CIS 325 Final Project
Submitted by: Anurag Yadav | Abhishek Mote

GitHub
[Click to Access Repo]

Human Gender Detection from Facial Images Using Convolution Neural Network

Introduction

Human gender detection, a part of facial recognition, has received extensive attention because of its different applications. However, previous research works on gender detection have been accomplished based on various static body features, for example, face, eyebrow, hand shape, body shape, fingernail, etc. In this project, we have built a human gender classification model using the Convolution Neural Network (CNN) by implementing it on the UTKFace dataset. The application has been created and deployed as a web service and is accessible over the internet for use.

To implement our system, a pre-processing technique was first applied to each image using image processing. Then, the pre-processed image is passed through the Convolution, RELU, and Pooling layer for feature extraction. Finally, a fully connected layer and a classifier are applied in the classification part of the image. The entire project was trained & evaluated on the UTKFace dataset. The result shows the highest accuracy of 89.27%.

Methodology

We have utilized a CNN (Convolutional Neural Network) architecture in our gender detection model. CNN, which is a deep learning algorithm, is capable of distinguishing images from their characteristics. CNN is generally used for image analysis, segmentation, classification, medical image analysis, photo and video recognition, etc. In this project, we first applied an image processing technique as pre-processing on images to transform the raw data into an efficient and useful format. Later, the CNN architecture was applied. Here, it has been decomposed into two parts:

- Feature Extraction
- Classification

The Convolution and the pooling layers perform the feature extraction of images that extract information from the input for decision making. Finally, a fully connected layer serves as the classification part.

Dataset

The UTKFace dataset is a well-known large-scale face dataset with a vast age range (from 0 to 116 years old). The dataset contains approximately 20,000 face images with age,

[Click to Access Repo]

gender, and ethnicity annotations. In addition, the photos cover a wide range of poses, facial expressions, lighting, occlusion, resolution, etc.

[Dataset Source: https://susanqq.github.io/UTKFace/]

The following code line was used to extract the.tar images dataset:

```
import os
import tarfile

fname = "crop_part1.tar.gz"

cwd=os.getcwd().replace("\\","/")

if fname.endswith("tar.gz"):

    tar = tarfile.open(fname, "r:gz")
    tar.extractall(cwd+"/Data/")
    tar.close()
```

Pre-Processing

Pre-processing of the image generally eliminates low-frequency background noise, normalizes the intensification of the individual practical image, removes reflection of light to get rid of the image noise, and prepares the face image for better feature extraction. We have first resized the images into 96 × 96 dimensions in our system. Then We transformed the image into an array of pixel values. Finally, each pixel value of the array is converted to float and divided by the value 255.0 so that all the resulting pixel values range between 0 and 1.

Feature Extraction

In Convolutional Neural Network (CNN), the Convolution and the Pooling layer perform the feature extraction. In our model, these layers are defined as follows:

I. The convolution layer contains 32 filters with a 3 × 3 kernel. Here RELU is used as the activation function followed by batch normalization.

II. The POOL layer uses a 3 × 3 pool size to reduce spatial dimension from 96 × 96 to 32 × 32. A dropout with 0.25 is used in our network architecture which disconnects nodes arbitrarily from layer to layer.

III. Next, the convolution and ReLU layers are applied twice before applying another POOL layer. This operation of multiple convolutional and ReLU layers allows the model to learn a richer set of features. Here-
    - The filter size is being increased from 32 to 64.
    - The max-pooling size is decreased from 3 × 3 to 2 × 2, so spatial dimensions don't get reduced too quickly. Dropout is again performed at this stage.

IV. Again, the convolution and ReLU layers are applied twice before applying another POOL layer. The filter size is increased to 128. And 25% dropout of the nodes is executed in this step to reduce overfitting.

Classification

Fully Connected and RELU operation is performed, and a sigmoid classifier is used for classification. Here:

V. RELU and batch normalization with dense (1024) defines the fully connected layer where dropout is executed for the last time. This time 50% of the node is being dropped during training.

VI. Finally, the sigmoid function is used as a classifier to return the predicted probabilities for each class label.

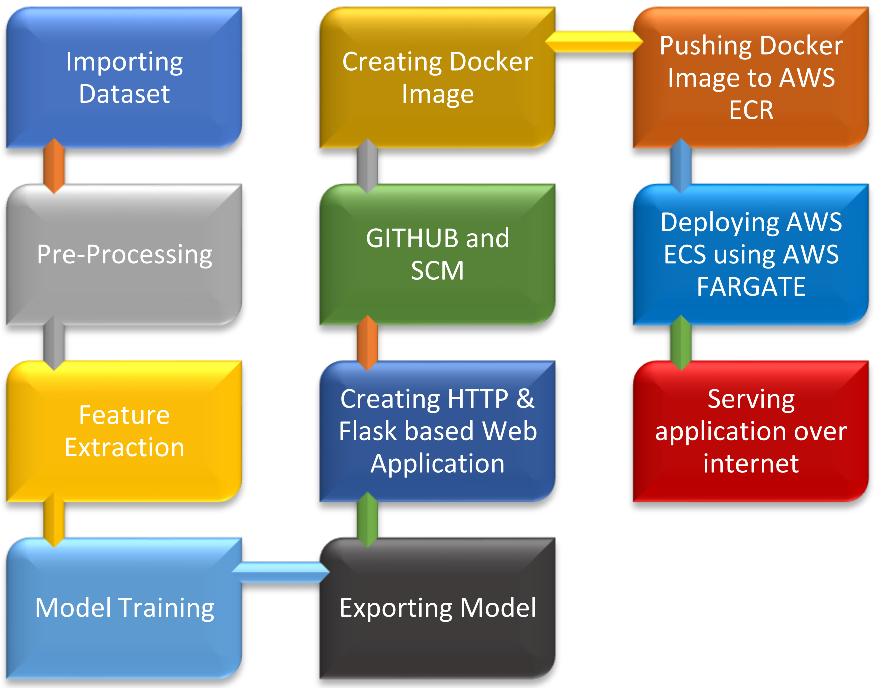
CIS 325 Final Project

Submitted by: Anurag Yadav | Abhishek Mote

GitHub

[Click to Access Repo]

