### **Assignment: Continuous Integration (CI) with Jenkins**

#### **Task Overview:**

In this task, you will set up a Continuous Integration (CI) environment using Jenkins. This includes installing and configuring Jenkins, creating Jenkins jobs, and integrating Jenkins with a version control system. Additionally, you will create a CI/CD pipeline using Jenkinsfile, configure parameterized builds, build triggers, and set up automated testing and notifications. Finally, you will initialize a Git repository, commit your Jenkins configurations, and share the Git URL.

#### **Instructions:**

##### **1. Set Up Jenkins**

Install Jenkins:

* Install Jenkins on your local machine or a cloud server (e.g., AWS, Azure, GCP).
* Document the installation process with screenshots and notes on any issues encountered and how they were resolved.

Initial Configuration:

* Complete the initial setup wizard.
* Install suggested plugins.
* Create an admin user and configure basic security settings.
* Document each step with screenshots and explanations.

##### **2. Explore Jenkins Plugins**

Explore and Install Essential Plugins:

* Research and install at least 5 plugins that extend Jenkins functionality. These should include:
  + Git Plugin
  + Pipeline Plugin
  + Email Extension Plugin
  + JUnit Plugin
  + Any other plugin of your choice
* Document the purpose and usage of each plugin with screenshots.

##### **3. Create and Configure Jenkins Jobs**

Create a Freestyle Project:

* Create a simple "Hello World" Jenkins freestyle job.
* Configure the job to run a shell script that prints "Hello, World!" to the console.
* Document the job creation process with screenshots and script details.

Parameterized Builds:

* Modify the "Hello World" job to accept a parameter (e.g., a name) and print "Hello, [name]!".
* Document the changes and provide screenshots showing the job running with different parameters.

Build Triggers:

* Configure the job to run every 5 minutes using cron syntax.
* Document the configuration with screenshots and explain how build triggers work.

##### **4. Integrate Jenkins with Version Control Systems**

Configure Jenkins with Git:

* Set up a new Git repository on GitHub or any other Git hosting service.
* Create a simple project (e.g., a Java application) in the repository.
* Configure Jenkins to clone the repository and build the project.
* Document the configuration process with screenshots and details of any issues encountered.

Jenkins Pipeline as Code:

* Create a Jenkinsfile in your Git repository that defines the CI/CD pipeline.
* The pipeline should include stages for building, testing, and deploying the application.
* Configure Jenkins to use the Jenkinsfile and run the pipeline.
* Document the Jenkinsfile content and pipeline execution with screenshots and explanations.

##### **5. Putting It All Together**

Create a Comprehensive CI Job:

* Create a new Jenkins job that pulls code from your Git repository, runs tests, and deploys the application.
* Use parameters to allow different deployment environments (e.g., dev, staging, production).
* Set up email notifications to alert you of build successes or failures.
* Document the job configuration and execution process with screenshots.

Automated Testing:

* Integrate a testing framework (e.g., JUnit for Java projects) into your Jenkins pipeline.
* Ensure that tests are run automatically as part of the build process and that results are reported.
* Document the setup and results with screenshots and explanations.

##### **6. Initialize Git and Push to Git**

Initialize Git Repository:

* Initialize a Git repository in your project directory using git init.
* Add all Jenkins configuration files to the repository using git add ..
* Commit the changes to the repository using git commit -m "Initial commit".
* Create a remote repository on a Git hosting service like GitHub, GitLab, or Bitbucket.
* Add the remote repository URL using git remote add origin <remote\_repository\_url>.
* Push the commits to the remote repository using git push -u origin master.

#### **Deliverables:**

1. Documentation:
   * A detailed report covering all parts of the assignment.
   * Include screenshots, code snippets, and explanations for each step.
   * Discuss any challenges faced and how you overcame them.
2. Source Code:
   * Provide the link to your Git repository containing the project and Jenkinsfile.
3. Jenkins Configuration:
   * Export and include any relevant Jenkins job configurations.
4. Git Repository URL:
   * Share the URL of the Git repository containing all Jenkins configurations.

#### **Submission Guidelines:**

Submit all Jenkins configuration files, documentation, and Git repository URL through the provided platform or repository. Ensure that your documentation is comprehensive and well-structured, providing clear explanations of each CI/CD component and its configuration.

#### **Evaluation Criteria:**

* Correctness and completeness of the Jenkins configurations.
* Proper utilization of Jenkins plugins and adherence to best practices in CI/CD.
* Effective setup of Jenkins jobs and pipeline.
* Proper configuration of version control integration and automated testing.
* Successful initialization and configuration of Git repository.
* Clarity and quality of documentation, including detailed explanations and organization.

#### **Useful Links:**

* Jenkins Installation
* Jenkins Plugins
* Jenkins Freestyle Project
* Jenkins Pipelines
* Jenkins Git Integration

Good luck with your assignment!