Bachelor of Technology

Computer Engineering Sem : 7

01CE0717 – DevOps Essentials

Marwadi University
Marwadi Chandarana Group

NAAC A+

Unit : 3 – Continuous Integration Using Jenkins

Content

- Install & Configure Jenkins, Jenkins Architecture Overview, Creating a Jenkins Job, Configuring a Jenkins job,

- Introduction to Plugins, Adding Plugins to Jenkins, Commonly used plugins (Git Plugin, Parameter Plugin, HTML Publisher, Copy Artifact and Extended choice parameters),

- Configuring Jenkins to work with java, Git and Maven, Creating a Jenkins Build and Jenkins workspace

Introduction to Jenkins

- Jenkins is a open source automation tool which allows contiguous integration (CI). It is written in Java.
- Jenkins build and test our software projects continuously which becomes easy for developers to integrate the changes in the project.
- Jenkins can integrate all types of development lifecycle processes including build, document, test, package, deploy and so on.
- Jenkins is a popular tool in DevOps community because it allows to automate the software development process, thereby reducing the cost of manual testing and increasing quality.

Features of Jenkins

- Easy Installation
- Easy configuration
- Free open source
- Available plugin
- Easy distribution

Introduction to Jenkins

Advantages of Jenkins

- Free to use
- Build automation
- Rich plugin ecosystem
- Platform independent
- User friendly and easy to install

Disadvantages of Jenkins

- Developer centric
- Maintenance overhead
- Changes in the settings creates problem

Install and Configure Jenkins

- Installing Jenkins on Windows
- Download Jenkins MSI Installer:
  - Goto the website: https://www.jenkins.io/download/
- Go to the Jenkins download page and select the Windows link to download the MSI installer for the latest version of Jenkins.
- Install Java Development Kit (JDK):

Install and Configure Jenkins

- Download and install JDK 8 from the OpenJDK project. Note the installation directory.
- Set Environmental Variables:
- Open the Jenkins web interface and go to the configure system page. Add the Java Home variable with the path to the JDK installation directory.
- Run the Jenkins Installer:

