

BANK

Telemarketing Campaign analysis



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.

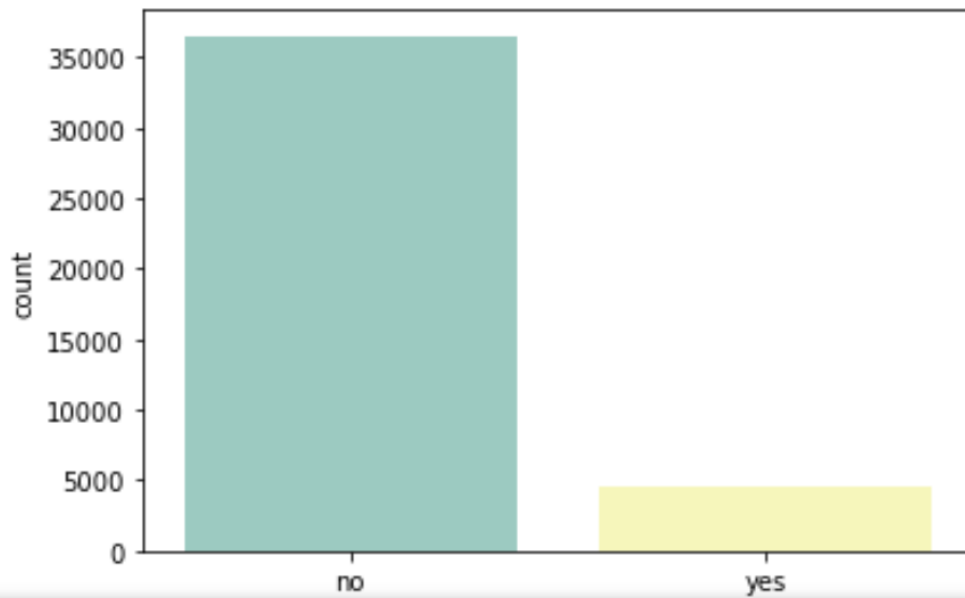
GOAL: The classification goal is to predict if the client will subscribe a term deposit (target variable y).

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.

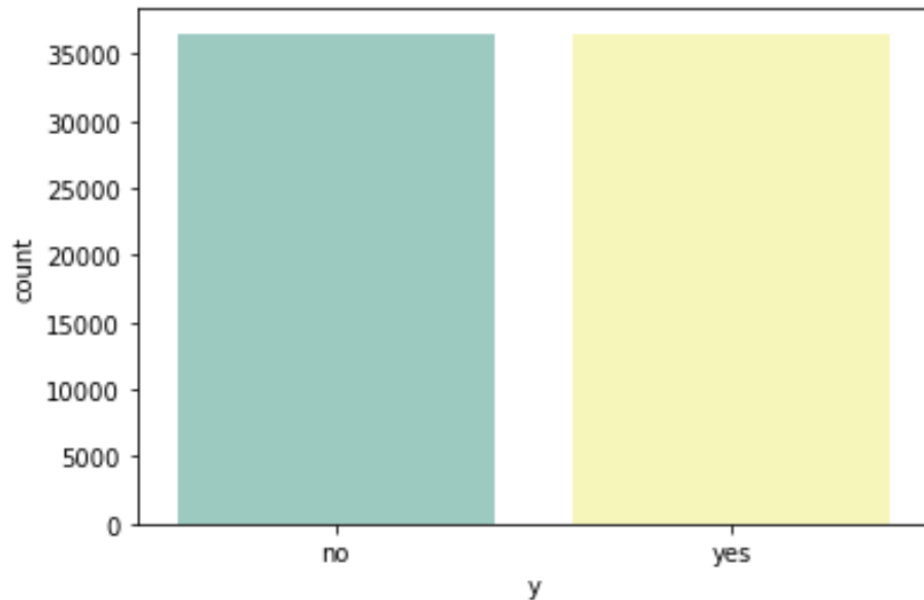
HIGHLIGHTS:

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 is 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:

