Lab : Project Structuring, Version Control, Docker, and Azure Container Registry Integration

## Objective

In this lab, you will implement an end-to-end workflow for an AIOps project. You will:

- Create a modular and structured project folder on your Azure VM.  
- Develop a Python script to detect log anomalies.  
- Generate a sample log file for testing.  
- Initialize a Git repository and push your project to GitHub.  
- Create a Dockerfile to containerize your application.

## Environment and Tools

You will use the following resources and tools:

- Azure Virtual Machine (Ubuntu)  
- Azure Bastion  
- Azure Cloud Shell (Bash)  
- Git and GitHub  
- Docker  
- Python 3 and pip

You will also use the username and password provided by the trainer to access the Azure VM.

## Step 1: Connect to Your Azure VM Using Bastion

1. Go to https://portal.azure.com and log in.  
2. Navigate to Virtual Machines > aiops-dev-vm-<yourGitHubUsername>.  
3. Click Connect > Bastion > Use Bastion.  
4. Enter the provided username and password to access the terminal.

Explanation:  
Bastion provides secure access without exposing SSH ports to the internet.

## Step 2: Create the Project Folder Structure

Commands:  
mkdir -p ~/aiops-lab/{data,logs,scripts,models,docker}  
cd ~/aiops-lab

Explanation:  
Creates a structured folder tree where:  
- 'data' is for datasets  
- 'logs' is for system/application logs  
- 'scripts' holds Python scripts  
- 'models' for ML models  
- 'docker' for Docker configurations

## Step 3: Create a Sample Log File for Testing

Commands:  
cd ~/aiops-lab/logs  
nano sample.log

Paste the following:  
2024-01-01 12:00:00 INFO Application started  
2024-01-01 12:05:10 ERROR Connection timed out  
2024-01-01 12:06:22 WARNING Memory usage high  
2024-01-01 12:07:45 EXCEPTION Null pointer dereferenced  
2024-01-01 12:08:30 INFO Process completed

Explanation:  
Simulates a log file that includes anomalies like ERROR and EXCEPTION.

## Step 4: Write the Python Script for Log Anomaly Detection

Commands:  
cd ~/aiops-lab/scripts  
nano log\_anomaly\_detector.py

Paste the Python script:  
import sys

def detect\_anomalies(log\_file):  
 with open(log\_file, 'r') as f:  
 for line in f:  
 if 'error' in line.lower() or 'exception' in line.lower():  
 print(f"Anomaly Detected: {line.strip()}")

if \_\_name\_\_ == '\_\_main\_\_':  
 if len(sys.argv) != 2:  
 print("Usage: python log\_anomaly\_detector.py <log\_file\_path>")  
 sys.exit(1)  
 detect\_anomalies(sys.argv[1])

Test it:  
python3 log\_anomaly\_detector.py ../logs/sample.log

Expected Output:  
Anomaly Detected: 2024-01-01 12:05:10 ERROR Connection timed out  
Anomaly Detected: 2024-01-01 12:07:45 EXCEPTION Null pointer dereferenced

Explanation:  
Detects anomalies in log lines by searching for keywords.

## Step 5: Initialize Git and Push Project to GitHub

Commands:  
cd ~/aiops-lab  
git init  
git remote add origin https://github.com/<yourGitHubUsername>/aiops-lab-<yourGitHubUsername>.git  
echo "# AIOps Log Anomaly Detector" > README.md  
git add .  
git commit -m "Initial commit"  
git push origin main

Explanation:  
Pushes your local folder and versioned code to GitHub.

## Step 6: Create a Dockerfile

Commands:  
cd ~/aiops-lab/docker  
nano Dockerfile

Paste:  
FROM python:3.10-slim  
WORKDIR /app  
COPY ../scripts/log\_anomaly\_detector.py .  
COPY ../logs/sample.log /logs/sample.log  
ENTRYPOINT ["python", "log\_anomaly\_detector.py", "/logs/sample.log"]

Explanation:  
Creates a lightweight container to run the Python script.

## Step 7: Build Docker Image Locally

Commands:  
cd ~/aiops-lab  
docker build -t log-anomaly-detector:v1 -f docker/Dockerfile .

Explanation:  
Builds a local image using the Dockerfile.

## Lab Completion Checklist

- Connected to VM and created structured folders  
- Wrote and tested a Python log anomaly detector  
- Pushed project to GitHub with Git