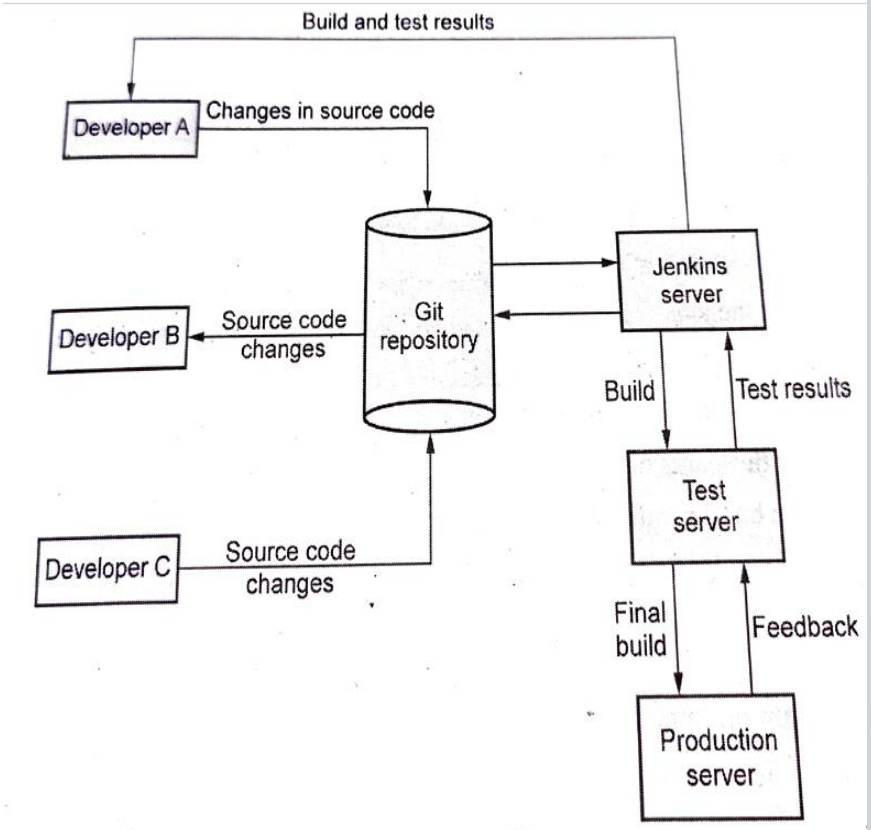
Install and Configure Jenkins

- **Configure Jenkins:**
- Open Jenkins in a web browser at http://localhost:8080.
- Enter the randomly generated password from C:\Program Files\Jenkins file jenkins.err.log
- Install suggested plugins by clicking the Install suggested plugins button.
- Configure the Jenkins administrator details and save the changes.
- Start using Jenkins by clicking the Start using Jenkins button

Jenkins Architecture Overview

Jenkins Workflow

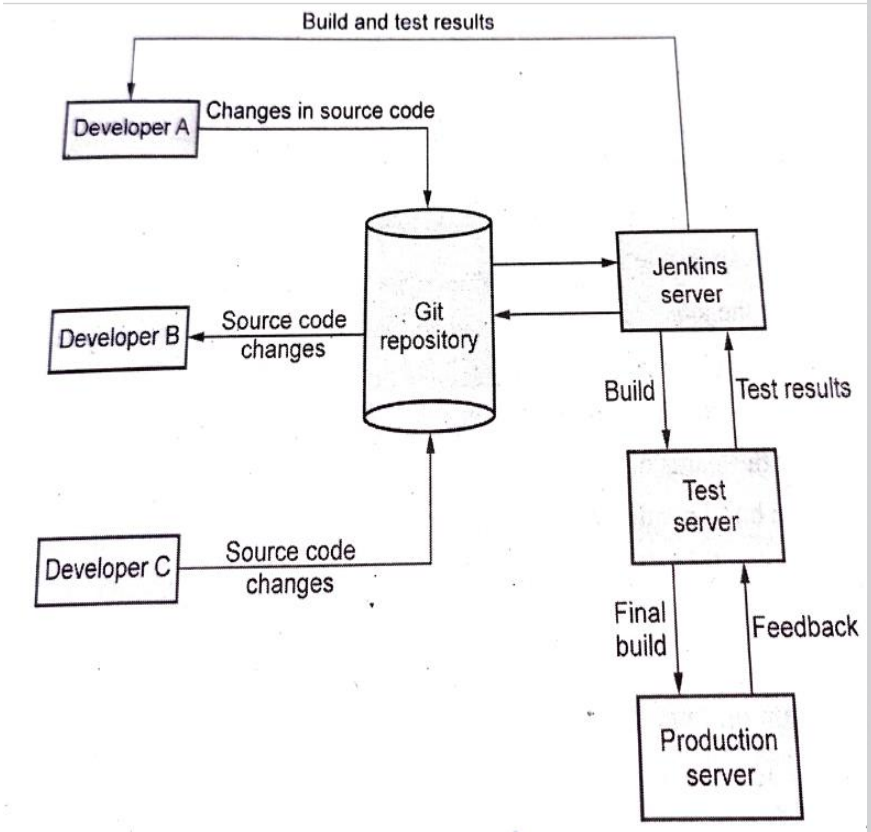
- Developers **commit the code** to source code **repository** (GitHub).
- The Jenkins CI server checks the repository at regular intervals and pulls any newly available code.
- The Build Server **builds** the code into an executable file.
- If the build fails, feedback is sent to the developers.
- If the build is successful, then Jenkins server deploys the build on **test server**.



Jenkins Architecture Overview

Jenkins Workflow

- The test server tests the build and generates the feedback. The Jenkins server gets this feedback and notifies about build and test results to developers.
- If everything is perfect, then build is deployed on **production server**.
- In all the above activities, Jenkins server continuously verifies the code repository for changes made in the source code. The above activities are repeated continuously.