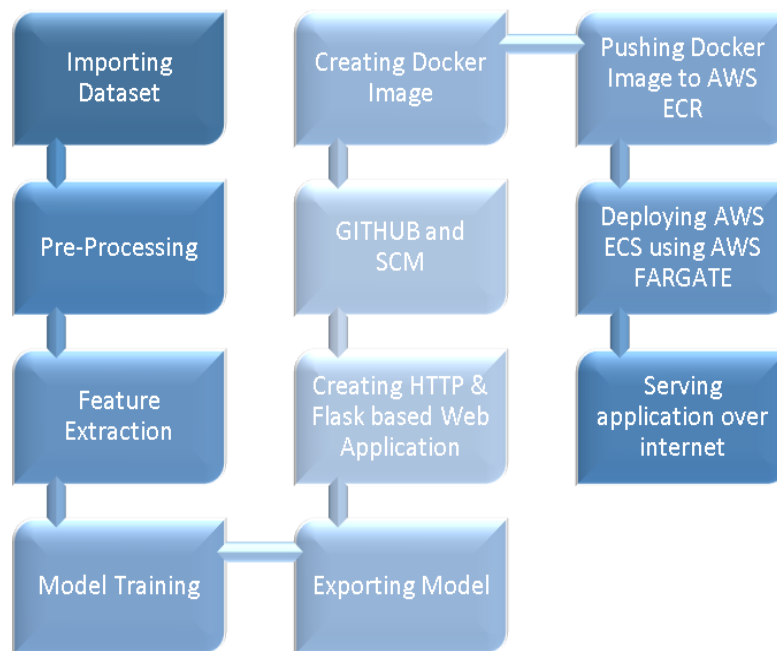
Numeric data type: banking dataset: salary, balance, duration and age.

