

Jenkins

Why Jenkins is Required in Development?

Continuous Development: Continuously monitor the Git source code repository and create a new build if changes are detected.

Continuous Deployment: Retrieve the latest build from the Git repository and deploy it to the testing environment.

Continuous Delivery: Retrieve the latest build from the Git repository and deploy it to the UAT environment.

Why Jenkins is Required in Automation?

Continuous Integration: Execute Selenium test scripts in the testing environment.

1. Installation Steps

a. Download and Install Jenkins

b. Precondition 1: Install Plugins

- Navigate to:

Jenkins → Manage Jenkins → Manage Plugins → Select "Available"

- Search and install the following plugins:

- Maven Integration Plugin
- GitHub Integration Plugin
- Pipeline Plugin

c. Precondition 2: Configure Environment Variables

- Navigate to:

Jenkins → Manage Jenkins → Global Tool Configuration

- Set the following paths:

- **JDK:** *C:\Program Files\Java\jdk-21*
- **Maven:** *D:\apache-maven-3.9.10-bin\apache-maven-3.9.10*
- **Git:** *C:\Program Files\Git\bin\git.exe*

2. Use of Jenkins in Testing

Jenkins is a Continuous Integration (CI) tool that monitors framework builds in Git and triggers actions based on changes.

Advantages of Jenkins

a. Three Execution Levels:

- **On Demand:** Start execution manually based on user request.
- **On Schedule:** Automatically trigger execution at scheduled times.
- **Poll SCM:** Continuously monitor the SCM (e.g., GitHub) and trigger execution upon new builds or test scripts.

b. Email Notifications:

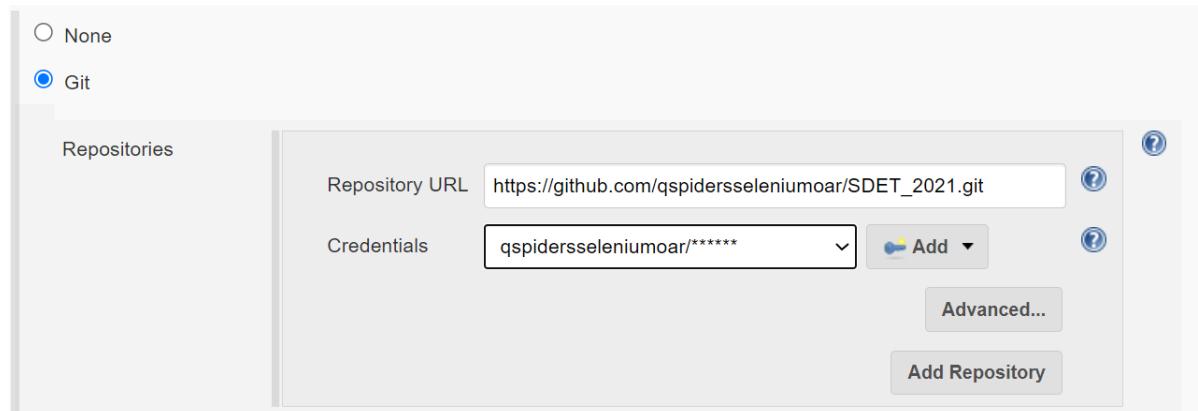
- Sends execution reports upon completion.
 - Alerts users if builds break due to compilation errors.
 - c. **Runtime Parameters:** Supports dynamic input during job execution.
 - d. **Pipeline Jobs:** Enables sequential execution of multiple jobs.
-

3. How to Create a Job in Jenkins

1. Log in to Jenkins.
 2. Click **New Item**.
 3. Select **Maven Project** and enter a job name.
 4. Click **OK**.
-

4. Linking Selenium Framework (GitHub) to Jenkins

1. Log in to Jenkins.
 2. Navigate to the desired job.
 3. Click **Configure**.
 4. Under **Source Code Management**, select **Git** and enter repository details.
 5. Under **Build**, specify the *pom.xml* path and goals (e.g., *clean test*).
 6. Click **Save**.
-



⇒ Scroll down , go to build Division & provide the POM.xml path & goal

The screenshot shows the Jenkins job configuration interface. In the 'Build' section, the 'Root POM' field is set to 'SeleniumFramework/pom.xml' and the 'Goals and options' field is set to 'test'. There is an 'Advanced...' button at the bottom right.

⇒ Click on “save” button

5. Executing a Jenkins Job

1. Log in to Jenkins.

2. Open the job.
3. Click **Build Now**.

6. Viewing Job Results

1. Navigate to the job.
2. Under **Build History**, click the latest build.
3. Select **Console Output** to view execution logs.

7. Scheduling a Job

1. Navigate to the job's **Configure** page.
2. Check **Build Periodically**.
3. Enter a cron schedule (e.g., `0 2 * * *` for daily at 2 AM).
 - o **Format:** `MINUTE HOUR DOM MONTH DOW`
 - **MINUTE:** 0-59
 - **HOUR:** 0-23
 - **DOM:** 1-31 (Day of Month)
 - **MONTH:** 1-12
 - **DOW:** 0-7 (0 and 7 = Sunday)



8. Configuring Poll SCM

1. Navigate to the job's **Configure** page.
2. Check **Poll SCM**.
3. Enter a cron schedule (e.g., `*/5 * * * *` to check every 5 minutes).

What is a Jenkins Pipeline?

A pipeline executes multiple jobs sequentially (e.g., Build → Deploy → Test → Deliver).

Note: Ensure the **Pipeline Plugin** is installed.

Example Pipeline:

Job 1 (Build) → Job 2 (Deploy) → Job 3 (Smoke Test) → Job 4 (Regression Test) → Job 5 (Delivery)

