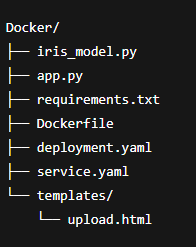
**### Deployment**

**Create a new folder: Docker**

****

**Create a folder named Docker.**

**Open the folder in VS Code**

**Create a file named iris\_model.py**

**Step 1: Train and save LogisticRegression model(iris\_model.py)**

**from** sklearn.datasets **import** load\_iris

**from** sklearn.linear\_model **import** LogisticRegression

**import** pickle

**import** joblib

*# Train model*

iris **=** load\_iris()

X, y **=** iris.data, iris.target

clf **=** LogisticRegression()

clf.fit(X, y)

*# Save as pickle*

**with** open("iris\_model.pkl", "wb") **as** f:

    pickle.dump(clf, f)

*# Alternative: save as joblib*

joblib.dump(clf, "iris\_model.joblib")

**Create another file inside the same folder app.py**

**Step 2: Prepare your Flask app(app.py)**

*# HTML Form Page*

@app.route('/form')

**def** form():

**return** render\_template('upload.html')

*# Handle Form Submission*

@app.route('/predict\_form', methods**=**['POST'])

**def** predict\_form():

    sepal\_length **=** float(request.form['sepal\_length'])

    sepal\_width  **=** float(request.form['sepal\_width'])

    petal\_length **=** float(request.form['petal\_length'])

    petal\_width  **=** float(request.form['petal\_width'])

    features **=** np.array([[sepal\_length, sepal\_width, petal\_length, petal\_width]])

    prediction **=** model.predict(features)[0]

**return** **f**"Predicted Class: {prediction}"

**if** \_\_name\_\_ **==** "\_\_main\_\_":

    app.run(host**=**"0.0.0.0", port**=**5000)

**Step 3: Create requirements.txt**

Flask==2.3.2

numpy

scikit-learn

**Step 4: Create Dockerfile inside the same folder**

**FROM** python:3.11

**WORKDIR** /app

**COPY** requirements.txt .

**RUN** pip install --no-cache-dir -r requirements.txt

**COPY** . .

**EXPOSE** 5000

**CMD** ["python", "app.py"]

**Step 5: Deployment .yaml (inside same folder)**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: welcome-app-deployment

spec:

  replicas: 2

  selector:

    matchLabels:

      app: welcome-app

  template:

    metadata:

      labels:

        app: welcome-app

    spec:

      containers:

      - name: welcome-app

        image: welcome-app:latest

        imagePullPolicy: Never   *# <<<<<<<<<< Add this line*

        ports:

        - containerPort: 5000

**Step 6: service.yaml (inside the same folder)**

apiVersion: v1

kind: Service

metadata:

  name: welcome-app-service

spec:

  selector:

    app: welcome-app

  type: NodePort

  ports:

    - protocol: TCP

      port: 5000

      targetPort: 5000

      nodePort: 30007

**Step 7: Create another folder in VS code: (templates)**

**Step 8: Create a file inside templates folder (upload.html)**

**<!DOCTYPE *html*>**

**<html>**

**<head>**

**<title>Batch Predict</title>**

**</head>**

**<body>**

**<h2>Enter Iris features:</h2>**

**<form *action*="/predict\_form" *method*="post">**

**Sepal Length: <input *type*="text" *name*="sepal\_length"><br>**

**Sepal Width: <input *type*="text" *name*="sepal\_width"><br>**

**Petal Length: <input *type*="text" *name*="petal\_length"><br>**

**Petal Width: <input *type*="text" *name*="petal\_width"><br>**

**<input *type*="submit" *value*="Predict">**

**</form>**

**</body>**

**</html>**

**Step 8: Open Docker Desktop**

**Step 9: Build Docker image(cmd in VS)**

docker build -t welcome-app .

**Step 10: Run the Docker container**

docker run -d -p 5000:5000 welcome-app

Check that it is running:

docker ps

**Step 11: Test your Flask app**

* Home page: http://localhost:5000/ → “Welcome to Welcome-App!”
* About page: http://localhost:5000/about → Info text

**Step 12: Test the ML model**

In **Command Prompt**, run:

curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d "{\"features\": [5.1, 3.5, 1.4, 0.2]}"

Expected output:

{"prediction":[0]}

**Step 13:**

**python app.py**

**Step 14:**

**Open the form in browser.**

**http://localhost:5000/form**

### Ubuntu

cd ~/docker

sudo docker build -t ml-docker-app .

sudo docker images

sudo docker run -p 5000:5000 ml-docker-app

Click on the link and Check.

sudo ufw allow 5000

curl -X POST <http://localhost:5000/predict> \  
     -H "Content-Type: application/json" \  
     -d '{"features":[5.1,3.5,1.4,0.2]}'

The response would be 0 or 1 or 2