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

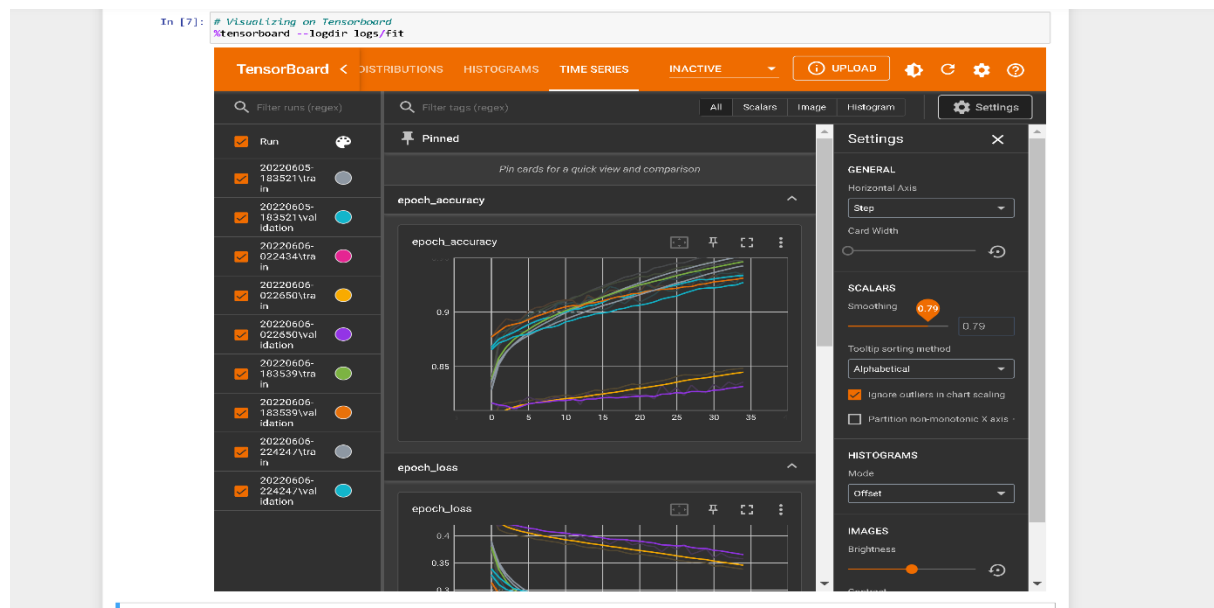
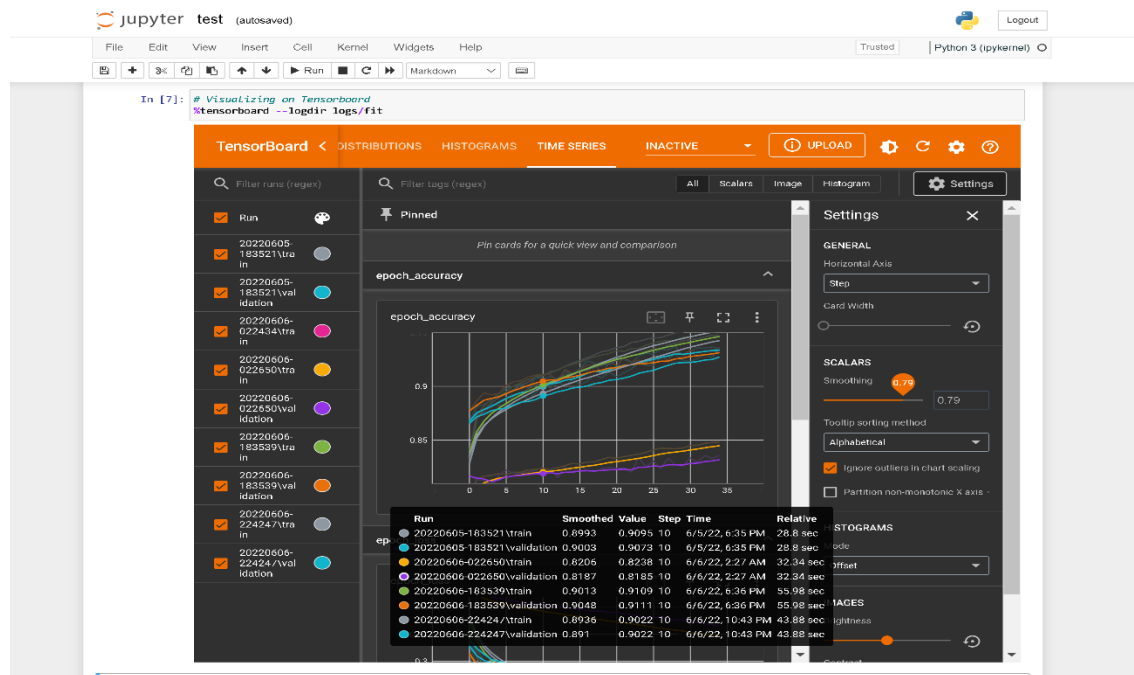
**DEPENDENCIES:**