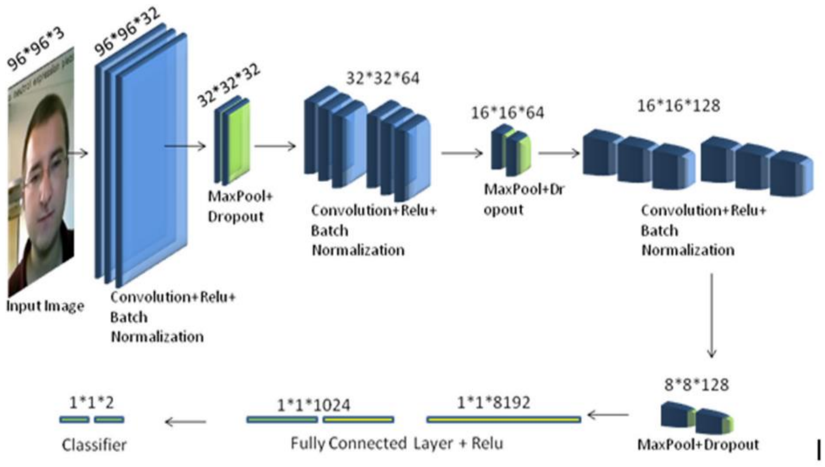
Project Workflow



Importing Dataset
Creating Docker Image
Pushing Docker Image to AWS ECR
Pre-Processing
GITHUB and SCM
Deploying AWS ECS using AWS FARGATE
Feature Extraction
Creating HTTP & Flask based Web Application
Serving application over internet
Model Training
Exporting Model

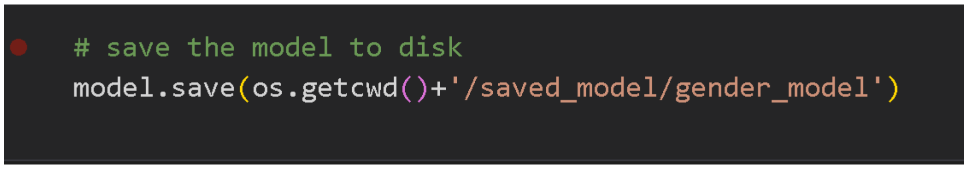
TRAINING MODEL

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.

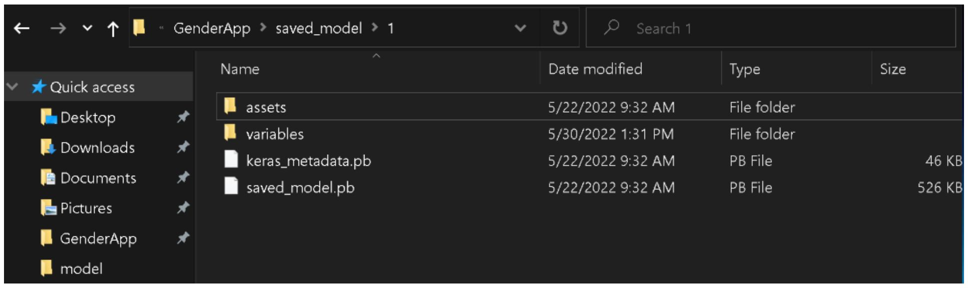


EXPORTING MODEL

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.



```
# save the model to disk
model.save(os.getcwd()+'/saved_model/gender_model')
```



GenderApp › saved_model › 1

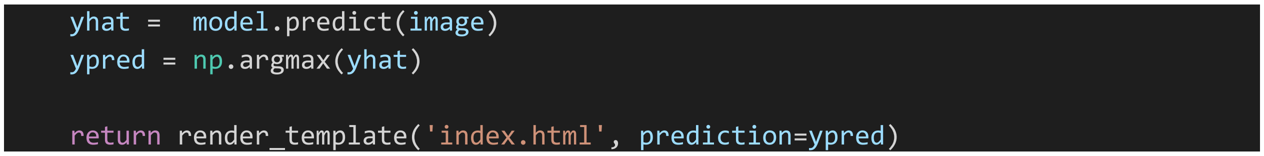
Name	Date modified	Type	Size
assets	5/22/2022 9:32 AM	File folder	
variables	5/30/2022 1:31 PM	File folder	
keras_metadata.pb	5/22/2022 9:32 AM	PB File	46 KB
saved_model.pb	5/22/2022 9:32 AM	PB File	526 KB

CREATING HTTP & FLASK BASED WEB APPLICATION

In order to create a Gender Detection web application, we have created two different files:

- Flask for the back-end engine - app.py
- HTML for the front-end - /templates/index.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.



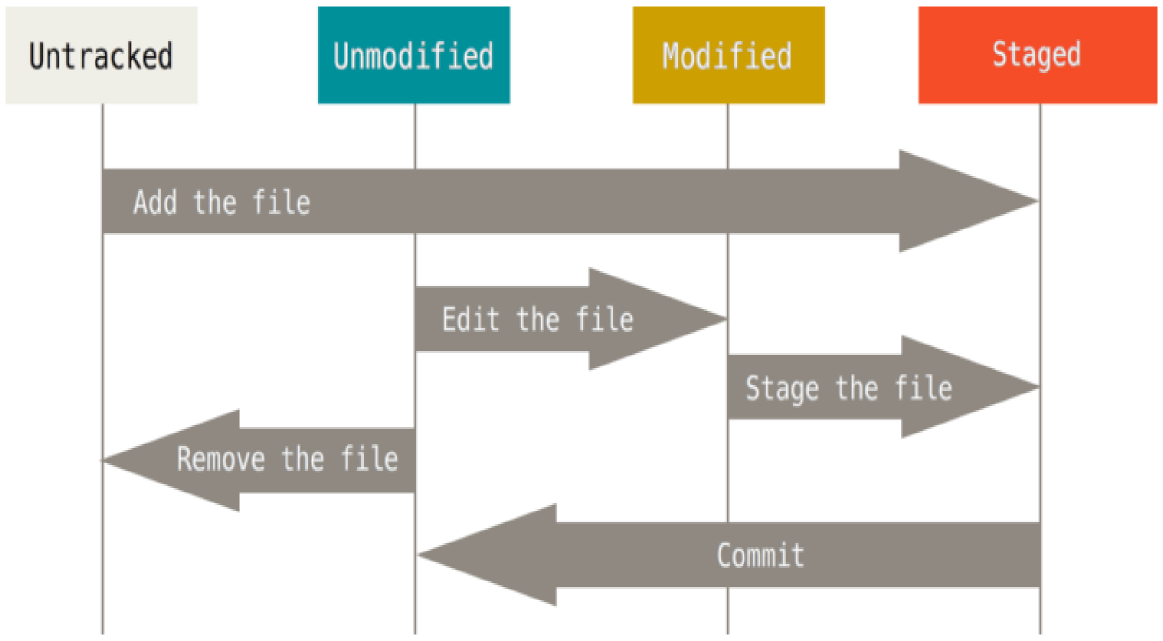
```
    yhat =  model.predict(image)
    ypred = np.argmax(yhat)

    return render_template('index.html', prediction=ypred)
```

```
        {% if prediction == 1 %}
            <p class="text-center" style="color:#e82561;font-size: 25px;">The
Person is a Female {{prediction}}</p>
        {% elif prediction == 0 %}
            <p class="text-center" style="color:#28c0e5;font-size: 25px;">The
Person is a Male {{prediction}}</p>
        {% endif %}
```

