

Lesson-End Project

Sending Test Reports Using Email Notification

Project agenda: To configure Jenkins pipeline, which compiles code, execute JUnit

Description: Imagine a development team using Jenkins for automated email notifications of test reports, improving code quality and speed. Upon code commits, Jenkins triggers a pipeline that compiles, tests, and emails test results via an SMTP setup. This quick feedback loop is vital for agile practices, ensuring that developers immediately identify and address issues.

Tools required: Jenkins

Prerequisites: You must have Jenkins and GitHub access in the lab to proceed.

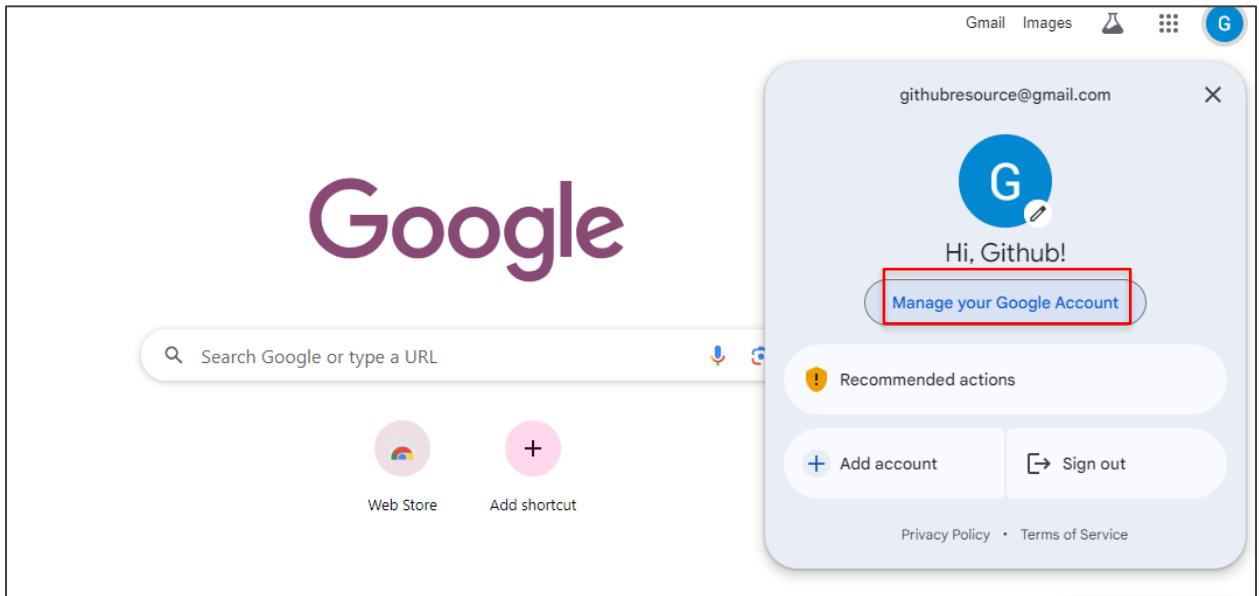
Expected deliverables: A fully configured Jenkins pipeline that compiles code, executes JUnit tests, and sends detailed test reports via email upon each build's completion

Steps to be followed:

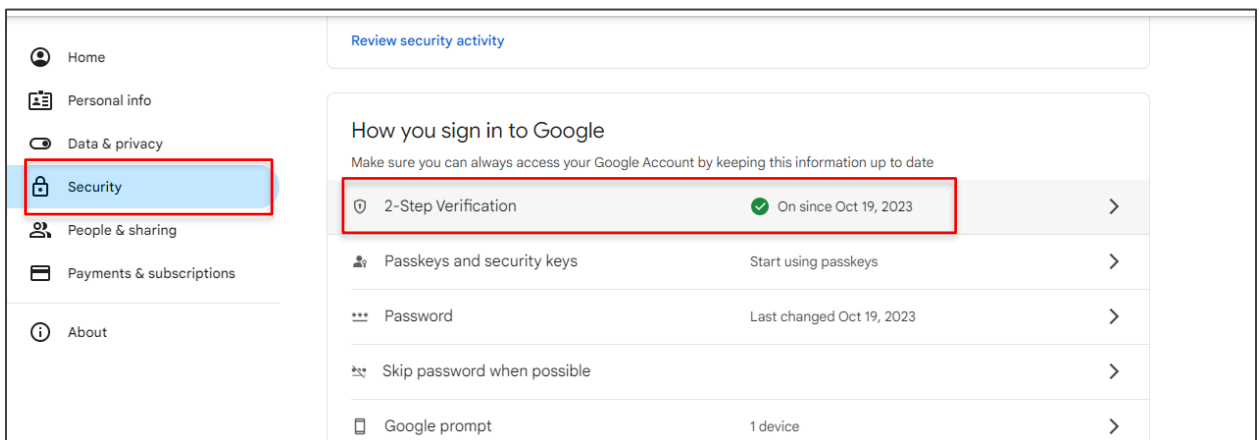
1. Create an app password for SMTP configuration
2. Configure SMTP configurations in Jenkins
3. Configure Maven in Jenkins

Step 1: Create an app password for SMTP configuration

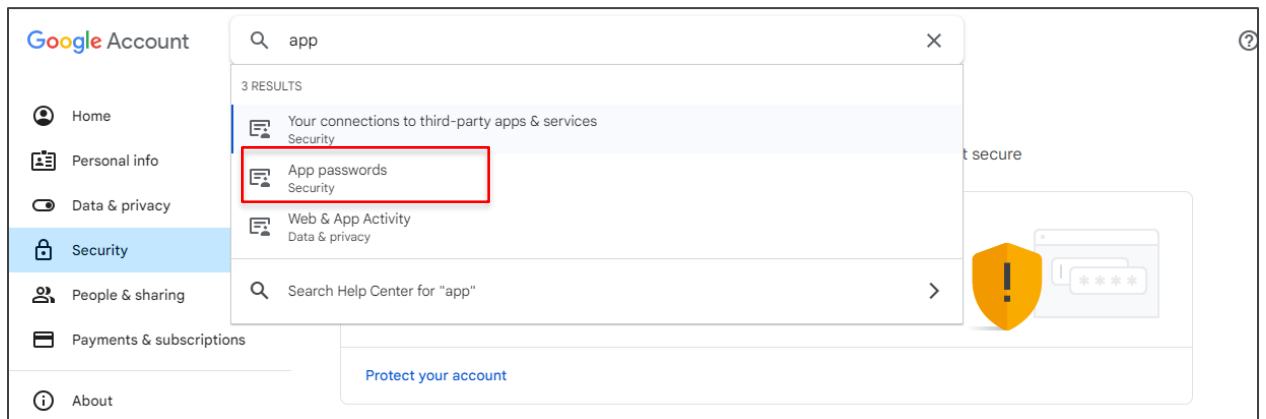
1.1 Navigate to your Google account and click on **Manage your Google Account**



