



Week 5: Hosting Web-App on AWS

Name: Ashish Sasanapuri

Batch Code: LISUM26

Submission Date: 04/11/2023

Submitted To: GitHub

1. Introduction

The aim of this project is to deploy the ML model designed in the Week 4's task on cloud. The process involves containerizing the Flask web-app and deploying it on AWS.

The additional files used in this project involves:

- **Dockerfile** – The Dockerfile contains the script to create and maintain the Docker Image.
- **dg_ml_deploy.pem** – A set of security credentials consisting a pair of public-private keys used for authenticating connection to EC2 instances on AWS.

2. Containerizing the Flask Web-app

This section will describe the process involved in using Docker for wrapping the web-app in a container. Containerization helps in deploying the application along with its dependencies and runtime environment on any machine irrespective of the environment and running via a single command run.

2.1. Installing Docker

Docker is one such platform which provides the functionalities of wrapping the application in a package. Install Docker from the [website](#).

Get Started with Docker

Build applications faster and more securely with Docker for developers

[Learn how to install Docker](#)[Download for Windows](#)

2.2. Create a Dockerfile

The below Dockerfile contains the commands for creating the Docker directory, installing the required files and libraries and provide a port to listen to the container. Save the Dockerfile in the project directory.

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /DataGlacier/Week5

# Copy the current directory contents into the container
COPY Scripts/ /DataGlacier/Week5/Scripts/
COPY model.pkl /DataGlacier/Week5/
COPY requirements.txt /DataGlacier/Week5/

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Expose the port 5000
EXPOSE 5000

# Entrypoint for executing the app
ENTRYPOINT ["python", "Scripts/app.py"]
```

2.3. Build the Docker Image

Redirect to the directory where the Dockerfile resides and build the Docker image using below command.

```
docker build -t diab-pred .
```

```

D:\GitUpload\Data-Glacier\Week 5>docker build -t diab-pred .
[+] Building 3.6s (12/12) FINISHED
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                  0.0s
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 585B                             0.0s
=> [internal] load metadata for docker.io/library/python:3.8-slim 3.5s
=> [auth] library/python:pull token for registry-1.docker.io   0.0s
=> [1/6] FROM docker.io/library/python:3.8-slim@sha256:9187d27fd8f222a181292f24f8e7d6b22419d46bd9cc4506adbdf2dcf 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 325B                                  0.0s
=> CACHED [2/6] WORKDIR /DataGlacier/Week5                     0.0s
=> CACHED [3/6] COPY Scripts/ /DataGlacier/Week5/Scripts/      0.0s
=> CACHED [4/6] COPY model.pkl /DataGlacier/Week5/              0.0s
=> CACHED [5/6] COPY requirements.txt /DataGlacier/Week5/       0.0s
=> CACHED [6/6] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:d7560f9af3753713cad30cd846dcd98fe96ce8d8e764e3c640375ae8121bd0f5 0.0s
=> => naming to docker.io/library/diab-pred                    0.0s

```

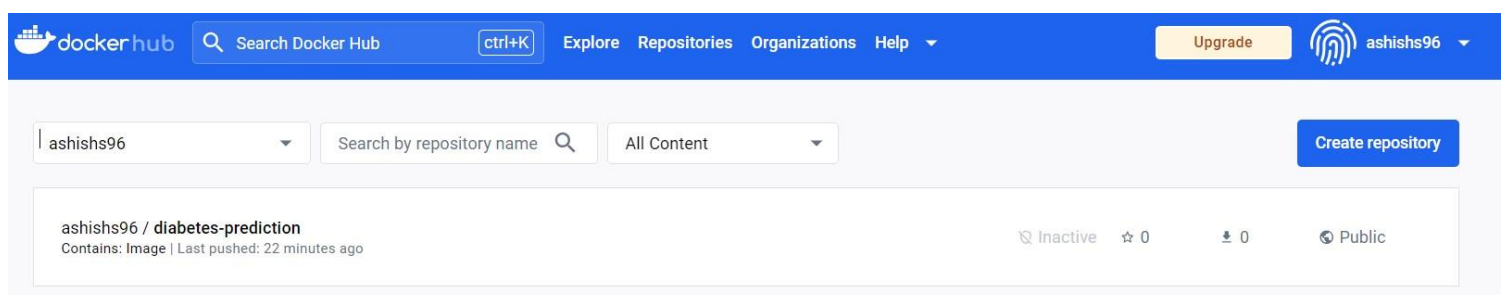
The Docker Desktop shows the below Docker image named **diab-pred**.

<input type="checkbox"/>	Name	Tag	Status
<input type="checkbox"/>	diab-pred d7560f9af375	latest	Unused

2.4. Push the Docker Image to repository

Since we have created the Docker Image locally, we need to push it to a public (or private) repository to use it on different environments.

The initial step taken is to create a repository on Docker Hub. We create a repository named **diabetes-prediction** and keep it's accessibility as **Public** for the purpose of this project.



We need to successfully login to the docker to establish a connection for pushing the image to the repository.

