FARGATE task has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.



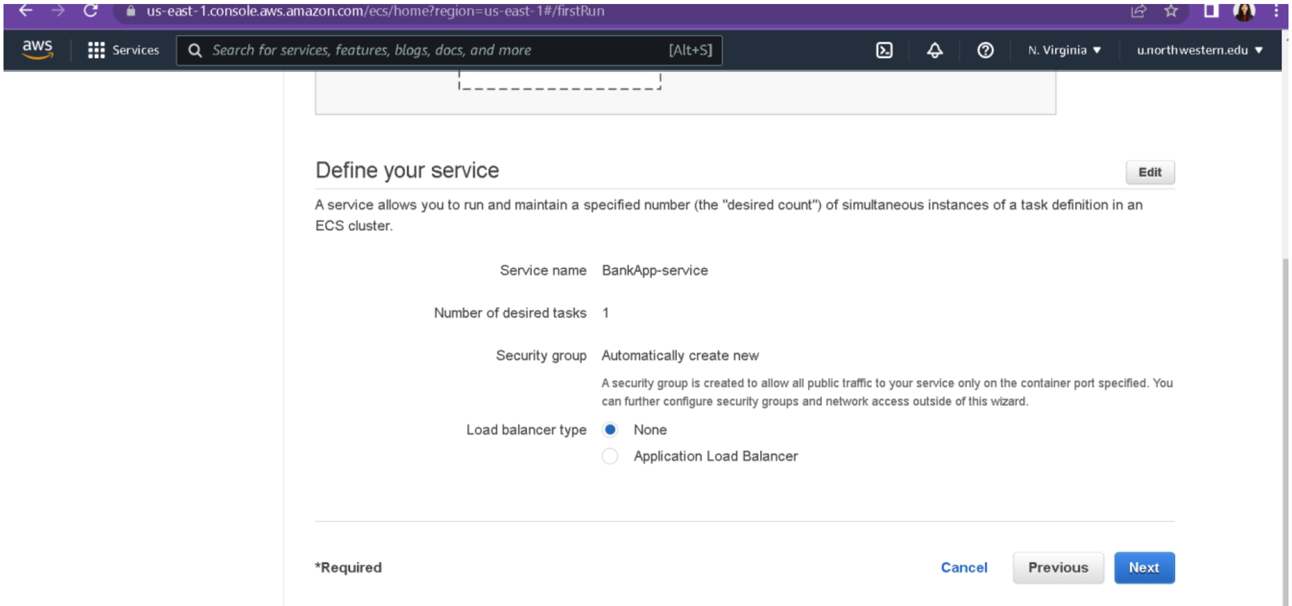
SUBMITTED BY-SOMI POOREY

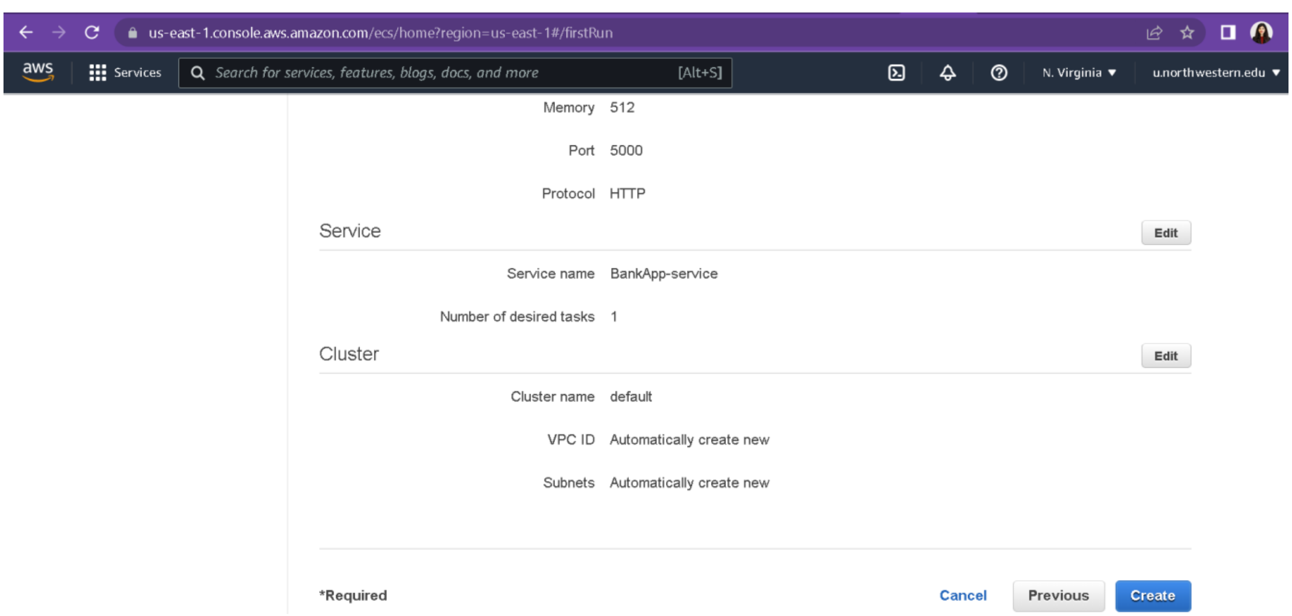
https://github.com/somipoorey/Final_Exam





