



technology

Learning and Knowledge Management

JENKINS Tool Overview

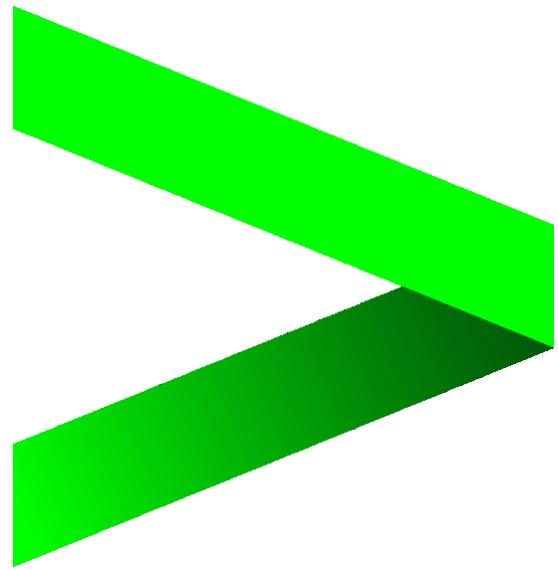


Table of Contents

1.	Exercise 2.1: Install Jenkins and Access the Dashboard	3
2.	Exercise 2.2: Configure Jenkins	8
3.	Exercise 2.3: Install Plugins	11
4.	Exercise 3.1: Build Job.....	14
5.	Exercise 4.1: Create a Job and Analyze Code in Jenkins	22
6.	Exercise 5.1: Create a Job to Deploy WAR File in Jenkins.....	33
7.	Exercise 6.1: Test application with Selenium.....	38
8.	Exercise 7.1: Create Job Pipeline	44

Module 2

Exercise 2.1: Install Jenkins and Access the Dashboard

Scenario

Perform activities to install Jenkins and to get started with tasks

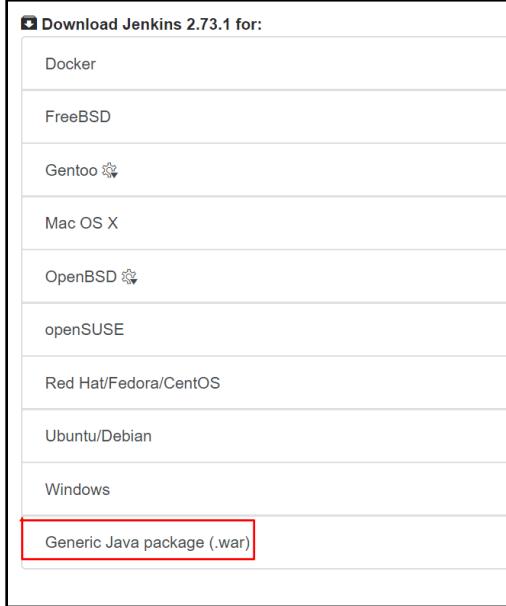
Prerequisite: Java 8(jdk1.8), 512 MB RAM

Walkthrough

1. Download and install the war file
2. Set the initial admin password
3. Install suggested plugins
4. Create first admin user and start accessing the Jenkins dashboard

Steps

1. Download and install the war file

1	<ol style="list-style-type: none">a. Navigate to https://jenkins.io/download/b. Download Generic Java package (.war) and save it into your local disk. Example: C:\Softwares 
2	Open Command prompt and set the JENKINS_HOME directory by executing the below command: C:\Softwares>set JENKINS_HOME=C:\Jenkins
3	Install Jenkins in the port 8082 by executing the command as highlighted in the below screen java -jar jenkins.war --httpPort=8082 Note: By default, Jenkins get started in the port number 8080.

The command prompt displays the following message, when Jenkins starts running successfully

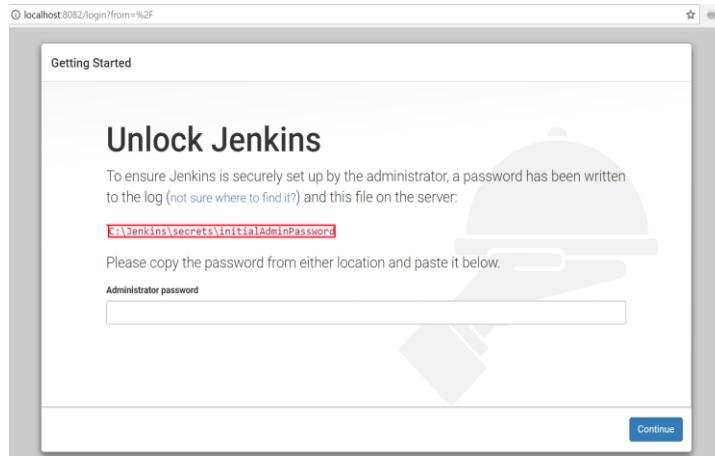
```
Oct 04, 2017 3:22:17 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Oct 04, 2017 3:22:18 PM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Oct 04, 2017 3:22:19 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Oct 04, 2017 3:22:19 PM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
Oct 04, 2017 3:22:21 PM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Oct 04, 2017 3:22:21 PM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata. 17,796 ms
```

2. Set the initial admin password

1

You can access Jenkins in the path: <http://localhost:8082>

To start working with Jenkins dashboard, first set the administrator password as provided in the location:
C:\Jenkins\Secrets\initialAdminPassword



2

Copy the admin password provided in the location and paste it as shown below and click **Continue**

	<p>Getting Started</p> <h2>Unlock Jenkins</h2> <p>To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:</p> <pre>C:\Jenkins\secrets\initialAdminPassword</pre> <p>Please copy the password from either location and paste it below.</p> <p>Administrator password</p> <p>.....</p> <p>Continue</p>	
--	--	--

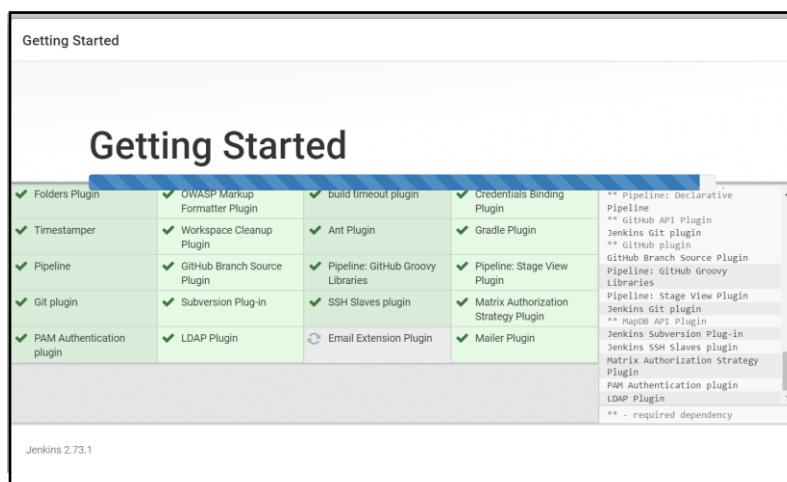
The Jenkins displays the following screen to install plugins before you start accessing the dashboard

<p>Getting Started</p> <h2>Customize Jenkins</h2> <p>Plugins extend Jenkins with additional features to support many different needs.</p> <div style="display: flex; justify-content: space-around;"> <div style="background-color: #e0f2fd; padding: 10px; width: 45%;"> <p>Install suggested plugins</p> <p>Install plugins the Jenkins community finds most useful.</p> </div> <div style="background-color: #fff9c4; padding: 10px; width: 45%;"> <p>Select plugins to install</p> <p>Select and install plugins most suitable for your needs.</p> </div> </div> <p>Jenkins 2.73.1</p>
--

3. Install suggested plugins

1	<p>Click Install suggested plugins</p> <table border="1"> <tr> <td> <p>Getting Started</p> <h2>Customize Jenkins</h2> <p>Plugins extend Jenkins with additional features to support many different needs.</p> <div style="display: flex; justify-content: space-around;"> <div style="background-color: #e0f2fd; padding: 10px; width: 45%;"> <p>Install suggested plugins</p> <p>Install plugins the Jenkins community finds most useful.</p> </div> <div style="background-color: #fff9c4; padding: 10px; width: 45%;"> <p>Select plugins to install</p> <p>Select and install plugins most suitable for your needs.</p> </div> </div> <p>Jenkins 2.73.1</p> </td> </tr> </table>	<p>Getting Started</p> <h2>Customize Jenkins</h2> <p>Plugins extend Jenkins with additional features to support many different needs.</p> <div style="display: flex; justify-content: space-around;"> <div style="background-color: #e0f2fd; padding: 10px; width: 45%;"> <p>Install suggested plugins</p> <p>Install plugins the Jenkins community finds most useful.</p> </div> <div style="background-color: #fff9c4; padding: 10px; width: 45%;"> <p>Select plugins to install</p> <p>Select and install plugins most suitable for your needs.</p> </div> </div> <p>Jenkins 2.73.1</p>
<p>Getting Started</p> <h2>Customize Jenkins</h2> <p>Plugins extend Jenkins with additional features to support many different needs.</p> <div style="display: flex; justify-content: space-around;"> <div style="background-color: #e0f2fd; padding: 10px; width: 45%;"> <p>Install suggested plugins</p> <p>Install plugins the Jenkins community finds most useful.</p> </div> <div style="background-color: #fff9c4; padding: 10px; width: 45%;"> <p>Select plugins to install</p> <p>Select and install plugins most suitable for your needs.</p> </div> </div> <p>Jenkins 2.73.1</p>		

The following screen appears to indicate the default plugins that are installed successfully:



4. Create first admin user and start accessing the Jenkins dashboard

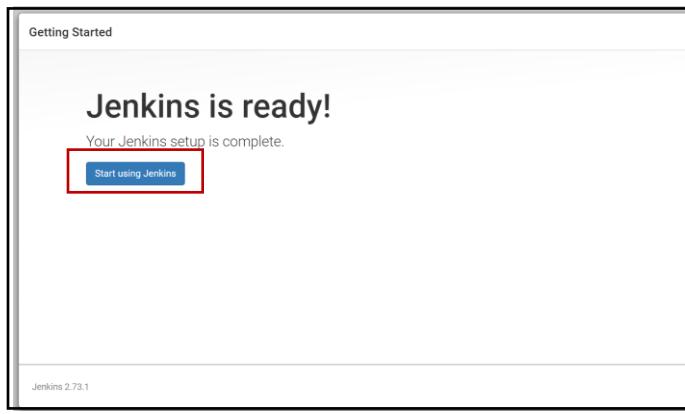
1

Enter the respective details for the fields, and click **Save and Finish** to complete Jenkins setup

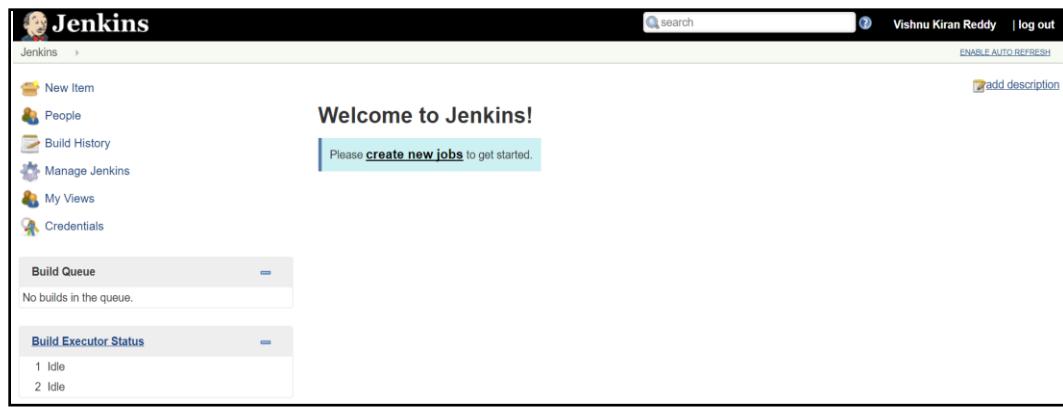
A screenshot of the 'Create First Admin User' form. It contains fields for Username, Password, Confirm password, Full name, and E-mail address. At the bottom right, there are 'Continue as admin' and 'Save and Finish' buttons, with 'Save and Finish' being highlighted by a red box.

2

Click **Start Using Jenkins**, when the setup is complete



You can now start using the Jenkins dashboard



Exercise 2.2: Configure Jenkins

Scenario

Perform activities to configure Jenkins with JDK, Git, Maven and Ant

Prerequisite: Working knowledge of JDK, GIT, Maven and Ant

Note:

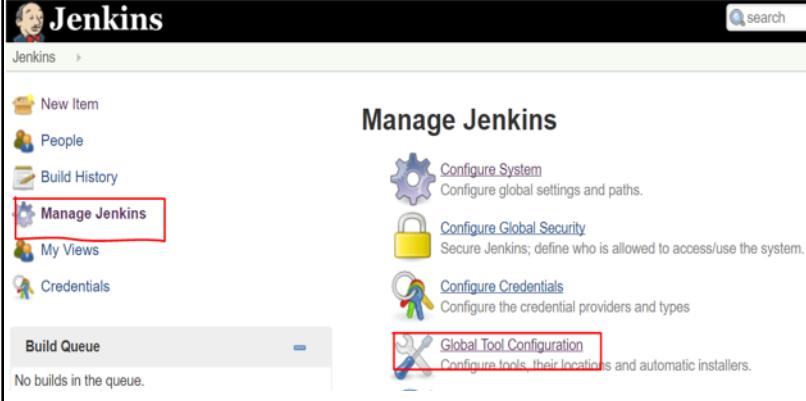
1. The Software's will be shared by faculty.(Git,Maven,Ant,JDK).
2. The JDK, Git, Maven and Ant to be stored in the C:\Softwares of the local system.