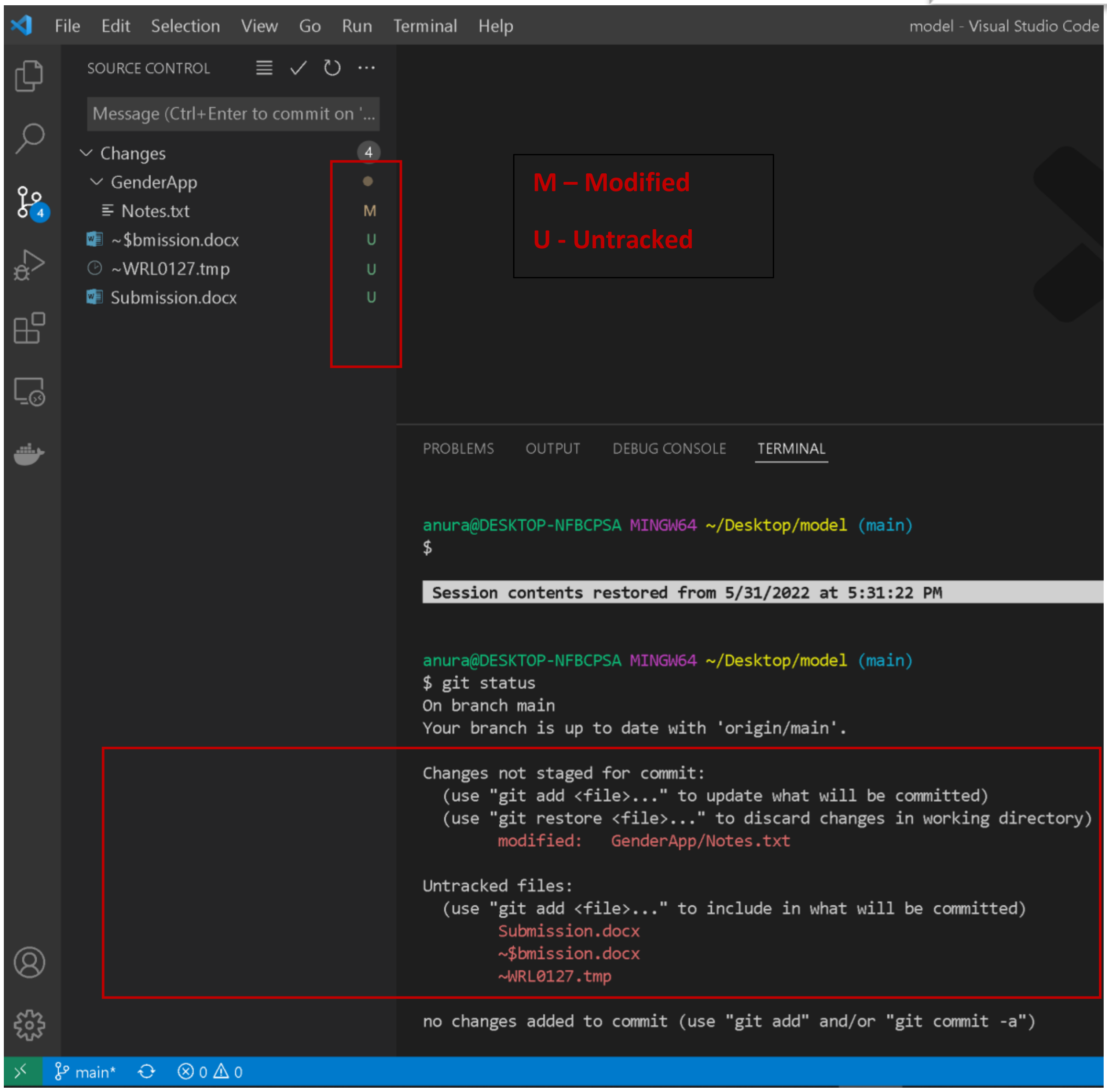
GITHUB & SCM

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.



[Click to Access Repo]


File Edit Selection View Go Run Terminal Help
model - Visual Studio Code
SOURCE CONTROL
Message (Ctrl+Enter to commit on '...
Changes 4
GenderApp
Notes.txt M
~$bmission.docx U
~WRL0127.tmp U
Submission.docx U
M – Modified
U - Untracked
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
main*

```
anura@DESKTOP-NFBCPSA MINGW64 ~/Desktop/model (main)
$

 Session contents restored from 5/31/2022 at 5:31:22 PM

anura@DESKTOP-NFBCPSA MINGW64 ~/Desktop/model (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   GenderApp/Notes.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Submission.docx
        ~$bmission.docx
        ~WRL0127.tmp

no changes added to commit (use "git add" and/or "git commit -a")
```