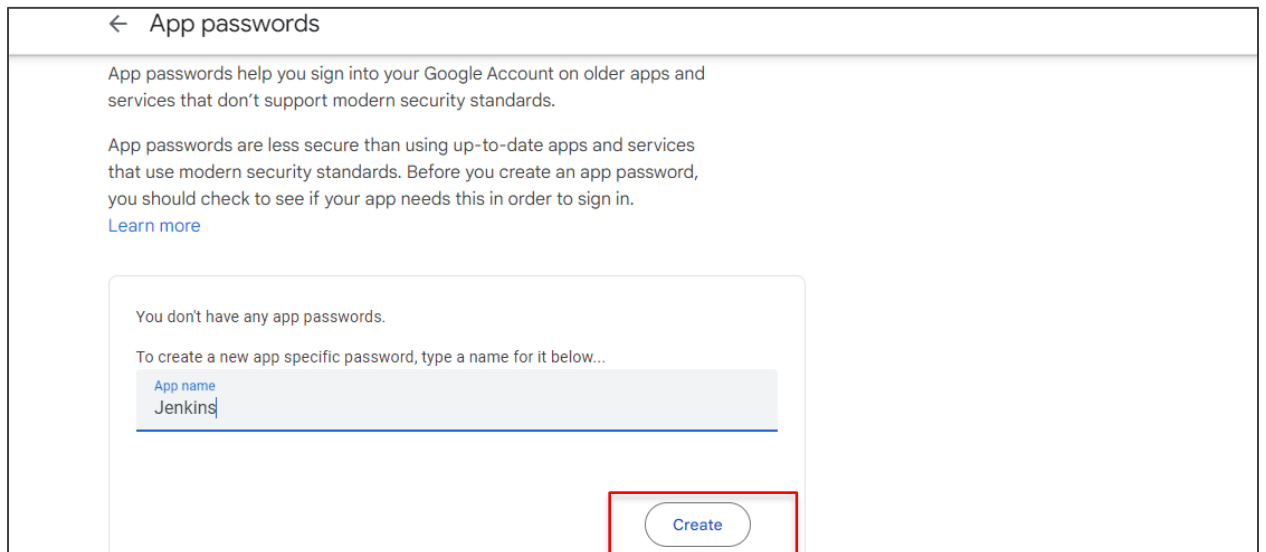
1.2 Now, navigate to **Security** and make sure you have **2-Step Verification** enabled



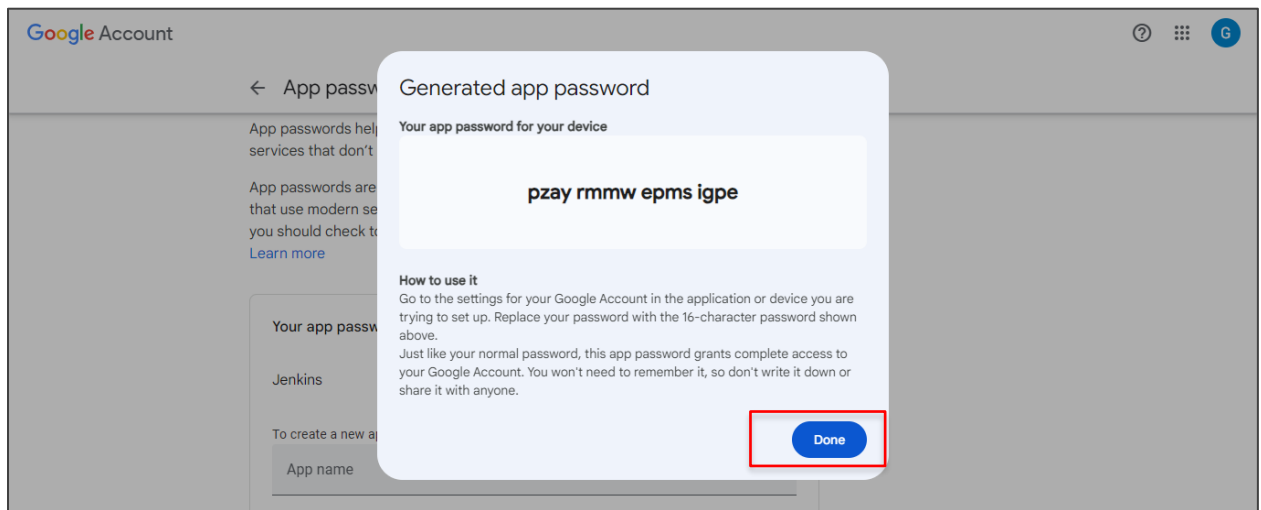
1.3 In **Security**, search for **App passwords** and click on it



1.4 Provide the app name as **Jenkins** and click on **Create**

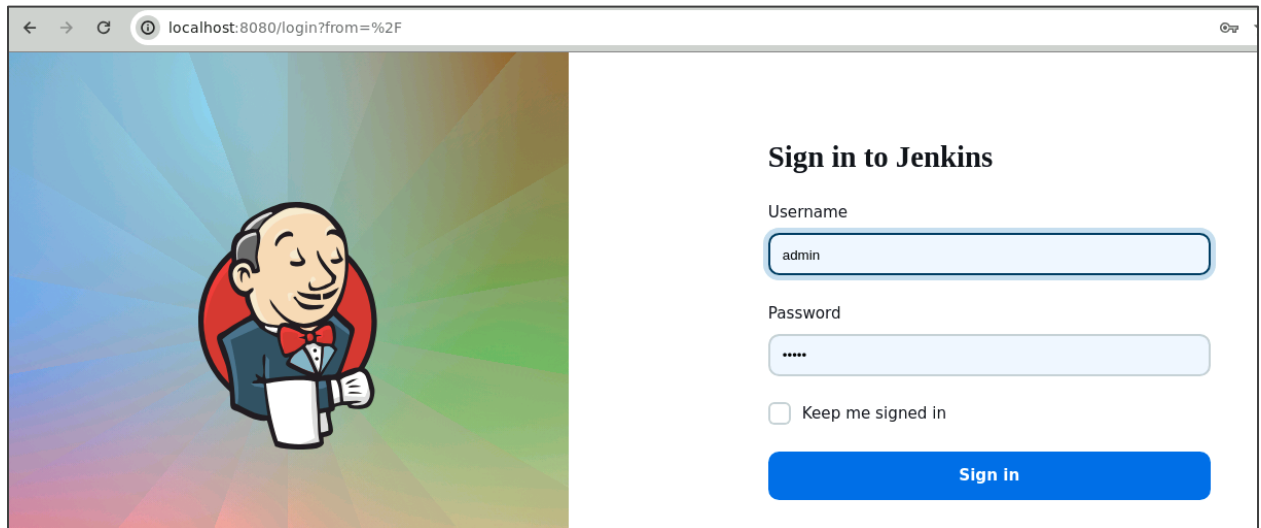


- 1.5 It will generate the app password. Copy the password and save it for future reference, then click on **Done**.



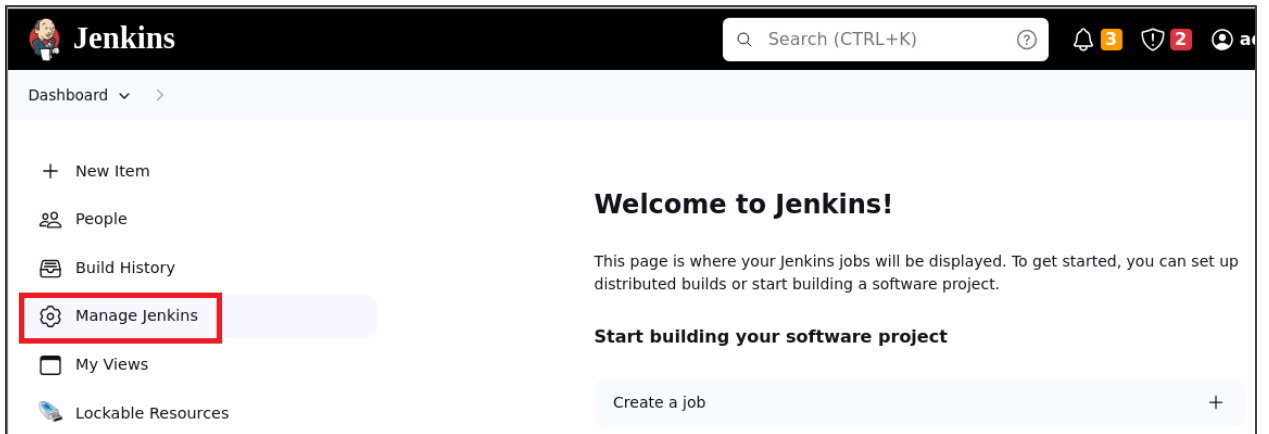
Step 2: Configure SMTP configurations in Jenkins