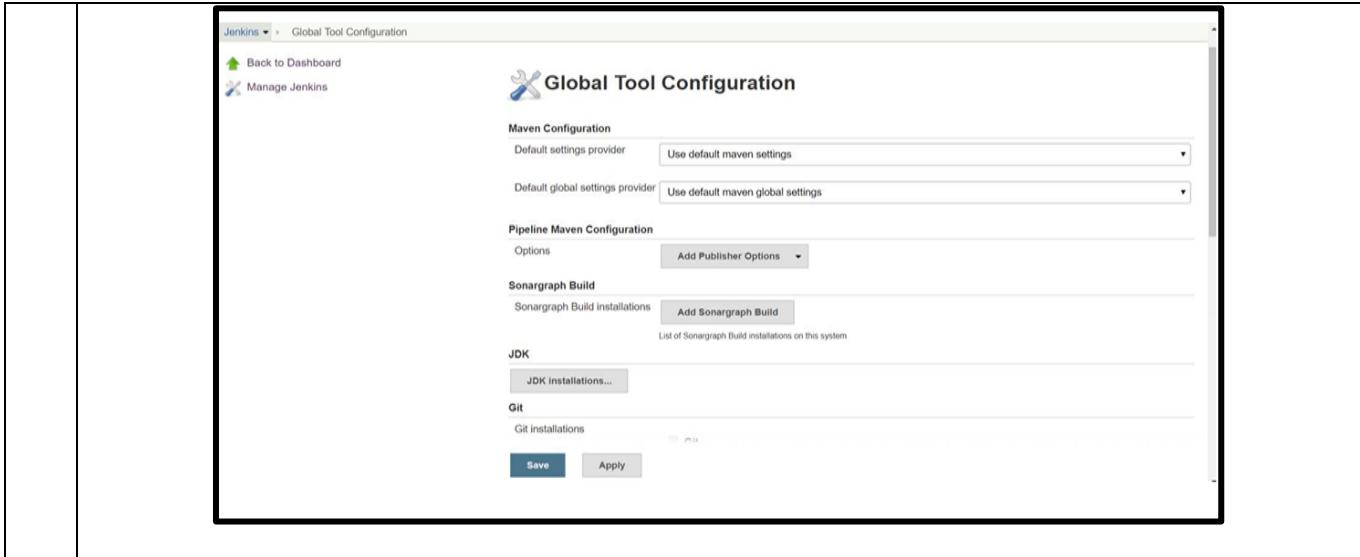
Walkthrough

1. Access Global Tool Configuration section
2. Configure Jenkins with JDK
3. Configure Jenkins with Git
4. Configure Jenkins with Maven
5. Configure Jenkins with Ant and save all the configurations

Steps

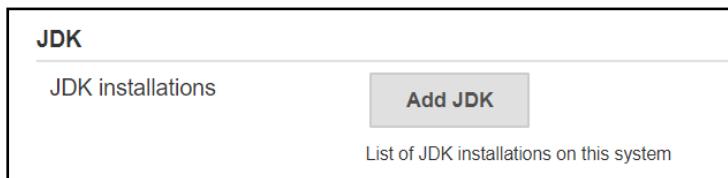
1. Access Global Tool Configuration section

<p>1</p>	<p>Open the Jenkins dashboard in the URL http://localhost:8082</p>
<p>2</p>	<p>Navigate to Manage Jenkins → Global Tool Configuration option</p>  <p>This results in the Global Tool Configuration page, where you can choose the components to configure Jenkins with.</p>

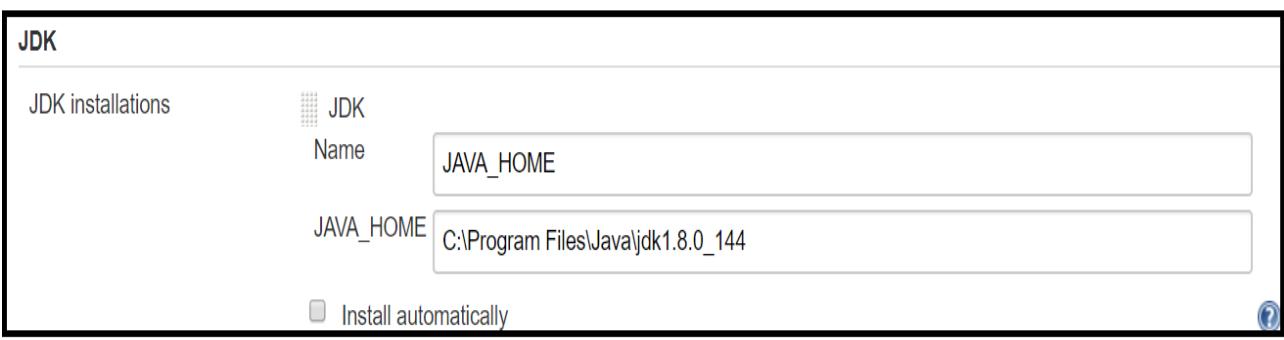


2. Configure Jenkins with JDK

- 1 a. In the Global Tool Configuration page, navigate to the **JDK** section
b. Click **Add JDK**



- 2 a. Uncheck the **Install automatically** checkbox
b. Specify the name and path of the JDK installed in the system in the JAVA_HOME field as shown below



3. Configure Jenkins with Git

- 1 a. Navigate to the **Git** section
b. Click **Add Git**

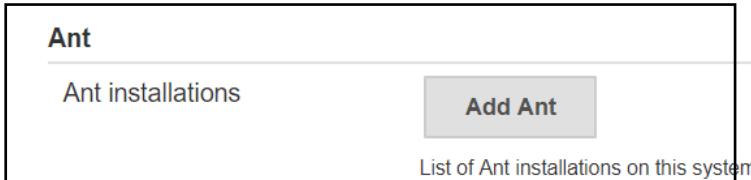
2	<p>a. Uncheck the Install automatically checkbox b. Specify the name and path of the Git installed in the Path to Git executable field as shown below</p>

4. Configure Jenkins with Maven

1	<p>a. Navigate to the Maven section b. Click Add Maven</p>
2	<p>a. Uncheck the Install automatically checkbox b. Specify the name and path of the Maven stored in your local system in the M2_HOME field as shown below</p>

--	--

5. Configure Jenkins with Ant and save all the configurations

1	<ul style="list-style-type: none"> a. Navigate to the Ant section b. Click Add Ant  <p>List of Ant installations on this system</p>
2	<ul style="list-style-type: none"> a. Uncheck the Install automatically checkbox b. Specify the name and path of the Ant installed in the ANT_HOME field as shown below 
3	Click the Save button to save the JDK, Git, Maven and Ant configurations

Exercise 2.3: Install Plugins

Scenario

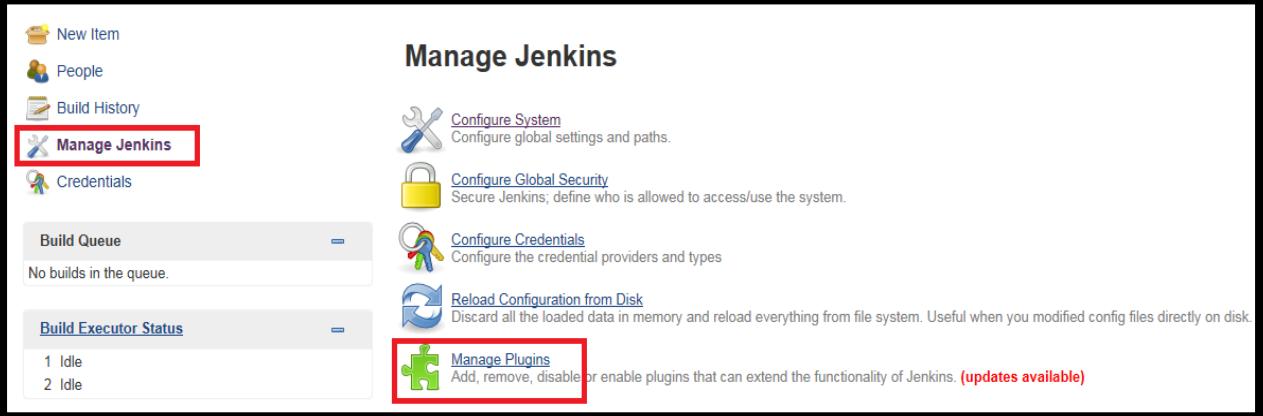
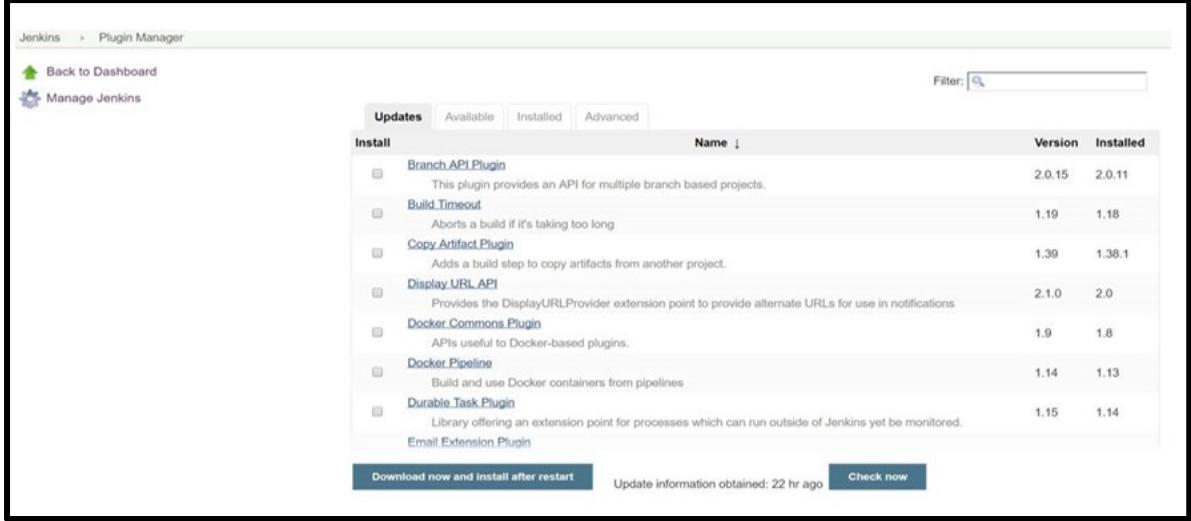
Perform activities to install plugins that facilitates to perform tasks in Jenkins.

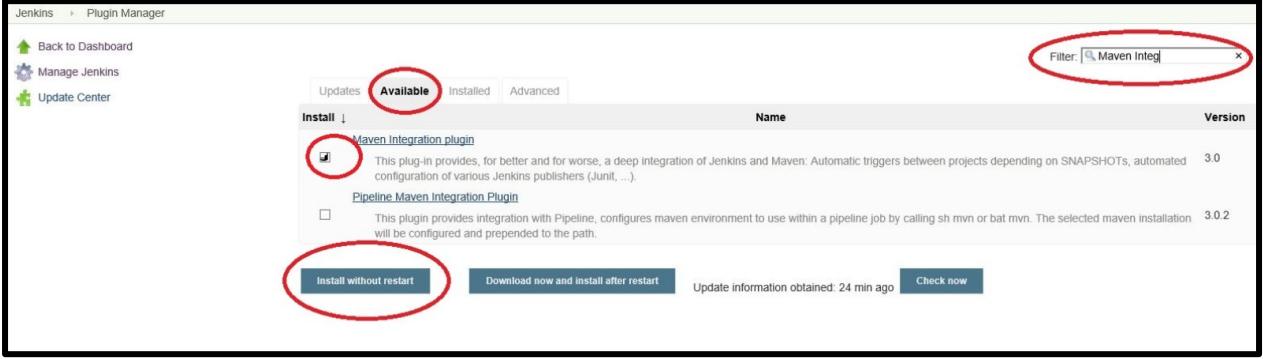
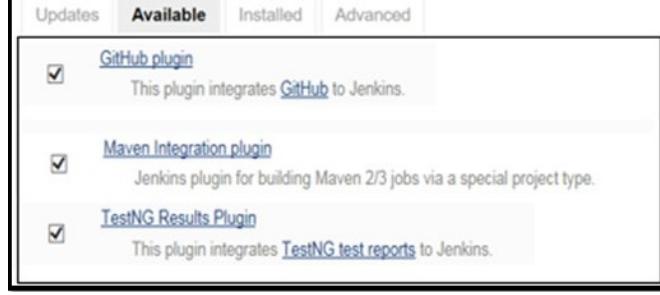
Walkthrough

1. Search and install the required plugins

Steps:

1	In Jenkins dashboard, navigate to Manage Jenkins → Manage Plugins option.
---	--

	 <p>You will see the Plugin Manager page</p> 
2	<ol style="list-style-type: none"> In the Plugin Manager page, click the Available tab Search for the required plugins Check the box for the required plugins Click the Install without restart tab

	 <p>The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'Maven Integ'. Below it, the 'Maven Integration plugin' is listed with a checked checkbox. A second plugin, 'Pipeline Maven Integration Plugin', is listed with an unchecked checkbox. At the bottom, there are two buttons: 'Install without restart' (which is highlighted with a red oval) and 'Download now and install after restart'.</p>
	<p>Note: Select the plugins shown below to install:</p>  <p>The screenshot shows the Jenkins Plugin Manager interface with three specific plugins selected for installation: 'GitHub plugin', 'Maven Integration plugin', and 'TestNG Results Plugin'. Each selected plugin has a checked checkbox next to its name and a brief description below it.</p>

End of Module 2

Module 3

Exercise 3.1: Build Job

Scenario

Perform activities to build job in Jenkins.

Prerequisite:

1. Working knowledge of GIT and Maven
2. The project to build should be created in GitHub server (The steps are included below)
3. JDK, Git, Maven and Ant should be stored in the C:\Softwares of the local system to perform this exercise.

Note:

1. This exercise uses GitHub server to store source code and Maven to build project
2. The source code will be shared by faculty
3. The ProjectSourceCode should be stored in C:\Softwares of the local system.