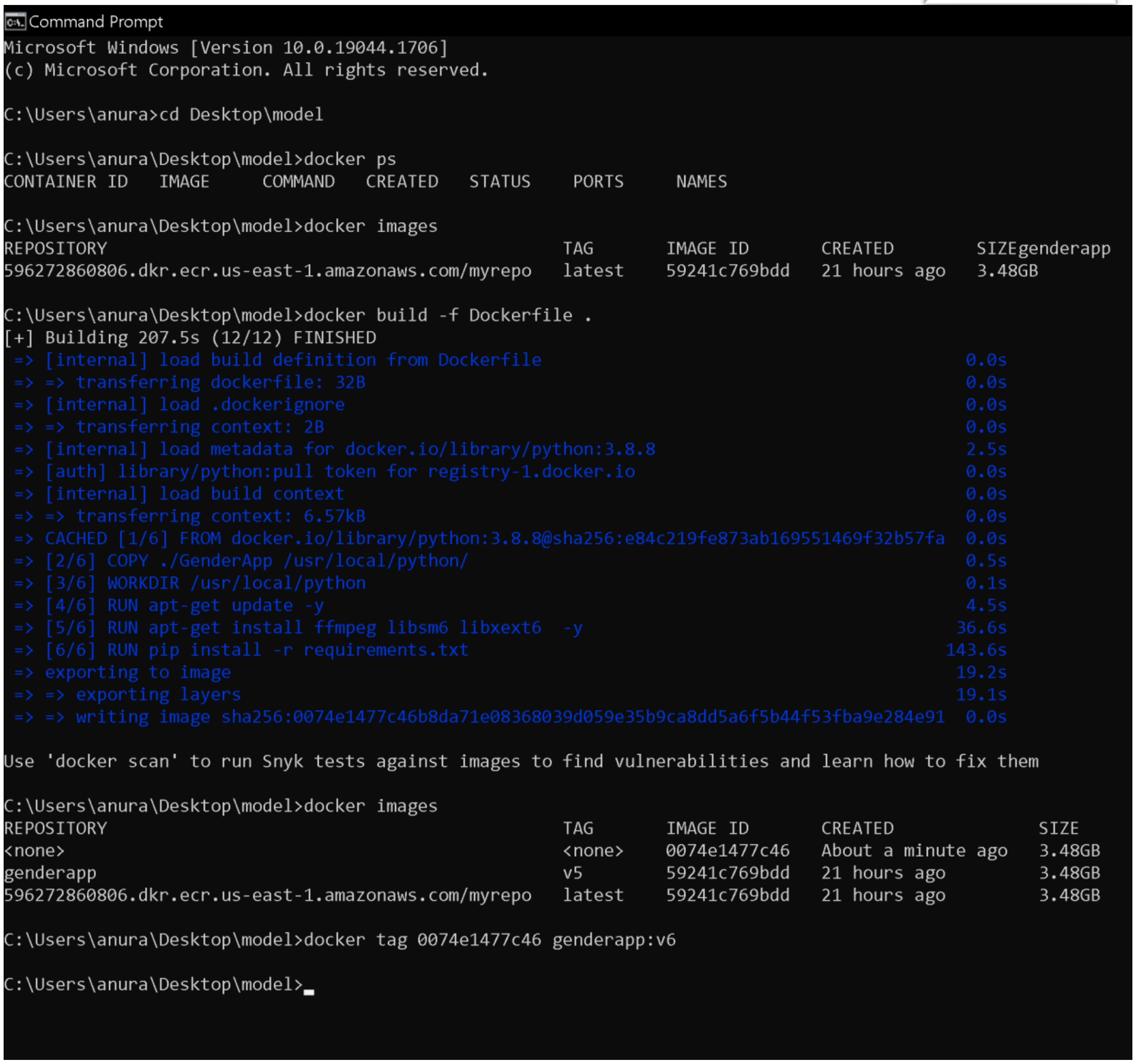
CREATING A CUSTOM IMAGE FROM DOCKER

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 GENDER_DETECTION <IMAGE_NAME: VERSION>* command.

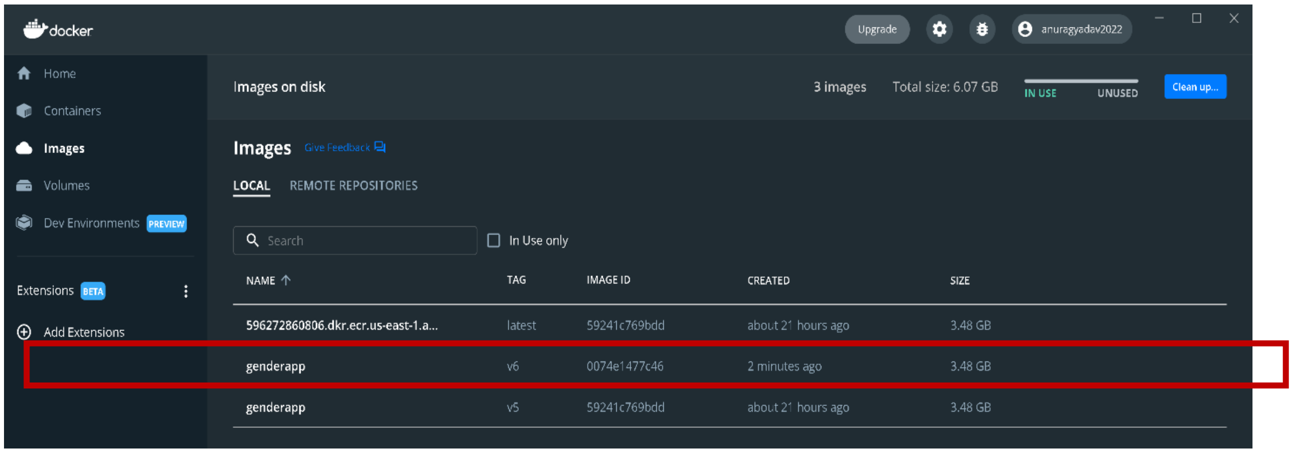
The application can be seen running at *localhost:5000* in the last image.

GitHub
[Click to Access Repo]



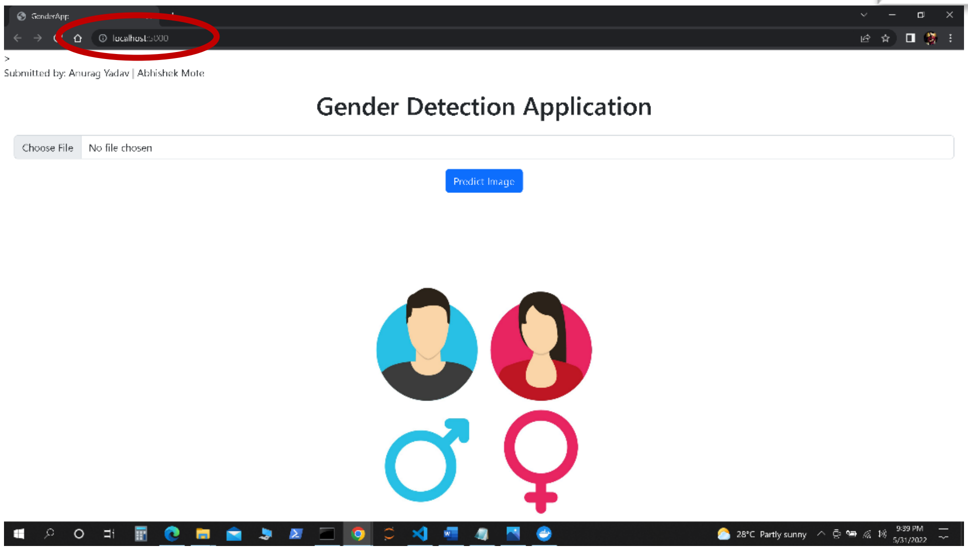
Command Prompt

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anura>cd Desktop\model

C:\Users\anura\Desktop\model>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\anura\Desktop\model>docker images
REPOSITORY                                         TAG       IMAGE ID       CREATED        SIZEgenderapp
596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo   latest    59241c769bdd   21 hours ago   3.48GB

