JENKINS

1] Create a Continuous Integration Pipeline with Jenkins and Github

Lab scenario

You are working for a team that is struggling to scale their finance product because their software releases are taking a lot of time. You have been asked to create an appropriate solution that can quickly build and test any new code changes added to the codebase by developers. The architecture group thinks that building a Continuous Integration Pipeline using Jenkins and Github will allow the developers to easily build and test their code with continuous scaling, which will lead to faster software releases . Your boss has asked you to build a proof-of-concept.

Objectives

* Implement a continuous integration pipeline
* Integrate GitHub with Jenkins.
* Set up Git Commit Status to push the status of a build to developers

Requirement

* Understanding of CI/CD Pipelines
* Basic understanding of Git and Linux CLI
* Basic understanding of Jenkins.

What you'll learn

* Creating Private Git Repositories in GitHub.
* Configuring GitHub Repository with Jenkins
* Continuous integration pipeline

**Your project assignment**

Hi SHARUKHAN! Thanks for working on this project for our team.

The Development team has raised an issue. All the developers are storing their code locally and are running the test in a local environment. Running the overall integration tests in locally takes a lot of time and are special build environment needs to be set up and maintained in their local workstation. This approach is a time-consuming task for the developers and also the overall software release cycle. They would like to just push the code to a central Git repository and the rest should be taken care of automatically by a more stable pipeline.

I want you to create a Continues Integration Pipeline using GitHub and Jenkins that can achieve this. Developer should also be able to see in GitHub whether their commits to the branch has failed or succeeded in the tests that were run by Jenkins.

All that the developers must be required to do is to commit the application code in the Git repository, and the rest steps of building code and reporting the failure status in GitHub must be taken care of by the pipeline service automatically.

Developers have provided you with a sample application code to build a POC pipeline. The application code is stored in demo.cpp file. This code is based on C++. Developers have also provided you with the necessary commands required to compile the code in a Linux environment, these instructions can be found in the file base-prerequisite.txt

**Tasks**

**1] Sign-Up for a New GitHub Account**

You have to sign up for a free Github account to be able to create a new private Git repository**.**

### 2] Create a New Private Git Repository

Create a new private Git repository where an application code can be stored.

**3] Commit the Sample C++ Code to Git Repository**

The sample C++ code that the developers have provided to you needs to be committed as part of the private Git repository. Commit the contents of the demo.cpp file within your existing repository.

### 4] Create New Personal Access Token in GitHub

Create a Personal Access Token for Jenkins to be able to authenticate to GitHub.

token : - ghp\_DsuIOM3HWXFpkKLt7VQlyco9zzePXU0DvGn8

**5] Configure Jenkins**

Jenkins should be able to connect to the GitHub repository and update the commit status (success/failure) as well as poll the repository. Configure Jenkins with appropriate GitHub credentials and Test the Connections

### 6] Create a new Freestyle Project in Jenkins

Create a new Jenkins Freestyle project to set up a continuous integration pipeline with an appropriate source, build, and post-build stages.

### 7] Testing Build Failure Scenario

Test a failure scenario in the build process and verify the outcome and whether Jenkins updates the appropriate commit status in Git repository.

**8] Clean Up**