Walkthrough

1. Install Git
2. Create project in GitHub server
3. Create job in Jenkins and configure it with the GitHub project
4. Configure Source Code Management with Git
5. Configure build job using Maven
6. Execute build
7. View build results

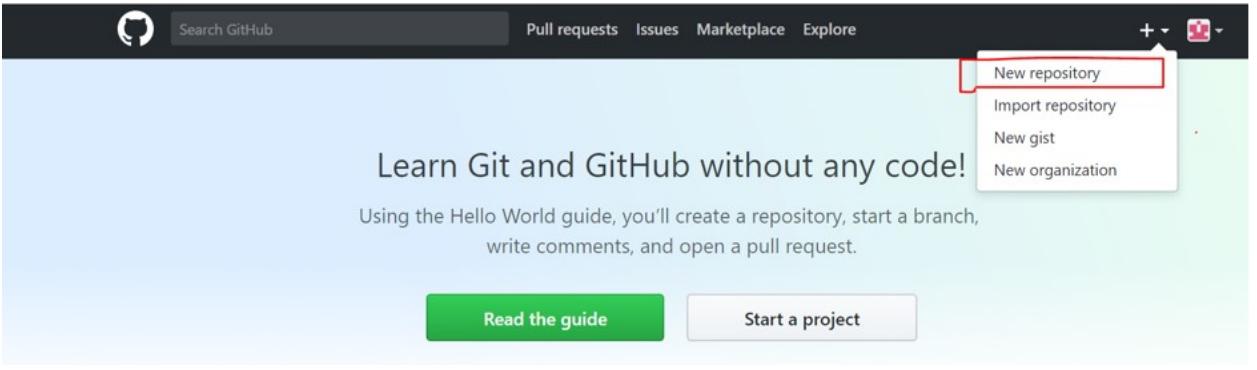
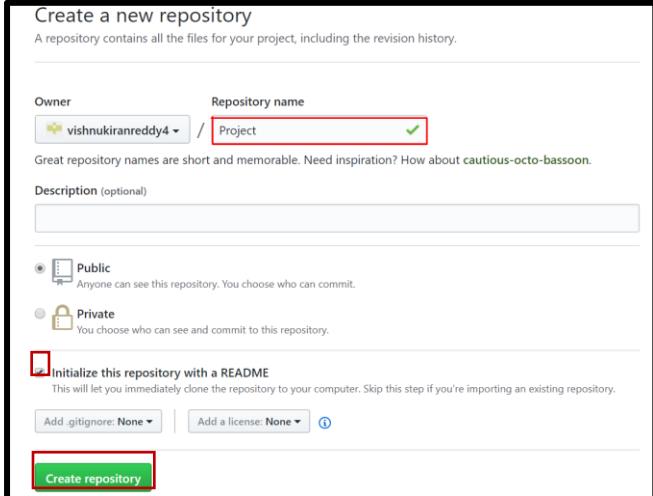
Steps

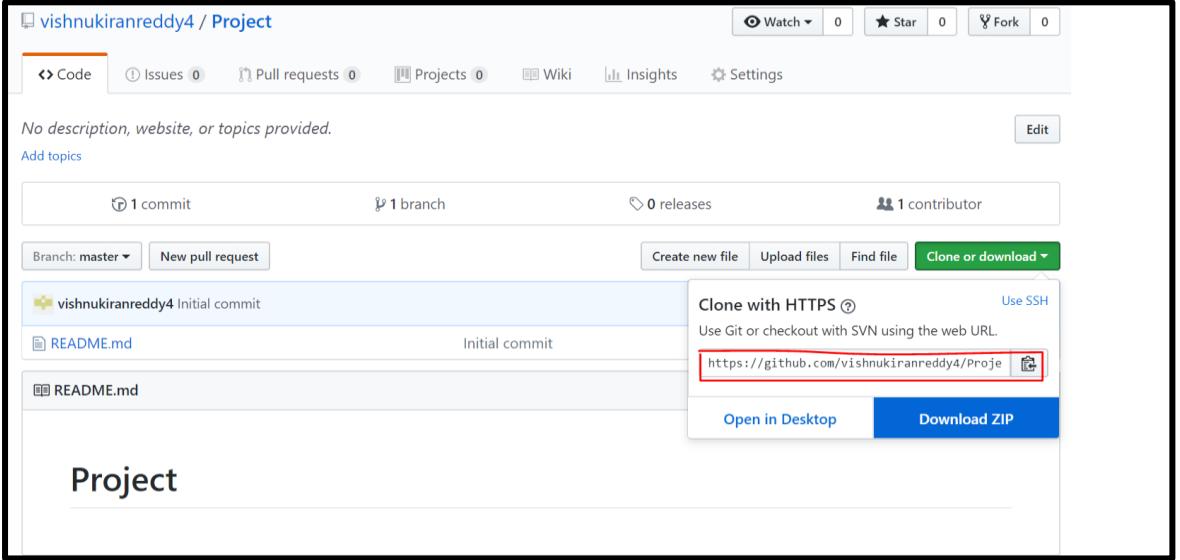
1. Install Git

1	Following are the steps to install Git: <ol style="list-style-type: none">a. Click on Git Download Git from the below URLb. https://git-for-windows.github.io/c. Start installationd. Select “Git bash” and “Git GUI”e. Select “Use Git from windows command prompt”f. Keep remaining options as default and click Install button.g. If Git installation is completed successfully, then follow these steps.h. Navigate and select Git Bash
---	---

2. Create project in GitHub server

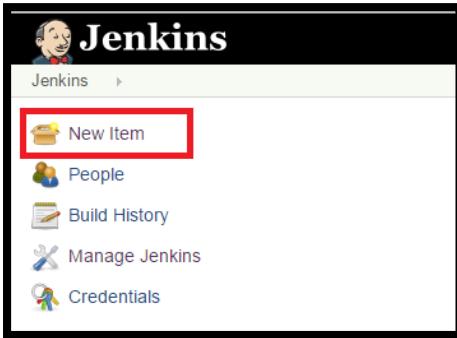
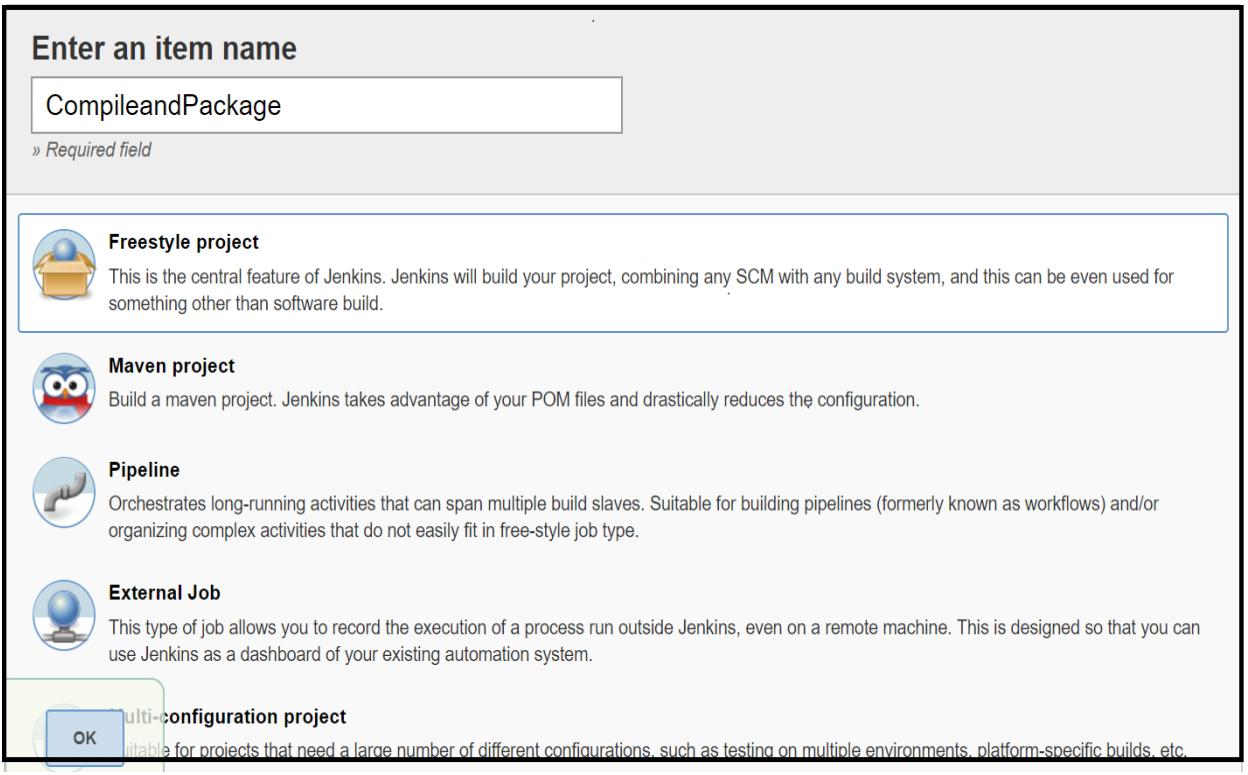
1	Sign in to the GitHub server <ol style="list-style-type: none">i. You are required to sign up to GitHub at https://github.com/
---	--

	<p>j. While signup, please provide your mail id as an email address. k. (Confirmation link will be sent). l. Once the account is created, login to the account.</p>
2	<p>Create repository</p> <p>a. Click New repository from the drop-down list of the GitHub home page</p>  <p>b. Specify the name (example: Project) in the Repository name field and click the checkbox next to Initialize this repository with README option</p> <p>c. Click the Create Repository button to create repository in GitHub server</p> 
3	<p>Clone the repository</p> <p>Clone the project that you created by using the git clone command. The path to the project can be found in GitHub server: The path looks as shown below as per your credentials:</p> <pre>git clone https://github.com/vishnukiranreddy4/Project.git</pre>

	 <p>Project</p>
4	<p>Navigate to Git bash on your workstation and make sure that you have changed the directory to the right directory where you want to clone your project and execute the git clone command(Create a folder JenkinsPractice in C drive)</p> <ul style="list-style-type: none"> a. cd /c/JenkinsPractice b. git clone https://github.com/vishnukiranreddy4/Project.git
5	<p>Change to your Git repository directory:</p> <ul style="list-style-type: none"> • cd Project
6	<p>Copy the project content into the "Project" folder manually from the path C:\Softwares\ProjectSourceCode\Module3 without using Git command Note: Copy the code which is present in ProjectSourceCode\Module3 folder(C:\Softwares\ProjectSourceCode\Module3) into Project folder(Cloned from the GitHub Server).</p>
7	<p>After copying content, navigate to git bash and execute the below commands</p> <ul style="list-style-type: none"> • git add . • git status • git config --global user.email <gitloginemail> For Ex: git config --global user.email "vishnureddy224@gmail.com" • git commit -m "Adding project content" • git push origin master <p>Now, the project is pushed to GitHub server Note: Enter password which you have given while creating the account in GitHub.</p>

3. Create job in Jenkins and configure it with the GitHub project

1	Access Jenkins Dashboard using the URL http://localhost:8082
2	Select New Item from the menu as highlighted in the below image.

	
3	<p>a. Type the job name as 'CompileandPackage' in the Enter an item name field and select Freestyle Project as the project type as shown below:</p> <p>b. Click OK to create job</p> 
4	<p>This action leads to the Project Configuration page, where you can configure settings for build activity</p> <p>To configure job with GitHub project, perform the following tasks:</p> <p>a. In the Project Configuration page, under the General tab, type the details in the Description field</p>

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Project name: CompileandPackage

Description: Compiling and Packaging the code.

[Plain text] [Preview]

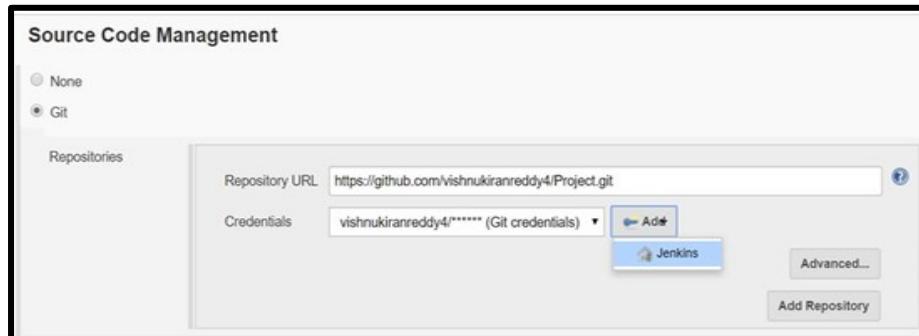
Discard old builds GitHub project

4. Configure Source Code Management (SCM) with Git

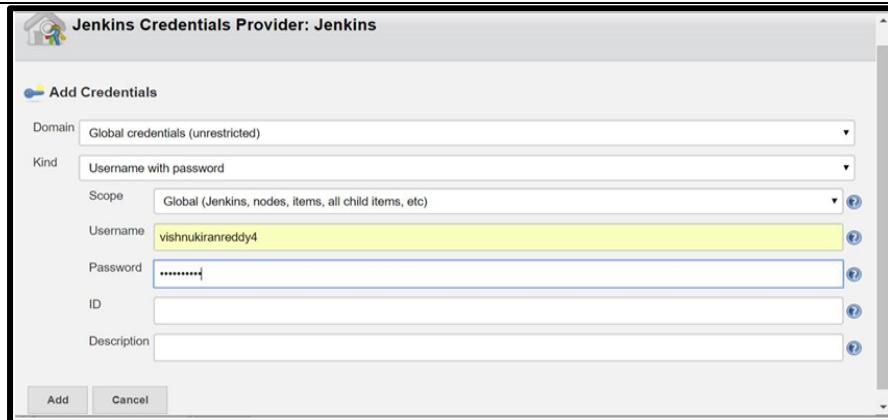
1. a. In the Project Configuration page, navigate to the **Source Code Management** section
 b. Select **Git**
 c. Specify the path in **Repository URL**(Copy your Repository URL from GitHub Server and specify here)



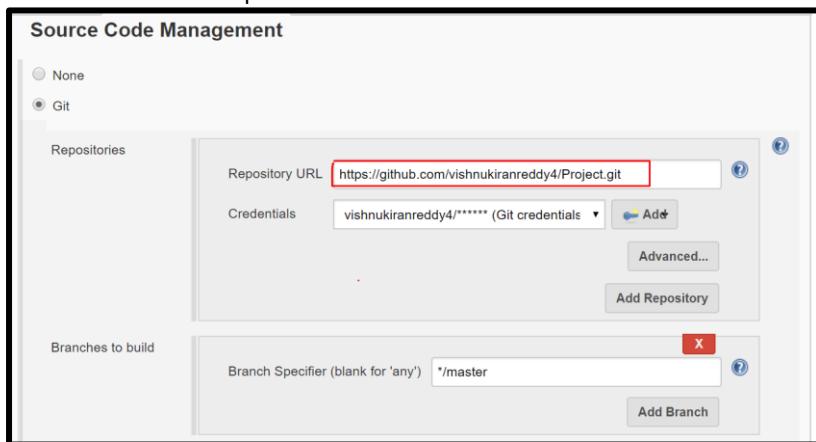
2. Add credentials:
 a. Click **Add** button to add username and password of GitHub account
 b. Click **Jenkins**



This leads to the **Jenkins Credentials Provider** page.
 c. Provide username and password of the GitHub server
 d. Click **Add**