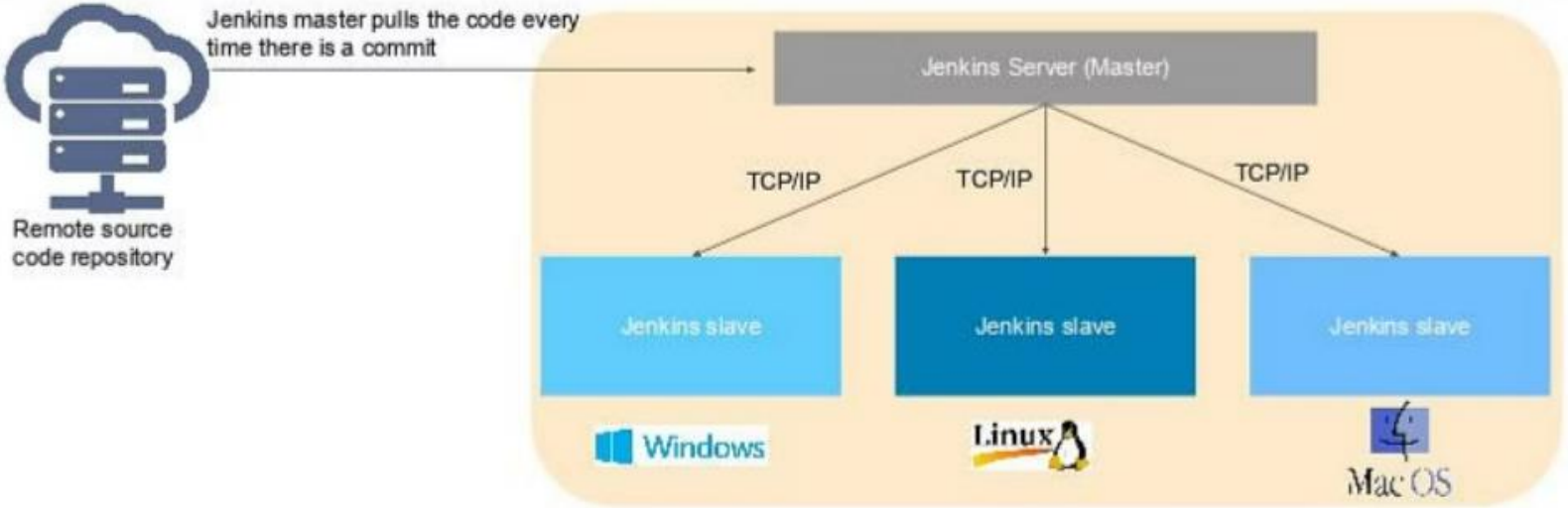


Jenkins Architecture Overview

- Jenkins architecture is distributed architecture. It manages the distributed builds using Master-Slave architecture.
- The TCP/IP protocol is used to communicate between Master and Slave.
- The Jenkins architecture comprises of two components:
  1) Jenkins Master/Server
  2) Jenkins Node/Slave/Build Server

Jenkins Architecture Overview

Jenkins Master-Slave Architecture



- Jenkins master distributes its workload to all the slaves
- On request from Jenkins master, the slaves carry out builds and tests and produce test reports

Jenkins Architecture Overview

Jenkin Master

- The Master is responsible for –
  i. Scheduling the jobs
  ii. Assigning them to slaves
  iii. Sending builds to slaves for execution
  iv. To monitor the status of every slave
  v. Retrieve the build results from slaves
  vi. Display the results obtained from slaves on console.

Jenkins Slave

- It run on remote server. The building jobs dispatched by the Master are executed by slaves.
- Jenkins slaves are compatible with all the Operating Systems. The Jenkins slave can be configured on any server including Windows, Linux and Mac.