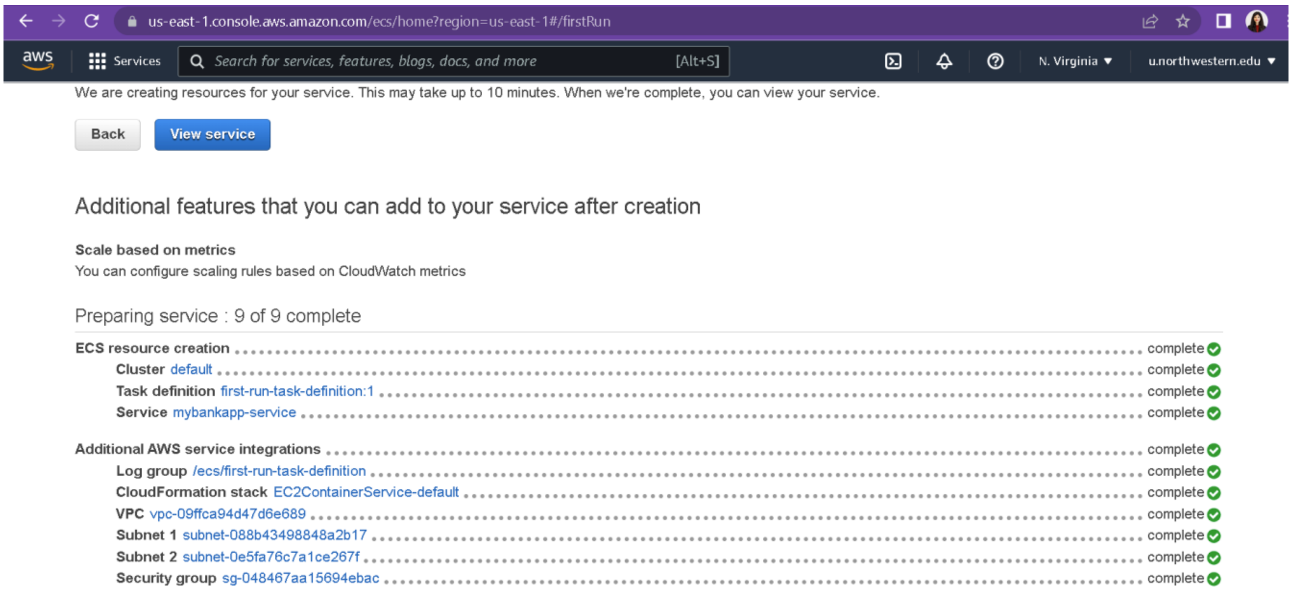
SUBMITTED BY-SOMI POOREY https://github.com/somipoorey/Final_Exam