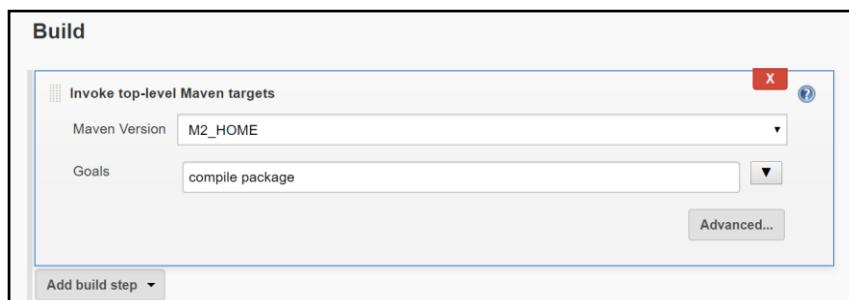


- e. Select the **credentials** from the drop-down list



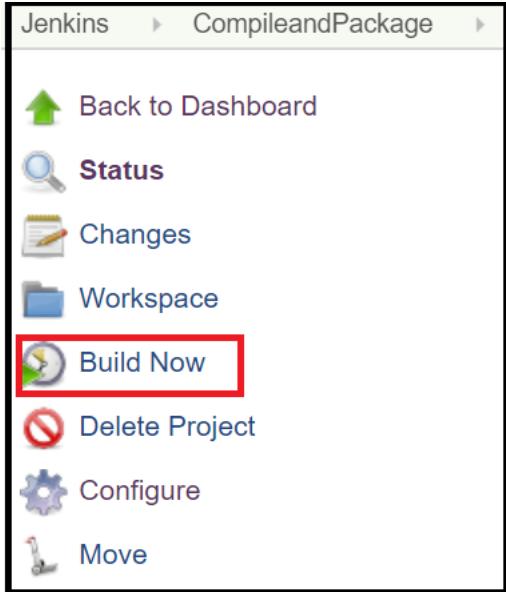
5. Configure build job using Maven

- 1 Now, build job by using the Maven commands as follows:
- Navigate to the **Build** section
 - Choose **Invoke top level Maven targets** from the **Add build step** drop-down menu
 - Specify the Maven version as shown in the screen capture below
 - Type the target name as **compile package** against the **Goals** field



Now, the configuration set up is complete to perform build activity

6. Execute build

1	<p>After the configurations are complete, execute build job manually as follows:</p> <ol style="list-style-type: none"> Click the Save button in the Project Configuration page Navigate to the Jenkins dashboard and select CompileandPackage project Click Build Now to schedule the build to execute immediately  <p>The artifacts (jar/war/ear files) are created on successful build of the project</p>
---	--

7. View build results

1	<p>To view build results, click #1, the build number under Build History of the job in the Jenkins dashboard</p> 
2	<p>Click Console Output</p>



This action will retrieve build results and display them in the console:

```
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [C:\Jenkins\workspace\CompileandPackage\target\spring-petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Jenkins\workspace\CompileandPackage\src\main\webapp]
[INFO] Webapp assembled in [5124 msecs]
[INFO] Building war: C:\Jenkins\workspace\CompileandPackage\target\petclinic.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 57.817 s
[INFO] Finished at: 2018-01-09T15:45:31+05:30
[INFO] Final Memory: 29M/269M
[INFO] -----
Finished: SUCCESS
```

End of Module 3

Module 4

Exercise 4.1: Create a Job and Analyze Code in Jenkins

Perform activities to analyze code using SonarQube Scanner in Jenkins

Prerequisite:

1. Java8(jdk1.8) should be installed in the system.
2. SonarQube and Sonar Scanner to be stored in the C:\Softwares of the local system
3. The project in GitHub server used in Exercise 3.1 is considered in this exercise to perform code analysis

Note:

1. SonarQube and SonarQube Scanner will be provided by faculty during session
2. The SonarQube 6.5 version is used in this exercise

Walkthrough

1. Install SonarQube
2. Install SonarQube Scanner plugin
3. Configure Jenkins with SonarQube
4. Configure Jenkins with SonarQube Scanner
5. Create job in Jenkins and configure it with the GitHub project
6. Configure SCM with Git
7. Configure job using Maven.
8. Perform code analysis using SonarQube Scanner
9. View analysis results

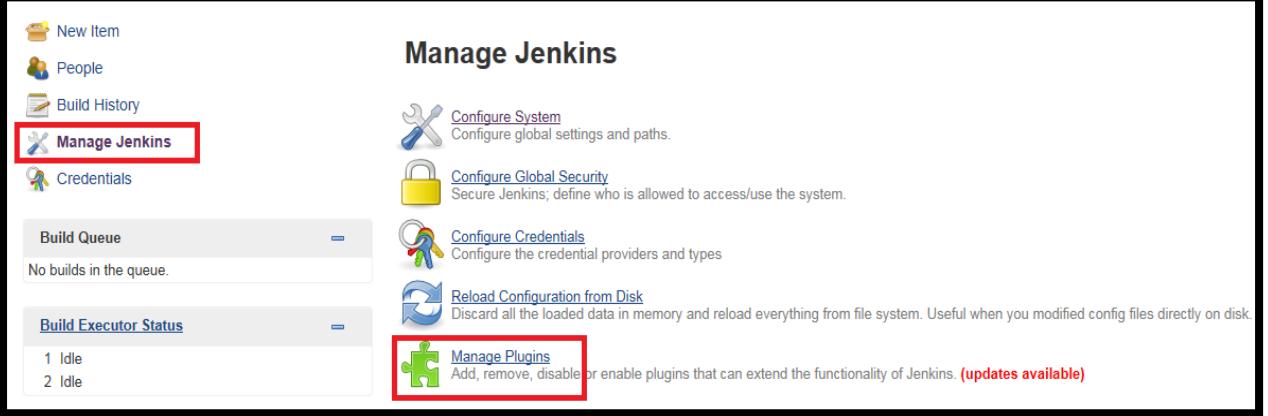
Steps

1. Install SonarQube

1	Navigate to C:\Softwares\sonarqube-6.5\bin\windows-x86-64 folder
---	---

	<p>PC > OSDisk (C:) > Softwares > sonarqube-6.5 > bin > windows-x86-64</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Date modified</th> <th>Type</th> <th>Size</th> </tr> </thead> <tbody> <tr> <td>lib</td> <td>11/12/2017 7:03 PM</td> <td>File folder</td> <td></td> </tr> <tr> <td>InstallNTService</td> <td>10/5/2017 3:40 PM</td> <td>Windows Batch File</td> <td>2 KB</td> </tr> <tr> <td>StartNTService</td> <td>10/5/2017 3:40 PM</td> <td>Windows Batch File</td> <td>2 KB</td> </tr> <tr> <td>StartSonar</td> <td>10/5/2017 3:40 PM</td> <td>Windows Batch File</td> <td>2 KB</td> </tr> <tr> <td>StopNTService</td> <td>10/5/2017 3:40 PM</td> <td>Windows Batch File</td> <td>2 KB</td> </tr> <tr> <td>UninstallNTService</td> <td>10/5/2017 3:40 PM</td> <td>Windows Batch File</td> <td>2 KB</td> </tr> <tr> <td>wrapper</td> <td>10/5/2017 3:40 PM</td> <td>Application</td> <td>216 KB</td> </tr> </tbody> </table>	Name	Date modified	Type	Size	lib	11/12/2017 7:03 PM	File folder		InstallNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB	StartNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB	StartSonar	10/5/2017 3:40 PM	Windows Batch File	2 KB	StopNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB	UninstallNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB	wrapper	10/5/2017 3:40 PM	Application	216 KB
Name	Date modified	Type	Size																														
lib	11/12/2017 7:03 PM	File folder																															
InstallNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB																														
StartNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB																														
StartSonar	10/5/2017 3:40 PM	Windows Batch File	2 KB																														
StopNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB																														
UninstallNTService	10/5/2017 3:40 PM	Windows Batch File	2 KB																														
wrapper	10/5/2017 3:40 PM	Application	216 KB																														
2	<ol style="list-style-type: none"> Click on StartSonar to start the SonarQube server. Once we click on StartSonar, SonarQube server will be up and running. <pre>jvm 1 2017.11.12 18:51:28 INFO app[] [o.s.a.SchedulerImpl] Process[web] is up jvm 1 2017.11.12 18:51:28 INFO app[] [o.s.a.p.JavaProcessLauncherImpl] Launch pr e1.8.0_151\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Xmx512m -Xms128m ava.io.tmpdir=C:\Java\sonarqube-6.5\temp -cp ./lib/common/*;./lib/server/*;./lib/ce/* h2-1.3.176.jar org.sonar.ce.app.CeServer C:\Java\sonarqube-6.5\temp\sq-process7585436 jvm 1 2017.11.12 18:51:35 INFO app[] [o.s.a.SchedulerImpl] Process[ce] is up jvm 1 2017.11.12 18:51:35 INFO app[] [o.s.a.SchedulerImpl] SonarQube is up</pre> <ol style="list-style-type: none"> You can access SonarQube in the path: http://localhost:9000 <p>Note: By default Sonarqube starts with port number is 9000.</p> <p>The screenshot shows the SonarQube homepage. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, and Quality Gates. A search bar is also present. Below the header, there's a summary section titled "Continuous Code Quality" with a "Log in" button and a "Read documentation" link. To the right, it shows "2 Projects Analyzed" with counts for Bugs (0), Vulnerabilities (0), and Code Smells (7). Below this, there's a "Multi-Language" section listing supported programming languages: Java, C/C++, C#, COBOL, ABAP, HTML, RPG, JavaScript, Objective C, XML, VB.NET, PL/SQL, Flex, Python, Groovy, PHP, Swift, Visual Basic, PL/I, and Perl. A note states that over 20+ languages are supported.</p> <ol style="list-style-type: none"> Log in to the SonarQube server with the following credentials: Username: admin Password: admin 																																

2. Install SonarQube Scanner plugin

1	In Jenkins dashboard, navigate to Manage Jenkins → Manage Plugins option.
	
2	<p>a. Click the Available tab under Plugin Manager page and then search and select SonarQube Scanner for Jenkins plugin</p> <p></p> <p>b. Click Install without restart to install the plugin</p>

3. Configure Jenkins with SonarQube

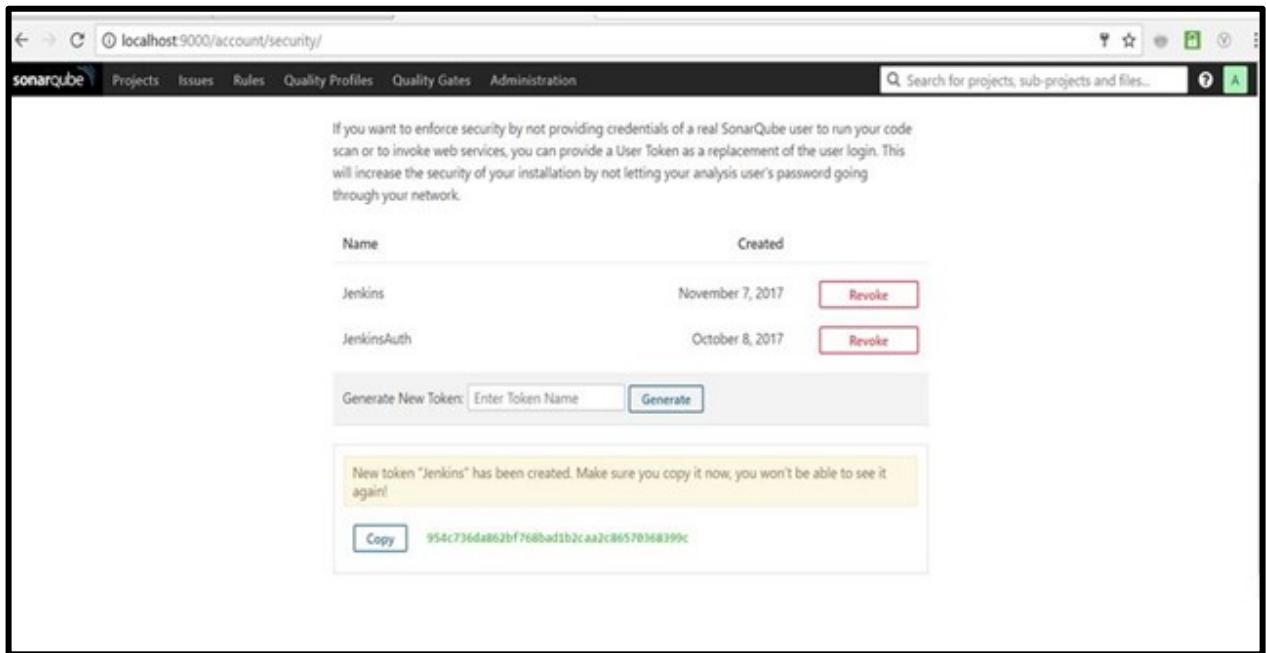
1	Open the Jenkins dashboard in the URL http://localhost:8082
2	Navigate to Manage Jenkins → Configure System

This results in the **Configuration** page, where you can choose the components to configure Jenkins with.

- 3
- In the Configuration page, navigate to the **SonarQube Servers** section
 - Click the box next to **Enable injection of SonarQube server configuration as build environment variables**
 - Click **Add SonarQube** and provide required details as shown below:

Note: **Server authentication token** is generated in SonarQube server. To generate token, perform the following steps:

- | | |
|--|--|
| | <ol style="list-style-type: none"> Access SonarQube server by using the link http://localhost:9000 Log in to the SonarQube server with the following credentials: username: admin, password: admin Click A at the top right-hand corner Navigate to My Account→Security Specify the token name and then click the Generate button to generate the token |
|--|--|

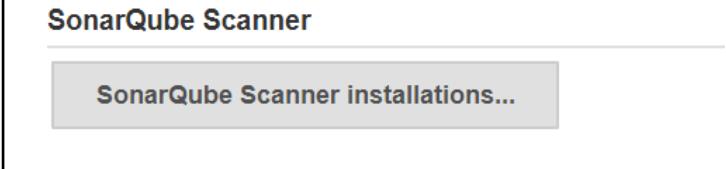
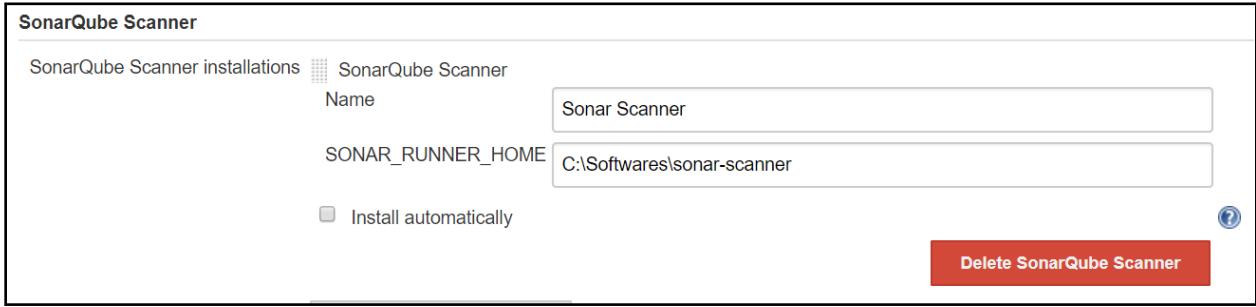


Copy the token and use it while configuring SonarQube server in Jenkins

- | | |
|---|---|
| 5 | Click the Save button to save the SonarQube server configuration |
|---|---|

4. Configure Jenkins with SonarQube Scanner

1	Open the Jenkins dashboard in the URL http://localhost:8082
2	<p>Navigate to Manage Jenkins → Global Tool Configuration option</p> <p>The screenshot shows the Jenkins 'Manage Jenkins' dashboard. On the left, there is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is highlighted with a red box), 'My Views', and 'Credentials'. Below the sidebar, there is a 'Build Queue' section stating 'No builds in the queue.' On the right, there is a main content area titled 'Manage Jenkins' with several configuration options listed:</p> <ul style="list-style-type: none"> Configure System: Configure global settings and paths. Configure Global Security: Secure Jenkins; define who is allowed to access/use the system. Configure Credentials: Configure the credential providers and types. Global Tool Configuration: Configure tools, their locations and automatic installers.