C:\Users\anura\Desktop\model>docker build -f Dockerfile .
[+] Building 207.5s (12/12) FINISHED
 => [internal] load build definition from Dockerfile                                         0.0s
 => => transferring dockerfile: 32B                                                          0.0s
 => [internal] load .dockerignore                                                            0.0s
 => => transferring context: 2B                                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.8.8                              2.5s
 => [auth] library/python:pull token for registry-1.docker.io                                0.0s
 => [internal] load build context                                                            0.0s
 => => transferring context: 6.57kB                                                          0.0s
 => CACHED [1/6] FROM docker.io/library/python:3.8.8@sha256:e84c219fe873ab169551469f32b57fa  0.0s
 => [2/6] COPY ./GenderApp /usr/local/python/                                                0.5s
 => [3/6] WORKDIR /usr/local/python                                                          0.1s
 => [4/6] RUN apt-get update -y                                                              4.5s
 => [5/6] RUN apt-get install ffmpeg libsm6 libxext6  -y                                    36.6s
 => [6/6] RUN pip install -r requirements.txt                                              143.6s
 => exporting to image                                                                      19.2s
 => => exporting layers                                                                     19.1s
 => => writing image sha256:0074e1477c46b8da71e08368039d059e35b9ca8dd5a6f5b44f53fba9e284e91  0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\anura\Desktop\model>docker images
REPOSITORY                                         TAG       IMAGE ID       CREATED              SIZE
<none>                                             <none>    0074e1477c46   About a minute ago   3.48GB
genderapp                                          v5        59241c769bdd   21 hours ago         3.48GB
596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo   latest    59241c769bdd   21 hours ago         3.48GB

C:\Users\anura\Desktop\model>docker tag 0074e1477c46 genderapp:v6

C:\Users\anura\Desktop\model>_
```



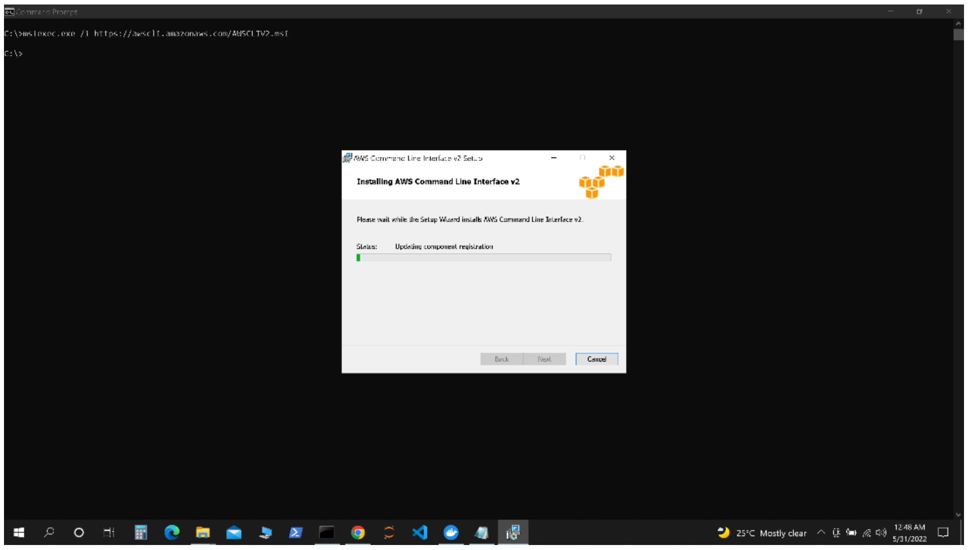
NAME	TAG	IMAGE ID	CREATED	SIZE
596272860806.dkr.ecr.us-east-1.a...	latest	59241c769bdd	about 21 hours ago	3.48 GB
genderapp	v6	0074e1477c46	2 minutes ago	3.48 GB
genderapp	v5	59241c769bdd	about 21 hours ago	3.48 GB

GitHub
[Click to Access Repo]



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 AWS ECR.

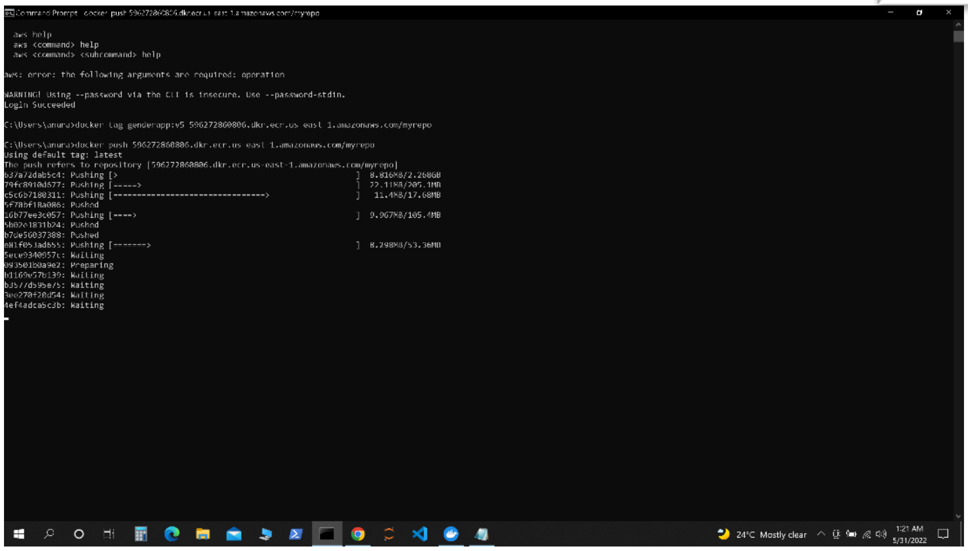


```
C:\>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi

C:\>
```

Installing AWS Command Line Interface v2

Please wait while the Setup Wizard installs AWS Command Line Interface v2.

Status: Updating component registration



```
Command Prompt - docker push 596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo

  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\anura>docker tag genderapp:v5 596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo

C:\Users\anura>docker push 596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo
Using default tag: latest
The push refers to repository [596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo]
637a72dab5c4: Pushing [>                                                  ]  8.816MB/2.268GB
79fc8910d677: Pushing [----->                                             ]  22.11MB/205.1MB
c5c6b7180311: Pushing [-------------------------------->                  ]   11.4MB/17.68MB
5f78bf18a086: Pushed
16b77ee3c057: Pushing [---->                                              ]  9.967MB/105.4MB
5b02e1831b24: Pushed
b7de56037388: Pushed
e81f053ad655: Pushing [------->                                           ]  8.298MB/53.36MB
5ece9340957c: Waiting
093501b0a9e2: Preparing
b1169e57b139: Waiting
b3577d595e75: Waiting
3ee270f20d54: Waiting
4ef4adca5c3b: Waiting
```

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.

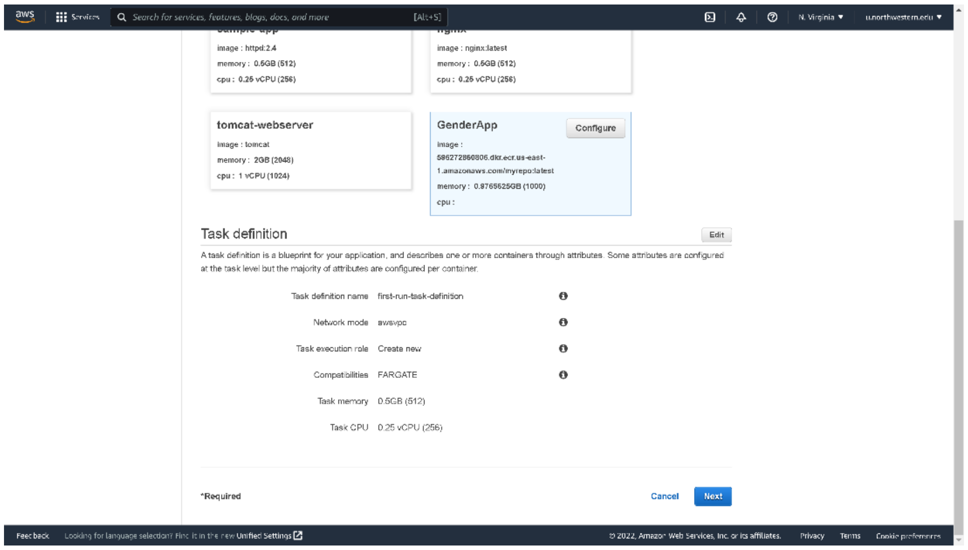


image : httpd:2.4
memory : 0.5GB (512)
cpu : 0.25 vCPU (256)

image : nginx:latest
memory : 0.5GB (512)
cpu : 0.25 vCPU (256)

tomcat-webserver
image : tomcat
memory : 2GB (2048)
cpu : 1 vCPU (1024)

GenderApp
image : 596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo:latest
memory : 0.9765625GB (1000)
cpu :

Task definition

A task definition is a blueprint for your application, and describes one or more containers through attributes. Some attributes are configured at the task level but the majority of attributes are configured per container.

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)

*Required

Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
Configure
Edit
Cancel
Next
© 2022, Amazon Web Services, Inc. or its affiliates.

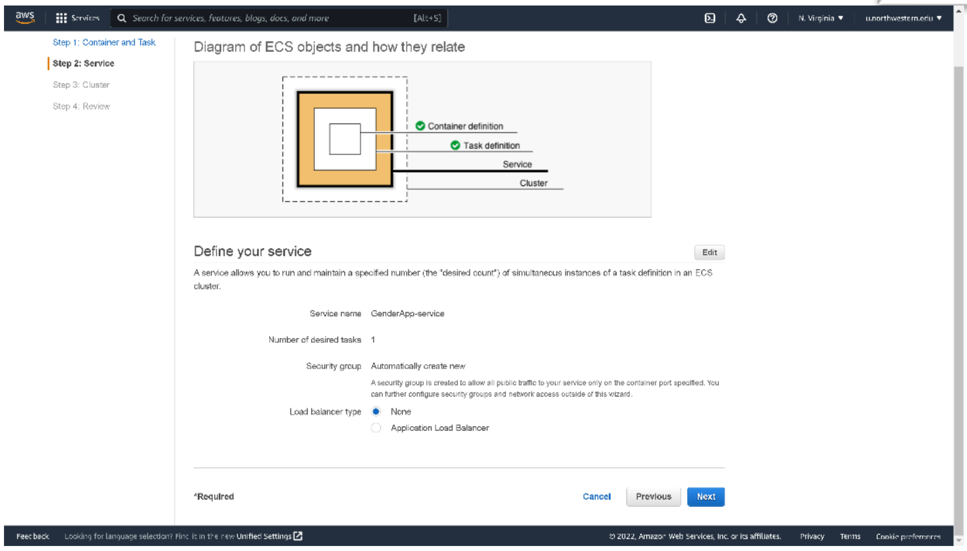

Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
u.northwestern.edu
Step 1: Container and Task
Step 2: Service
Step 3: Cluster
Step 4: Review
Container definition
Task definition
Service
Cluster
Edit
Cancel
Previous
Next
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Diagram of ECS objects and how they relate

Define your service

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	GenderApp-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

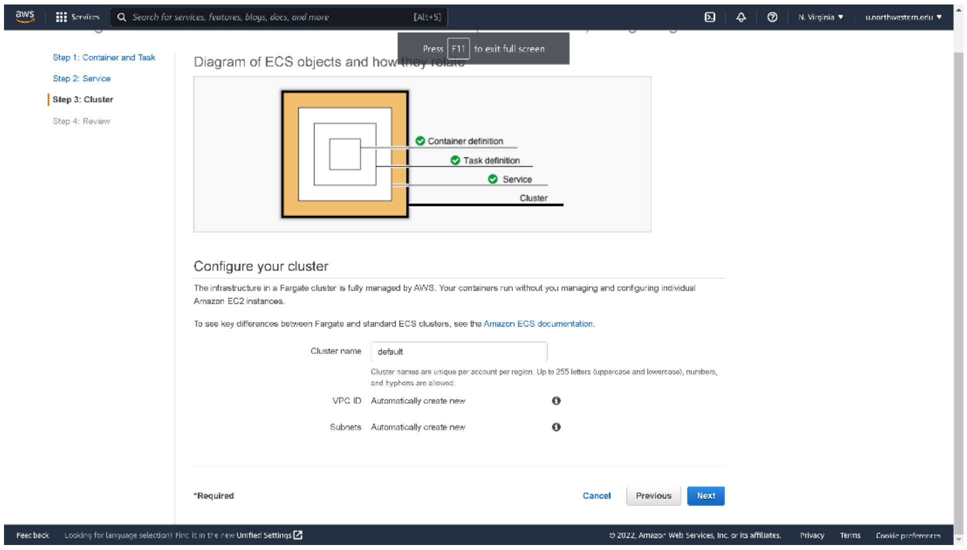

Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
u.northwestern.edu
Press F11 to exit full screen
Step 1: Container and Task
Step 2: Service
Step 3: Cluster
Step 4: Review
Container definition
Task definition
Service
Cluster
Cancel
Previous
Next
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Diagram of ECS objects and how they relate

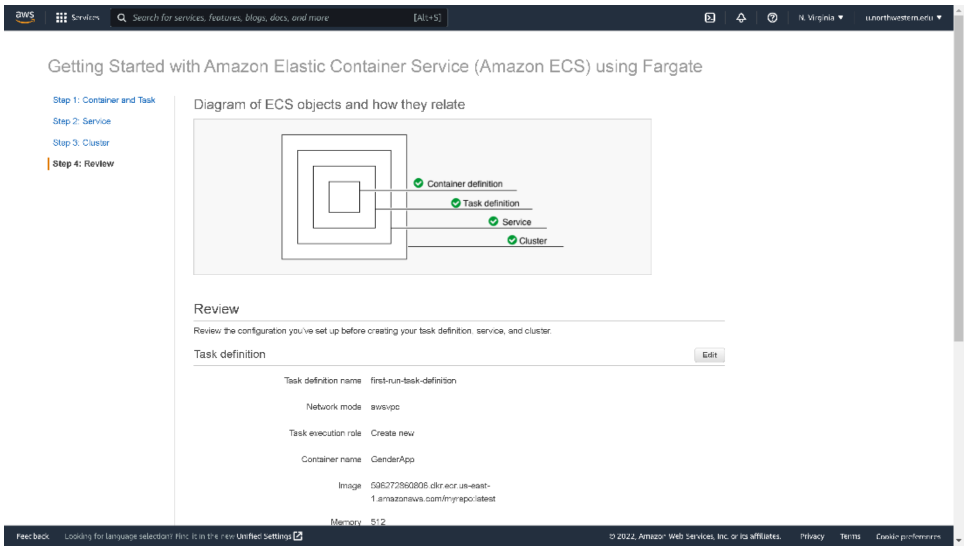
Configure your cluster

The infrastructure in a Fargate cluster is fully managed by AWS. Your containers run without you managing and configuring individual Amazon EC2 instances.

To see key differences between Fargate and standard ECS clusters, see the Amazon ECS documentation.

Cluster name	default Cluster names are unique per account per region. Up to 255 letters (uppercase and lowercase), numbers, and hyphens are allowed.
VPC ID	Automatically create new
Subnets	Automatically create new

*Required


Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
u.northwestern.edu
Step 1: Container and Task
Step 2: Service
Step 3: Cluster
Step 4: Review
Container definition
Task definition
Service
Cluster
Edit
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Getting Started with Amazon Elastic Container Service (Amazon ECS) using Fargate

Diagram of ECS objects and how they relate

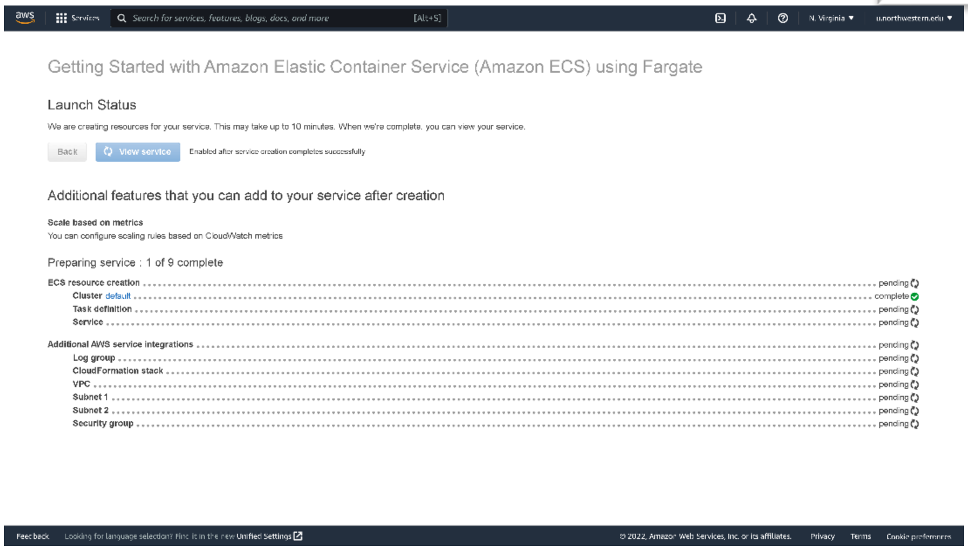
Review

Review the configuration you've set up before creating your task definition, service, and cluster.

Task definition

Task definition name	first-run-task-definition
Network mode	awsvpc
Task execution role	Create new
Container name	GenderApp
Image	596272860806.dkr.ecr.us-east-1.amazonaws.com/myrepo:latest
Memory	512

[Click to Access Repo]



Getting Started with Amazon Elastic Container Service (Amazon ECS) using Fargate

Launch Status

We are creating resources for your service. This may take up to 10 minutes. When we're complete, you can view your service.

Back | View service | Enabled after service creation completes successfully

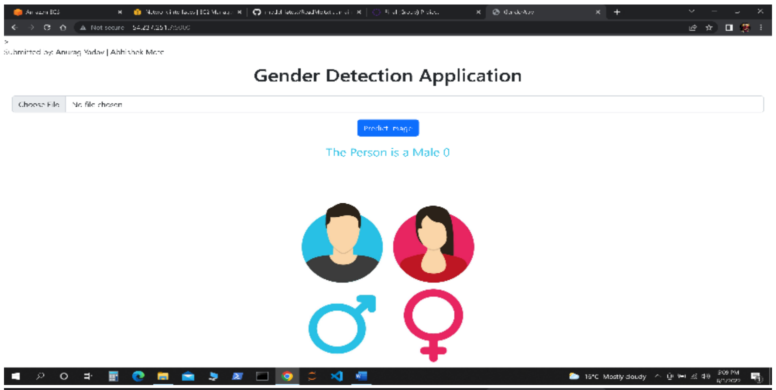
Additional features that you can add to your service after creation

Scale based on metrics
You can configure scaling rules based on CloudWatch metrics

Preparing service : 1 of 9 complete

