**DOCUMENT ON GIT AND JENKINS**

**1.Introduction**

**1.1 Overview**

In today’s software development lifecycle, continuous integration and continuous delivery (CI/CD) are essential for delivering quality software efficiently.

Two powerful tools used to achieve this are Git and Jenkins.

Git is a distributed version control system that helps teams track changes in source code, collaborate efficiently, and maintain version history.

Jenkins is an open-source automation server used to automate building, testing, and deploying applications.

Together, Git and Jenkins form the backbone of modern DevOps practices.

**2. Git Workflow**

**2.1 What is Git?**

Git allows multiple developers to work on the same project simultaneously without overwriting each other’s work. It stores snapshots of code versions and enables easy rollback or branching.

**2.2 Git Workflow Diagram**

developer

Working directory

Staging Area

Local Repo

Remote Repo

**2.3 Git Commands and Explanation**

|  |  |
| --- | --- |
| **Command** | **Description** |
| git init | Initializes a new Git repository. |
| git add . | Adds all files to the staging area. |
| git commit -m “message” | Commits the changes to the local repository. |
| git remote add origin<URL> | Connects local repo to remote (GitHub). |
| git push -u origin main | Pushes local commits to the GitHub repository. |
| git status | Shows current status of working directory. |

**2.4 Steps to Perform Git Operations**

1. Open Git Bash or Command Prompt in your project folder.

2. Run the following commands:

   In git bash:

   git init

git config --global user.name “Srihitha”

git config –global user.email “Sai.Srihitha.Reddy.Karnati@gds.ey.com”

   git add .

   git commit -m "Initial commit"

cd “Path to git repository”

git gui

3. Verify files on GitHub.

4. All the files are pushed into the repository.

**3. Jenkins Workflow**

**3.1 What is Jenkins?**

Jenkins is a Continuous Integration (CI) tool that automates the building and testing of software projects, enabling developers to detect problems early.

It can be integrated with Git to automatically trigger builds whenever new code is pushed to the repository.

**3.2 Jenkins Workflow Diagram**

Developer pushes code

GitHub Repo

Jenkins CI/CD

Server builds and tests app

Test Results and artifacts

**3.3 Jenkins Setup Steps**

**1. Download and Install Jenkins**

   \* Go to [<https://www.jenkins.io/download>](<https://www.jenkins.io/download>)

   \* Install using the Windows Installer package.

   \* Once installed, open Jenkins Dashboard via [<http://localhost:8080>](<http://localhost:8080>)

**2. Unlock Jenkins**

   \* Copy the admin password from the file path shown.

   \* Paste it to log in for the first time.

**3. Install Suggested Plugins**

   \* Select install suggested plugins.

   \* Wait for installation to complete.

**4. Create a New Job**

   \* Click New Item → Enter a project name (e.g., Git Jenkins Project)

   \* Select Freestyle Project → Click \*OK\*

**5. Configure Git Repository**

   \* User Source Code Management, choose \*\*Git\*\*

   \* Enter your GitHub repository URL (e.g. `<https://github.com>/<username>/<repo>.git`)

   \* Provide credentials if your repo is private

**6. Add Build Step**

   \* Under Build Steps, select Execute Windows batch command

   \* Add commands like:

     bash

     echo "Building the project..."

**7. Save and Build**

   \* Click Save

   \* Click Build Now

   \* Check Console Output for successful build logs.

**3.4 Jenkins GUI Explanation**

|  |  |
| --- | --- |
| **Section** | **Purpose** |
| Dashboard | Shows all jobs and build status. |
| Configure | To set up Git repository and build steps. |
| Build Now | Manually triggers a build. |
| Console Output | Displays build logs in real-time. |
| Workspace | Stores code fetched from GitHub. |

**4. Integration of Git and Jenkins**

Git stores source code and tracks version history.

Jenkins connects to the Git repository to automatically pull the latest code and execute builds.

You can configure a poll SCM trigger (e.g., `H/5 \* \* \* \*`) or a webhook from GitHub so Jenkins automatically builds whenever new code is pushed.

**Workflow Summary:**

1. Developer commits and pushes code to GitHub.

2. Jenkins detects the change.

3. Jenkins pulls the code and runs the build.

4. Jenkins provides build/test results.

**5. Flow of Operations**

**Developer → Git Bash → GitHub → Jenkins → Build → Test Results → Deployment**

This automation ensures:

\* Early detection of bugs

\* Faster integration

\* Continuous feedback

\* Reliable software delivery

**6. Advantages**

|  |  |
| --- | --- |
| **Git** | **Jenkins** |
| Tracks changes and versions | Automates build/test process |
| Enables teamwork and branching | Supports CI/CD |
| Open source and lightweight | Integrates with 1000+ plugins |
| Ensures code reliability | Improves delivery speed |

**7. Conclusion**

Through this mini project, we learned how Git helps in version control and how Jenkins automates the process of building and testing software.

Together, they establish a strong foundation for Continuous Integration (CI) and Continuous Delivery (CD), enabling faster and more reliable software development.