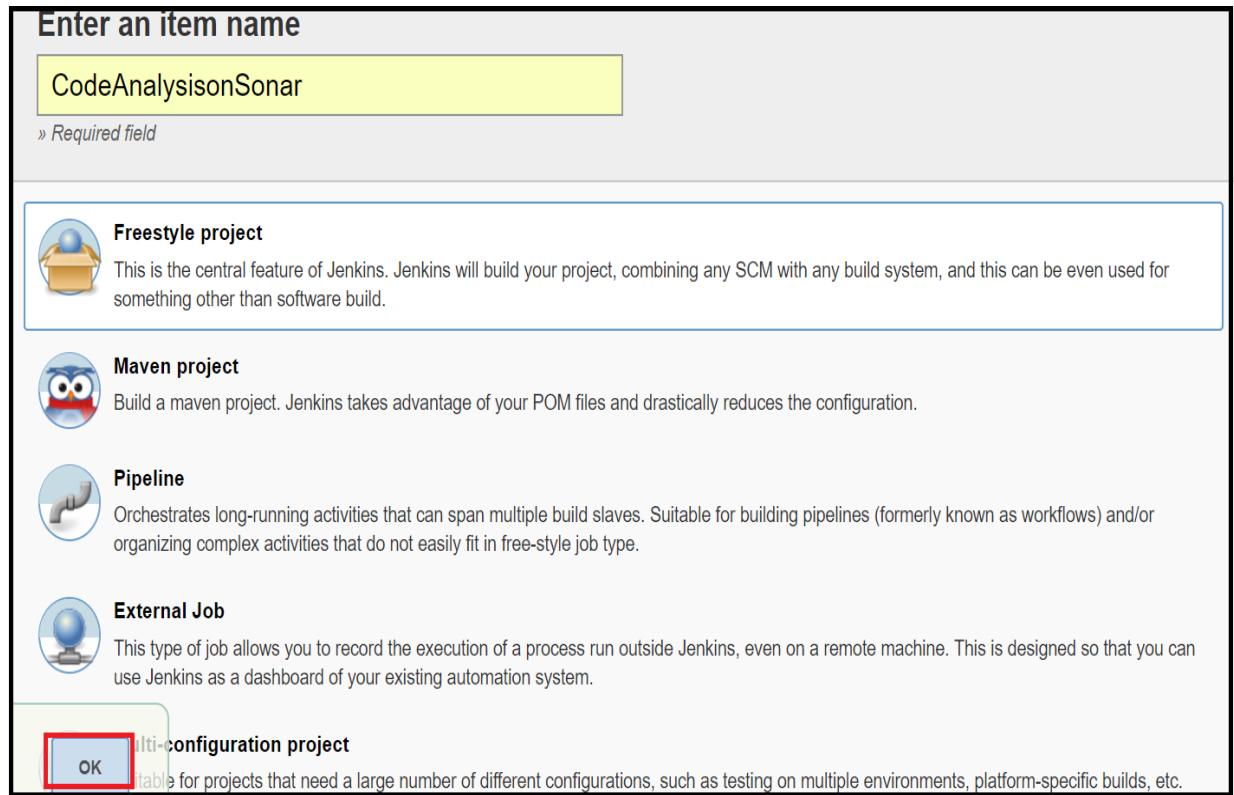
	This results in the Global Tool Configuration page, where you can choose the components to configure Jenkins with.
3	In the Global Tool Configuration page, navigate to the SonarQube Scanner section and Click SonarQube Scanner Installations
	
4	<ul style="list-style-type: none"> a. Uncheck the box next to Install automatically b. Specify the name and path of the SonarQube Scanner in the SONAR_RUNNER_HOME field. The example is shown in the following capture: 
5	Click the Save button to save the SonarQube Scanner configuration

5. Create job in Jenkins and configure it with the GitHub project

1	Access Jenkins Dashboard using the URL http://localhost:8082
2	Select New Item from the menu as highlighted in the below image.



- 3 c. Type the job name as 'CodeAnalysisonSonar' in the **Enter an item name** field and select **Freestyle Project** as the project type as shown below:
d. Click **OK** to create job



Enter an item name

CodeAnalysisonSonar
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

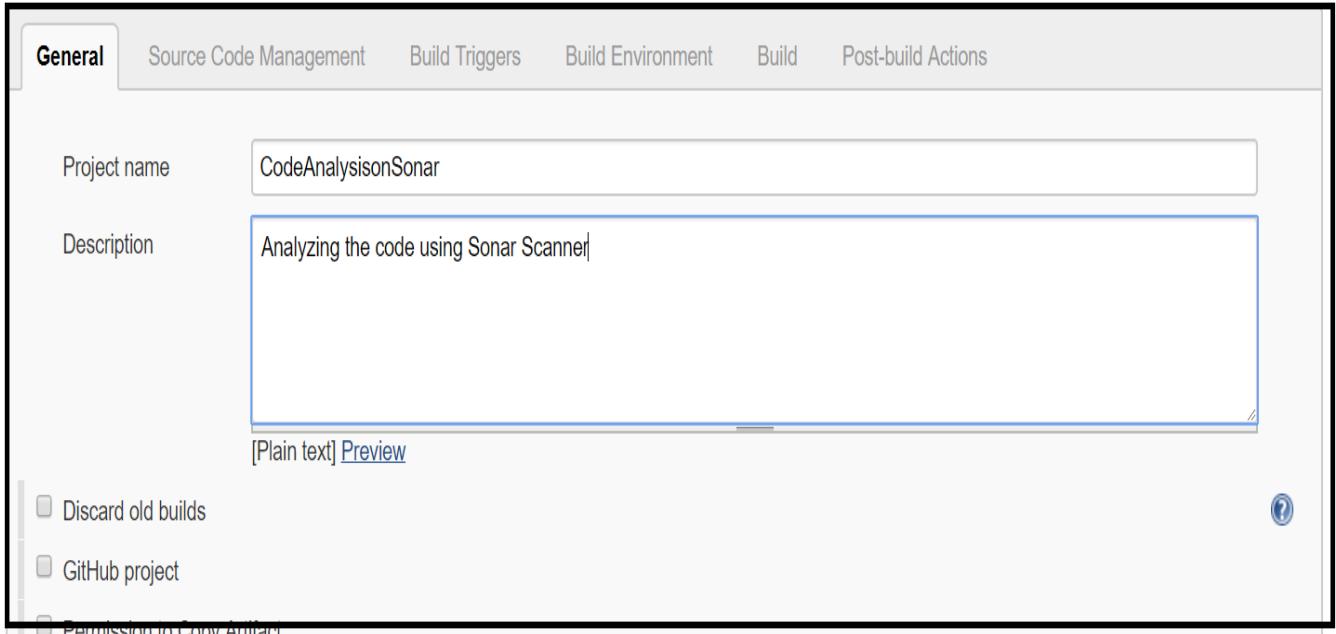
External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

This action leads to the Project Configuration page, where you can configure settings for build activity

- 4 To configure job with GitHub project, perform the following tasks:
b. In the **Project Configuration page**, under the **General** tab, type the details in the **Description** field



General Source Code Management Build Triggers Build Environment Build Post-build Actions

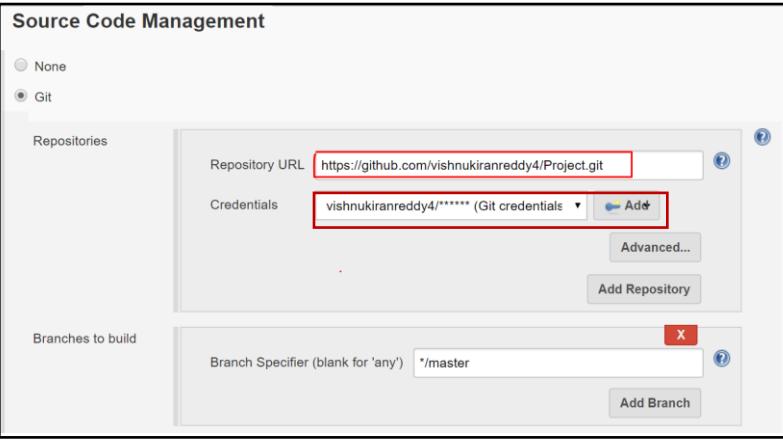
Project name: CodeAnalysisonSonar

Description: Analyzing the code using Sonar Scanner

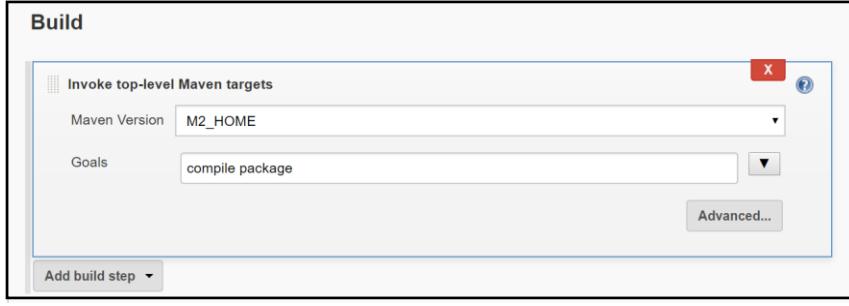
[Plain text] [Preview](#)

Discard old builds [?](#)
 GitHub project
 Permission to Copy Artifact

6. Configure SCM with Git

1	<ul style="list-style-type: none"> a. In the Project Configuration page, navigate to the Source Code Management section b. Select Git
2	<ul style="list-style-type: none"> a. Specify the path in Repository URL b. To add credentials, click Add button to add username and password of GitHub account c. Select the credentials from the drop-down list 

7. Configure job using Maven

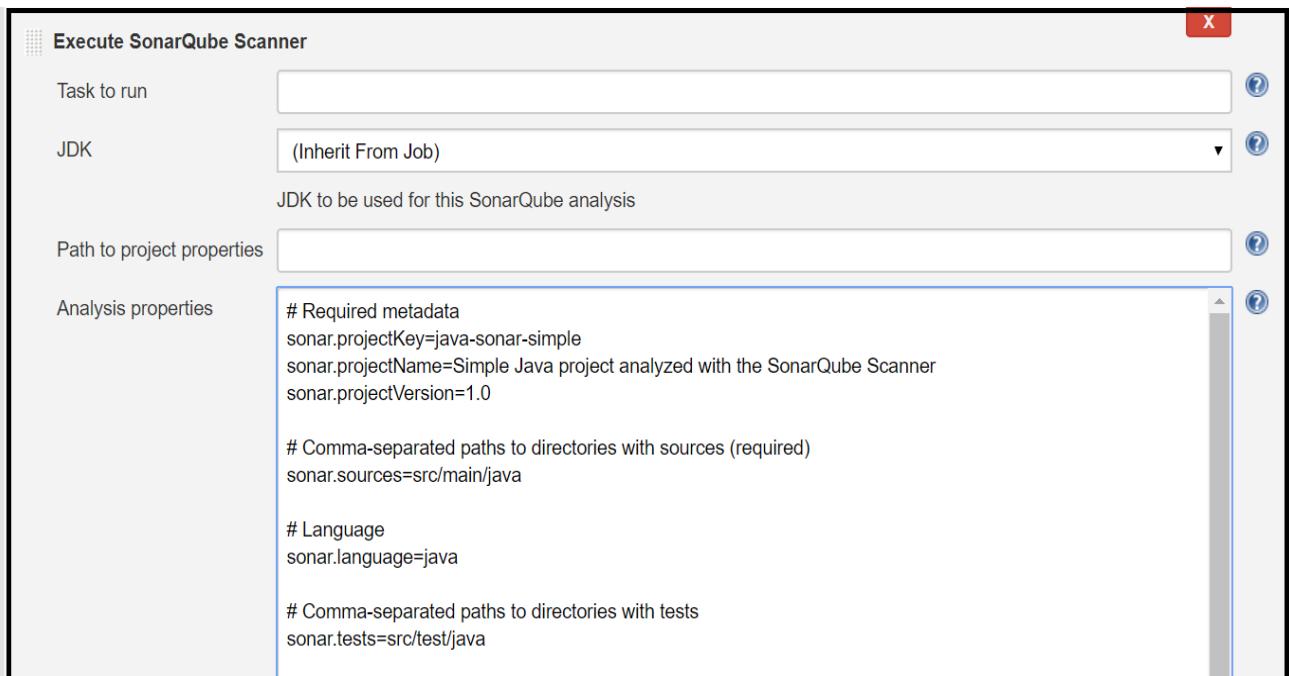
1	<p>Configure the job by using the Maven commands as follows:</p> <ul style="list-style-type: none"> a. Navigate to the Build section b. Choose Invoke top level Maven targets from the Add build step drop-down menu c. Specify the Maven version as shown in the screen capture below d. Type the target name as compile package against the Goals field 
---	---

8. Perform code analysis

1	<p>To perform code analysis by using SonarQube Scanner, follow the steps below:</p> <ul style="list-style-type: none"> a. Navigate to the Build section
---	---

- b. Choose **Execute SonarQube Scanner** from the **Add build step** drop-down menu
 c. Provide the details for **Analysis Properties** as shown below

Note: Copy the properties from the **sonar-project** file present in the **ProjectSourceCode\Module3**.