```
ECS resource creation ............................ pending
    Cluster default .............................. complete
    Task definition .............................. pending
    Service ...................................... pending
Additional AWS service integrations .............. pending
    Log group .................................... pending
    CloudFormation stack ......................... pending
    VPC .......................................... pending
    Subnet 1 ..................................... pending
    Subnet 2 ..................................... pending
    Security group ............................... pending
```

RESULTS:



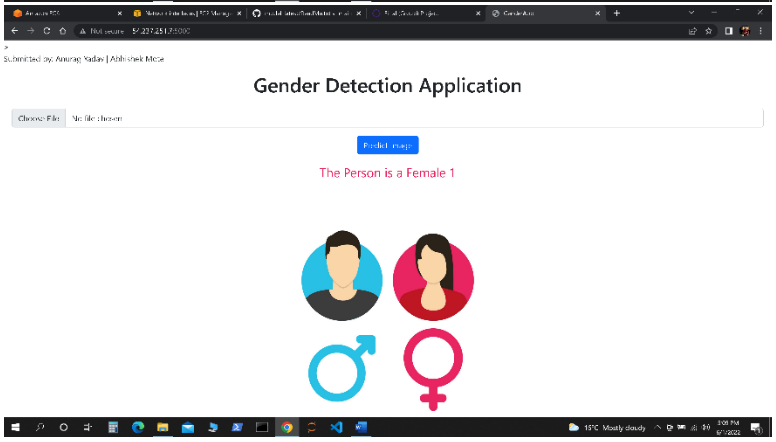
Submitted by: Anurag Yadav | Abhishek Mote

Gender Detection Application

Choose File | No file chosen

Predict Image

The Person is a Male 0



Submitted by: Anurag Yadav | Abhishek Mote

Gender Detection Application

Choose File | No file chosen

Predict Image

The Person is a Female 1

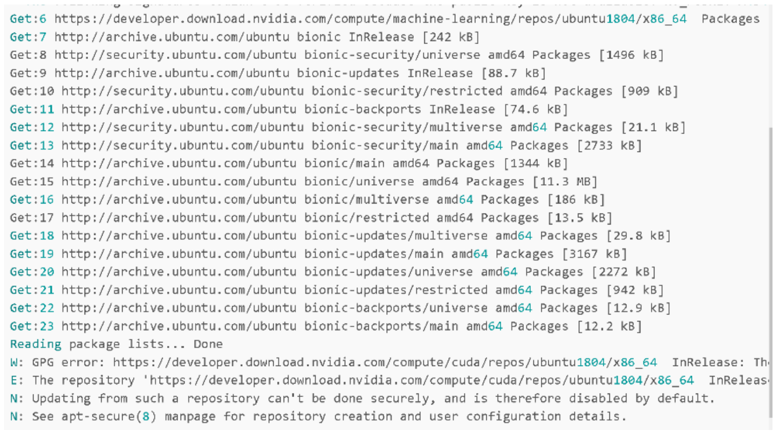
Access this Application over Internet: http://54.237.251.7:5000/

Demo Video: https://drive.google.com/file/d/1Da8pqNhX1ZPzoGYwfcrE0ElZwA-PC0-v/view?usp=sharing

GitHub
[Click to Access Repo]

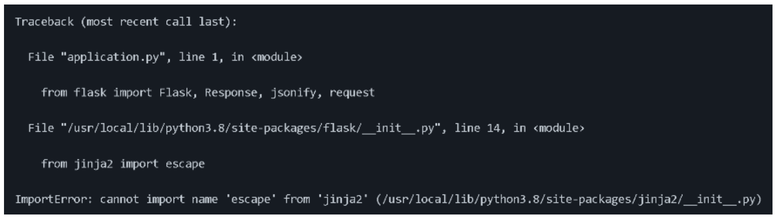
CHALLENGES / ERRORS ENCOUNTERED:

1. Invalid public key for CUDA apt repository