- 2.1 Sign in to **Jenkins** using your credentials

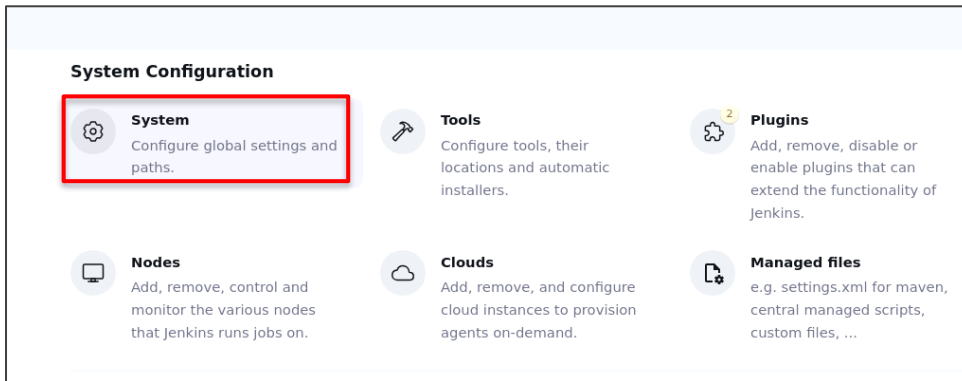


Note: The credentials for accessing Jenkins in the lab are Username: **admin** and Password: **admin**.

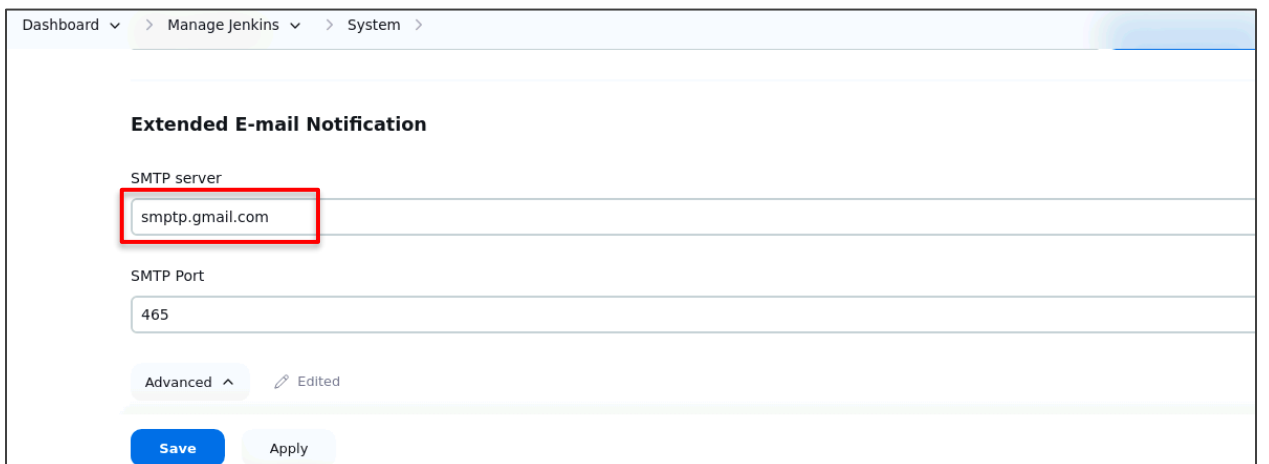
2.2 Navigate to **Manage Jenkins** in the Jenkins dashboard



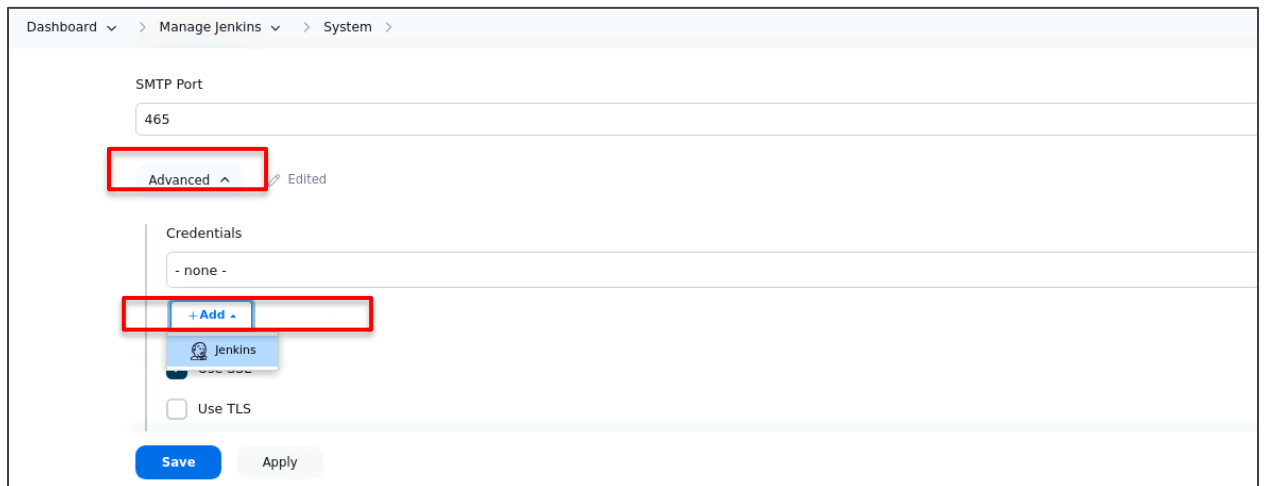
2.3 Now, click on **System** under **System Configuration**



2.4 In the **System** section, scroll down to **Extend E-mail Notification** section and provide **smtp.gmail.com** as the **SMTP server** and **465** as **SMTP Port**