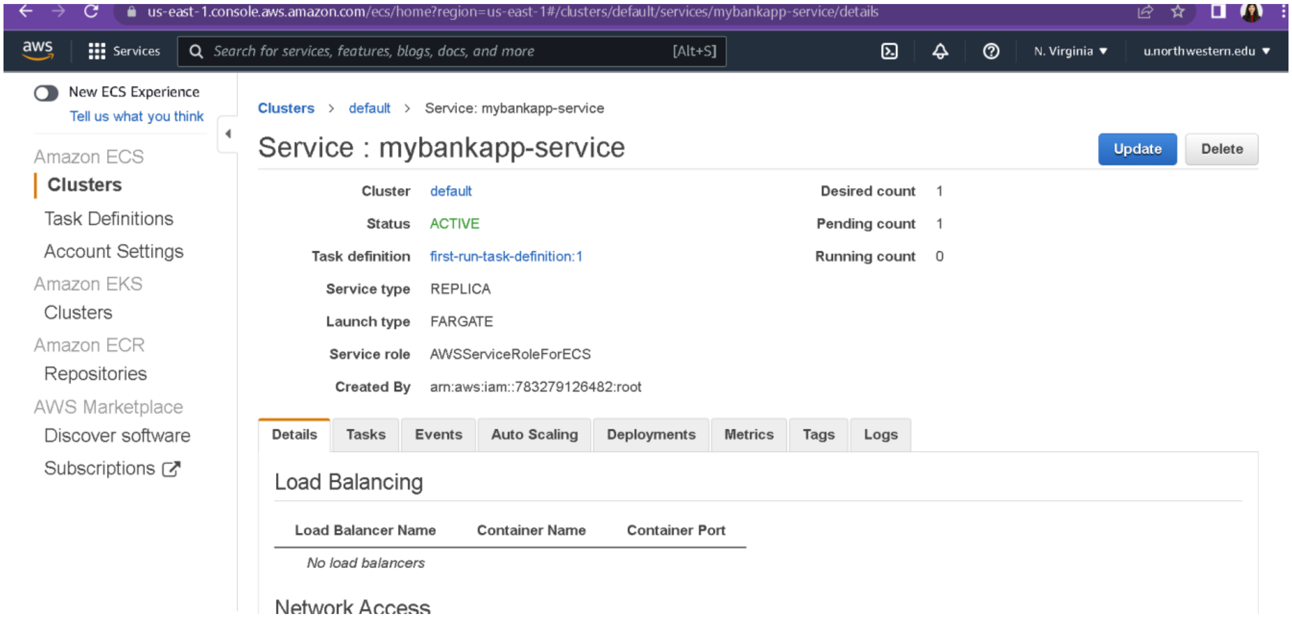


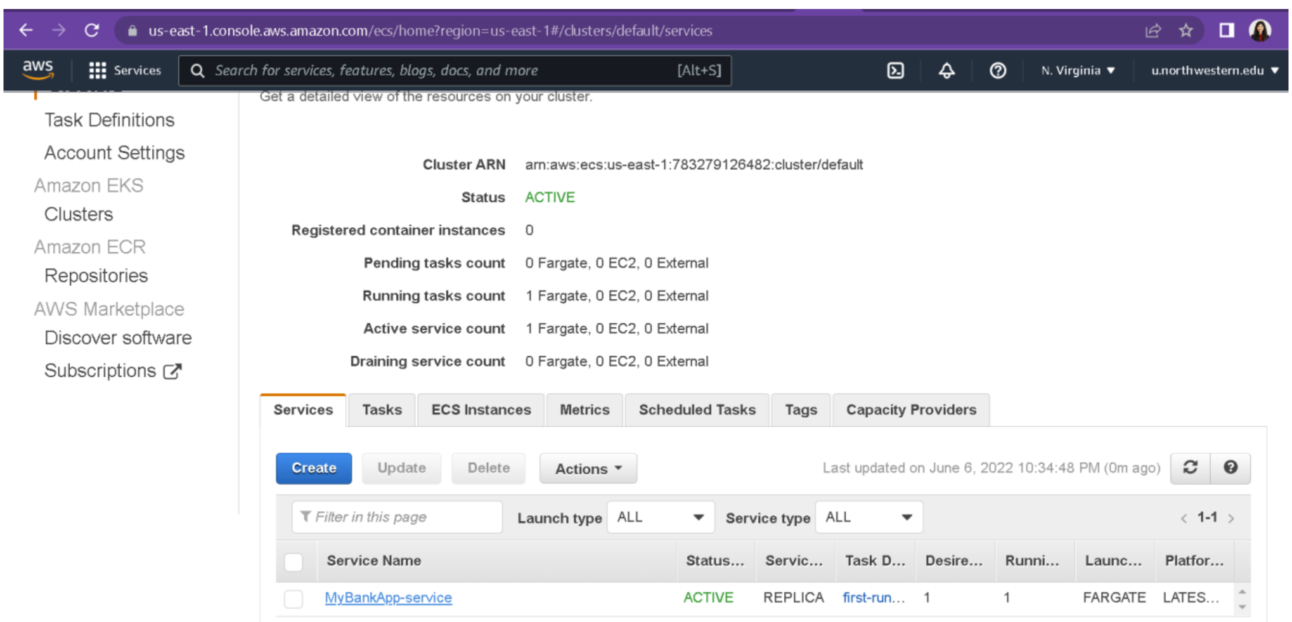
SUBMITTED BY-SOMI POOREY https://github.com/somipoorey/Final_Exam

