Contents

[Model Deployment Using Cloud Services 1](#_Toc176796211)

[Setting up the environment 2](#_Toc176796212)

[Use the curl command to check if the /predict endpoint is working 3](#_Toc176796213)

[Screen shots of the web ui 3](#_Toc176796214)

[Containerizing the application using Docker 5](#_Toc176796215)

[Building the docker image 5](#_Toc176796216)

[Running the docker container 5](#_Toc176796217)

[Posting to the flask endpoint using curl 5](#_Toc176796218)

[Login to Azure from command line 6](#_Toc176796219)

[az login 6](#_Toc176796220)

[Create a resource group on azure 6](#_Toc176796221)

[Login to the newly created registry 6](#_Toc176796222)

[Build the docker image 6](#_Toc176796223)

[Tag and push the image to azure container registry 7](#_Toc176796224)

[Check if the image is present in the registry 7](#_Toc176796225)

[Find the credentials of the m3registry 7](#_Toc176796226)

[Create an azure container instance 8](#_Toc176796227)

[Check if azure instance is created 8](#_Toc176796228)

[Check if the container has started 9](#_Toc176796229)

[Use the ipaddress in a curl request 9](#_Toc176796230)

[Following error will be seen 10](#_Toc176796231)

[Application needs to be configured with Flask CORS 11](#_Toc176796232)

[Using the h2o model 11](#_Toc176796233)

[Use this curl command to test the model 11](#_Toc176796234)

# Model Deployment Using Cloud Services

# Setting up the environment

A black background with white text

Description automatically generated

Start the Flask application

python main.py

A screen shot of a computer

Description automatically generated

# Use the curl command to check if the /predict endpoint is working

A black screen with white text

Description automatically generated

# Screen shots of the web ui

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Containerizing the application using Docker

## Building the docker image

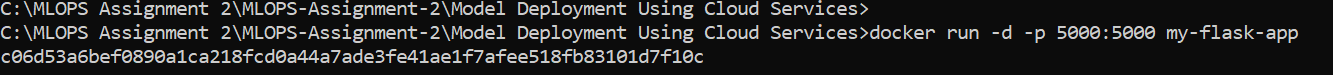
**docker build -t my-flask-app .**

A computer screen shot of blue text

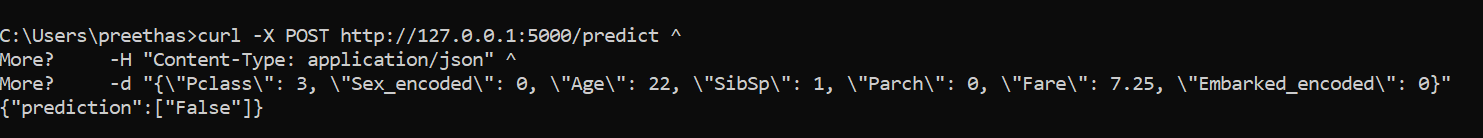
Description automatically generated

## Running the docker container

docker run -d -p 5000:5000 my-flask-app



## Posting to the flask endpoint using curl



## Login to Azure from command line

## az login

## Create a resource group on azure

An Azure Resource Group is a container that holds related resources for an Azure solution.

Resource Management: It provides a way to manage and organize related resources as a

single entity.You can control access to resources within the resource group by applying role-

based access control (RBAC) to the resource group. Resource groups support the deployment