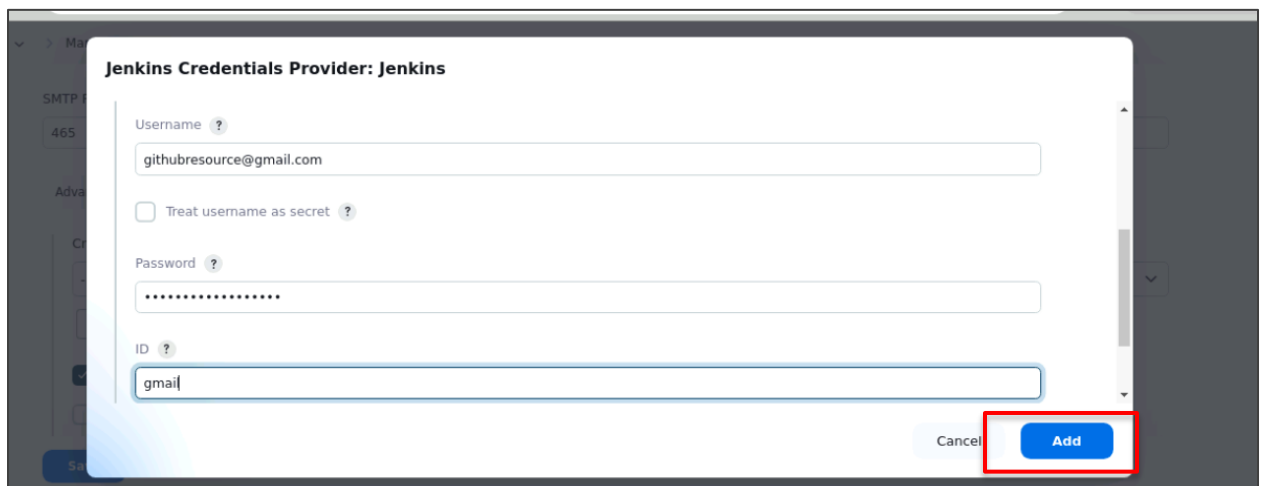


2.5 Click on **Advanced**, and in the **Credentials**, click on **Add** to add your credentials



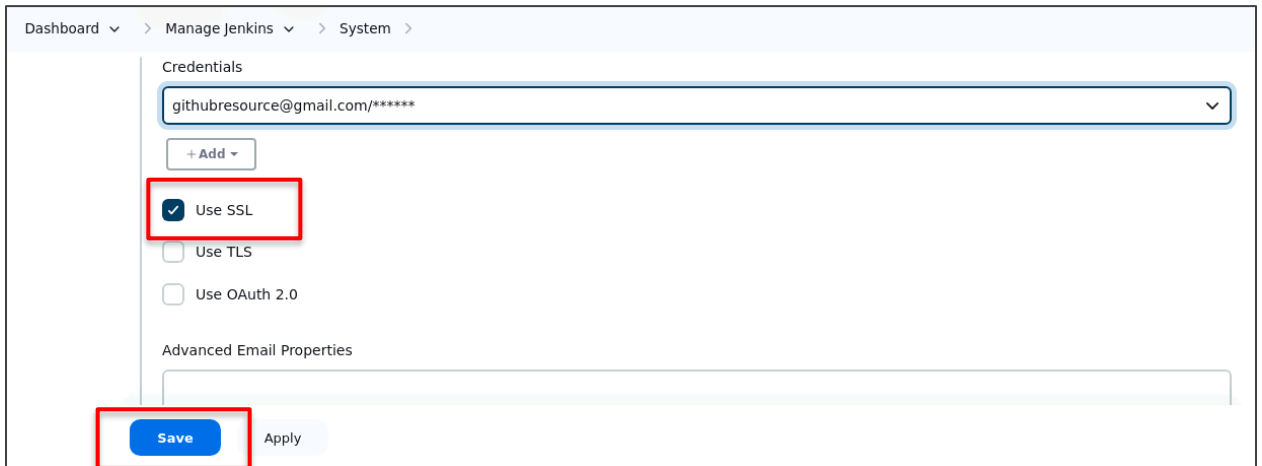
The screenshot shows the Jenkins configuration page for SMTP. The breadcrumb trail at the top is "Dashboard > Manage Jenkins > System > SMTP". The "SMTP Port" is set to "465". Below this, the "Advanced" tab is selected and highlighted with a red box. Under the "Credentials" section, a dropdown menu shows "- none -", and the "+ Add +" button is highlighted with a red box. Below the dropdown, there is a "Jenkins" icon and a "Use TLS" checkbox. At the bottom, there are "Save" and "Apply" buttons.

2.6 In the **Username**, provide your email ID; for **Password**, add the previously created app password, and then for **ID**, write **gmail** and click on **Add**



The screenshot shows a modal dialog titled "Jenkins Credentials Provider: Jenkins". It contains three input fields: "Username" with the value "githubresource@gmail.com", "Password" with masked characters, and "ID" with the value "gmail". There is a checkbox labeled "Treat username as secret" which is unchecked. At the bottom right, there are "Cancel" and "Add" buttons, with the "Add" button highlighted by a red box.

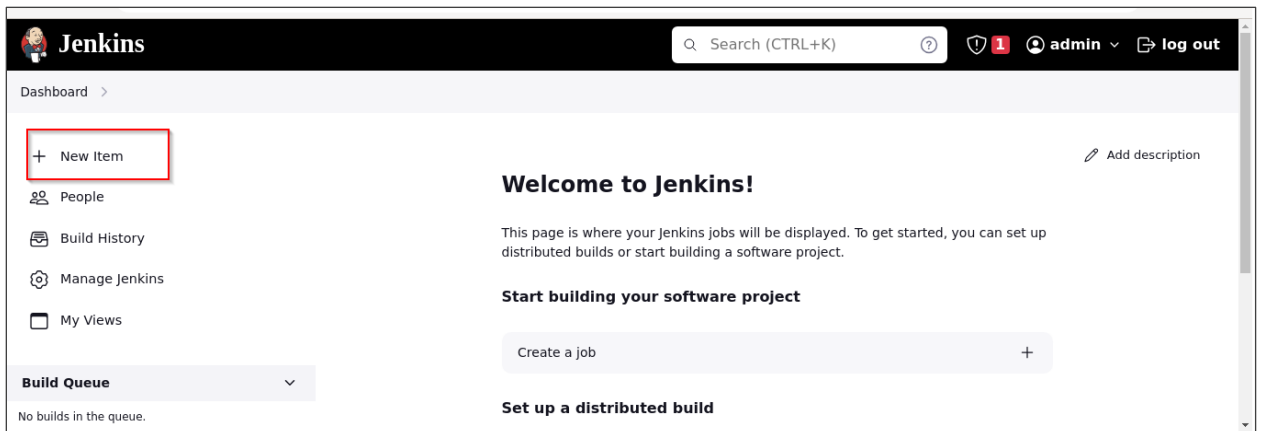
2.7 Select **Use SSL** and click on **Save** to save the configurations



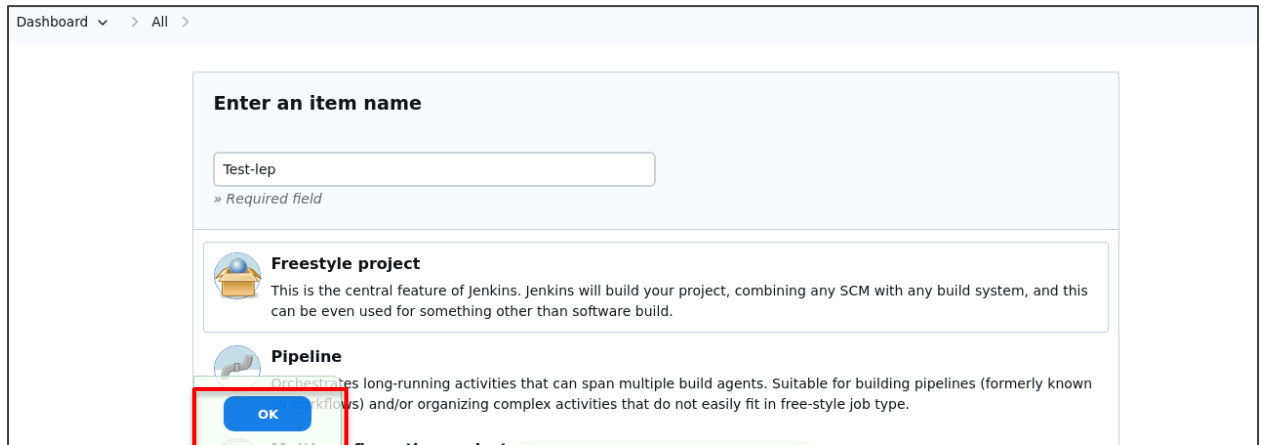
The screenshot shows the Jenkins 'Manage Jenkins' > 'System' > 'Credentials' configuration page. A text field contains the credential 'githubresource@gmail.com/*****'. Below it, the 'Use SSL' checkbox is checked and highlighted with a red box. Other options like 'Use TLS' and 'Use OAuth 2.0' are unchecked. At the bottom, the 'Save' button is highlighted with a red box, next to an 'Apply' button. The breadcrumb trail at the top reads 'Dashboard > Manage Jenkins > System >'.

