Flask App Deployment (2-Tier DevOps Project)

This project demonstrates a complete **2-tier application deployment pipeline**
using **Flask + MySQL + Docker + Jenkins**.

It shows how a simple web application can be containerized, built using CI/CD,
and deployed automatically.

Architecture

Frontend / App Layer:
- Flask application
- HTML template UI

Database Layer:
- MySQL container
- message table initialization

DevOps Stack:
- Docker
- Docker Compose
- Jenkins Pipeline

Project Structure

```
.
├── app.py
├── Dockerfile
├── docker-compose.yml
├── Jenkinsfile
├── requirements.txt
├── message.sql
└── templates/
```

Let's go through the steps:

First create your EC2 instance. Then you should start with the steps.

Step 1: Install all dependencies: Git | Docker | Docker-compose

```
sudo apt-get update
sudo apt-get install git docker.io docker-compose-v2
```

Step 2: start and enable docker

```
sudo systemctl start docker
sudo systemctl enable docker
```

Step 3: add user to docker group(to run docker commands without sudo)

```
sudo usermod -aG docker $USER
newgrp docker
```

Step 4: install Jenkins

First install Java:

```
sudo apt update
sudo apt install fontconfig openjdk-21-jre
java -version
```

Then Jenkins: you can get the command from Jenkins official website itself:

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt update
sudo apt install jenkins
```

Step 5: Start and enable Jenkins

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

Step 6: Let's run the container without Jenkins to make sure all your images are build correctly

Your Docker file should look like this:

```
# Use an official Python runtime as the base image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# install required packages for system
RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y gcc default-libmysqlclient-dev pkg-config \
```

```
    && rm -rf /var/lib/apt/lists/*

# Copy the requirements file into the container
COPY requirements.txt .

# Install app dependencies
RUN pip install mysqlclient
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Specify the command to run your application
CMD ["python", "app.py"]
```

Your docker-compose.yml file should look like this

```
version: "3.8"

services:
mysql:
user: "${UID}:${GID}"
image: mysql:5.7
container_name: mysql
environment:
MYSQL_ROOT_PASSWORD: root
MYSQL_DATABASE: devops
MYSQL_USER: admin
MYSQL_PASSWORD: admin
volumes:
- ./mysql-data:/var/lib/mysql
- ./message.sql:/docker-entrypoint-initdb.d/message.sql
networks:
- twotier
healthcheck:
test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-proot"]
interval: 10s
timeout: 5s
retries: 5
start_period: 60s

flask-app:
build:
context: .
container_name: flask-app
ports:
- "5000:5000"
environment:
MYSQL_HOST: mysql
MYSQL_USER: root
```

```
MYSQL_PASSWORD: root
MYSQL_DB: devops
depends_on:
- mysql
networks:
- twotier
restart: always
healthcheck:
test: ["CMD-SHELL", "curl -f http://localhost:5000/health || exit 1"]
interval: 10s
timeout: 5s
retries: 5
start_period: 30s

networks:
twotier:
```

Then run the command:

```
docker compose up -d
```

Check the images, containers with these command

```
docker images
docker ps
```

If the build failed check the log

```
docker logs <container id \ name>
```

Now you can check if the app is working or not:

```
Get your EC2’s pubilc ip with the port 5000:5000
like - http://34.228.170.129:5000/
```

Now Let’s move to the Jenkins setup:

get the initial password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Grant the permissions:

```
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
```

Now login to jenkins:

Make sure your ec2 instance configured with the security rules.

Now set up your Jenkins:
Before anything add your GitHub cred to Jenkins.

- Manage Jenkins
- Credentials
- Add GitHub domain
- add your credentials

Now Create a pipeline:

- New item
- give a name, choose pieline, next
- In the triggers section choose GitHub hook trigger for GITScm pooling
- choose pipeline script from scm
- provide your repo link, use the credentials
- now build it, open console output to see what’s is going on

Congratulation your APP is deployed on AWS with Jenkins successfully.