Web Application Development and CI/CD Pipeline Implementation

Approach for Each Task

1. Setting Up the Web Application

I used **Python with Flask** to create a simple web application with a **/health** endpoint. This endpoint returns a **200 OK** status to confirm the app is running correctly.

Steps Taken:

- Set up a basic web server using Flask.
- Created a **/health** route that returns a JSON response.
- Tested the application locally to ensure it runs without errors.

2. Version Control

I used **Git and GitHub** to manage the source code efficiently.

Steps Taken:

- Initialized a Git repository using **git init.**
- Committed the initial code and pushed it to a **public GitHub repository.**

3. CI/CD Pipeline Setup

I configured a **Jenkins CI/CD pipeline** to automate the build, test, and deployment process.

Steps Taken:

- **Created a Jenkins job to perform the following steps:**
    - **Clone Repository:** Pull the latest code from GitHub.
    - **List Files:** Check the project structure.
    - **Install Dependencies:** Install required packages.
    - **Pull Docker Image:** Fetch the necessary image from DockerHub.
    - **List Docker Images:** Verify available Docker images.
    - **Run Docker Image:** Start the application in a container.
    - **Run Tests:** Ensure the /health endpoint works properly.
- Configured Jenkins to automatically trigger builds on code updates.

4. Deployment

I deployed the application on **AWS EC2.**

Steps Taken:

- Set up a deployment workflow in Jenkins.
- Configured environment variables and cloud service settings.
- Successfully deployed and accessed the **/health** endpoint.

5. Documentation

To ensure clarity I created a **README.md** with the following details:

- Project overview
- Steps to run the application locally
- Instructions for triggering the CI/CD pipeline
- Link to the deployed application
- Screenshots of successful execution

Challenges Faced and Resolutions

1. Pipeline Configuration Issues

Challenge: Jenkins builds failed due to missing dependencies.

Solution: Updated the pipeline script to install dependencies before running tests and deployment.

2. Cloud Deployment Errors

Challenge: AWS EC2 deployment failed due to incorrect authentication settings.

Solution: Generated new API keys and updated Jenkins credentials.

3. Testing Failures

Challenge: The **/health** endpoint returned a **404 error** due to incorrect routing.

Solution: Fixed the routing issue and corrected the endpoint configuration.

4. Debugging with ChatGPT

Challenge: Encountered various errors during the process.

Solution: Used ChatGPT to troubleshoot and resolve some of the issues efficiently**.**

Lessons Learned

- **CI/CD Makes Deployment Easier:** Automating build, test, and deployment saved time and effort.
- **Version Control Matters:** Structuring commits properly made code management smoother.
- **Debugging is Essential:** Checking logs helped in fixing cloud deployment issues.
- **Good Documentation Helps:** A well-structured README makes collaboration easier.

This assignment gave me hands-on experience in DevOps helping me understand CI/CD, cloud deployment, and troubleshooting in a real-world scenario.