Creating the cluster



SUBMITTED BY-SOMI POOREY https://github.com/somipoorey/Final_Exam

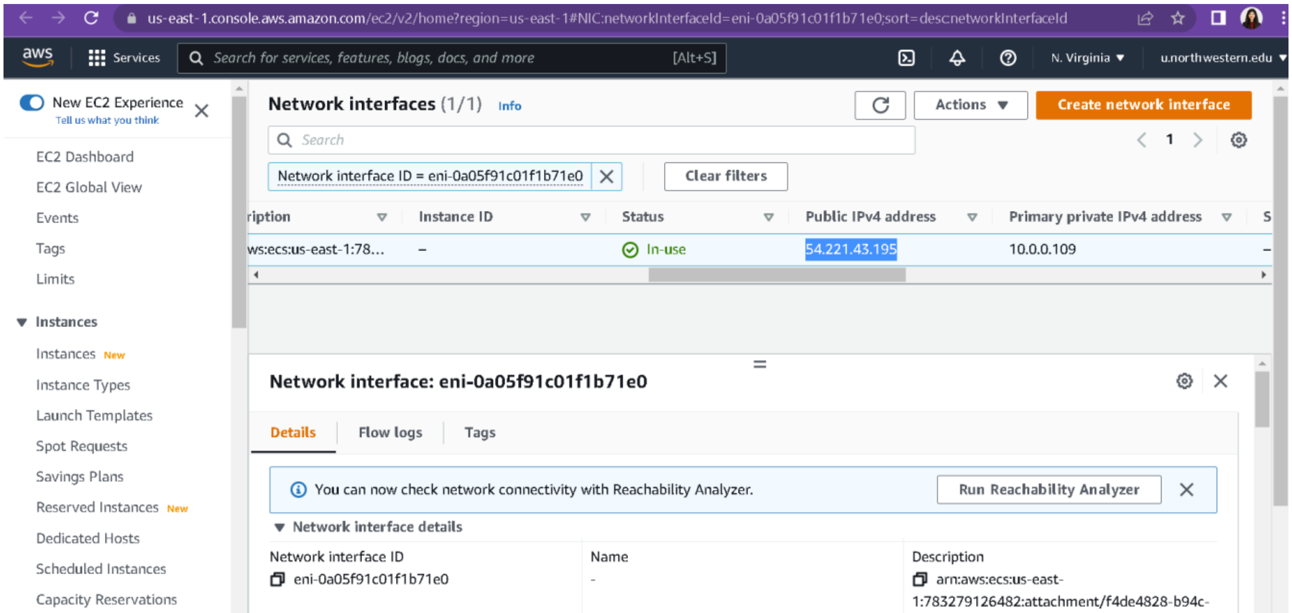




As you can see the Status is ACTIVE and there's 1 Running Task

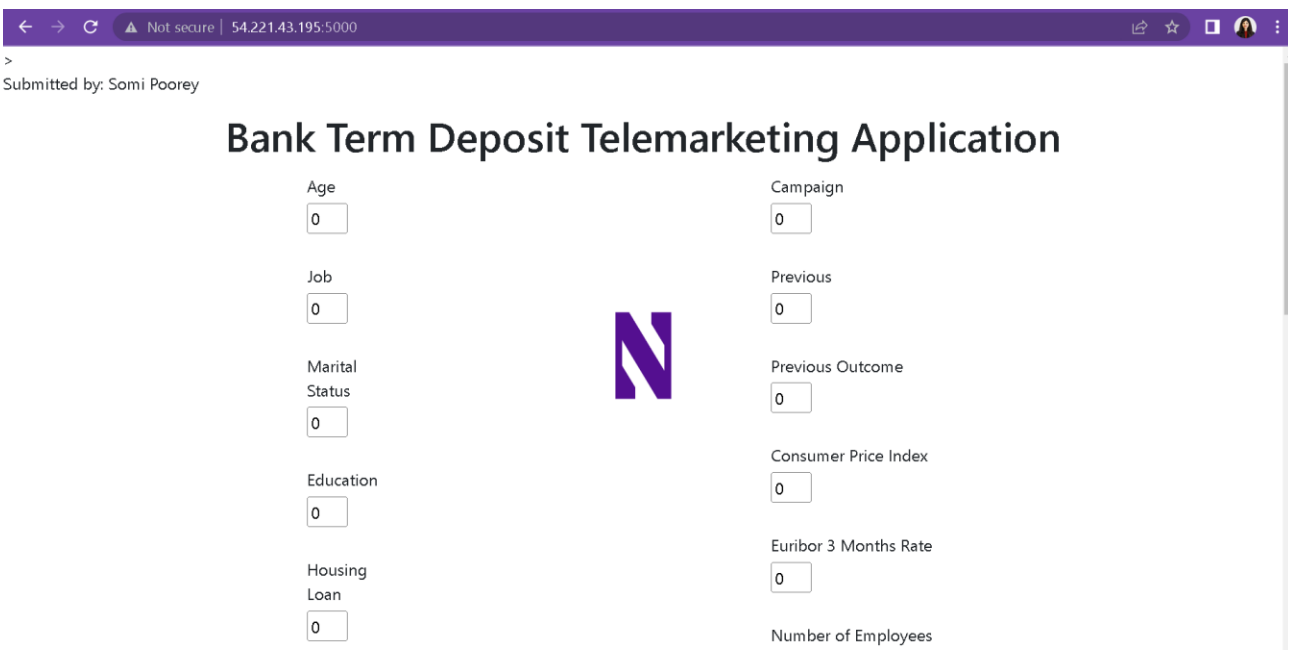
CIS-325 FINAL EXAM

SUBMITTED BY-SOMI POOREY https://github.com/somipoorey/Final_Exam



IP ADDRESS 54.221.43.195:5000

PREDICTION:



Customer Churn Prediction

Age: 0

Job: 0

Marital Status: 0

Education: 0

Housing Loan: 0

OtherLoan: 0

Default:

Contact: 2

Day Of Week: 0

Duration: 0

Campaign: 0

Previous: 0

Previous Outcome: 0

Consumer Price Index: 0.0

Euribor 3 Months Rate: 0.0

Number of Employees: 0

Passed Days: 0

Employment Variation Rate: 0.0

Consumer Confidence Index: 0.0

