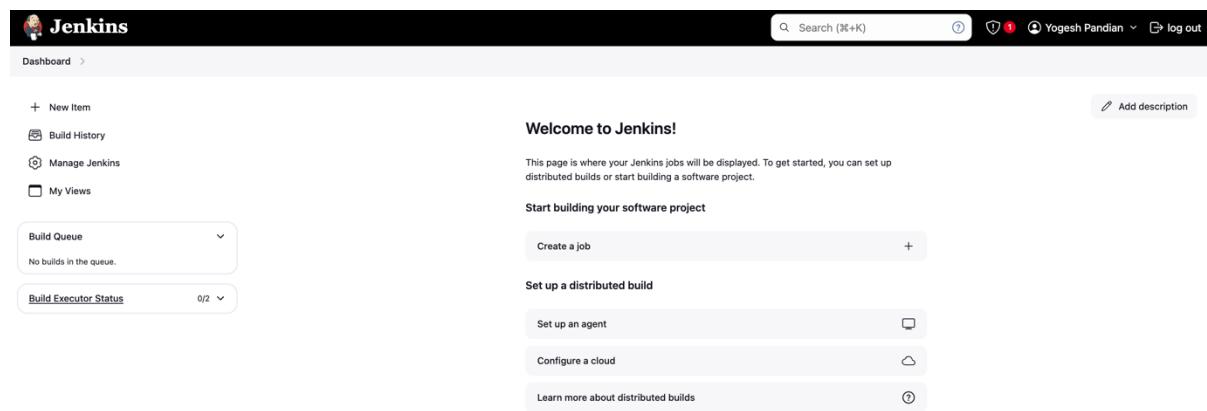


Jenkins Configuration Guide for Testers

One-Time Configuration

After installing Jenkins, open the Jenkins dashboard and ensure Jenkins is running. To begin, access **Manage Jenkins** from the main dashboard.

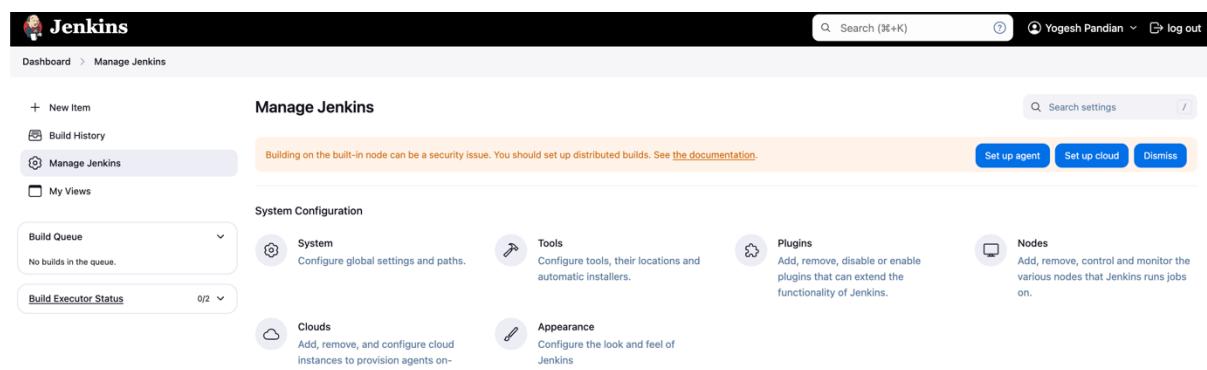


The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, search bar (Search (⌘+K)), user info (Yogesh Pandian), and log out button.
- Left Sidebar:** Links for New Item, Build History, Manage Jenkins (highlighted in orange), and My Views.
- Build Queue:** Shows "No builds in the queue."
- Build Executor Status:** Shows "0/2".
- Welcome to Jenkins! Section:** Title "Welcome to Jenkins!", subtext "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.", and a "Start building your software project" section with links for "Create a job", "Set up a distributed build", "Set up an agent", "Configure a cloud", and "Learn more about distributed builds".
- Right Sidebar:** "Add description" link.

Manage Jenkins

Within **Manage Jenkins**, you'll find various configuration options. Select **System Configuration** to access important settings. From here, go to **Plugins** to configure the necessary tools and integrations.



The screenshot shows the Manage Jenkins page with the following elements:

- Header:** Jenkins logo, search bar (Search (⌘+K)), user info (Yogesh Pandian), and log out button.
- Left Sidebar:** Links for New Item, Build History, Manage Jenkins (highlighted in orange), and My Views.
- Build Queue:** Shows "No builds in the queue."
- Build Executor Status:** Shows "0/2".
- Manage Jenkins Section:** Title "Manage Jenkins", subtext "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).", and buttons for "Set up agent", "Set up cloud", and "Dismiss".
- System Configuration Section:** Sub-sections for System, Tools, Plugins, Clouds, Appearance, and Nodes.