- 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

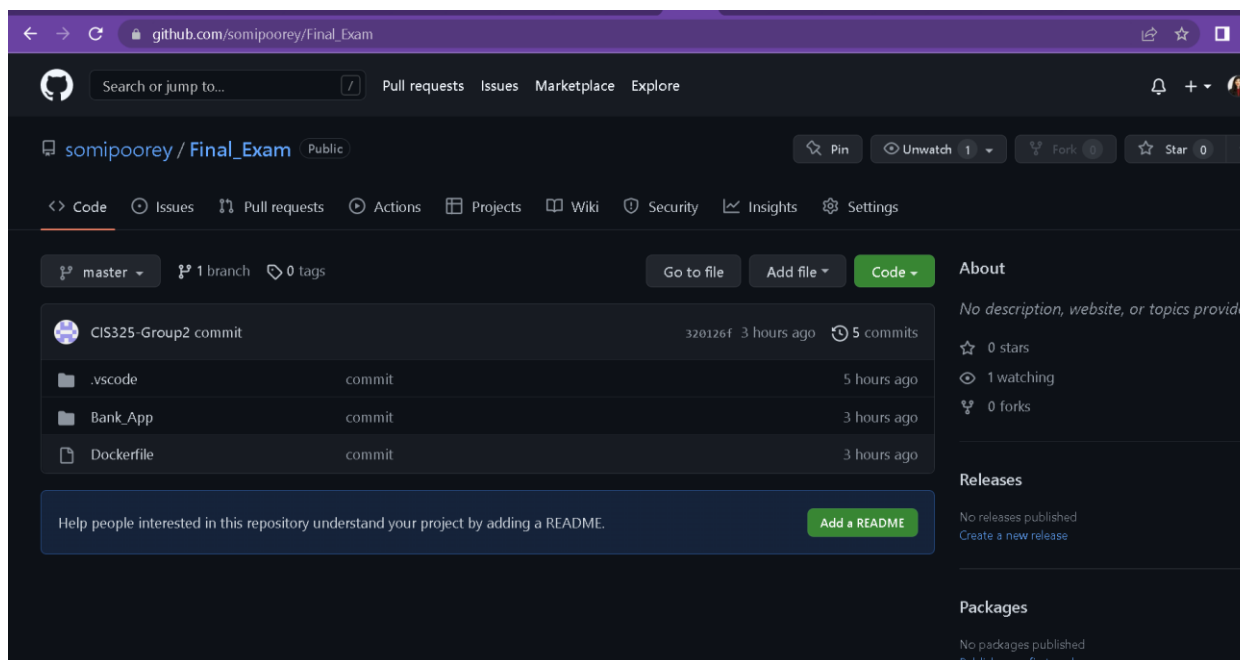
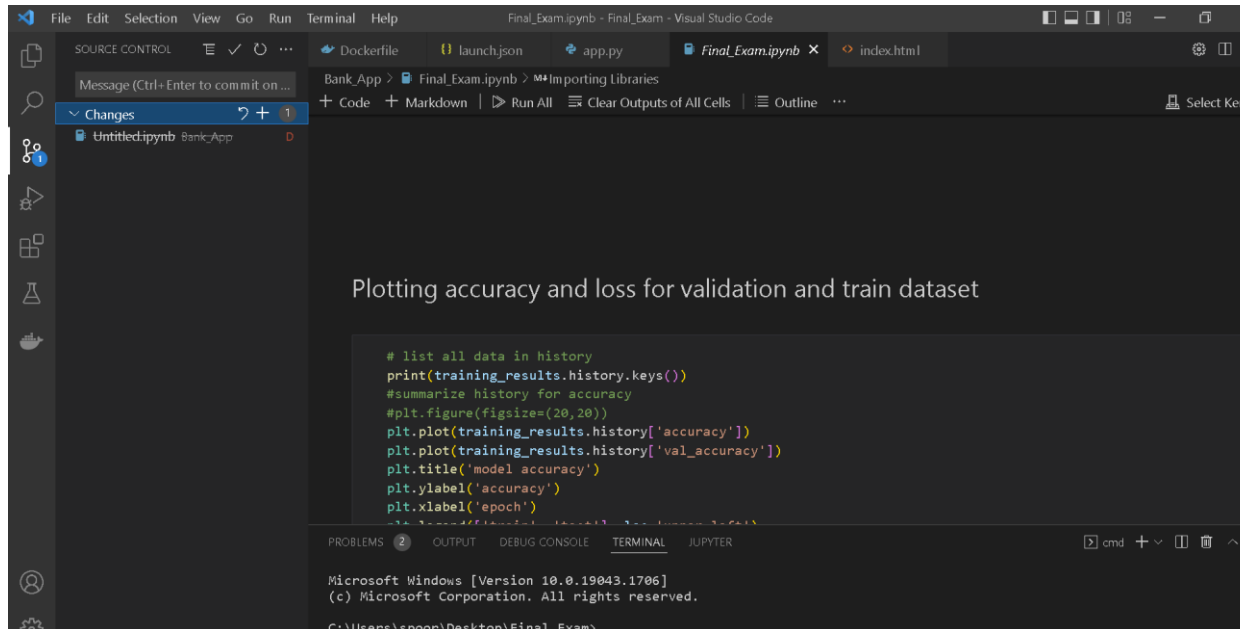
The front-end HTML acts as a medium to interact with and accept input from the user, who then receives predictions from the model. First, the POST request is received from the HTML. Then, when a request is received, the ML model is loaded, and the input file is pre-processed according to the steps described 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.