of resources in a structured manner using templates

**az acr create --resource-group MyResourceGroup --name m3registry --sku Basic**

A computer screen with text on it

Description automatically generated

## Login to the newly created registry

**az acr login --name m3registry**

A black screen with white text

Description automatically generated

## Build the docker image

**docker build -t m4img1:latest .**

## Tag and push the image to azure container registry

**docker tag m4img1 m3registry.azurecr.io/m4img1:latest**

**docker push m3registry.azurecr.io/ m4img1:latest**

A screen shot of a computer

Description automatically generated

## Check if the image is present in the registry

**az acr repository list --name m3registry --output table**

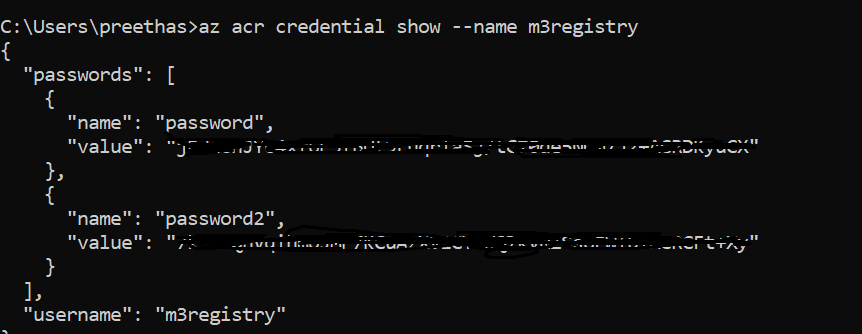
A black screen with white text

Description automatically generated

## Find the credentials of the m3registry

**az acr credential show --name m3registry**

az acr credential show --name m3registry

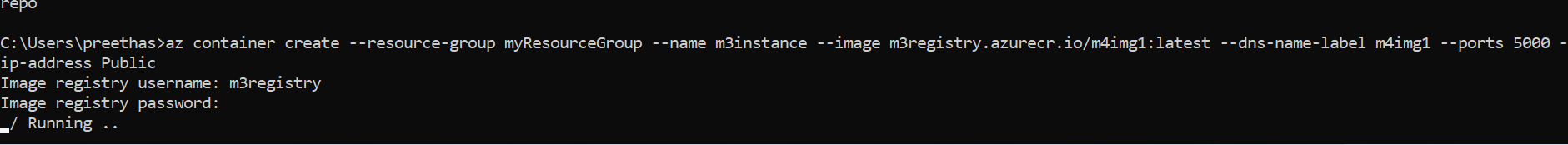


# Create an azure container instance

Azure Container Instances (ACI) is a managed service that allows you to run Docker containers

directly on Azure without needing to manage the underlying virtual machines or orchestration

**az container create --resource-group myResourceGroup --name m3instance --image m3registry.azurecr.io/m4img1:latest --dns-name-label m4img1 --ports 5000 --ip-address Public**



# Check if azure instance is created

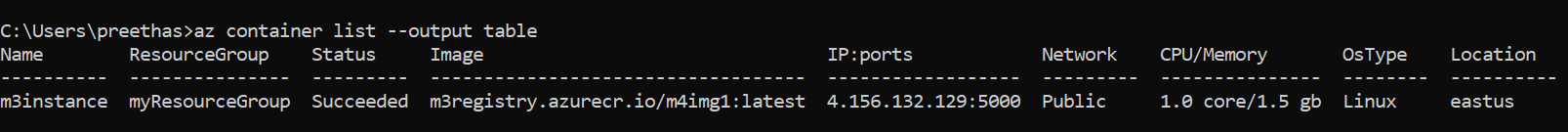
**az container show --resource-group myResourceGroup --name m3instance**

A computer screen with white text

Description automatically generated

## Check if the container has started

**az container list --output table**

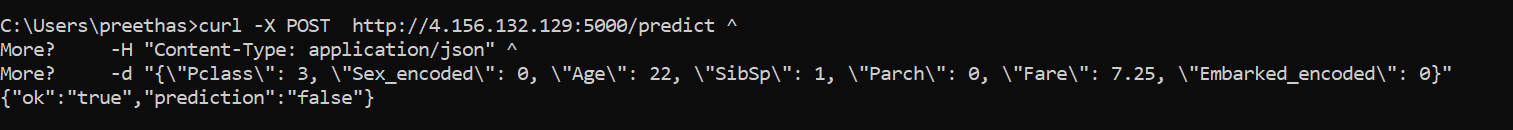


## Use the ipaddress in a curl request

**curl -X POST http://4.156.132.129:5000/predict ^**

**-H "Content-Type: application/json" ^**

**-d "{\"Pclass\": 3, \"Sex\_encoded\": 0, \"Age\": 22, \"SibSp\": 1, \"Parch\": 0, \"Fare\": 7.25, \"Embarked\_encoded\": 0}"**



Use the above url **http://4.156.132.129:5000/predict** in javascript code

A screenshot of a computer program

Description automatically generated

## Following error will be seen

Access to fetch at 'http://4.156.132.129:5000/predict' from origin 'http://10.233.26.39:5000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

CORS is a security feature implemented by browsers to prevent malicious websites from making unauthorized requests to your server. To allow your server to respond to requests from different origins, you need to configure CORS properly.

## Application needs to be configured with Flask CORS

A screen shot of a computer code

Description automatically generated

# Using the h2o model

Run the main\_h20.py file using the command python main\_h2o

A screenshot of a computer program

Description automatically generated

## Use this curl command to test the model

curl -X POST http://127.0.0.1:5000/predict ^

-H "Content-Type: application/json" ^

-d "{\"Pclass\": 3, \"Sex\": \"male\", \"Age\": 22, \"SibSp\": 1, \"Parch\": 0, \"Fare\": 7.25, \"Embarked\": \"C\"}"