Installing Necessary Plugins

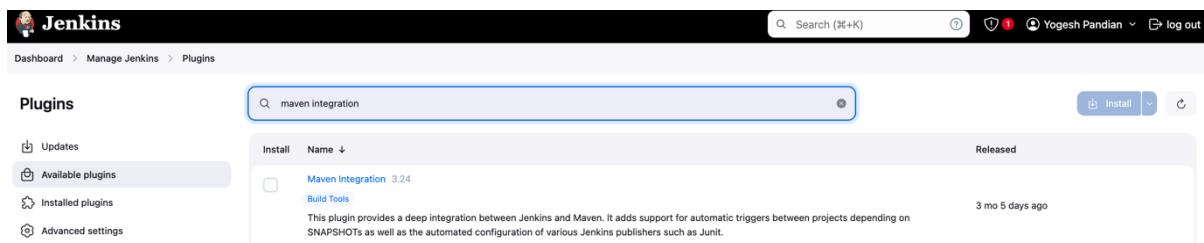
In Jenkins, plugins are essential for integrating third-party tools such as **Maven**, **Git**, and **JDK**. These tools are crucial for managing your build process and connecting Jenkins to external systems like **GitHub**.

Key Plugins for Test Automation:

- **Maven**: Manages the build lifecycle of your project.
- **Git**: Handles version control.
- **GitHub**: Integrates with GitHub repositories for source code management.
- **JDK**: Configures Java for running your build and automation tasks.

To install plugins:

1. Go to **Manage Jenkins > Plugins**.
2. Under the **Available** tab, search for the **Maven Integration Plugin**.
3. Select the plugin and click **Install**.



Same way install all mentioned key plugins. Once the plugins are installed, ensure you've set the correct paths for **Maven**, **Git**, and **JDK**. These paths are necessary for running your automation scripts in Jenkins.

Configuring Tool Paths

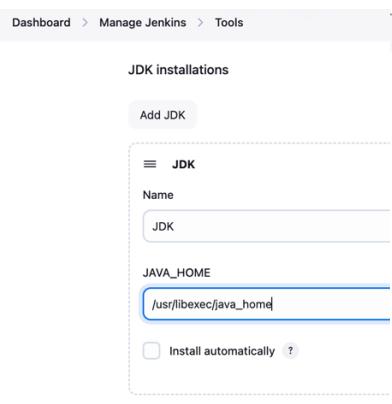
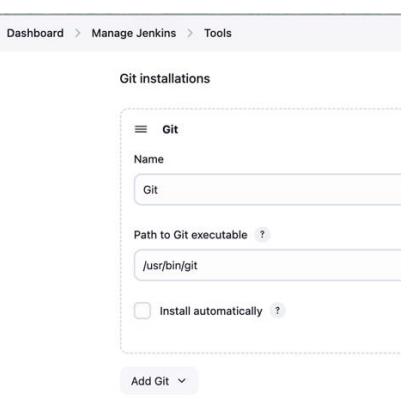
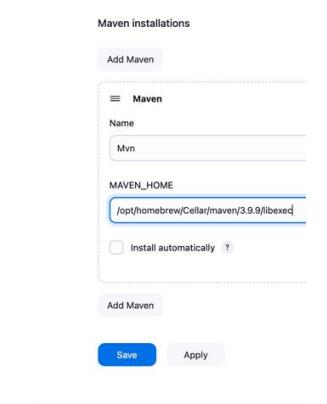
Now that the necessary plugins are installed, you need to set the paths for **Maven**, **Git**, and **JDK**. This step is essential for Jenkins to recognize these tools.

To do this, navigate to **Manage Jenkins > System Configuration** and click on **Tools**. Here, you can configure the locations for each tool.

Example Paths:

- **JDK Path:**
`/opt/homebrew/opt/openjdk@17/libexec/openjdk.jdk/Contents/Home`
- **Maven Path:**
`/opt/homebrew/Cellar/maven/3.9.9/libexec`
- **Git Path:**
`/usr/bin/git`

These paths are essential for Jenkins to locate the tools required to run your automation.

JDK Path	Git Path	Maven Path
		

Summary of One-Time Setup

So far, completed the following setup steps:

- Installed Jenkins and configured the initial setup.
- Installed key **plugins**, including Maven, Git, GitHub, and JDK.
- Set the **paths** for Jdk, Maven, and Git in **System Configuration > Tools**.

Running a Maven Project from GitHub

Now that Jenkins is set up, let's configure it to run a Maven project from GitHub.

Creating a New Jenkins Job

1. On the **Jenkins Dashboard**, click **New Item**.
2. Select **Maven Project** and provide a name for your project.

New Item

Enter an item name

RestAssuredFramework

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Configuring Source Code Management

1. In the project configuration page, scroll down to **Source Code Management**.
2. Select the **Git** option.
3. Enter the **GitHub repository URL** for your project:
<https://github.com/AutomatedMind/RestAssuredFrameworkPractice.git>

The screenshot shows the Jenkins project configuration interface. The left sidebar has links for General, Source Code Management (which is selected), Build Triggers, Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The main panel is titled 'Source Code Management' and shows the 'Git' option selected. It includes fields for 'Repository URL' containing 'https://github.com/AutomatedMind/RestAssuredFrameworkPractice.git' and a 'Credentials' dropdown set to '- none -'. There is also an 'Advanced' section at the bottom.

