**Jenkins Assignments**

**Assignment 1:**

1. Create two servers in AWS:
   1. Jenkins-Server (Install Jenkins, Docker) -> t2.medium/t2.large
   2. Application-Server1 (Install Docker, Provide access to ubuntu user) -> t2.micro

[Note: docker run hello-world

If you get the permission denied error, after giving the accesses, please reboot the server -> sudo reboot -> Wait for 2 min]

1. In the Jenkins-Server after installing Docker and Jenkins, allow access to **Jenkins** and **Ubuntu** user to run the docker commands.
2. Create a pipeline.
   1. Fork the repository.
   2. Create Jenkinsfile.
   3. Stages:
      1. Clean Workspace
      2. Code Checkout
      3. Build Docker Image
      4. Push to DockerHub (You need to have a DockerHub account)
      5. Deploy to Server
3. Important Plugins to Install:
   1. Docker pipeline
   2. SSH Agent
   3. Pipeline: Stage View
4. Credentials to store:
   1. Dockerhub\_Credentials -> Username & Password Format
   2. SSH Key – Copy the whole SSH key. -> SSH Key with Username format

**Assignment 2:**

1. Create three servers in AWS:
   1. Jenkins-Server (Install Jenkins, Docker) -> t2.large
   2. Sonarqube-Server (Install SonarQube) -> t2.medium
   3. Application-Server1 (Install Docker) -> t2.micro
2. In the Jenkins-Server after installing Docker and Jenkins, allow access to **Jenkins** and **Ubuntu** user to run the docker commands.
3. Install the SonarQube in the Sonarqube-Server. Steps to follow:
   1. Update the packages (sudo apt update)
   2. Install the Java
   3. Wget the URL - <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.8.100196.zip>
   4. Install the Unzip package (sudo apt install unzip)
   5. Unzip the Zip file
   6. Go inside the bin folder -> Go inside the OS distribution -> ./sonar.sh console
   7. Login into Sonarqube -> username: admin, password: admin
   8. Set a new password.
   9. Create a manual project. -> Choose the steps.
   10. Save the Project Key and Project Name in a Notepad++.
4. Generate a token in SonarQube:
   1. Click on Profile icon -> Security.
   2. Click on Add and provide the name: Jenkins-Token, -> Global -> Expiry: 30 days.
   3. Click on Generate the copy the token to future reference.
5. Extra Steps to perform for the SonarQube:
   1. Click on Manage Jenkins -> Tools -> Add the tools in the SonarQube Scanner.
   2. Click on Manage Jenkins -> System -> Add the information in the SonarQube Server.
6. Create the pipeline.
   1. Trigger webhook from Github to Jenkins, so that whenever we push some code, it will automatically trigger the pipeline to build.
   2. Trigger webhook from SonarQube to Jenkins, so that Jenkins should know about the status of the Code Analysis, whether the Quality Gate passed or Failed.
   3. Stages:
      1. Clean Workspace
      2. Code Checkout
      3. Static Code Analysis -> SonarQube
      4. Sonarqube Quality Gate -> Trigger webhook at Jenkins
      5. Run Test Cases
      6. Build docker image
      7. Push to DockerHub. (Need DockerHub Account)
      8. Deploy to Server
7. Plugins to Install:
   1. Docker pipeline
   2. SSH Agent
   3. Pipeline: Stage View
   4. SonarQube Scanner
   5. SSH2 Easy
8. Credentials to Store:
   1. Dockerhub\_Credentials
   2. SSH-Key
   3. Sonar-Token -> The Token that we have generated in the Sonarqube, we need to store the token here.

**Assignment 3:**

1. We will use the above pipeline script to learn the **BUILD WITH PARAMETERS** concept.
2. Create 8 servers in AWS:
   1. Jenkins-Server
   2. SonarQube-Server
   3. Dev-Server-1
   4. QA-Server-1
   5. UAT-Server-1
   6. UAT-Server-2
   7. Prod-Server-1
   8. Prod-Server-2
   9. Prod-Server-3
3. We will create Parameters in Jenkins:
   1. **Environment**: [‘DEV’, ‘QA’, ‘UAT’, ‘Prod’]
   2. Based on the environment, the servers will be automatically populated, using the **Active Choices plugin**. [We need to install this plugin in Jenkins]. This will be check-box inputs, where user can choose the checkbox and it will be deployed into the same servers, where user have chosen to be deploy the application.
   3. After the installation of the Plugin, restart the Jenkins.
   4. **Branch**: [‘development’, ‘main’, ‘Other’]
   5. If **Others**, please specify the branch name. Why does this needed -> development branch will be deployed into DEV, main branch will be deployed into QA. Whenever any new release comes, we will create another branch and use that branch to deploy into UAT and Prod environment.
4. Once the setup has been done, we will test this pipeline in Jenkins.