
Devops-Assignment3

Warda Bibi 2020517

Wardah Tariq 2020519

# Introduction

In this project, we are streamlining the deployment process through automation, involving developers and a QA engineer. The goal is to create two dedicated servers: one for testing, accessible to developers and the QA engineer, and the other for staging, where clients can review changes.

The workflow begins with developers modifying the codebase to meet client requirements. Upon completion, they trigger a workflow that deploys the changes to the testing server. Here, the QA engineer and developers assess the codebase for any issues and ensure all features function correctly.

If no issues are identified, the QA engineer initiates another workflow to deploy all changes to the staging server. This enables the client to view the modifications and provide feedback. This iterative process continues whenever new features are developed or modifications are made to the codebase.

Both testing and staging servers run applications within Docker containers. The user can access the applications through port mapping, allowing mapping from port 3000 of the Docker backend application to port 3000 of the servers. Additionally, the backend application from Docker port 3000 is mapped to host port 8080. This approach enhances the efficiency of the development and testing lifecycle.

## 1-Repository Setup and Docker Image Creation

We initiate the process by cloning the repository and creating an empty repository on GitHub to host the assignment files. Our application, lacking a database, requires containers only for the backend and frontend. A Docker image, based on NodeJs, is crafted to host the application. The Docker Compose file, named "docker-compose.yaml," orchestrates and manages these containers. Two services, frontend and backend, are defined. Volumes are incorporated to synchronize local changes with containerized environments.

## 2-EC2 Instance  Setup

New EC2 instances are created for this assignment.

## 3-Workflow

The project is built using the command `npm run build-react`. To execute test cases on React, the workflow includes the command `npm run test-react`.

1. Starting the Server & Application
   - Launch EC2 instances to connect via public IPs.
   - Create a new branch (e.g., "Feature1") and modify the code.
   - Push changes to the "Feature1" branch on the "origin" server.

2. Pull Request and Testing Deployment
   - Generate a pull request from the "Feature1" branch, triggering the "Deploy To Testing" workflow.
   - Workflow success sends emails to developers and QA (Quality Assurance).
   - QA verifies testing changes deployed on the testing server.

3. Merge Changes and Staging Deployment
   - QA merges "testing" branch with "master" via pull request.
   - This triggers the "Deploy to Staging" workflow, deploying changes to the staging server.
   - Workflow success prompts an email to the QA.

4. Client Verification
   - The client views the application on the staging server by entering the server's public IP in a web browser.
   - Both the "testing" and "staging" servers run identical applications at this point.

This automated deployment process ensures efficient collaboration between developers, QA, and clients, with continuous integration and seamless version control.

## 5-Issues Faced:

**AWS Free Tier Limit:**

- Encountered challenges due to exceeding the AWS free tier limits.

**Deploying Docker Containers on Servers:**

- Challenges experienced during the deployment of Docker containers on servers.

# 6-FlowDaigram