**Lab 9+10**

**Application deployment on Cloud**

Phase 1: Develop the ML Model

**Colab Notebook:**[**https://tinyurl.com/giki-ds-session1**](https://tinyurl.com/giki-ds-session1)

**Dataset:**[**https://tinyurl.com/giki-ds-session1-data**](https://tinyurl.com/giki-ds-session1-data)

1. Create a simple Flask API
2. Set Up AWS Free Tier Account
3. Deploy on EC2 Instance

**Launch an EC2 Instance**

* Go to EC2 dashboard
* Click "Launch Instance"
* Choose "Ubuntu Server 20.04 LTS (HVM)" (free tier eligible)
* Select "t2.micro" instance type
* Create new key pair (save the .pem file securely)
* Configure security group to allow HTTP (80), HTTPS (443), and custom TCP (5000)
* Launch instance

1. Connect to your EC2 Instance

**//Bash**

**chmod 400 your-key-pair.pem**

**ssh -i "your-key-pair.pem" ubuntu@your-ec2-public-dns**

1. Setup the Environment

**//Bash**

**# Update packages**

**sudo apt update && sudo apt upgrade -y**

**# Install Python and pip**

**sudo apt install python3 python3-pip -y**

**# Install required packages**

**pip3 install flask scikit-learn joblib gunicorn**

1. Transfer your files to EC2

**//bash**

**scp -i "your-key-pair.pem" app.py iris\_model.joblib ubuntu@your-ec2-public-dns:/home/ubuntu/**

1. Run the Application

**//bash**

**# Test running directly**

**python3 app.py**

**# For production, use Gunicorn:**

**gunicorn -w 4 -b 0.0.0.0:5000 app:app**

1. Set Up Reverse Proxy using Nginx

**//bash**

**sudo apt install nginx -y**

**sudo systemctl start nginx**

**sudo nano /etc/nginx/sites-available/your-app**

**//nginx**

**server {**

**listen 80;**

**server\_name your-ec2-public-dns;**

**location / {**

**proxy\_pass http://127.0.0.1:5000;**

**proxy\_set\_header Host $host;**

**proxy\_set\_header X-Real-IP $remote\_addr;**

**}**

**}**

1. Set Up Process Management

**//bash**

**sudo npm install pm2 -g**

**pm2 start gunicorn -- -w 4 -b 127.0.0.1:5000 app:app**

**pm2 startup**

**pm2 save**

1. Deployment Test

**//bash**

**curl -X POST http://your-ec2-public-dns/predict \**

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

**-d '{"features": [5.1, 3.5, 1.4, 0.2]}'**

TASK 2:

Take your manual 7:

Update the following files accordingly

* 1. requirement.txt

Flask==2.0.1

boto3==1.26.1

* 1. App.py

|  |
| --- |
| * from flask import Flask, render\_template, request, redirect, url\_for * import boto3 * from datetime import datetime * import os * app = Flask(\_\_name\_\_) * # AWS S3 Configuration * AWS\_ACCESS\_KEY\_ID = os.getenv('AWS\_ACCESS\_KEY\_ID') * AWS\_SECRET\_ACCESS\_KEY = os.getenv('AWS\_SECRET\_ACCESS\_KEY') * S3\_BUCKET\_NAME = 'your-bucket-name' # Create this in AWS S3 first * def save\_to\_s3(name, email): * s3 = boto3.client( * 's3', * aws\_access\_key\_id=AWS\_ACCESS\_KEY\_ID, * aws\_secret\_access\_key=AWS\_SECRET\_ACCESS\_KEY * ) * # Create a unique filename * timestamp = datetime.now().strftime("%Y%m%d%H%M%S") * filename = f"form-submissions/{timestamp}\_{name.replace(' ', '\_')}.txt" * # File content * content = f"Name: {name}\nEmail: {email}" * # Upload to S3 * s3.put\_object( * Bucket=S3\_BUCKET\_NAME, * Key=filename, * Body=content * ) * @app.route("/") * def home(): * return render\_template("index.html") * @app.route("/submit", methods=["POST"]) * def submit(): * name = request.form["name"] * email = request.form["email"] * save\_to\_s3(name, email) * return redirect(url\_for("home")) * if \_\_name\_\_ == "\_\_main\_\_": * app.run(host="0.0.0.0", port=5000) |

Update docker-compose.yml