Month: 0

The customer will

NOT SUBSCRIBE TO A TERM DEPOSIT

Back

Customer Churn Prediction

Age: 40

Job: 0

Marital Status: 0

Education: 0

Housing Loan: 0

OtherLoan: 2

Default:

Contact: 0

Day Of Week: 0

Duration: 0

Campaign: 1

Previous: 0

Previous Outcome: 0

Consumer Price Index: 0.0

Euribor 3 Months Rate: 0.0

Number of Employees: 0

Passed Days: 0

Employment Variation Rate: 0.0

Consumer Confidence Index: 0.0

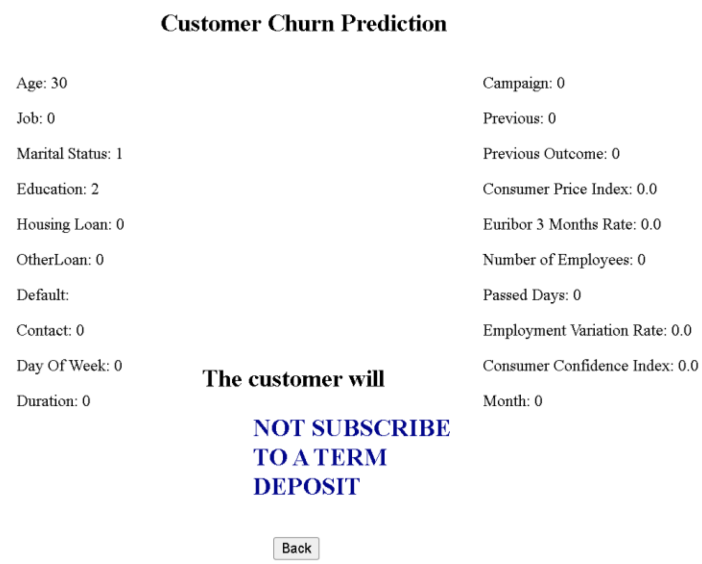
Month: 0

The customer will

NOT SUBSCRIBE TO A TERM DEPOSIT

Back

https://github.com/somipoorey/Final_Exam



I TRIED WITH DIFFERENT PARAMETERS LIKE DIFFERENT AGE, LOAN, EDUCATION, MARITAL STATUS BUT I GOT THE CUSTOMER WILL NOT SUBSCIBE FOR ALL THE ABOVE