Step 3: Configure Maven in Jenkins

3.1 Navigate back to Jenkins **Dashboard** and click on **New Item**



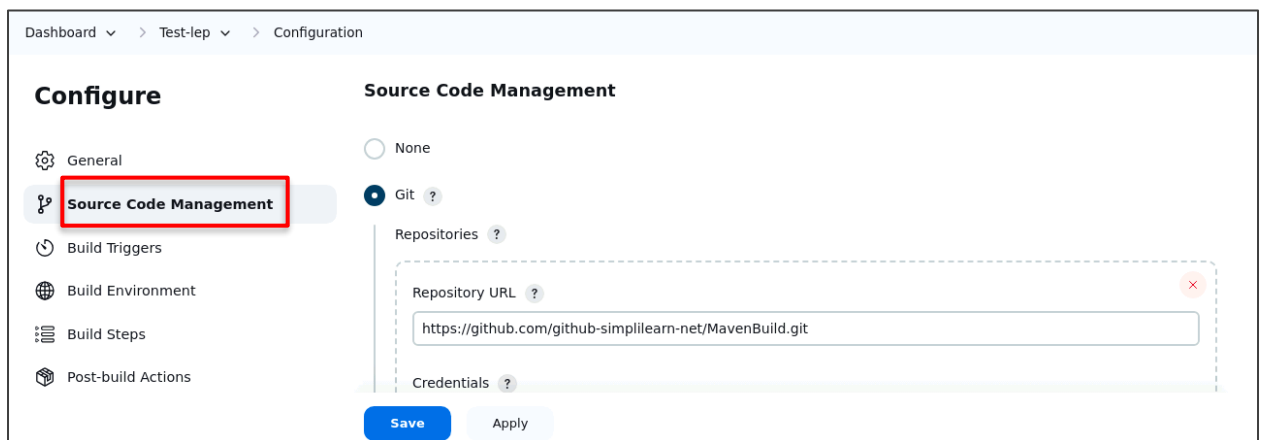
3.2 Enter the name of your project as **Test-lep**, select **Freestyle project**, then click on **OK**



The image shows the 'Enter an item name' dialog box in Jenkins. At the top, there is a breadcrumb trail: 'Dashboard > All >'. Below this, the title 'Enter an item name' is displayed. A text input field contains 'Test-lep', with a red asterisk and the text '» Required field' below it. Underneath the input field, there are two options: 'Freestyle project' and 'Pipeline'. The 'Freestyle project' option is selected, indicated by a blue radio button. Below these options, there is a blue 'OK' button, which is highlighted with a red rectangular box.

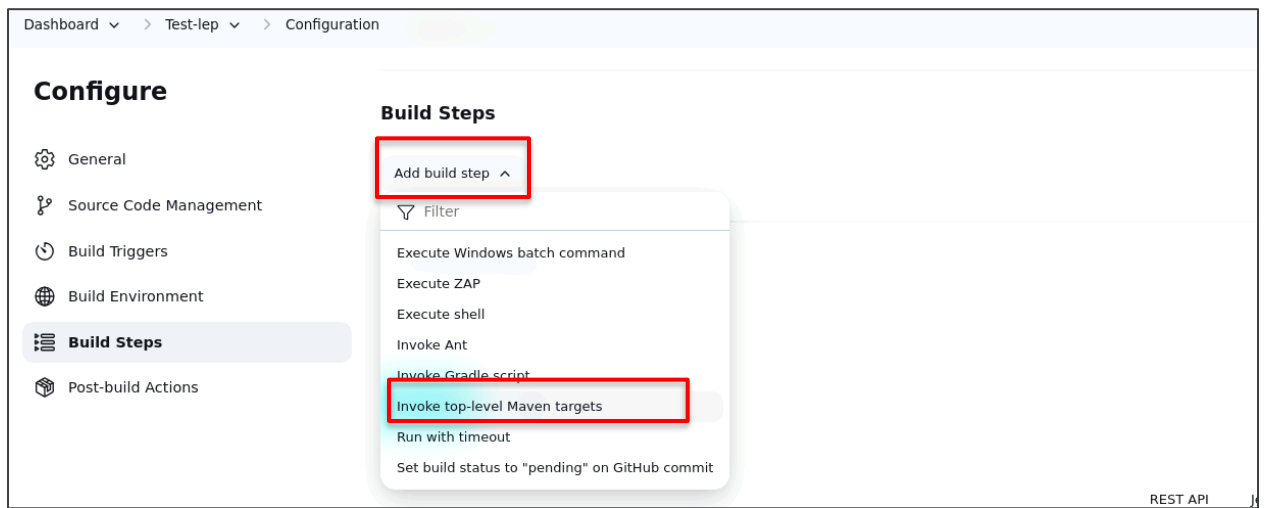
3.3 Navigate to **Source Code Management** and provide the below Git repository under **Repository URL**:

<https://github.com/github-simplilearn-net/MavenBuild.git>

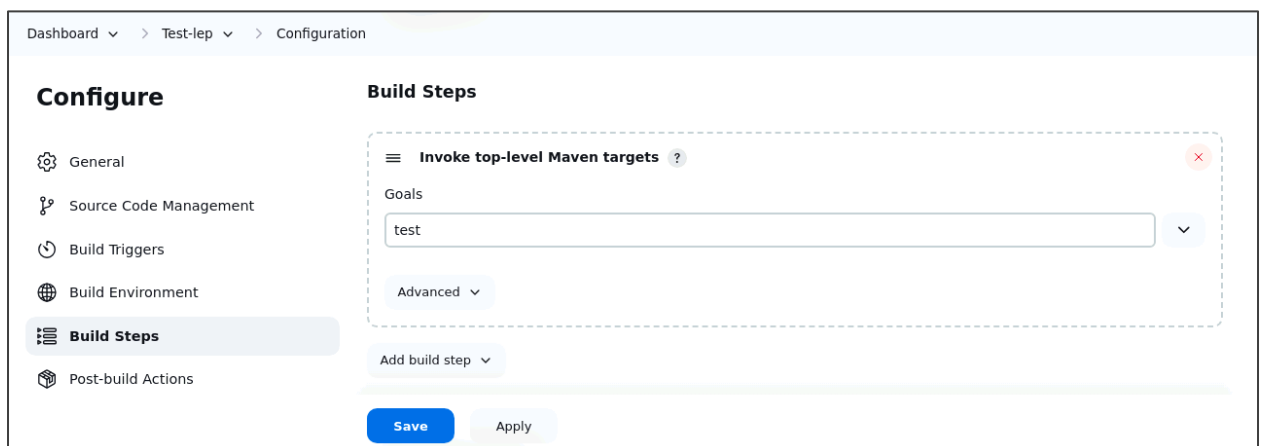


The image shows the 'Configure' page for the 'Test-lep' project in Jenkins. The breadcrumb trail at the top is 'Dashboard > Test-lep > Configuration'. On the left side, there is a 'Configure' section with a list of tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions'. The 'Source Code Management' tab is selected and highlighted with a red rectangular box. On the right side, under the 'Source Code Management' heading, there are two radio buttons: 'None' and 'Git'. The 'Git' radio button is selected. Below this, there is a 'Repositories' section with a 'Repository URL' input field. The URL 'https://github.com/github-simplilearn-net/MavenBuild.git' is entered in this field. At the bottom of the page, there are two buttons: 'Save' and 'Apply'.