Creating Jenkins Job

- Access Jenkins Dashboard:
- Open a web browser and navigate to http://localhost:8080.
- Create a **New Item**:
- Click on New Item at the top left-hand side of the Jenkins dashboard.
- Enter Item Details:
- Enter the name of the item you want to create. For this example, we will use "HelloWorld".
- Select **Freestyle project** as the project type.
- Configure **Source Code Management**:
- Under Source Code Management, select Git.

Creating Jenkins Job

- Enter the URL of your Git repository. For example, if your repository is located at https://github.com/your_username/HelloWorld.git, enter this URL.
- Add **Build Steps**:
- Go to the Build section.
- Click Add build step and select **Execute Windows batch command** (if you are using Windows) or Execute shell command (if you are using Linux/macOS).
- Enter the commands you want to execute during the build process. For example, to compile and run a Java program, you can use:
- **Save** the Job:

Creating Jenkins Job

- Click **Apply** and then **Save** to save the job configuration.
- Build the Job:
- Click the **Build Now** button to run the job.
- Monitor the Build:
- After the build is complete, you can view the build status under Build History.
- Click on the **build number** to see the console output and verify the success of the build.

Configuring Jenkins Job

- **General Settings**
- **Name:** Provide a descriptive name for the job.
- **Types of Jobs:**
  1) **Freestyle project:** Jenkins freestyle projects allow users to automate simple jobs, such as running tests, creating and packaging applications, producing reports, or executing commands. Freestyle projects are repeatable and contain both build steps and post-build actions.
  2) **Maven project:** For maintaining and building the Maven projects using Jenkins, this type of job is used.
  3) **Pipeline:** The pipeline job is used when working on long running activities.
  4) **Multi-configuration project:** If require multiple configuration, the select this job.
  5) **GitHub organization:** If your project is hosted on GitHub, this job type allows Jenkins to automatically scan and build repositories within a GitHub organization.

Configuring Jenkins Job

- Description: Add a brief description of the job's purpose.
- Discard Old Builds: Configure how many old builds to keep.
- GitHub project: If the job is for a GitHub project, provide the project URL.
- Source Code Management
- Repository Type: Select the source control system, e.g., Git, Subversion, etc.
- Repository URL: Provide the URL of the source code repository.
- Credentials: If the repository is private, add the necessary credentials.

Configuring Jenkins Job

- **Build Triggers**
  - **Trigger builds remotely:** If you want to trigger your project built from anywhere anytime then you should select this option and provide an ***authorization token***.
  - **Build after other projects are built:** If your project depends on another project build then you should select this option and must specify the project name in ***projects to watch*** field section.
  - **Build periodically:** If you want to schedule your project build periodically then select this option.
    - You have to enter the schedule for periodic building using ***cron expression***.

Configuring Jenkins Job

- ***cron expression***:
- Each line consists of 5 fields separated by TAB or whitespace. Cron is command-line utility that allows the developers and system administrators to schedule the tasks that run in the background at regular intervals.
  - MINUTES – (0-59)
  - HOURS – (0-23)
  - DAYMONTH – (1-31)
  - MONTH – (1-12)
  - DAYWEEK – (0-7) where 0 and 7 are Sunday
  - The ***** specifies all the valid values
  - M-N defines a range of values
  - M-N/X or */X – defines the steps by intervals of X through the specified range

Configuring Jenkins Job

- ***cron expression***:
  - A, B, …., Z – enumerates multiple values
  - An empty line that start with a # symbol is treated as a comment

minute hour day (month) month day (week)

Example:

```
#every single minute
* * * * *

#every fifteen minute
*/15 * * * *
```

https://crontab.guru/examples.html

Configuring Jenkins Job

- **Build Triggers**
  - **GitHub hook trigger for GITScm polling:** A webhook is an HTTP callback, an HTTP POST that occurs when something happens through a simple event-notification via HTTP POST. GitHub wewbhooks in Jenkins are used to trigger the build whenever a developer commits something to the branch, on GitHub.
  - **Poll SCM:** Poll SCM periodically polls the SCM to check whether changes were made and builds the project if new commits were pushed since the last build.

