Development Operations
Assignment # 3

MEMBER1 REG#: ________________ NAME: ________________________________

MEMBER2 REG#: ________________ NAME: ________________________________

COURSE CODE: CS423 INSTRUCTOR: MUHAMMAD SAJID ALI

TOTAL MARKS: 100

CI/CD Pipeline with Git, GitHub Actions and Docker

This assignment is an extension of your previous assignment to build Continuous Integration and Continuous Delivery (CI/CD) pipeline of *containerized applications*, an automated approach to software development. You'll set up CI/CD pipeline using the popular tools and Docker and see the benefits they bring to the DevOps pipeline. This will be a group assignment and each group should consist of *two people* only.

Instructions

Generally, there are four types of environments in the software development process such as Development, Testing, Staging and Production. The **Development environment**, also known as Dev, is where the software develops, and all the updates take place at first. The software in this environment is usually updated with all changes when developers commit to the Dev branch. Once the developer is satisfied then he/she commits changes to the testing branch which deploys all updates on **Testing environment** for QAs to test new and changed code whether via automated or non-automated techniques. Thus, testers can ensure the quality of the code by finding any bugs and reviewing all bug fixes. However, if all tests pass, the test environment can automatically move the code to the next deployment **staging environment**. This environment is nearly the same as the production environment to ensure the software works correctly. The target here is to test the complete software and observe how it will look on the production server. Finally, after all tests, the changes are deployed to the **Production environment** for end users.

You as DevOps engineer are asked to automate this process from development to deployment of software on these environments.

Task 1: Remember, in the previous assignment you cloned a simple open-source application from GitHub that was developed using ReactJs and NodeJS. (CLO5, PLO9) **(40 marks)**

1. As you and your team will be making further changes to this application so you must initialize this cloned repository code to your new repository and push it to your GitHub repository. Add a second team member as a collaborator to your GitHub project.
2. You must make your repository public and add me as a collaborator to that repository as well.
3. Deploy both frontend and backend services on local machine as separate **containers** named *frontend* & *backend*. Create a separate container for **the database** if it is being used by your application.

Task 2: Create or use existing three EC2 instances on AWS for Dev, Testing and Staging environment. **(0 marks)**

1. Please follow instructions from *Task 2* of your previous assignment in case you are creating new EC2 instances.

Task 3: Automating application from development to deployment. (CLO3, PLO5) **(60 marks)**

NOTE:

A. You must deploy application services in a docker container in each environment.
B. *Frontend* and *backend* should run as a service in a separate container. Create a separate container for **the database** if it is being used by your application.
C. Create Dockerfile for each of the application to be containerized.
D. Use docker compose to deploy a multi-container application. You must think of creating images, networks and volumes for your application where required.

2. You or your team members will always *create features* or *fix branches* from the *main branch* to add something new or update anything existing in future. Create a pull request, once done with the changes on the feature or fix branch. This pull request must trigger a workflow in your GitHub actions which will deploy changes for QA to test on AWS testing server. There should also be a button on your workflow to trigger this workflow without any pull requests if needed. QA should be able to access the deployed project using the link: http://<Instance IP>. Your workflow must do the following:

   1. Build your project: Compile the source code and build the application or software. This step ensures that the code can be successfully compiled.

   2. Unit Testing: Run unit tests to verify that individual components or functions of the code work as expected. Unit tests are typically fast and focused on specific functionalities. (you must choose a project which has a few unit test cases, or you must create your own).

   3. Code Analysis/Linting: Perform static code analysis or linting to check for code quality, adherence to coding standards, and potential issues. This step helps maintain a consistent and high-quality codebase.

   4. Notification: Your workflow must send an email to me and the developer who created a pull request if the workflow fails. Moreover, it must send an email to the QA (to me) if it successfully deploys your change on the instance with all details how the QA can access the deployed app feature for testing.

3. Assume that the QA validates all the changes made in a *Testing* environment and everything looks fine. Now let's say QA confirm the pull request and merges the changes into *master/main* branch. This push must trigger a workflow in your GitHub actions which will deploy changes for client and team members to on AWS staging server. There should also be a button on your workflow to trigger this workflow without any push event if needed. All users should be able to access the deployed project using the link: http://<Instance IP>. Your workflow must do all the tasks you did in your previous workflow.

What to Hand In

1. Submit the zip file of your project with all created workflows.
2.
3. Submit the word document that includes.
   A. A short introduction about the project and your assignment objective.
   B. List down any challenges you have faced while deploying.
   C. A complete flow diagram of what you did in the assignment.
   D. A diagram that explains the deployment of your containerized application.
   E. Include screenshots for each significant action performed in each task and subtask. Ensure that the screenshots are accompanied by appropriate captions under the relevant task or subtask heading. If necessary, provide brief details about the context or outcome captured in each screenshot. The aim is to create a visually appealing and informative report with a professional look.