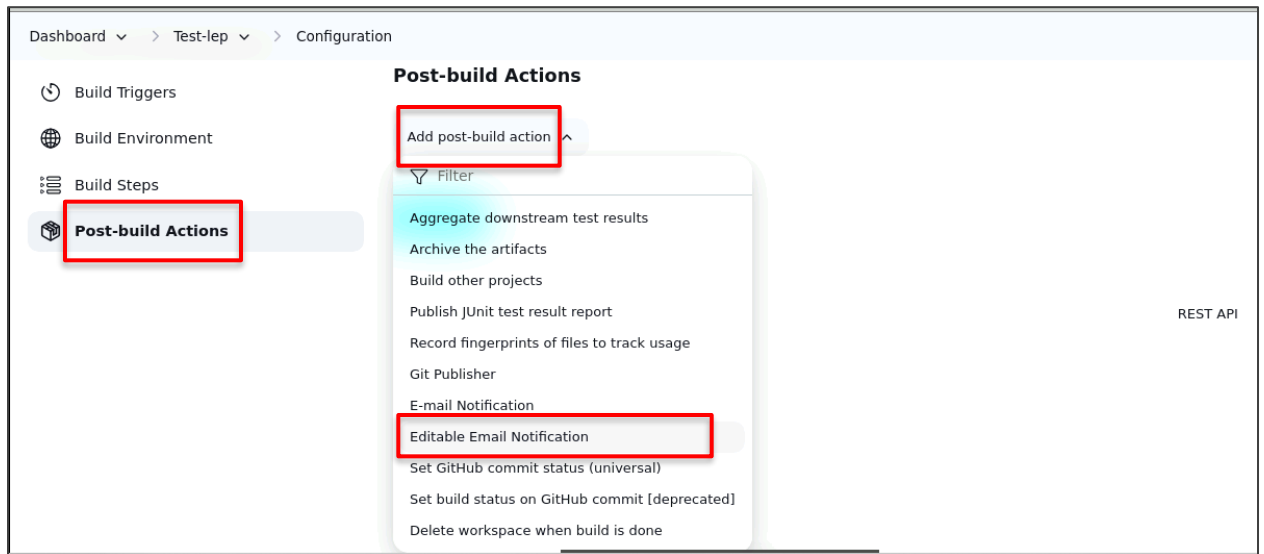
3.4 Navigate to **Build Steps**, click on **Add build step**, and select **Invoke top-level Maven targets**



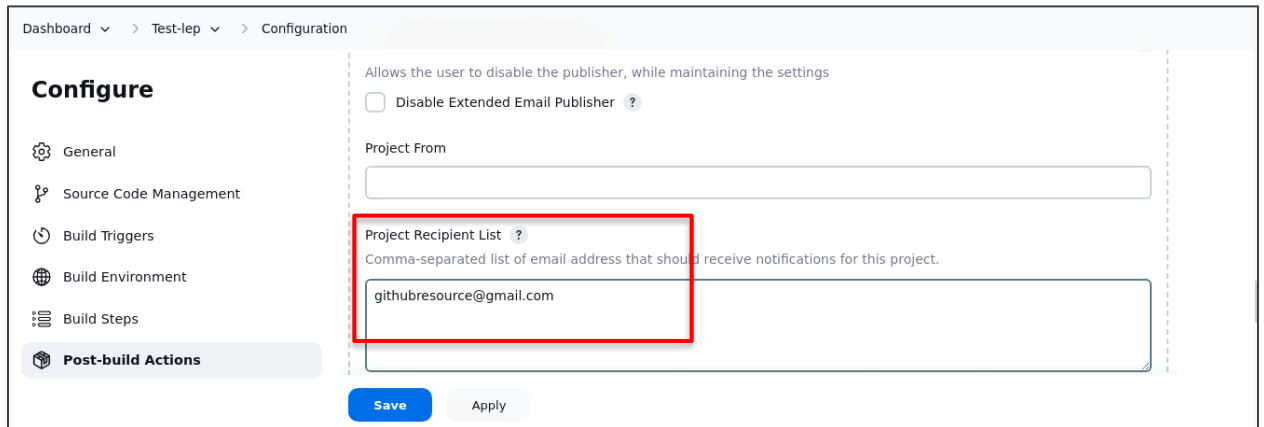
3.5 Write **test** in the **Goals**



3.6 Navigate to **Post-build Actions**, click on **Add post-build action**, and select **Editable Email Notification**



3.7 Under **Project Recipient List**, write your email address



3.8 Scroll down to **Attachments** and provide the below details:

target/surefire-reports/*.xml

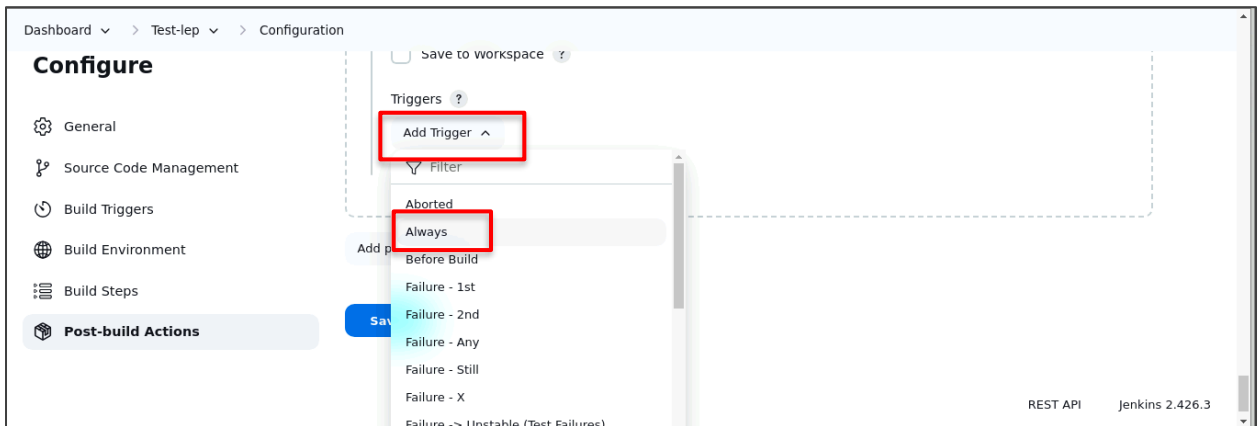
The screenshot shows the 'Configure' page for 'Test-lep' under the 'Configuration' tab. On the left, a sidebar lists various settings categories: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Attachments' section is active, displaying a text input field containing 'target/surefire-reports/*.xml', which is highlighted with a red rectangular box. Below this field, there are options for 'Attach Build Log' (set to 'Do Not Attach Build Log') and 'Content Token Reference'. At the bottom of the configuration area are 'Save' and 'Apply' buttons.

3.9 Click on **Advanced Settings**, scroll down to **Triggers**, and remove the existing trigger

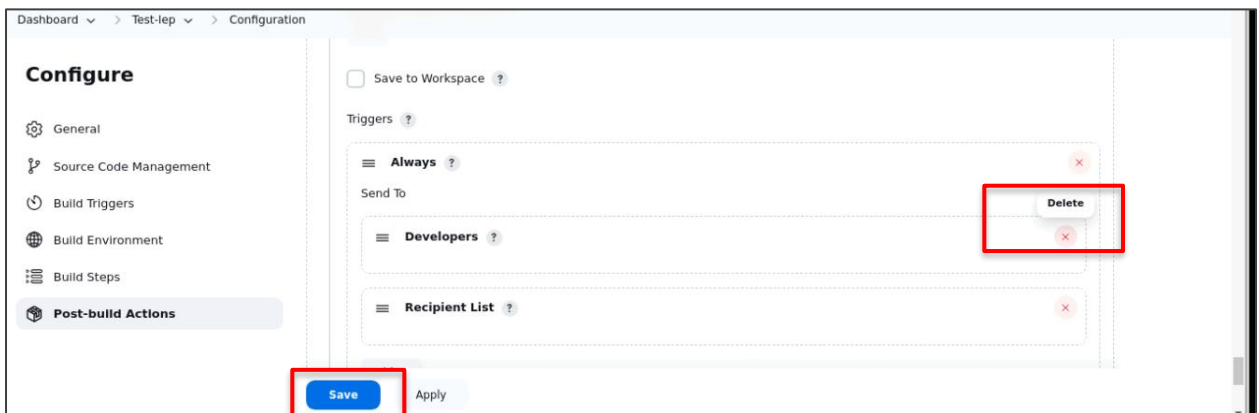
This screenshot shows the 'Configure' page with the 'Advanced Settings' dropdown menu highlighted by a red box. The sidebar on the left remains the same. The main content area shows the 'Advanced Settings' section, which includes a 'Content Token Reference' field and an 'Add post-build action' button. The 'Save' and 'Apply' buttons are at the bottom.