The screenshot shows the 'Execute SonarQube Scanner' configuration dialog. It includes fields for 'Task to run', 'JDK' (set to 'Inherit From Job'), 'Path to project properties', and 'Analysis properties'. The 'Analysis properties' field displays the following configuration:

```
# Required metadata
sonar.projectKey=java-sonar-simple
sonar.projectName=Simple Java project analyzed with the SonarQube Scanner
sonar.projectVersion=1.0

# Comma-separated paths to directories with sources (required)
sonar.sources=src/main/java

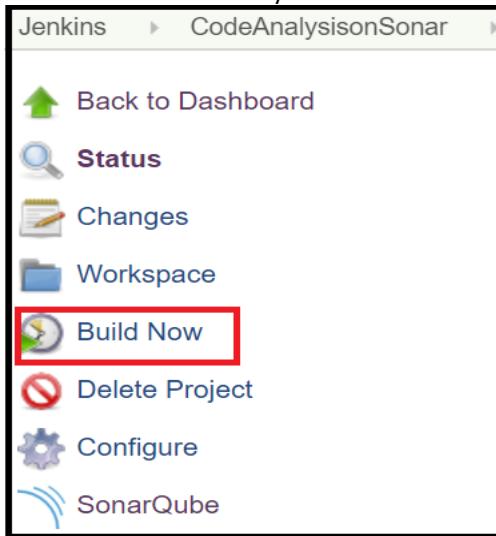
# Language
sonar.language=java

# Comma-separated paths to directories with tests
sonar.tests=src/test/java
```

Now, the configuration set up is complete to perform code analysis.

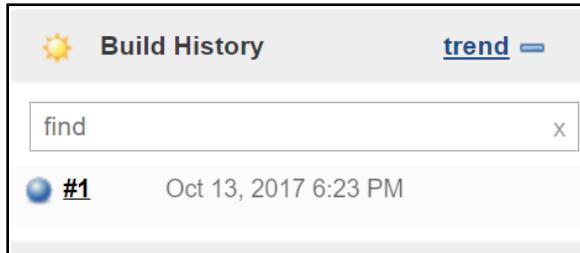
- d. Click the **Save** button in the Project Configuration page

- 2 After the configurations are complete, execute analysis manually as follows:
- Navigate to the Jenkins dashboard and select **CodeAnalysisonSonar** project
 - Click **Build Now** to execute code analysis

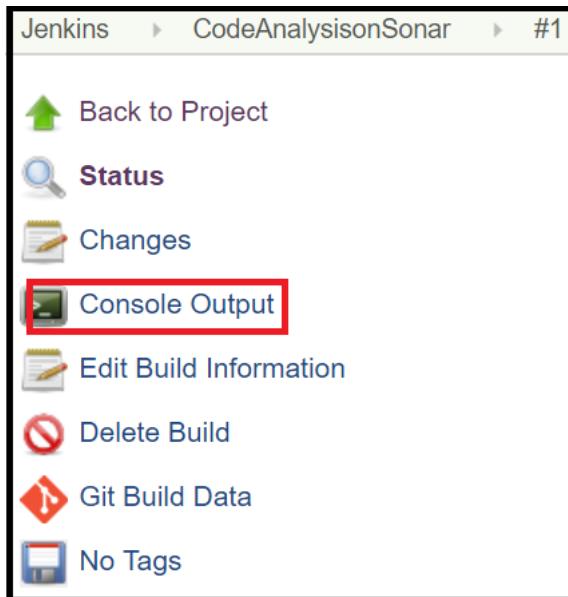


9. View analysis results

- 1 To view build results, click #1, the build number under **Build History** in the Jenkins dashboard for the project, Code Analysis on Sonar



- 2 Click **Console Output**



This action will retrieve results in console.

```

INFO: Analysis report generated in 864ms, dir size=253 KB
INFO: Analysis reports compressed in 343ms, zip size=164 KB
INFO: Analysis report uploaded in 56ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard/index/java-sonar-simple
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AwDafqqNEjf7MpilKSHo
INFO: Task total time: 13.411 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 15.119s
INFO: Final Memory: 56M/438M
INFO: -----
Finished: SUCCESS

```

- Access the above link in the console output to see the analysis results

The screenshot shows the SonarQube web interface. At the top, the navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. Below the navigation, a project card displays the name "Simple Java project analyzed w...". A horizontal menu bar below the card includes links for Home, Issues, Measures, Code, Activity, and Administration. The "Issues" link is underlined, indicating it is the active page. A prominent green button labeled "Passed" indicates the project's quality gate status. Below this, a section titled "Bugs & Vulnerabilities" shows two metrics: "0 Bugs" and "0 Vulnerabilities", each accompanied by a green circular icon containing the letter "A".

End of Module 4

*Module 5***Exercise 5.1: Create a Job to Deploy WAR File in Jenkins**

Perform activities to create a project and deploy it in Tomcat server

Prerequisite:

1. Add the **Deploy to Container Plugin** and **Copy Artifacts Plugin** to Jenkins to perform deployment
(Install the plugin as mentioned in the steps in the exercise 2.3)
2. Tomcat should be installed in your local system
3. The artifact created in Exercise 3.1 is considered in this exercise for deployment in Tomcat
4. A ‘Free Style Project’ named **DeploytoTomcat** should be created in Jenkins for deployment
(Please follow the steps in exercise 3.1 to create the job)

Note:

1. Tomcat will be shared by the faculty during the session.

The screenshot shows the Jenkins project configuration interface. The 'General' tab is active. In the 'Project name' field, the value 'DeploytoTomcat' is entered. The 'Description' field contains the text 'Deploying the war file on the tomcat server.'. Below these fields, there are links for '[Plain text]' and '[Preview]'. At the bottom left, there is a checkbox labeled 'Discard old builds' with an unchecked state. On the right side, there is a small blue help icon.

Walkthrough

1. Perform build activities to deploy the project
2. Perform post build tasks to deploy the project
3. Execute project deployment
4. View results

Steps

1. Perform build activities to deploy the project

1	In the Project Configuration page, navigate to the Build section
2	Choose Copy artifacts from another project from the Add build step drop-down list
3	<ol style="list-style-type: none"> a. Specify CompileandPackage as Project Name (i.e., Project from which we should copy the artifacts) b. Select Copy from WORKSPACE of the latest completed build from the drop-down list

Build

Copy artifacts from another project

Project name: CompileandPackage

Which build: **Copy from WORKSPACE of latest completed build**

Limitation Note:

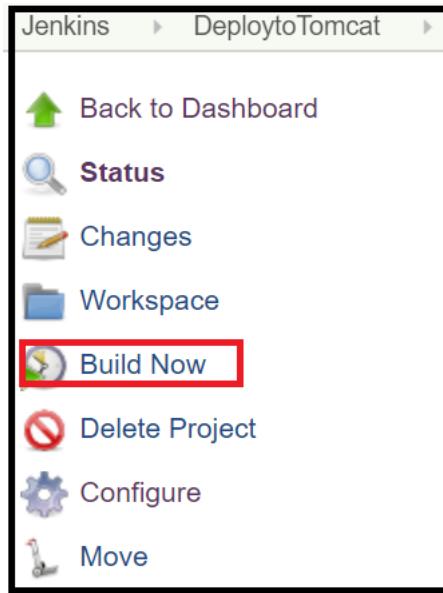
Artifacts to copy:

2. Perform post build tasks to deploy the project

1	In the Project Configuration page, navigate to the Post-build Actions section
2	<p>Configure Tomcat to deploy project</p> <ol style="list-style-type: none"> From the Add Post-build action drop-down list, choose Deploy war/ear to a container Specify WAR/EAR files as **/*.war Specify Context Path as petclinic.war Click Add Container drop-down menu and choose Tomcat 7.x and provide the details as shown below Click Add to add credentials of Tomcat and select it from the drop-down list <p>Post-build Actions</p> <p><input checked="" type="checkbox"/> Deploy war/ear to a container</p> <p>WAR/EAR files: **/*.war</p> <p>Context path: petclinic.war</p> <p>Containers: Tomcat 7.x</p> <p>Credentials: admin/*****</p> <p>Tomcat URL: http://localhost:8080</p> <p>Add Container</p> <p>Deploy on failure: <input type="checkbox"/></p> <p>Add post-build action</p> <p>f. Click Save to save the configurations</p>

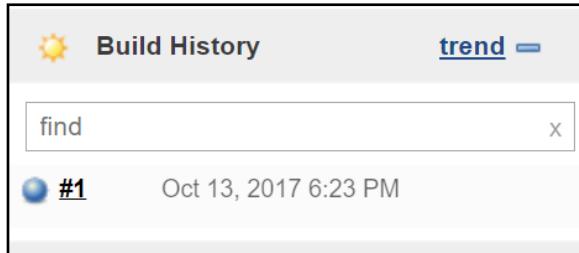
3. Execute project deployment

- 1 Now that the deployment configurations are complete, execute deployment job manually by following the below steps:
- In the Jenkins dashboard, navigate to the **DeploytoTomcat** project
 - Schedule the deployment to be executed immediately by clicking **Build Now**

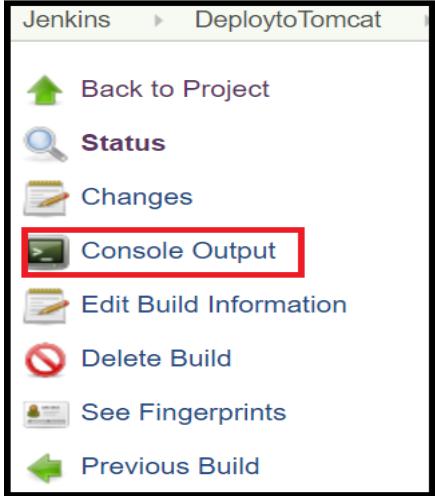
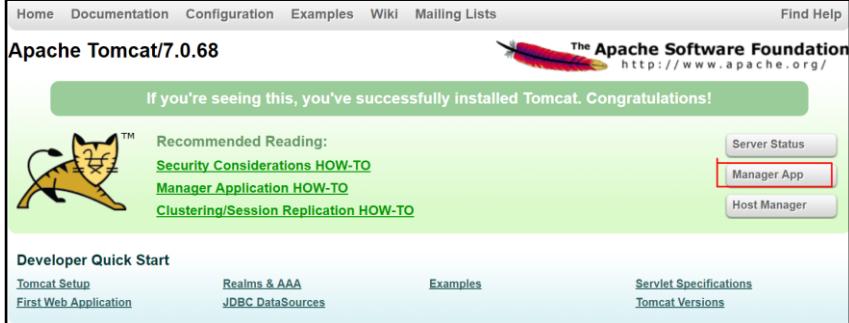


4. View results

- 1 To view build results, click #1, the build number under **Build History** in the Jenkins dashboard for the project, Deploy to Tomcat

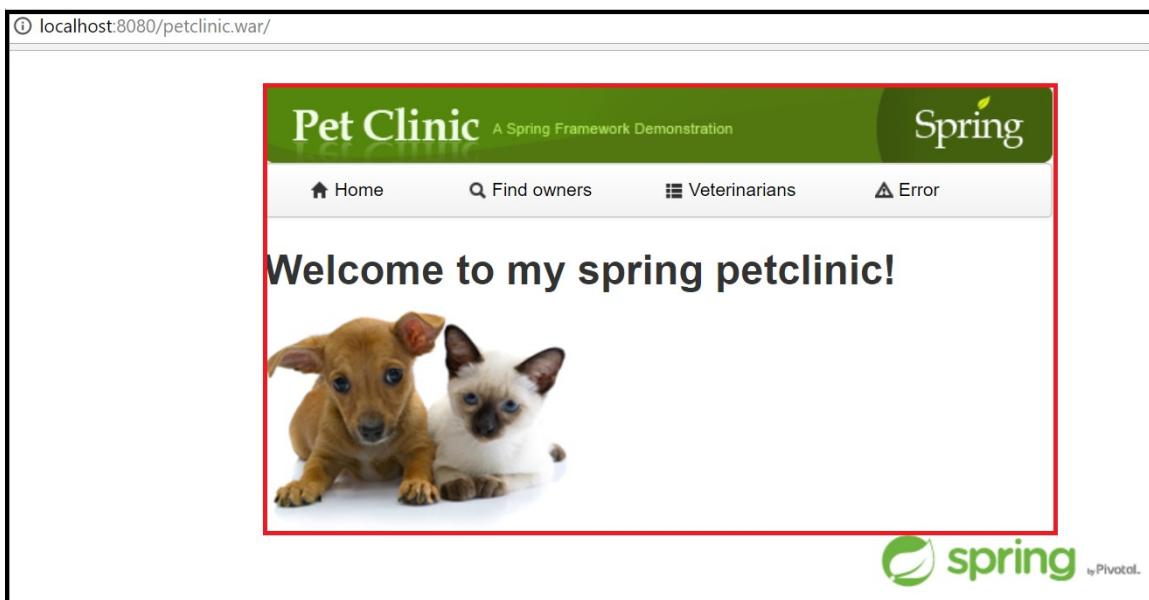


- 2 Click **Console Output**

	 <p>This action will retrieve results in console.</p>
3	<p>To verify deployment, perform the following tasks:</p> <ol style="list-style-type: none"> Go to browser and access the link http://localhost:8080 to see the Tomcat home page Click the Manager App icon to see the deployed war file (i.e., petclinic.war)  <p>c. Click the petclinic.war file to check the output</p>

Manager						
List Applications		HTML Manager Help		Manager Help		Server Status
Applications						
Path	Version	Display Name	Running	Sessions	Commands	
/	None specified	Welcome to Tomcat	true	0	Start	Stop Reload Undeploy
/docs	None specified	Tomcat Documentation	true	0	Start	Stop Reload Undeploy
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy	Expire sessions with idle ≥ 30 minutes
/petclinic.war	None specified	Spring PetClinic	true	0	Start Stop Reload Undeploy	Expire sessions with idle ≥ 30 minutes

You can see that the deployed application is up and running



End of Module 5

Module 6

Exercise 6.1: Test application with Selenium

Scenario

Perform activities to test with Selenium and view test results using TestNG in Jenkins

Prerequisite:

- The project will be provided by the faculty to test the application with selenium.
- Include the plugin, TestNG Results Plugin to view TestNG results
- Add **Selenium Plugin** to test the application with Selenium
- **Mozilla Firefox** and **Google Chrome** should be installed for testing

Note:

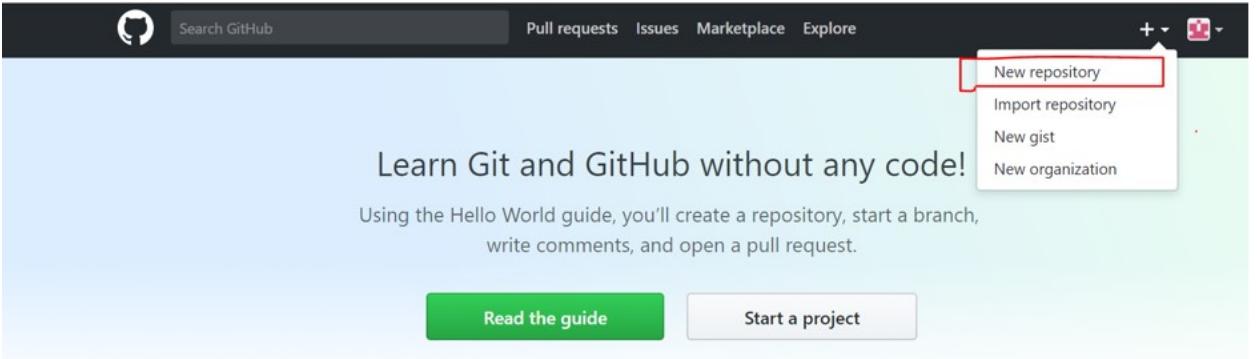
1. The source code will be shared by faculty
2. The ProjectSourceCode should be stored in C:\Softwares of the local system.