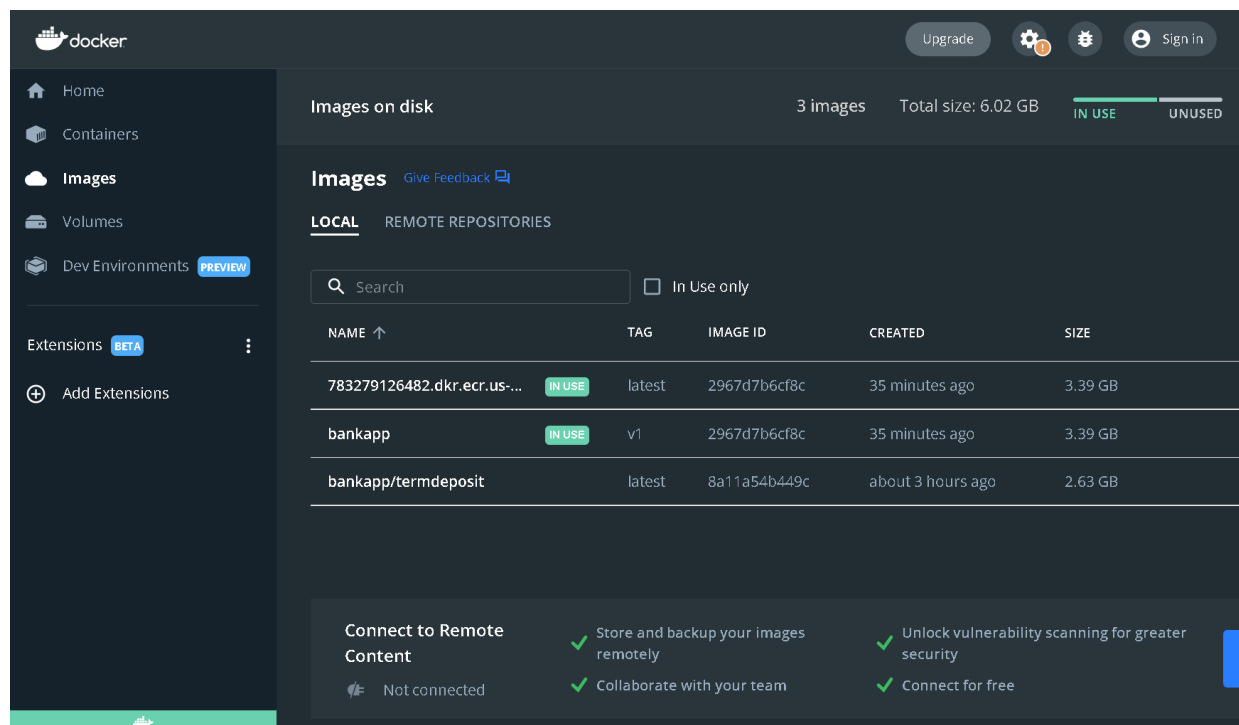


CREATING A CUSTOM IMAGE FROM DOCKER

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.


```

C:\Command Prompt - docker push 783279126482.dkr.ecr.us-east-1.amazonaws.com/mybankapp
1qcx0ZRIjDZS9VAAWQU2JHqspVU2NRTeraH1E3V1R4tWk8ZiVDOIWdExXUlh2dJnFuphJlDh5d2d2KzVQUVp2UnR4bWpyVMF8R3B0VG3vTFZqbBLS2trT0hZTXN1a3Z1TnFLY3d2b2lyVERTODBrOT
ZVdHpi1K1hZBT84QT11N12tiZFEVMI4t3QTh2YnY2SEYVhAVL2JHWHBRB3MOTVg2Ug1cUNURWhaSkFXaVpJh3RrZjVvOTR0Si4tWENhVynVgaU9AU24yVnhUTjdNZmNre21aRUZZZ1hSTTVva3VozWxkcz
p1ZG110VpZa0pUKzRLcXhuaG9IS2I0G41RURPWWVqUHFhUnZCcDJSN2N0eVFeTKzUzhodGF3UzB3Vgc3YnJ1NEp2Y1ZVa0hJY2xrWUtnRjlySC91b1I0a1Q2a3B0NDRRCVgxL1NFR1hiaG14emhFUK
Nq2ko2R3RPN1ZSc8d1bmrIdVRYdKwmsSxw6TK1HwUt1TjM4TWk1aEFubW1rcTraVnZQZ2k3ZEZUM25iMzg3eDRvbVvSTm3WbkF1RTQ5RmdnT2h6WGS3enVYdA1OVk4SUgrcKpJZmc9PSIsImRhdfFrZX
FJU0p1UnRkbTVuMjC2dXFIzWtYdW9VWFB1NVVGY2U5UnE4LzE0d0FBQJg0d2ZBwUpLb1pJaHZjTkFRY0dvRzh3Y1F7QkFEQm9CZ2tXaGtpRz13WEJCd0V3SGdZS11JWk1BV1VEQkFFdU1CRUVESi9PWE
VJQTdQckV3N3E3ZGEwbeJOUHBNWThJMD1jNkZ0L1BmZ1paT0HfVhNTUmExS69CL2xhQUNJmRfC21SYm5scm1WYXE2cUhuN1lYQzRPS3FmL2E4PSIsInZ1cnNpb24iOiIyIiwidHlwZSI6IHRBVEFfFS0
I6MTY1NDYxMTY5MT50X0E= 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
445f5b9da273: 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