This screenshot shows the 'Configure' page with the 'Triggers' section expanded. A red box highlights the 'Remove Trigger' button, which is located next to the 'Always' trigger. The 'Send To' section below it shows a 'Recipient List' with a red 'x' icon. The 'Save' and 'Apply' buttons are at the bottom.

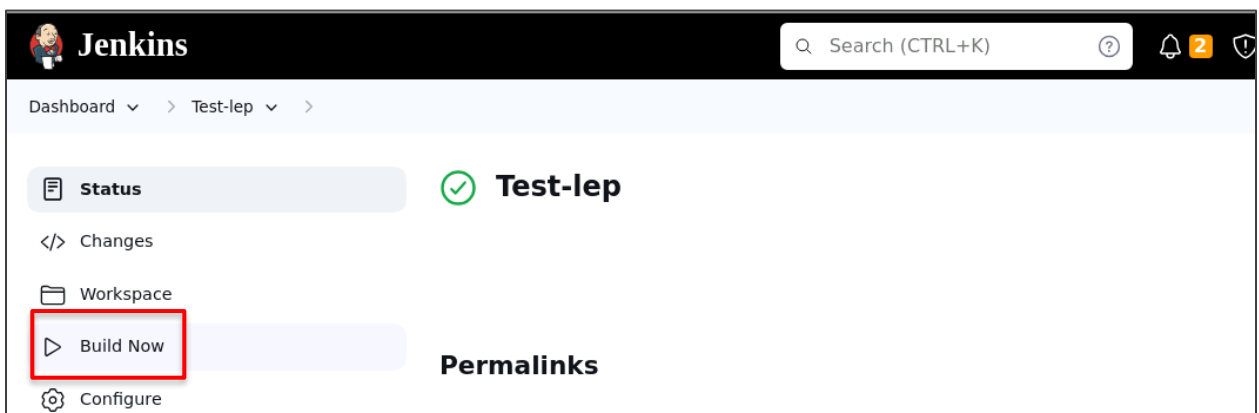
3.10 Click on **Add Trigger** and select **Always**



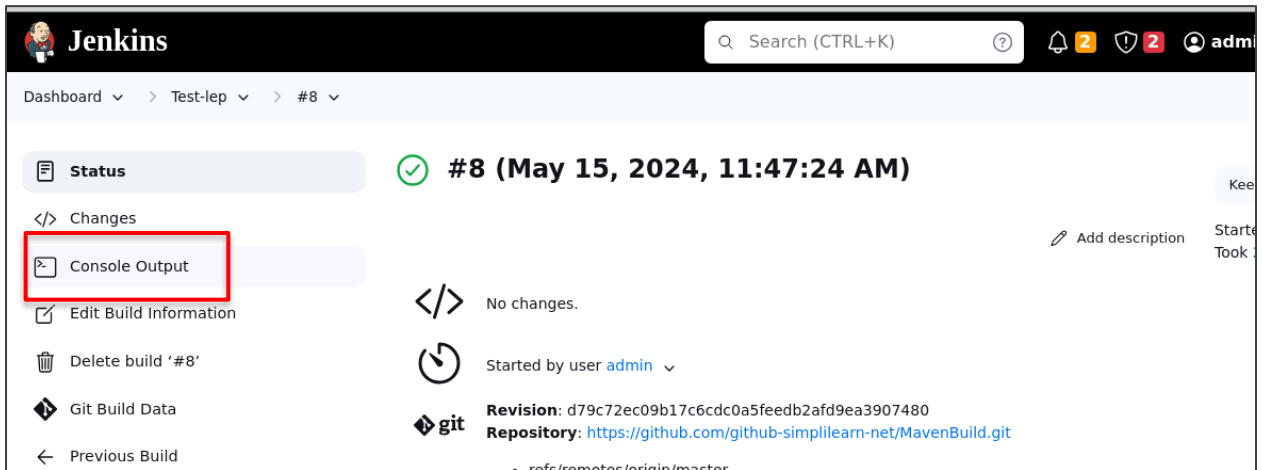
3.11 Delete the **Developers** and keep the **Recipient List**, then click on **Save**



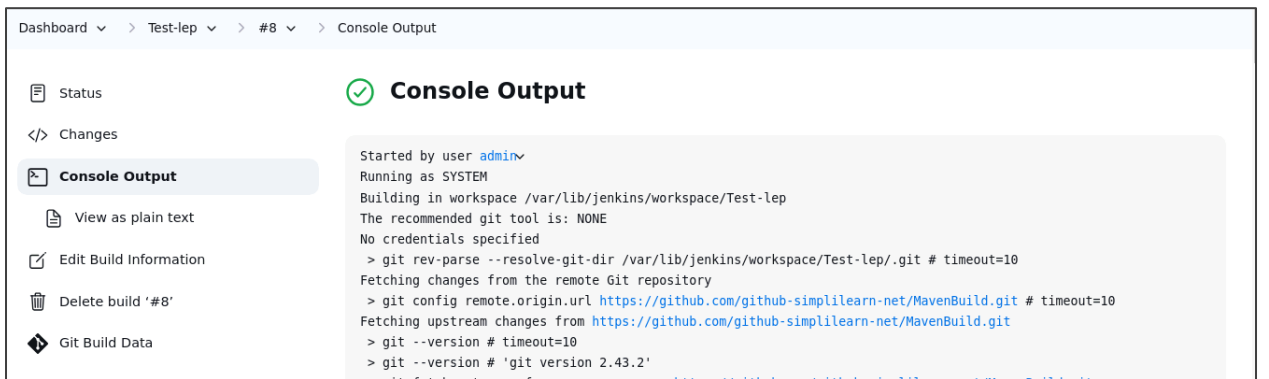
3.12 Now, click on **Build Now** to build the job



3.13 Once the job is built, click on **Console Output** to see the output




The screenshot shows the Jenkins job page for build #8. The left sidebar contains a list of links: Status, Changes, Console Output (highlighted with a red box), Edit Build Information, Delete build '#8', Git Build Data, and Previous Build. The main area displays the build status as 'Success' with a green checkmark and the timestamp 'May 15, 2024, 11:47:24 AM'. Below this, it shows 'No changes' and 'Started by user admin'. The revision is 'd79c72ec09b17c6cdc0a5feedb2afd9ea3907480' and the repository is 'https://github.com/github-simplilearn-net/MavenBuild.git'.