Walkthrough

1. Create project in GitHub server.
2. Create job in Jenkins and configure it with the GitHub project
3. Configure Source Code Management with Git
4. Configure build with Maven and test results with TestNG
5. View results using TestNG

Steps

1. Create project in GitHub server.

1	Sign in to the GitHub server
2	Create repository <ol style="list-style-type: none"> Click New repository from the drop-down list of the GitHub home page 

- e. Specify the name (example: Project-Testing) in the **Repository name** field and click the checkbox next to **Initialize this repository with README** option
- f. Click the **Create Repository** button to create repository in GitHub server

Create a new repository
A repository contains all the files for your project, including the revision history.

Owner: vishnukiranreddy4 / Repository name: Project-Testing

Great repository names are short and memorable. Need inspiration? How about [refactored-octo-tribble](#).

Description (optional):

Public: Anyone can see this repository. You choose who can commit.

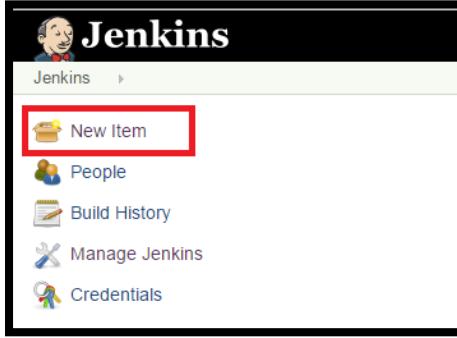
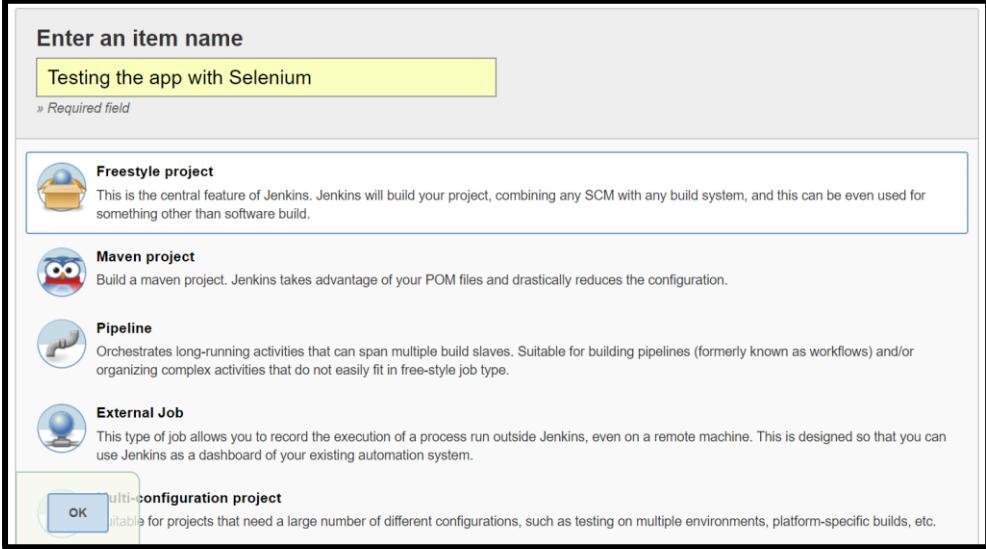
Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** | ⓘ

Create repository

3	<p>Clone the repository</p> <p>Clone the project that you created by using the git clone command. The path to the project can be found in GitHub server: The path looks as shown below as per your credentials:</p> <pre>git clone https://github.com/vishnukiranreddy4/Project-Testing.git</pre>
4	<p>Navigate to Git bash on your workstation and make sure that you have changed the directory to the right directory where you want to clone your project and execute the git clone command(Create a folder JenkinsPractice in C drive)</p> <ul style="list-style-type: none"> c. cd /c/JenkinsPractice d. git clone https://github.com/vishnukiranreddy4/Project-Testing.git
5	<p>Change to your Git repository directory:</p> <ul style="list-style-type: none"> • cd Project-Testing
6	<p>Copy the project content into the "Project-Testing" folder manually from the path C:\Softwares\ProjectSourceCode\Module6 without using Git command.</p> <p>Note: Copy the code which is present in ProjectSourceCode\Module6 folder(C:\Softwares\ProjectSourceCode\Module6) into Project-Testing folder(Cloned from the GitHub Server).</p>
7	<p>After copying content, navigate to git bash and execute the below commands</p> <ul style="list-style-type: none"> • git add . • git status • git commit -m "Adding project content" • git push origin master

	<p>Now, the project is pushed to GitHub server Note: Enter password which you have given while creating the account in GitHub.</p>
	<p>2. Create job in Jenkins and configure it with the GitHub project</p> <p>1 Select New Item in the Jenkins dashboard from the menu as highlighted in the below image.</p> 
3	<p>a. Type the job name as Testing the app with Selenium in the Enter an item name field and select Freestyle Project as the project type as shown below:</p> <p>b. Click OK to create job</p>  <p>This action leads to the Project Configuration page, where you can configure settings for testing activity</p>
4	<p>To configure job with GitHub project, perform the following tasks:</p> <p>a. In the Project Configuration page, under the General tab, type the details in the Description field</p>

3. Configure Source Code Management with Git

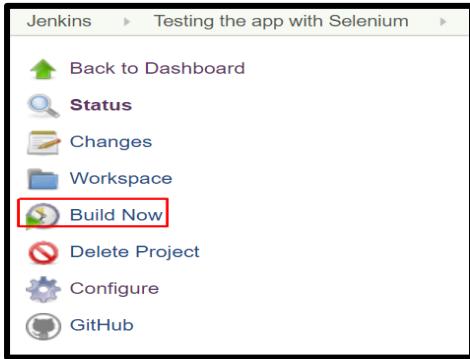
4	To configure SCM with GitHub project, perform the following tasks:
---	--

- a. In the **Project Configuration page**, under the **Source Code Management** section:
- Select **Git**
 - Specify Repository URL
 - Select the credentials from dropdown
 - Click **Save** to save the configuration

The screenshot shows the 'Source Code Management' section of the Jenkins project configuration. Under 'Repositories', 'Git' is selected. The 'Repository URL' field contains 'https://github.com/vishnukiranreddy4/Project-Testing.git'. The 'Credentials' dropdown is set to 'vishnukiranreddy4/***** (Git credentials)'. There are 'Advanced...' and 'Add Repository' buttons. Under 'Branches to build', the 'Branch Specifier' field contains '/master'. There is an 'Add Branch' button.

4. Configure build with Maven and test results with TestNG

- 1 To build job using maven commands, do the following steps.
- Navigate to **Build** section
 - Choose **Invoke top-level Maven targets** from **Add build step** drop-down list.
 - Specify the Maven version and type target name as shown below to execute clean and package goals in Maven
-
- The screenshot shows the 'Build' section of the Jenkins project configuration. It includes an 'Invoke top-level Maven targets' step. The 'Maven Version' is set to 'M2_HOME' and the 'Goals' field contains 'clean package'. There is an 'Advanced...' button and an 'Add build step' dropdown.
- 2
- In the Project Configuration page, navigate to the Post-build Actions section
 - From the Add post-build action drop-down items, select Publish TestNG Results
 - Specify the XML report pattern as shown, to save results during build execution

	
3	<p>Once configurations are completed, execute build job manually by following the below steps:</p> <ol style="list-style-type: none"> Click Save Schedule the build to be executed immediately by clicking the Build Now menu item 

5. View results using TestNG

1	<p>View the generated test results by following the below steps:</p> <ol style="list-style-type: none"> Select the execute build number Click TestNG Results link 
---	---

The test TestNG results are as shown below:

<h2>TestNG Results</h2>							
0 failures(±0)							
							4 tests(+4)
<h3>Failed Tests</h3>							
No Test method failed							
<h3>All Tests (grouped by their packages)</h3>							
hide/expand the table							
Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
springpetclinic_selenium.selenium	00:00:01.372	0	0	0	0	2	0
springpetclinic.selenium	00:00:01.454	0	0	0	0	2	0

End of Module 6

Module 7

Exercise 7.1: Create Job Pipeline

Scenario

Perform activities to create pipeline and configure jobs to execute in the Pipeline.

Prerequisite:

- Include the plugin, **Build Pipeline Plugin** in Jenkins configuration to create and work with Pipeline
- The jobs to include in the Pipeline should be created before creating Pipeline

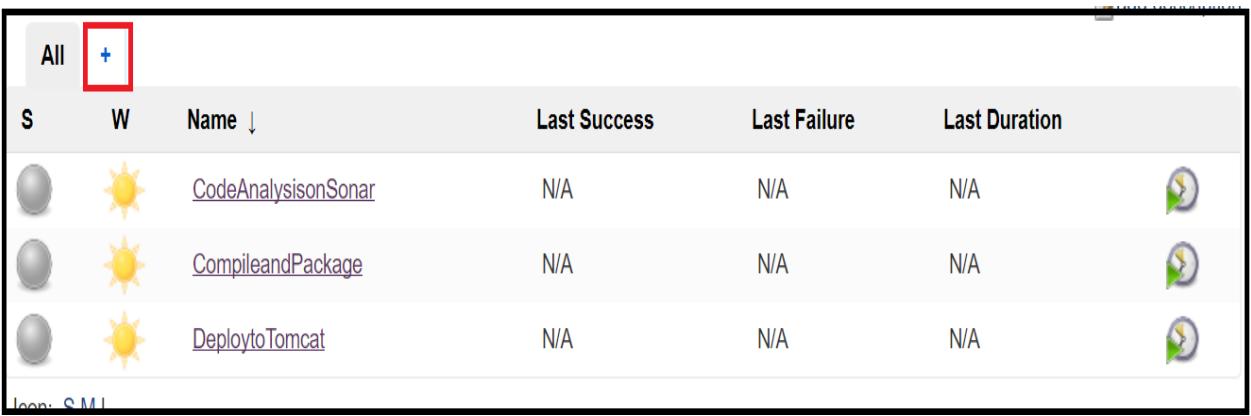
Note: This exercise considers 3 jobs, [CompileandPackage](#), [CodeAnalysisonSonar](#) and [DeploytoTomcat](#) to execute in the Pipeline

Walkthrough

- Create Pipeline
- Configure sequence of jobs in Pipeline
- Execute Pipeline – 1 Touch and 0 Touch deployment

Steps

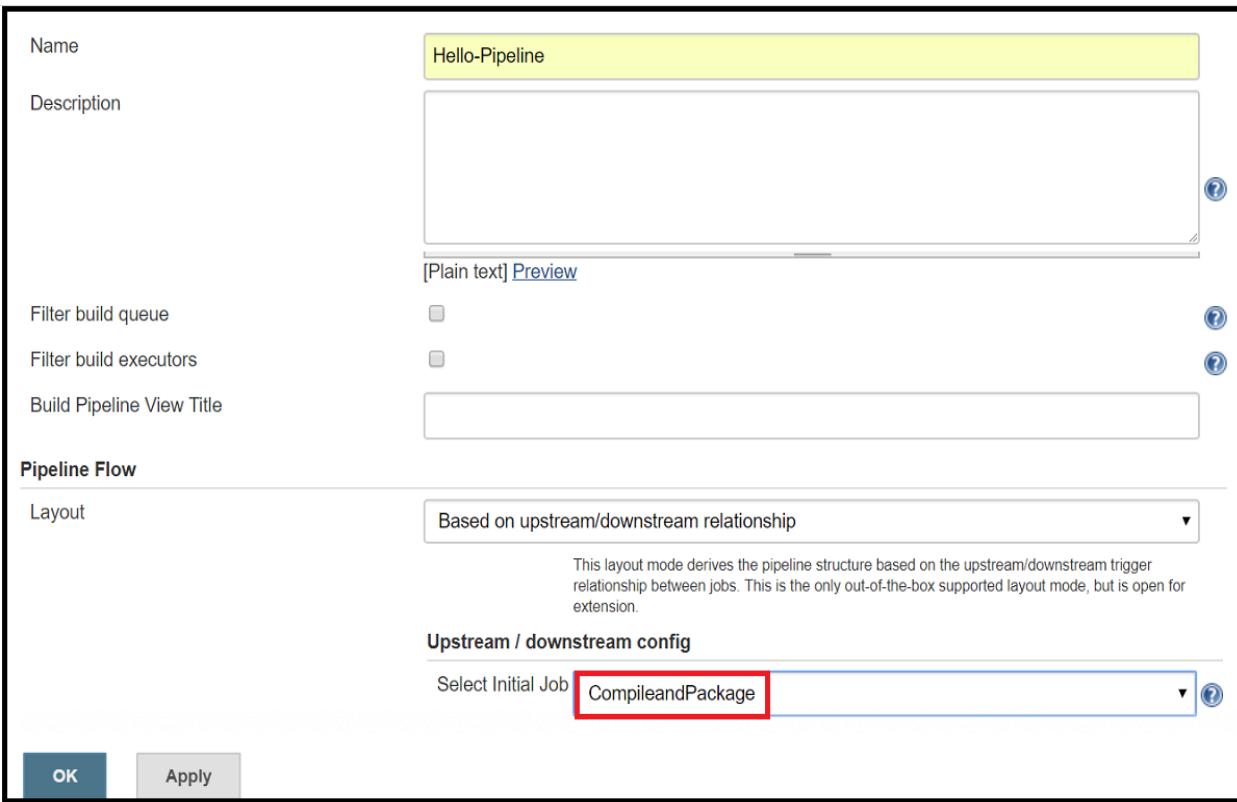
1. Create Pipeline

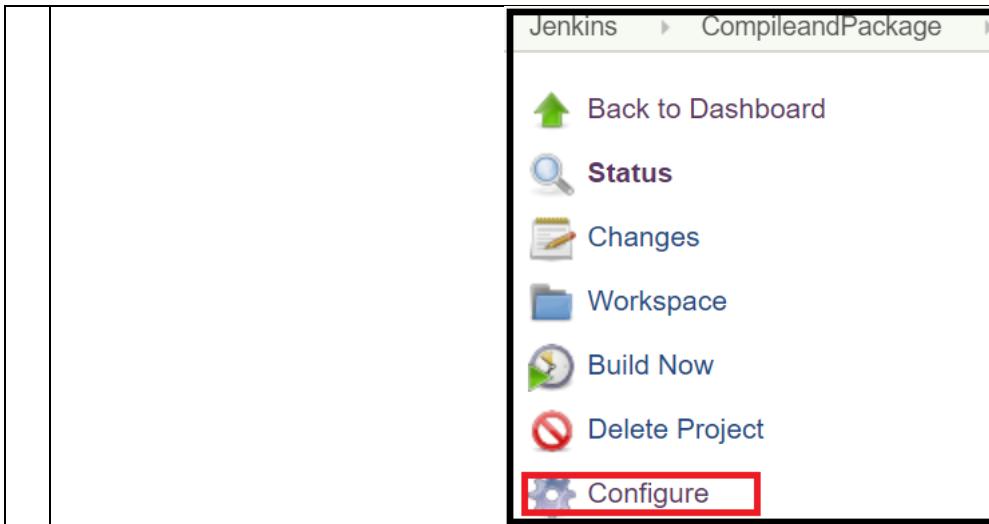
1	<p>To create a pipeline, click on the “+” option in the Jenkins dashboard</p> 
	<ol style="list-style-type: none"> To name the Pipeline, provide the details for View name as Hello-Pipeline Check the box next to Build Pipeline View Click OK

