**Important:**

In this lab, d23023 is used as a placeholder roll number to make unique names for some of the artefacts like DNS name. Search for d23023 and replace with your Praxis roll number.

In this lab, red font is used for commands. And in the commands, blue highlight is used for assigned names.

Why are some of the variable names so long? The intent was to help understand the flow and steps easily.

Hope you enjoy this lab.

**Step 1: Create and test the FastAPI App Locally**

**Set up python 3.10 environment**

conda create -n fastapi-env python=3.10

conda activate fastapi-env

Even if you already have an environment, it is a best practice to set up a new one for labs like this.

**install dependencies**

pip install fastapi uvicorn scikit-learn numpy pydantic

Create a working dir. In this lab it will be ref to as CWD.

In the CWD, place the files provided:

* train\_model.py
* main.py
* requirements.txt
* Dockerfile

**Execute the train\_model.py in a terminal:**

python train\_model.py

This will create the model pickle file that will be used by the FastAPI app.

Run the FastAPI app in the python environment like Anaconda Prompt terminal using uvicorn

uvicorn main:app --reload --host 0.0.0.0 --port 8000

**Test the API by navigating to the following URLs in a web browser:**

* http://localhost:8000/ – Should return the welcome message.
* http://localhost:8000/docs – Swagger UI for FastAPI apps
* http://localhost:8000/redoc – Redoc documentation for FastAPI app

**Test using cURL**

Open a new instance of terminal, execute the following command:

curl -X "POST" "http://localhost:8000/predict/" -H "Content-Type: application/json" -d "{\"x\": 2}"

The output should be: { "input": 2, "prediction": 3.5 }

**Free up port 8000:**

Close the app running at port 8000 in the terminal using Ctrl + C

**Step 2: Dockerize the FastAPI App**

In terminal, navigate to the working directory i.e. CWD

**Build Docker image**

Build the docker image by running this in terminal:  
docker build -t my-fastapi-app-image .

Verify this image is available by running by executing this command or by checking in images tab of Docker Desktop

docker images

**Run Docker Container**

Run the docker container using

docker run -d -p 8000:8000 --name my-fastapi-app-container my-fastapi-app-image

This starts the FastAPI app inside the container and exposes it on port 8000.

Verify if the container is running by running this in a terminal  
docker ps

**Test the API**

Try to access the app using the same methods as mentioned in step 1.

**How to close the running sessions?**

Stop docker container in Docker Desktop or in terminal using

docker stop my-fastapi-app-container

Step 3: Deploy App on Azure Container Instances

**Log in to Azure**

Execute this command CMD terminal:

az login

If you have already logged into azure in a browser, this step should just require you to click Enter again.

**create resource group**

az group create --name myResourceGroup --location eastus2

**Create Azure Container Registry (ACR)**

az acr create --resource-group myResourceGroup --name myacrd23023 --sku Basic

az acr update --name myacrd23023 --admin-enabled true

The ACR name has to be unique across Azure.

**Login to ACR registry**

az acr login --name myacrd23023

az acr credential show --name myacrd23023

This will display the username and two passwords.

**Tag and Push the Docker image to ACR**

docker tag my-fastapi-app-image myacrd23023.azurecr.io/my-fastapi-app-image:latest

docker push myacrd23023.azurecr.io/my-fastapi-app-image:latest

Get login-server name

az acr show --name myacrd23023 --query "loginServer" --output tsv

**Create a Container in Azure Container Instances (ACI)**

In the command below, paste the ACR password from the previous step keeping everything else as given.

az container create --resource-group myResourceGroup --name myapp-fastapi-containerd23023 --image myacrd23023.azurecr.io/my-fastapi-app-image:latest --os-type Linux --cpu 1 --memory 1 --registry-login-server myacrd23023.azurecr.io --registry-username myacrd23023 --registry-password "ZSwjriSudVSB7bXX/LqBus/vr2nXXnzrTF23g4+s/8+ACRDP/mUh" --dns-name-label myapp-fastapi-dnsd23023 --ports 8000 --location eastus2

**Get the IP address of the Azure Container Instance**

az container show --resource-group myResourceGroup --name myapp-fastapi-containerd23023 --query ipAddress.fqdn -o tsv

Output DNS-NAME expected is:

*myapp-fastapi-dns-d23023.eastus2.azurecontainer.io*

**Test the API**

* Open a browser and visit: http://<DNS-NAME>:8000/
* Example:
  + [http://myapp-fastapi-dns-d23023.eastus2.azurecontainer.io:8000/](http://myapp-fastapi-dns-d25001.eastus2.azurecontainer.io:8000/)
* Try this for swagger UI
  + http://myapp-fastapi-dnsd23023.eastus2.azurecontainer.io:8000/docs

Test the API using using cURL by running this in a terminal:

curl -X POST "http://myapp-fastapi-dns-d23023.eastus2.azurecontainer.io:8000/predict/" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"x\": 2}"

expected output: {"input":2.0,"prediction":3.5}