The screenshot shows the Jenkins Console Output page for build #8. The left sidebar contains a list of links: Status, Changes, Console Output (highlighted), View as plain text, Edit Build Information, Delete build '#8', and Git Build Data. The main area displays the console output, which includes the following text:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Test-lep
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Test-lep/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/github-simplilearn-net/MavenBuild.git # timeout=10
Fetching upstream changes from https://github.com/github-simplilearn-net/MavenBuild.git
> git --version # timeout=10
> git --version # 'git version 2.43.2'
```



The screenshot shows the Jenkins Console Output page for build #8, displaying the test results. The output includes the following text:

```
Running com.geekcap.vmturbo.ThingTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.041 sec

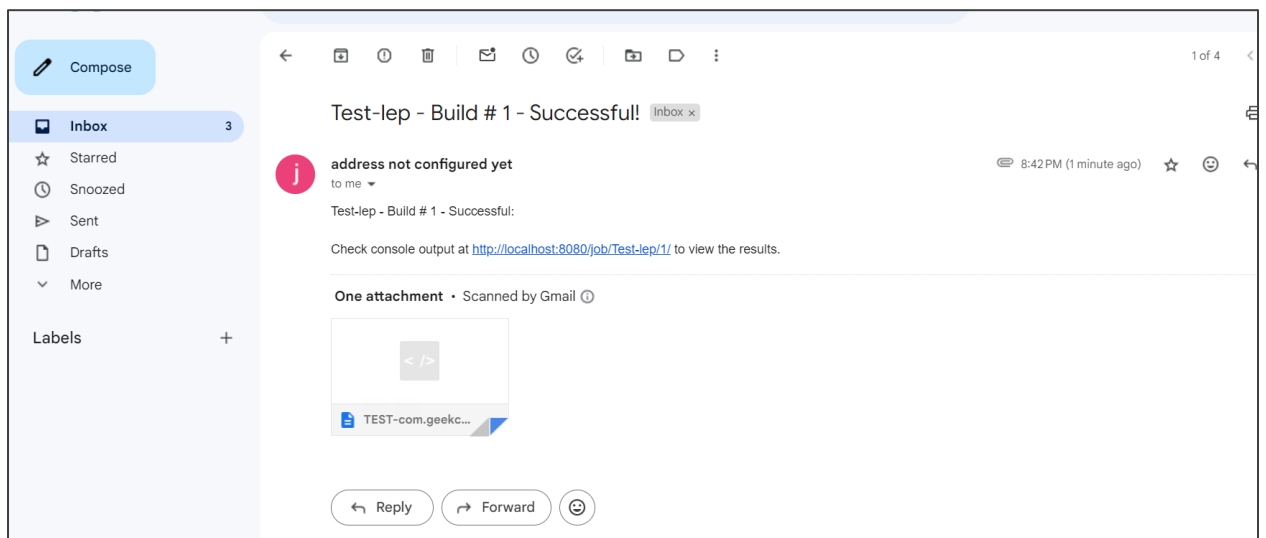
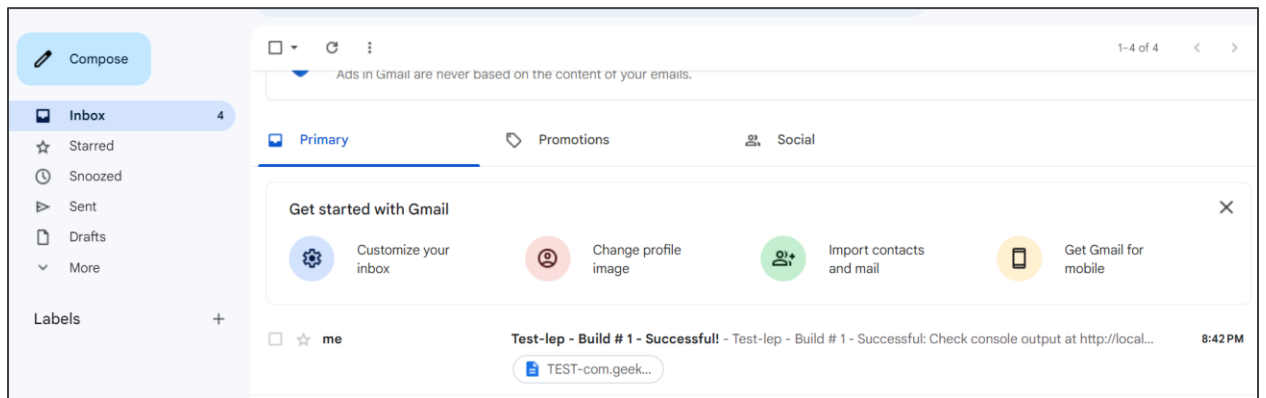
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

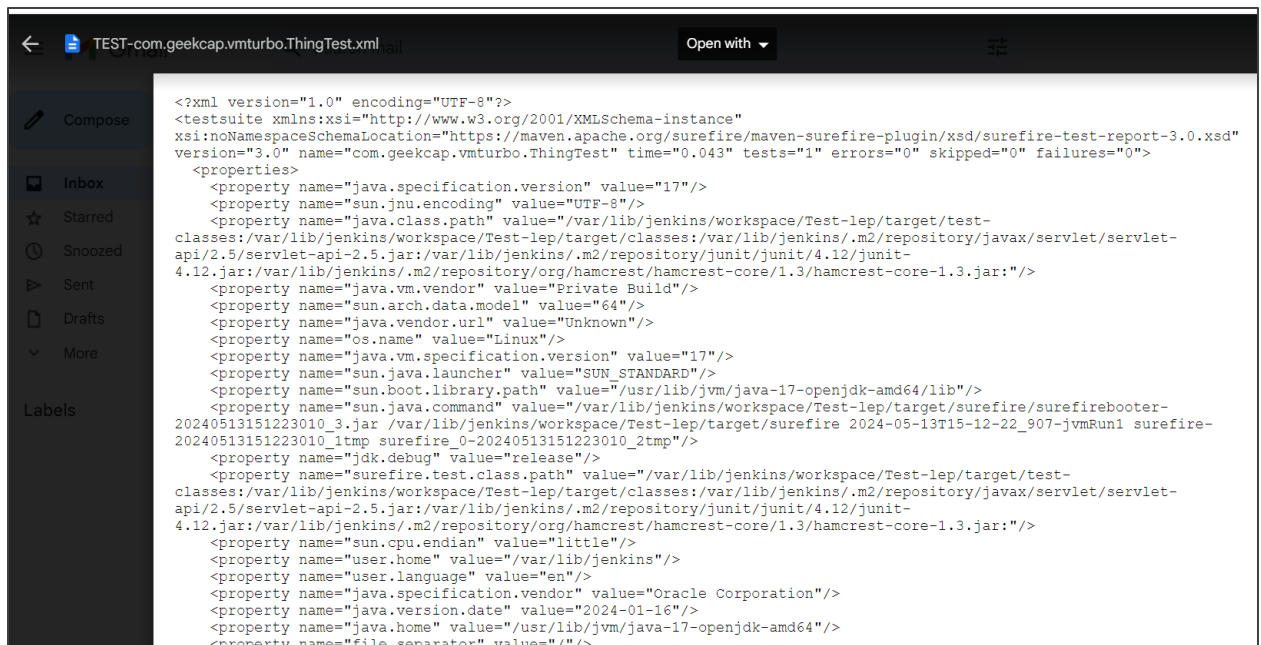
[1;34mINFO[1m] [1m-----[1m
[1;34mINFO[1m] [1;32mBUILD SUCCESS[1m
[1;34mINFO[1m] [1m-----[1m
[1;34mINFO[1m] [1mTotal time: 1.105 s
[1;34mINFO[1m] [1mFinished at: 2024-05-15T11:47:27Z
[1;34mINFO[1m] [1m-----[1m
Email was triggered for: Always
Sending email for trigger: Always
Sending email to: githubresource@gmail.com
Finished: SUCCESS
```

You can see the build is successful.

3.14 Navigate to your email account to validate if the email has been sent



You can see the email has been sent successfully.



You can see the Junit report.

By following these steps, you have successfully set up Maven in Jenkins to run JUnit test cases and automatically distribute the test reports via email notifications to specified recipients after each build.