```
Get:6 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  Packages
Get:7 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1496 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [909 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [21.1 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2733 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:16 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [29.8 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3167 kB]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2272 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [942 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [12.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [12.2 kB]
Reading package lists... Done
W: GPG error: https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRelease: Th
E: The repository 'https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InReleas
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

2. Flask failing to startup due to Jinja2 breaking change

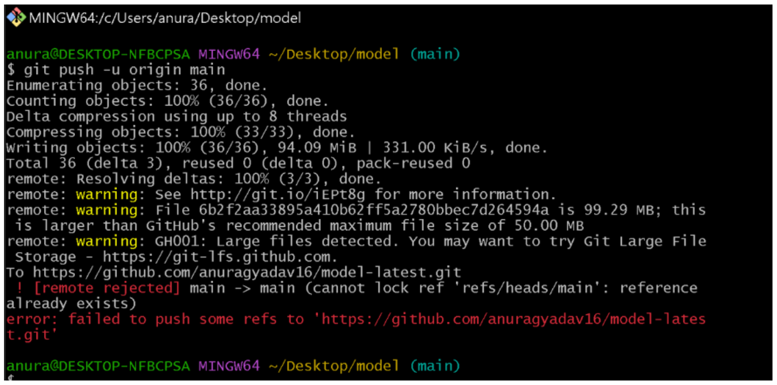


```
Traceback (most recent call last):

  File "application.py", line 1, in <module>

    from flask import Flask, Response, jsonify, request

  File "/usr/local/lib/python3.8/site-packages/flask/__init__.py", line 14, in <module>

    from jinja2 import escape

ImportError: cannot import name 'escape' from 'jinja2' (/usr/local/lib/python3.8/site-packages/jinja2/__init__.py)
```

3. Pip version error

```
WARNING: You are using pip version 20.1.1; however, version 20.2 is available. You
should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --
upgrade pip' command
```

4. Git size limit error



```
MINGW64:/c/Users/anura/Desktop/model

anura@DESKTOP-NFBCPSA MINGW64 ~/Desktop/model (main)
$ git push -u origin main
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
Delta compression using up to 8 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (36/36), 94.09 MiB | 331.00 KiB/s, done.
Total 36 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
remote: warning: See http://git.io/iEPt8g for more information.
remote: warning: File 6b2f2aa33895a410b62ff5a2780bbec7d264594a is 99.29 MB; this
 is larger than GitHub's recommended maximum file size of 50.00 MB
remote: warning: GH001: Large files detected. You may want to try Git Large File
 Storage - https://git-lfs.github.com.
To https://github.com/anuragyadav16/model-latest.git
 ! [remote rejected] main -> main (cannot lock ref 'refs/heads/main': reference
already exists)
error: failed to push some refs to 'https://github.com/anuragyadav16/model-lates
t.git'

anura@DESKTOP-NFBCPSA MINGW64 ~/Desktop/model (main)
```