```
C:\Users\Ashish S>docker login
Authenticating with existing credentials...
Login Succeeded
```

Before pushing the image, we set a tag to the Docker Image for identification. We use the **docker tag** command for this purpose and it helps identifying multiple versions of the image.

```
docker tag diab-pred:latest ashishs96/diabetes-prediction:v1.0
```

Post that, we push the image to the repository.

```
C:\Users\Ashish S>docker push ashishs96/diabetes-prediction:v1.0
The push refers to repository [docker.io/ashishs96/diabetes-prediction]
356416a15723: Pushed
20ffcf241913: Pushed
93c5e8d1030d: Pushed
694ef891967d: Pushed
e2908626ca8a: Pushed
06c9b2aef57b: Pushed
a839d81d2782: Pushed
26ee2f9cbcd4: Pushed
87579ac672ec: Pushed
ec983b166360: Pushed
v1.0: digest: sha256:5b0a1f8eae80e6383f7650bb001ed766dfd015be0fddaf62c0ff508db56af29c size: 2411
```

3. Hosting on cloud

We use Amazon Web Service (AWS) for deploying the local web-app on cloud as it provides an abstract environment consisting of underlying services and network security.

3.1. Create an instance on AWS

AWS provides various free and paid instances depending on the requirements of the deployment. We choose Elastic Compute Cloud (EC2) instance of the AWS compute services offering virtual servers for hosting on the cloud.

Navigating to the EC2 service, we launch an instance of EC2 and configure the environment.

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▼

Migrate a server ↗

Note: Your instances will launch in the Asia Pacific (Mumbai) Region

The underlying operating system (OS) and application server is provided by selecting one of the Amazon Machine Image (AMI). We chose the default free tier machine which is **Amazon Linux 2023** AMI.

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

🔍 Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux
aws


macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE Li
SUSE


Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-02e94b011299ef128 (64-bit (x86)) / ami-0a31c2002672717b6 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Amazon Linux 2023 AMI 2023.2.20231030.1 x86_64 HVM kernel-6.1

The next step is to select the instance type. Among all the instance types, EC2 provides **t2.micro** as the eligible free tier.

▼ Instance type [Info](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Linux base pricing: 0.0124 USD per Hour

On-Demand Windows base pricing: 0.017 USD per Hour

On-Demand RHEL base pricing: 0.0724 USD per Hour

On-Demand SUSE base pricing: 0.0124 USD per Hour

☒ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

The network configuration defines the protocols and ports for communication with the instance. We allow the HTTP and SSH traffic by setting the both options.

▼ Network settings [Info](#)

Edit

Network [Info](#)

vpc-0aa5c6bd92a1a989a

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

☐ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

We can also set the inbound rules for the EC2 instance.

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Remove

Type [Info](#)

ssh ▼

Protocol [Info](#)

TCP

Port range [Info](#)

22

Source type [Info](#)

Anywhere ▼

Source [Info](#)

🔍 Add CIDR, prefix list or security

0.0.0.0/0 ✕

Description - optional [Info](#)

e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, Multiple sources)

Remove

Type [Info](#)

HTTP ▼

Protocol [Info](#)

TCP

Port range [Info](#)

80

Source type [Info](#)

Custom ▼

Source [Info](#)

🔍 Add CIDR, prefix list or security

0.0.0.0/0 ✕

::/0 ✕

Description - optional [Info](#)

e.g. SSH for admin desktop

After configuring the required settings, we launch the service. We will get status of the instance creation.

[EC2](#) > [Instances](#) > Launch an instance

✔ Success

Successfully initiated launch of instance ([i-0fae8e2bd0e17b30e](#))

▼ Launch log

Initializing requests ✔ Succeeded

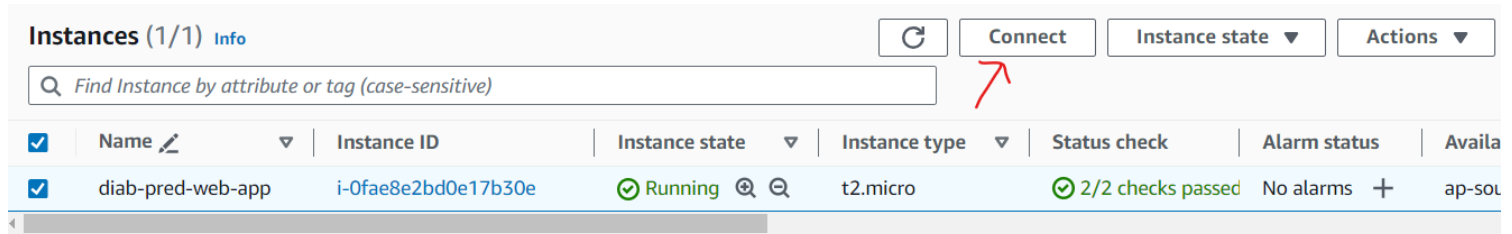
Creating security groups ✔ Succeeded

Creating security group rules ✔ Succeeded

Launch initiation ✔ Succeeded

3.2. Establishment of connection

Once the instance is created, we can establish the connection by navigating to the **Instances** section on the AWS console, selecting the created instance and clicking the connect button.



This opens a browser-based shell or a session manager for securely connecting to the instance without the need for opening inbound ports or managing SSH keys.