```

```

C:\Command Prompt

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: Pushed
25830c8108ad: Pushed
7868ca7ea887: Pushed
445f5b9da273: Pushed
5f70bf18a086: Pushed
ca4af606d3fc: Pushed
5b02e1831b24: Pushed
b7de56037388: Pushed
e81f053ad655: Pushed
5ece9340957c: Pushed
093501b0a9e2: Pushed
b1169e57b139: Pushed
b3577d595e75: Pushed
3ee270f20d54: Pushed
4ef4adca5c3b: Pushed
latest: digest: sha256:651332a2fb7982521b68c04211797d9e56e5e0a08785b10f1016896bdc741af5 size: 3484

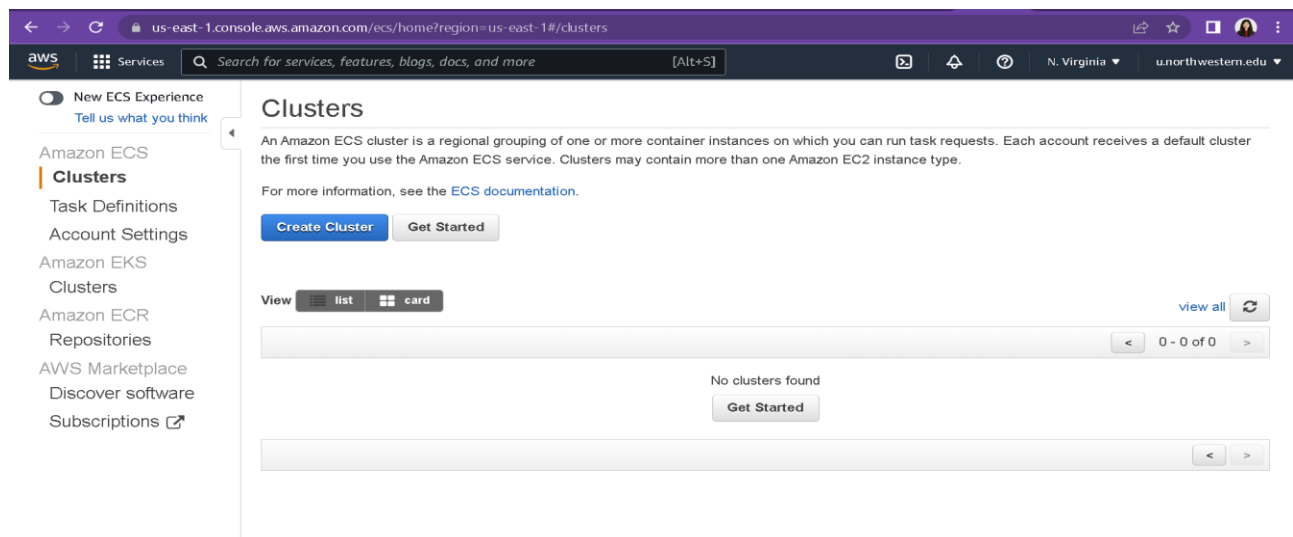
C:\Users\spoor\Desktop\Final_Exam>_

```

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.



