Development Operations
Assignment # 2

MEMBER1 REG#: _____   NAME: _____

MEMBER2 REG#: _____   NAME: _____

COURSE CODE: CS423   INSTRUCTOR: MUHAMMAD SAJID ALI

TOTAL MARKS: 100

CI/CD Pipeline with Git and Jenkins

The objective of this assignment is to get hands on with Continuous Integration and Continuous Delivery (CI/CD), an automated approach to software development. You'll set up CI/CD pipeline using the popular tools 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: Clone any simple open-source application from GitHub built using ReactJs and NodeJS.   **(25 marks)**

1.   Carefully follow all the instructions inside the Readme file to deploy an application on your machine for a demo.

2.   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.
3.   You must put your repository private and add me as a collaborator to that repository.

Task 2: Create three EC2 instances on AWS for Dev, Testing and Staging environment.   **(20 marks)**

1.   Give appropriate name such that it represents environment
2.   Each environment Amazon Machine Image (AMI) should be *Ubuntu Server 22.04 LTS (HVM)*
3.   All instances must share one common security group such that if you add/remove any rule, it should affect all instances.
4.   While creating instance, install *jdk 11* (for Jenkins) and Apache server.
5.   Allow SSH and HTTP traffic communication for all instances

6. Dev instance will also be your Jenkins master instance and others will act as an agent.
7. Connect agents with Jenkins master using SSH

Task 3: Automating application from development to deployment.   **(55 marks)**

1. If the developer pushes changes to the *dev* branch, then Jenkins must fetch from *dev* branch and deploy it on *Dev* or *Development* environment.

2. If you as a developer are satisfied, then commit changes to the *testing* branch.

3. If the developer pushes changes to *the testing* branch, then Jenkins must fetch from *the testing* branch and deploy it on to the *Testing* environment.

4. Assume that the QA manually validates all the changes made in a *Testing* environment and everything looks fine then, QA should be able to trigger a job in Jenkins to merge changes into *master/main* branch.

5. If the changes are pushed to *master/main* branch, then Jenkins will fetch *master/main* branch and deploy on the *Staging* environment.

What to Hand In

1. Submit the zip file of your Jenkins backup.
2. Submit the word document that includes
   A. Screenshots of deployed application after cloning
   B. Screenshots of the last 5 commits you and your team member have made
   C. List down any challenges you have faced while deploying
   D. Screenshots of any one EC2 instance while launching
   E. Screenshots of the security groups inbound rules
   F. Screenshots of the jobs failed to build (at least two)
   G. Screenshots of the jobs build successfully (at least two)
   H. Screenshots of the pipeline build successfully