	<p>View name <input type="text" value="Hello-Pipeline"/></p> <p><input checked="" type="radio"/> Build Pipeline View Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.</p> <p><input type="radio"/> List View Shows items in a simple list format. You can choose which jobs are to be displayed in which view.</p> <p><input type="radio"/> My View This view automatically displays all the jobs that the current user has an access to.</p> <p><input type="radio"/> Report Info A view with some information from Surefire, PMD, Findbugs and Checkstyle reports.</p> <p style="text-align: center;">OK</p>
--	---

This will successfully create a pipeline and Jenkins will display configurations screen immediately

2. Configure sequence of jobs in Pipeline

1	<p>Select the first job to execute in the Pipeline.</p> <ol style="list-style-type: none"> In this exercise, select CompileandPackage for the Select Initial Job field to indicate it as the first job to execute in the Pipeline, Hello-Pipeline as shown below: Click OK 
2	<p>After selecting the initial job, you configure the first job</p> <ol style="list-style-type: none"> Navigate to the Jenkins Dashboard Click the job, CompileandPackage. Click Configure



- 3 Next, to configure the second job, **CodeAnalysisonSonar** in the Pipeline,
a. Navigate to the **Post-build Actions** section in the Project Configuration page of the Compile and Package job

A screenshot of the "General" configuration page for a Jenkins job named "CompileandPackage". The "Post-build Actions" tab is selected and highlighted with a red box. The page shows the following fields:

- Project name: CompileandPackage
- Description: Compiling and Packaging the code.
- [Plain text] [Preview]
- Discard old builds
- GitHub project

- b. From the **Add post-build action** dropdown list, choose **Build other projects**
c. Specify the next job which we want to build after the successful execution of first job, Code Analysis on Sonar as shown below
d. Click **Save**

Post-build Actions

Build other projects

Projects to build **CodeAnalysisonSonar**

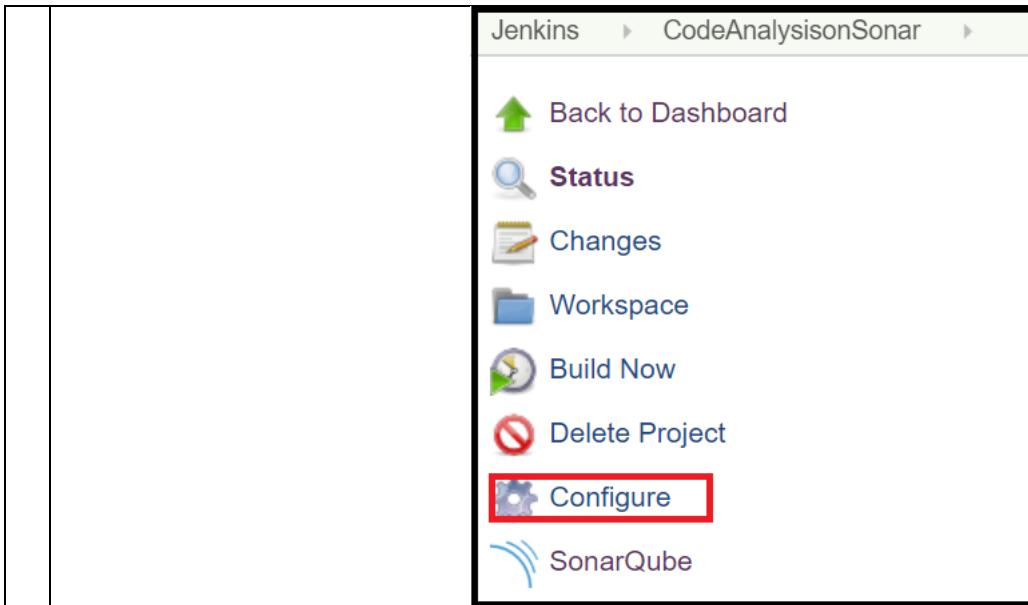
Trigger only if build is stable
 Trigger even if the build is unstable
 Trigger even if the build fails

Save **Apply**

e. Navigate to the Jenkins dashboard and click the job, **CodeAnalysisonSonar**

All	+				
S	W	Name ↓	Last Success	Last Failure	Last Duration
		CodeAnalysisonSonar	N/A	N/A	N/A
		CompileandPackage	N/A	N/A	N/A
		DeploytoTomcat	N/A	N/A	N/A

f. Click **Configure**



This action leads to the Configuration Page of the Code Analysis on Sonar job, from where you can configure the next job in Pipeline

- g. Similarly, by following the same steps mentioned above, configure the third job, DeploytoTomcat on the Project Configuration page of the CodeAnalysisonSonar job

3. Execute Pipeline – 1 Touch and 0 Touch deployment

- 1 After all the jobs are configured, you can execute jobs as per the specified configuration:

- a. Navigate to the Jenkins dashboard
- b. Click the **Hello-Pipeline** created
- c. Click the **Run** icon (This is for **1 touch** deployment)

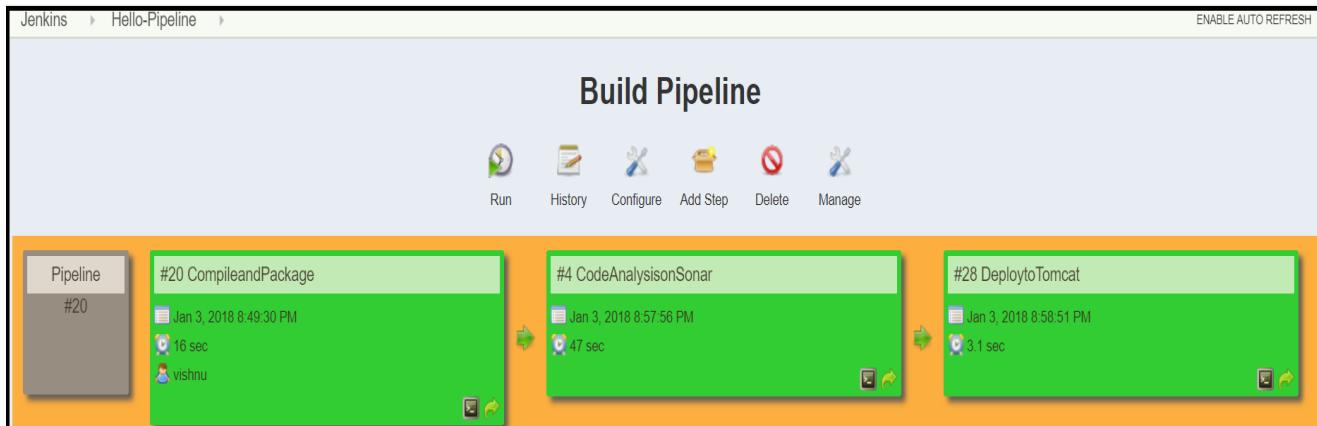
The screenshot shows the Jenkins dashboard with the 'Hello-Pipeline' pipeline listed. The pipeline consists of three stages: 'CodeAnalysisonSonar', 'CompileandPackage', and 'DeploytoTomcat'. Each stage has a green sun icon indicating success. The 'Run' icon for the 'CodeAnalysisonSonar' stage is highlighted with a red rectangular box.

S	W	Name ↓	Last Success	Last Failure	Last Duration
1		CodeAnalysisonSonar	N/A	N/A	N/A
2		CompileandPackage	N/A	N/A	N/A
3		DeploytoTomcat	N/A	N/A	N/A

- d. Click **Run**

The screenshot shows the 'Build Pipeline' page for the 'Hello-Pipeline'. At the bottom of the page, there is a toolbar with several icons: Run, History, Configure, Add Step, Delete, and Manage. The 'Run' icon is highlighted with a red rectangular box.

This action triggers jobs to execute in the specified sequential order.



- 2 After all the jobs are configured, you can execute jobs as per the specified configuration for **0 touch** deployment.
- Navigate to Jenkins dashboard -> CompileandPackage -> Configure
 - Go to Build Triggers section
 - Check the check box **Trigger builds remotely** and enter authentication token (any random text for example:ausdj4537)

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Authentication Token ?

Use the following URL to trigger build remotely: JENKINS_URL/job/1.Compile%20and%20Package/build?token=TOKEN_NAME
or /buildWithParameters?token=TOKEN_NAME

Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

Build after other projects are built ?

Build periodically ?

Gerrit event ?

- Click the **Save** button to save the configuration.
- Go to Jenkins dashboard. Click the **Full name** present in the top right corner and click **Configure**
- Click the **Show API Token** button
- Copy the **Name** and **API Token** and save in notepad.

h. Go to **Git Bash** and go to project repository

- cd Project
- cd .git
- cd hooks
- vi post-commit

In the vi editor press **p** to paste the below script.

```
#!/bin/sh
curl --user <jenkinslogin>:<API Token> http://localhost:8082/job/CompileandPackage/build?token= ausdj4537
echo "jenkins from an external script"
```

i. Type :wq and press enter

j. Go to **Project** folder(Cloned from GitHub Server) do some changes in the **README** file and push your changes to repository by referring the Exercise 3.1

k. Now you observe the pipeline that is triggered automatically

Appendix:

Steps to set up environment variable(jdk1.8):

1. Open Jenkins dashboard in the URL <http://localhost:8082>
2. Click on **Manage Jenkins → Configure System**

The screenshot shows the Jenkins Manage Jenkins dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins (which is highlighted with a red box), and Credentials. Below the sidebar are two collapsed sections: Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). On the right, there are several configuration links: Configure System (highlighted with a red box), Configure Global Security, Configure Credentials, Reload Configuration from Disk, Manage Plugins (with a note about updates available), System Information, and System Log.

3. Navigate to **Global properties** and check the box next to **Environment variables**.
 - Click on Add .
 - Specify the name and path of the JDK installed in the system in the JAVA_HOME field as shown below.

The screenshot shows the Jenkins Global properties screen. The 'Environment variables' checkbox is checked. A single entry is listed: Name: JAVA_HOME and Value: C:\Program Files\Java\jdk1.8.0_144. There are 'Add', 'Delete', and '?' buttons at the bottom.

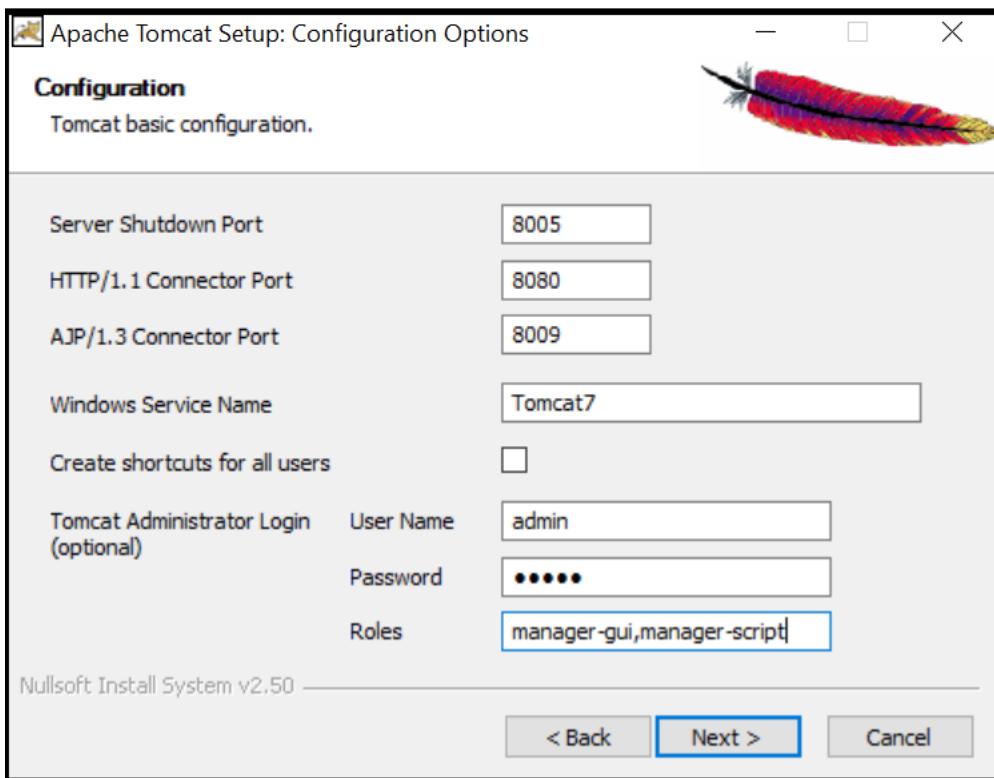
Steps to install Tomcat:

1. Navigate to **C:\Softwares\Apache-Tomcat** folder.
2. Start installation(apache-tomcat-7.0.68).
3. Add the following configurations as shown below:

User Name: admin

Password: admin

Roles: manager-gui,**manager-script**



4. Keep remaining options as default and click in Install button.
5. Once the installation is successfully completed, you can access tomcat in the path <http://localhost:8080>

Note: By default, Tomcat starts with port number 8080.

End of exercises