The screenshot displays the AWS Management Console interface for configuring an ECS task. The browser address bar shows the URL: `us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/firstRun`. The console header includes the AWS logo, a 'Services' menu, a search bar, and navigation links for 'N. Virginia' and 'u.northwestern.edu'.

Container definition

Choose an image for your container below to get started quickly or define the container image to use.

Four container image options are presented:

- sample-app**: image : httpd:2.4, memory : 0.5GB (512), cpu : 0.25 vCPU (256)
- nginx**: image : nginx:latest, memory : 0.5GB (512), cpu : 0.25 vCPU (256)
- tomcat-webserver**: image : tomcat, memory : 2GB (2048), cpu : 1 vCPU (1024)
- custom**: image : --, memory : --, cpu : -- (with a 'Configure' button)

Task definition

A task definition is a blueprint for your application, and describes one or more containers through attributes. Some attributes are

The 'Task definition' section shows a configuration table:

Task definition name	first-run-task-definition	?
Network mode	awsvpc	?
Task execution role	Create new	?
Compatibilities	FARGATE	?
Task memory	0.5GB (512)	
Task CPU	0.25 vCPU (256)	

The screenshot shows the 'Define your service' step in the AWS ECS console. The browser address bar indicates the URL is `us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/firstRun`. The AWS logo and 'Services' link are in the top left. A search bar contains the text 'Search for services, features, blogs, docs, and more'. The top right shows the region 'N. Virginia' and the user 'u.northwestern.edu'. The main content area has a title 'Define your service' with an 'Edit' button. Below the title is a description: 'A service allows you to run and maintain a specified number (the "desired count") of simultaneous instances of a task definition in an ECS cluster.' The configuration fields are: 'Service name' set to 'BankApp-service', 'Number of desired tasks' set to '1', 'Security group' set to 'Automatically create new' (with a note: 'A security group is created to allow all public traffic to your service only on the container port specified. You can further configure security groups and network access outside of this wizard.'), and 'Load balancer type' with 'None' selected (radio button) and 'Application Load Balancer' as an option. At the bottom, there is a '*Required' label and three buttons: 'Cancel', 'Previous', and 'Next'.

Define your service Edit

A service allows you to run and maintain a specified number (the "desired count") of simultaneous instances of a task definition in an ECS cluster.

Service name BankApp-service

Number of desired tasks 1

Security group Automatically create new

A security group is created to allow all public traffic to your service only on the container port specified. You can further configure security groups and network access outside of this wizard.

Load balancer type ☒ None ☐ Application Load Balancer

*Required Cancel Previous Next

The screenshot shows the 'Cluster' configuration step in the AWS ECS console. The browser address bar is the same as the previous screenshot. The main content area has a title 'Cluster' with an 'Edit' button. Below the title are the configuration fields: 'Cluster name' set to 'default', 'VPC ID' set to 'Automatically create new', and 'Subnets' set to 'Automatically create new'. At the bottom, there is a '*Required' label and three buttons: 'Cancel', 'Previous', and 'Create'.