```
aws Services [Alt+S]
#_
~\_ ##### Amazon Linux 2023
~~ \#####\
~~ \|###|
~~ \|#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '->
~~~ /
~~.-./ -/_/
~/m/'
Last login: Sat Nov 4 07:21:27 2023 from 13.233.177.4
[ec2-user@ip-172-31-43-57 ~]$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

We successfully login to docker using **docker login** command and run the below command to open the web-app.

```
[ec2-user@ip-172-31-43-57 ~]$ sudo docker run -p 80:5000 -d ashishs96/diabetes-prediction:v1.0
892d2839c158aefcb845d7e104dec1a14e8ffb957eca01548276065a6cc8cfed
```

4. Accessing the web-app

As the web-app connection has been established, we open the app using the **Public IP** address.

Instance: i-0fae8e2bd0e17b30e (diab-pred-web-app)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

▼ Instance summary [Info](#)

Instance ID
i-0fae8e2bd0e17b30e (diab-pred-web-app)

IPv6 address
-

Public IPv4 address
3.108.40.46 [open address](#)

Instance state
Running

Private IPv4 addresses
172.31.43.57

Public IPv4 DNS
ec2-3-108-40-46.ap-south-1.compute.amazonaws.com
[open address](#)

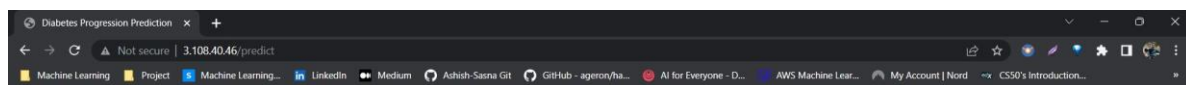
This will open the web app as below:



Predict Progression of Diabetes after 1 year from baseline

Note: All the values should be between 1 and -1 as the model accepts only these values.

Enter Age	<input type="text"/>
Enter Sex	<input type="text"/>
Enter BMI	<input type="text"/>
Enter Blood Pressure	<input type="text"/>
Enter Total Serum Cholesterol	<input type="text"/>
Enter Low-Density Lipoproteins	<input type="text"/>
Enter High-Density Lipoproteins	<input type="text"/>
Enter Total Cholesterol	<input type="text"/>
Enter Triglycerides Level	<input type="text"/>
Enter Blood Sugar Level	<input type="text"/>
<input type="button" value="Predict"/>	



Predict Progression of Diabetes after 1 year from baseline

Note: All the values should be between 1 and -1 as the model accepts only these values.

Enter Age	<input type="text" value="0.0045341"/>
Enter Sex	<input type="text" value="-0.044642"/>
Enter BMI	<input type="text" value="-0.006206"/>
Enter Blood Pressure	<input type="text" value="-0.015999"/>
Enter Total Serum Cholesterol	<input type="text" value="0.125019"/>
Enter Low-Density Lipoproteins	<input type="text" value="0.125198"/>
Enter High-Density Lipoproteins	<input type="text" value="0.019187"/>
Enter Total Cholesterol	<input type="text" value="0.034309"/>
Enter Triglycerides Level	<input type="text" value="0.032433"/>
Enter Blood Sugar Level	<input type="text" value="-0.005220"/>
<input type="button" value="Predict"/>	

Diabetes progression result is 138.001