Configuring Jenkins Job

- Poll SCM: Periodically check the repository for changes and trigger a build.
- GitHub hook trigger: Trigger a build when a GitHub push event occurs.
- Scheduled: Run the build on a schedule using a cron-like syntax.
- Build Environment
- Use secret text(s) or file(s): Inject sensitive data like API keys or passwords.
- Prepare an environment for the run: Set environment variables.
- Build
- Build Steps: Add the steps to build, test, and deploy the project.
- Execute shell commands
- Run Maven targets

Configuring Jenkins Job

- Invoke Ant targets
- Run system commands
- Post-build Actions
- Archive the artifacts: Save build artifacts for later use.
- Publish JUnit test result report: Publish test results.
- Publish to a Git repository: Push build artifacts to a Git repository.

Configuring Jenkins Job

- Trigger a parameterized build on other projects: Trigger downstream jobs.
- Advanced Project Options
- Restrict where this project can be run: Specify which agents can execute the job.
- Quiet period: Delay the start of a build for a specified time.
- Retry Count: Retry a failed build a specified number of times.

Introduction To Plugins

- Plugins are the primary means of enhancing Jenkins functionality to suit specific needs.

- There are over a thousand different plugins available that can integrate various build tools, cloud providers, analysis tools, and more.

- Plugins are packaged as self-contained ***.hpi*** files containing the necessary code, images, and resources.

Introduction To Plugins

Benefits of using Plugins

- ***Extended functionality:*** Plugins can be used to add new features to Jenkins. Many plugins are open-source and maintained by the Jenkins community. This means you can benefit from the contributions and improvements made by a large user base.

- ***Task automation:*** It can be used to automate various tasks related to software development such as building, testing, deployment of library files to executable environment.

Introduction To Plugins

Benefits of using Plugins

- ***Scalability:*** We can add or remove the plugins as per the requirements of our project and thereby Jenkins meets the requirements of organization.

- ***Security:*** Jenkins plugins are useful to make the job secure by integrating with security tools, vulnerability scanners.

- ***Continuous Improvement***: Jenkins is an automated tool. Developers continuously create new plugins or improve the existing ones. Thus continuous improvement is made with latest technologies.

Introduction To Plugins

Benefits of using Plugins

- ***Increased flexibility:*** Plugins can be used to increase the flexibility of Jenkins. For example,
  - Amazon EC2 plugin allows to deploy the application from cloud platform
  - Git plugin allows you to deploy the application from GitHub repository.

  There are various ways by which the desired activity can be carried out in Jenkins.

Introduction To Plugins

- Installing Plugins
- Plugins can be automatically downloaded with dependencies from the Jenkins Update Center.
- The simplest way to install plugins is through the Manage Jenkins > Plugins view in the web UI.
- Plugins can also be installed using the Jenkins CLI install-plugin command

Adding plugins to jenkins

- Using the Jenkins Plugin Manager
- Access the Jenkins Dashboard: Open a web browser and navigate to the Jenkins dashboard, typically at http://localhost:8080.
- Manage Jenkins: Click on the "Manage Jenkins" link on the left-hand side.
- Manage Plugins: In the "Manage Jenkins" section, click on the "Manage Plugins" option.
- Available Plugins: This will open the Jenkins Plugin Manager. Click on the "Available" tab to see the list of plugins that can be installed.

Adding plugins to jenkins

- Search and Select Plugins: Use the search bar to find the plugins you want to install. Select the checkboxes next to the plugins you want to install.
- Install Plugins: Click the "Install without restart" button to install the selected plugins. Jenkins will download and install the plugins.
- Restart Jenkins: If prompted, restart Jenkins to complete the installation.

Adding plugins to jenkins

Using the Jenkins CLI

Install the Jenkins CLI: Download the Jenkins CLI JAR file from the Jenkins website.