Cluster Edit

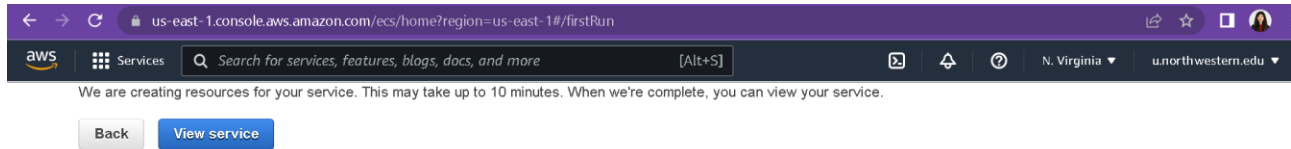
Cluster name default

VPC ID Automatically create new

Subnets Automatically create new

*Required Cancel Previous Create

Creating the cluster



Additional features that you can add to your service after creation

Scale based on metrics

You can configure scaling rules based on CloudWatch metrics

Preparing service : 9 of 9 complete

ECS resource creation	complete	✓
Cluster default	complete	✓
Task definition first-run-task-definition:1	complete	✓
Service mybankapp-service	complete	✓
Additional AWS service integrations	complete	✓
Log group /ecs/first-run-task-definition	complete	✓
CloudFormation stack EC2ContainerService-default	complete	✓
VPC vpc-09ffca94d47d6e689	complete	✓
Subnet 1 subnet-088b43498848a2b17	complete	✓
Subnet 2 subnet-0e5fa76c7a1ce267f	complete	✓
Security group sg-048467aa15694ebac	complete	✓

us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/default/services/mybankapp-service/details

Services Search for services, features, blogs, docs, and more [Alt+S]

New ECS Experience Tell us what you think

Amazon ECS

- Clusters
- Task Definitions
- Account Settings

Amazon EKS

- Clusters

Amazon ECR

- Repositories

AWS Marketplace

- Discover software
- Subscriptions

Clusters > default > Service: mybankapp-service

Service : mybankapp-service

Update Delete

Cluster default Desired count 1

Status ACTIVE Pending count 1

Task definition first-run-task-definition:1 Running count 0

Service type REPLICA

Launch type FARGATE

Service role AWSServiceRoleForECS

Created By arn:aws:iam::783279126482:root

Details Tasks Events Auto Scaling Deployments Metrics Tags Logs

Load Balancing

Load Balancer Name	Container Name	Container Port
No load balancers		

Network Access

us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/default/services

Services Search for services, features, blogs, docs, and more [Alt+S]

Get a detailed view of the resources on your cluster.

Cluster ARN arn:aws:ecs:us-east-1:783279126482:cluster/default

Status ACTIVE

Registered container instances 0

Pending tasks count 0 Fargate, 0 EC2, 0 External

Running tasks count 1 Fargate, 0 EC2, 0 External

Active service count 1 Fargate, 0 EC2, 0 External

Draining service count 0 Fargate, 0 EC2, 0 External

Services Tasks ECS Instances Metrics Scheduled Tasks Tags Capacity Providers

Create Update Delete Actions Last updated on June 6, 2022 10:34:48 PM (0m ago)

Filter in this page Launch type ALL Service type ALL

Service Name	Status...	Servic...	Task D...	Desire...	Runni...	Launc...	Platfor...
MyBankApp-service	ACTIVE	REPLICA	first-run...	1	1	FARGATE	LATES...

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

The screenshot shows the AWS Management Console for the us-east-1 region. The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, and Instances. The main content area displays 'Network interfaces (1/1)' with a table listing one interface: 'eni-0a05f91c01f1b71e0' with status 'In-use' and public IP address '54.221.43.195'. Below the table, the 'Details' tab for this interface is shown, including a 'Run Reachability Analyzer' button and a table with details like 'Network interface ID', 'Name', and 'Description'.

Description	Instance ID	Status	Public IPv4 address	Primary private IPv4 address
aws:ec2:us-east-1:78...	-	In-use	54.221.43.195	10.0.0.109

Network interface ID	Name	Description
eni-0a05f91c01f1b71e0	-	arn:aws:ec2:us-east-1:783279126482:attachment/f4de4828-b94c-...

IP ADDRESS 54.221.43.195:5000

PREDICTION:

The screenshot shows a web application titled 'Bank Term Deposit Telemarketing Application'. It features a large purple 'N' logo in the center. The application has two columns of input fields for user information. The left column includes fields for Age, Job, Marital Status, Education, and Housing Loan. The right column includes fields for Campaign, Previous, Previous Outcome, Consumer Price Index, Euribor 3 Months Rate, and Number of Employees. All input fields are currently empty.

Age	Campaign
0	0

Job	Previous
0	0

Marital Status	Previous Outcome
0	0

Education	Consumer Price Index
0	0

Housing Loan	Euribor 3 Months Rate
0	0

Number of Employees

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

Customer Churn Prediction

Age: 30

Job: 0

Marital Status: 1

Education: 2

Housing Loan: 0

OtherLoan: 0

Default:

Contact: 0

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**

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