Setting Up the Build Command

1. Scroll to the **Build** section.
2. Under **Goals and Options**, enter the following command:
`test`
This tells Maven to run the `mvn test` command, which will execute the tests defined in the `pom.xml` file.
3. Save the configuration.

Build

Root POM ?
pom.xml

Goals and options ?
test

Advanced ▾

Project Configuration Summary

Here's a quick recap of the steps to configure the project:

- Create a new **Maven Project** and provide a name.
 - In **Source Code Management**, select **Git** and provide the **GitHub project URL**.
 - Under **Build**, add the Maven command test in the **Goals and Options** field.
 - Click **Save** to complete the configuration.
-

Running the Project

After configuring your project, it will appear on the **Jenkins Dashboard**. To trigger a build:

1. Click on the project name to open the project's specific dashboard.
2. From there, click **Build Now** to run the project.

Project Dashboard

The screenshot shows the Jenkins dashboard with the following details:

- Header:** Jenkins logo, Search bar (Search (⌘+K)), User icon (Yogesh Pandian), Log out button.
- Left Sidebar:** New Item, Build History, Manage Jenkins, My Views. A "Build Queue" section indicates "No builds in the queue." A "Build Executor Status" section shows "0/2" executors.
- Main Content:** A table view for the "RestAssuredFramework" project. The columns are S (Status), W (Workload), Name (RestAssuredFramework), Last Success (N/A), Last Failure (N/A), and Last Duration (N/A). There is a "More" button (>) at the end of the table row.
- Bottom:** A footer with icons for S, M, L and three dots.

Project Specific Dashboard

The screenshot shows the Jenkins Project Specific Dashboard for the Maven project 'RestAssuredFramework'. The top navigation bar includes links for 'Dashboard', 'RestAssuredFramework', 'Status' (which is currently selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Maven project', 'Modules', and 'Rename'. A search bar at the top right contains the placeholder 'Search (⌘+K)'. On the right side, there are buttons for 'Add description' and 'log out'. The main content area is titled 'Maven project RestAssuredFramework'. It features a sidebar with options: 'Builds' (selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Maven project', 'Modules', and 'Rename'. Below the sidebar, a 'Permalinks' section has a link to 'Add description'. The 'Builds' section displays a table with three rows:

Build	Duration
#5	02:05
#4	02:00
#3	01:29

The first build (#5) is marked with a green checkmark and is expanded, while the others are collapsed. The expanded view shows the build number, duration, and a dropdown arrow.

Viewing Build Details

After triggering the build, you can view important details about the build process, such as:

- Who executed the build
- When the build was executed
- The time it took to complete
- The overall build status

Click on the **Console Output** to see the detailed logs and any errors that occurred during the build.

Build Details

The screenshot shows the Jenkins Build Details page for build #5. The top navigation bar includes the Jenkins logo and the path: Dashboard > RestAssuredFramework > #5. On the left, a sidebar lists various build actions: Status (highlighted), Changes, Console Output, Edit Build Information, Delete build '#5', Timings, Git Build Data, Test Result, Redeploy Artifacts, See Fingerprints, and Previous Build. The main content area displays the build status as green with a checkmark, indicating success (#5 (29 Jan 2025, 02:05:53)). It shows the build was started by user Yogesh Pandian. The run spent information includes waiting, build duration, and total time from scheduled to completion. The Git details show revision 45048a03a123c1b08a2e2a3ce0d3248a8c2767ad and repository https://github.com/AutomatedMind/RestAssuredFrameworkPractice.git. The Test Result section indicates no failures. A summary table for Module Builds shows one entry for RestAssuredFramework with a green checkmark and 16 sec duration.

Module	Status	Duration
RestAssuredFramework	Success	16 sec

Console Output

```
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 14.10 s -- in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[JENKINS] Recording test results
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  16.375 s
[INFO] Finished at: 2025-01-29T02:06:20+05:30
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /Users/apple/.jenkins/workspace/RestAssuredFramework/pom.xml to org.example/RestAssuredFramework/1.0-SNAPSHOT/RestAssuredFramework-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

Checking Build Status

Once the build completes, you'll see a green checkmark if the build passed. If it fails, you can find detailed error messages in the **Console Output** to help diagnose the issue.

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. Below the navigation bar is a search bar and user information for 'Yogesh Pandian'. A central table displays a single project: 'RestAssuredFramework'. The table has columns for Status (S), Warning (W), Name (sorted), Last Success, Last Failure, and Last Duration. The project 'RestAssuredFramework' is listed with a green circle icon, a cloud icon, and the name 'RestAssuredFramework'. Its last success was 2 min 56 sec ago, build #5. Its last failure was 8 min 16 sec ago, build #4. The last duration was 30 sec. Below the table are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2).

With these steps, we can successfully set up Jenkins to run Maven projects from GitHub. Now we know how to **configure Jenkins**, **install necessary plugins**, **link your GitHub repository**, and **run your automation tests** using **Maven commands**.