Run the Install Command: Use the following command to install a plugin:
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin <plugin-name>

Replace <plugin-name> with the name of the plugin you want to install.

Install Dependencies: If the plugin has dependencies, Jenkins will automatically install them.

Restart Jenkins: Restart Jenkins to complete the installation.

Commonly used plugins

1. **Git plugin**

2. **Docker plugin**

3. **Amazon EC2 plugin**

4. **SonarQube plugin:** It is an open-source tool used for continuous code quality inspection. The Jenkins monitoring plugin allows you to integrate SonarQube into Jenkins so that you can easily analyze a code while running a Jenkins job that comes with SonarQube execution.

5. **Jira plugin:** It is an open-source plugin that integrates Jenkins with Atlassian Jira Software, enabling the DevOps teams more visibility into the development pipeline.

Commonly used plugins

1. Git plugin for Jenkins

- Among the extensive list of plugins for Jenkins, the Git plugin holds a prominent position.
- As the name suggests, it facilitates essential git functions for Jenkins projects.
- It offers Git operations such as pulling, fetching, checking out, branching, listing, merging, tagging, and pushing repositories.
- Git plugin has the functionality, performance, **security**, and flexibility that the DevOps teams need.

Commonly used plugins

1. **Git plugin for Jenkins**

- It serves as a Distributed Version Control DevOps tool that supports distributed non-linear workflows by providing data assurance for developing quality software.

- Moreover, it enables access to GitHub as a Source Code Management (SCM) system, which acts as a repository browser for many other providers.

Commonly used plugins

2. Parameter plugin for Jenkins

- The Parameter Plugin allows users to define parameters that can be passed to Jenkins jobs. This enables dynamic job configuration based on user input or other criteria.
- **Types of Parameters:**
- The plugin supports various parameter types, including:
  - String Parameter
  - Boolean Parameter
  - File Parameter
  - Choice Parameter
  - Credentials Parameter
  - Multi-line String Parameter
  - Password Parameter
  - Run Parameter

Commonly used plugins

The Html publisher plugin:

- **Purpose:**
- The HTML Publisher Plugin enables Jenkins to display HTML reports generated by your build processes directly in the Jenkins user interface.

- This is particularly useful for showing test results, code coverage reports, and other documentation.

- **Key Features:**
- Publish HTML Reports: It allows you to specify one or more HTML directories to be published as part of the build results.

- Customizable Reports: You can customize the report's title and specify which index file to display as the landing page.

Commonly used plugins

The Html publisher plugin:

Multiple Reports: Supports publishing multiple HTML reports from different directories in a single job.

Usage:
After installing the plugin, you can configure it in your Jenkins job by adding a post-build action to publish HTML reports.

Should specify the directory where the HTML files are located and the index file that should be displayed.

Commonly used plugins

Copy Artifact plugin:
The Copy Artifact plugin allows you to copy artifacts (files generated by a build) from one Jenkins job to another. This is particularly useful for sharing build outputs between jobs in a pipeline.

Key Features:
Artifact Selection:
You can specify which artifacts to copy based on various criteria, such as build number, timestamp, or specific file patterns.

Parameterized Builds: The plugin supports parameterized builds, allowing you to dynamically determine which artifacts to copy based on input parameters.

Commonly used plugins

Copy Artifact plugin:

Integration with Pipelines: It can be easily integrated into both freestyle projects and pipeline jobs.

Usage:

After installing the plugin, you can add a build step in your job configuration to "Copy artifacts from another project".

You will need to specify the project name, the criteria for selecting the build, and the artifacts to copy.

Commonly used plugins

Extended Choice Parameter Plugin

Purpose:
The Extended Choice Parameter plugin enhances the parameterization capabilities of Jenkins jobs by allowing more complex parameter types beyond the standard options.

Key Features:
Multi-Select:
Allows users to select multiple options from a list.
Checkboxes and Radio Buttons: Provides options to display parameters as checkboxes or radio buttons.

Commonly used plugins

Extended Choice Parameter Plugin

File Parameter: Enables users to upload files as parameters.

Dynamic Choices: Can generate choices dynamically from a script or a URL
.

Usage:

After installing the plugin, you can add extended choice parameters in the job configuration under the "This project is parameterized" section.

You can specify the type of parameter (e.g., multi-select, checkbox) and provide the options for users to choose from.

Configuring Jenkins to work with java

Step 1: Install Java

Download Java:
Visit the Oracle JDK download page or AdoptOpenJDK to download the appropriate version of the JDK for your operating system.

Install Java:
Follow the installation instructions for your operating system (Windows, macOS, or Linux).

Set Environment Variables (Windows):
Set the JAVA_HOME environment variable to point to the JDK installation directory.
Add %JAVA_HOME%\bin to the PATH environment variable.

Verify Installation:
Open a command prompt and run java -version to ensure Java is installed correctly.

Configuring Jenkins to work with java

Step 2: Install Jenkins

Download Jenkins:
Download the Jenkins installer from the Jenkins website.

Install Jenkins:
Follow the installation instructions for your operating system.

Start Jenkins:
After installation, start Jenkins and access it via http://localhost:8080 in your web browser.

Configuring Jenkins to work with java

Step 3: Configure Jenkins to Use Java

Manage Jenkins:
Go to Manage Jenkins > Global Tool Configuration.

Add JDK:
Under the JDK section, click Add JDK.
Provide a name (e.g., "JDK 11") and specify the path to the JDK installation.
You can also check the "Install automatically" option if you want Jenkins to manage the installation.

Setting up Git in Jenkins

Install Git Plugin:
Go to Manage Jenkins > Manage Plugins and install the "Git Plugin".

Configure Git:
Go to Manage Jenkins > Global Tool Configuration.
Under the Git section, configure the path to your Git installation.

Setting up Maven in Jenkins

Install Maven Plugin:
Go to Manage Jenkins > Manage Plugins and install the "Maven Integration" plugin.

Configure Maven:
Go to Manage Jenkins > Global Tool Configuration.
Under the Maven section, configure the path to your Maven installation.

Creating a Jenkins build and Jenkins workspace

Step 1: Install Jenkins

Download Jenkins:
Visit the Jenkins website and download the appropriate installer for your operating system.
Install Jenkins:
Follow the installation instructions for your OS. After installation, Jenkins typically runs on http://localhost:8080.

Step 2: Create a New Jenkins Job

Access Jenkins Dashboard:
Open your web browser and navigate to http://localhost:8080.
Create a New Item:
Click on the New Item link on the left-hand side of the Jenkins dashboard.
Select Project Type:
Enter a name for your project and select Freestyle project or Pipeline, then click OK.

Creating a Jenkins build and Jenkins workspace

Step 3: Configure the Job
For a Freestyle Project:

General Configuration:
Under the General tab, you can add a project description.
Source Code Management:
If using Git, select Git and provide the repository URL.

Build Triggers:
Configure triggers if you want the build to start automatically (e.g., Poll SCM).

Creating a Jenkins build and Jenkins workspace

Step 3: Configure the Job

Build Steps:

Under the Build section, click Add build step and choose Execute Windows batch command or Execute shell.

Enter the commands you want Jenkins to execute (e.g., for a Maven build, you might enter mvn clean package).

Post-build Actions:
You can add actions such as archiving artifacts or sending notifications.

Creating a Jenkins build and Jenkins workspace

Step 4: Configure Jenkins Workspace

Workspace Location:

Jenkins creates a workspace for each job by default. You can find the workspace at JENKINS_HOME/jobs/<job-name>/workspace.

You can configure a custom workspace directory in the job configuration under the Advanced Project Options section by specifying a different path.

Creating a Jenkins build and Jenkins workspace

Step 5: Build the Project

Build Now:

After configuring the job, click the Build Now link on the left side of the job page to start the build.

View Build Output:

Click on the build number in the Build History section to view the console output and check for any errors.

THANK YOU