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

|  |  |
| --- | --- |
| 1 | Access Jenkins Dashboard using the URL [**http://localhost:8082**](http://localhost:8082) |
| 2 | Select **New Item** from the menu as highlighted in the below image. |
| 3 | 1. Type the job name as ‘CompileandPackage’ in the **Enter an item name** field and select **Freestyle Project** as the project type as shown below: 2. Click **OK** to create job   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Compile.png  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:   1. In the **Project Configuration page**, under the **General** tab, type the details in the **Description** field   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Git.png |

1. Configure Source Code Management (SCM) with Git

|  |  |
| --- | --- |
| 1 | 1. In the Project Configuration page, navigate to the **Source Code Management** section 2. Select **Git** 3. Specify the path in **Repository URL(**Copy your Repository URL from GitHub Server and specify here**)** |
| 2 | Add credentials:   1. Click **Add** button to add username and password of GitHub account 2. Click **Jenkins**     This leads to the **Jenkins Credentials Provider** page.   1. Provide username and password of the GitHub server 2. Click **Add**      1. Select the **credentials** from the drop-down list |

1. Configure build job using Maven

|  |  |
| --- | --- |
| 1 | Now, build job by using the Maven commands as follows:   1. Navigate to the **Build** section 2. Choose **Invoke top level Maven targets** from the **Add build step** drop-down menu 3. Specify the Maven version as shown in the screen capture below 4. Type the target name as **compile package** against the **Goals** field     Now, the configuration set up is complete to perform build activity |

1. Execute build

|  |  |
| --- | --- |
| 1 | After the configurations are complete, execute build job manually as follows:   1. Click the **Save** button in the Project Configuration page 2. Navigate to the Jenkins dashboard and select **CompileandPackage** project 3. Click **Build Now** to schedule the build to execute immediately   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Build.png  The artifacts (jar/war/ear files) are created on successful build of the project |

1. View build results

|  |  |
| --- | --- |
| 1 | To view build results, click **#1**, the build number under **Build History** of the job in the Jenkins dashboard |
| 2 | Click **Console Output**  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Console.png  This action will retrieve build results and display them in the console:  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CO-Build.png |

1. 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 | 1. Type the job name as ‘CodeAnalysisonSonar’ in the **Enter an item name** field and select **Freestyle Project** as the project type as shown below: 2. Click **OK** to create job   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Code.png  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:   1. In the **Project Configuration page**, under the **General** tab, type the details in the **Description** field   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Codeanalysis.png |

|  |  |
| --- | --- |
| 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 | 1. Specify **CompileandPackage** as **Project Name** (i.e., Project from which we should copy the artifacts) 2. Select **Copy from WORKSPACE of the latest completed build** from the drop-down list   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Artifact.png |

1. Perform code analysis

|  |  |
| --- | --- |
| 1 | To perform code analysis by using SonarQube Scanner, follow the steps below:   1. Navigate to the **Build** section 2. Choose **Execute SonarQube Scanner** from the **Add build step** drop-down menu 3. Provide the details for **Analysis Properties** as shown below   **Note:** Copy the properties from the **sonar-project** file present in the **ProjectSourceCode\Module3.**  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\SOnar.png  Now, the configuration set up is complete to perform code analysis.   1. Click the **Save** button in the Project Configuration page |
| 2 | After the configurations are complete, execute analysis manually as follows:   * 1. Navigate to the Jenkins dashboard and select **CodeAnalysisonSonar** project   2. Click **Build Now** to execute code analysis   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\test.png |

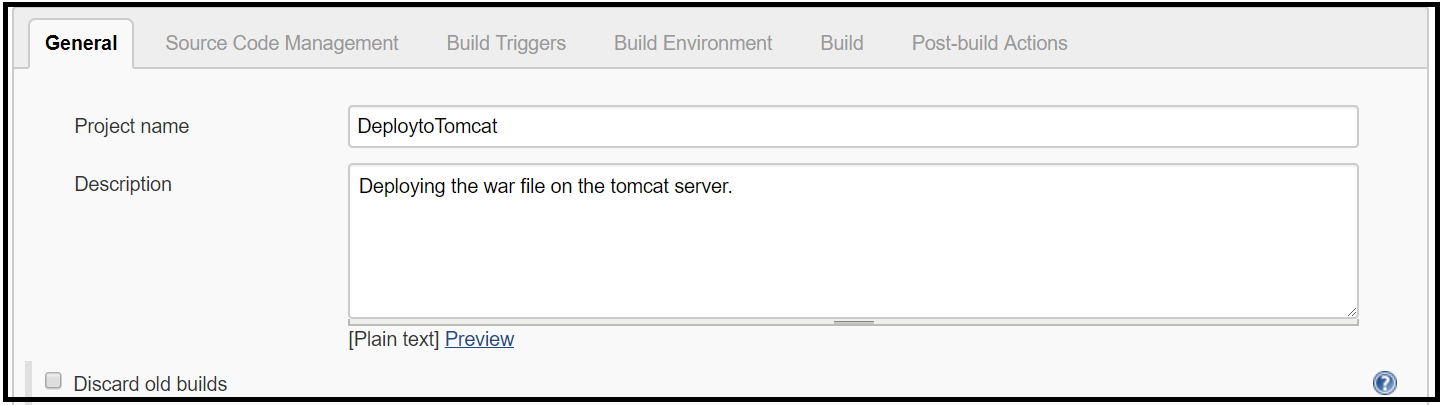
1. View analysis results

|  |  |
| --- | --- |
| 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**  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\test1.png  This action will retrieve results in console.  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CO-CA.PNG   * Access the above link in the console output to see the analysis results   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Sonarqube.png |

1. 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.



## 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 | 1. Specify **CompileandPackage** as **Project Name** (i.e., Project from which we should copy the artifacts) 2. Select **Copy from WORKSPACE of the latest completed build** from the drop-down list   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Artifact.png |

1. Perform post build tasks to deploy the project

|  |  |
| --- | --- |
| 1 | In the Project Configuration page, navigate to the **Post-build Actions** section |
| 2 | Configure Tomcat to deploy project   1. From the **Add Post-build action** drop-down list, choose **Deploy war/ear to a container** 2. Specify **WAR/EAR files** as **\*\*/\*.war** 3. Specify **Context Path** as petclinic.war 4. Click **Add Container** drop-down menu and choose **Tomcat 7.x** and provide the details as shown below 5. Click **Add** to add credentials of Tomcat and select it from the drop-down list   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Deploy-pA.PNG   1. Click **Save** to save the configurations |

1. Execute project deployment

|  |  |
| --- | --- |
| 1 | Now that the deployment configurations are complete, execute deployment job manually by following the below steps:   1. In the Jenkins dashboard, navigate to the **DeploytoTomcat** project 2. Schedule the deployment to be executed immediately by clicking **Build Now**   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Deploy1.png |

1. 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**  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Deploy2.png  This action will retrieve results in console. |
| 3 | To verify deployment, perform the following tasks:   1. Go to browser and access the link <http://localhost:8080> to see the Tomcat home page 2. Click the **Manager App** icon to see the deployed war file (i.e., petclinic.war)      1. Click the **petclinic.war** file to check the output   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\OP.PNG  You can see that the deployed application is up and running  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Pet.png |

End of Module 5

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

|  |  |
| --- | --- |
| 1 | Select **New Item** in the Jenkins dashboard from the menu as highlighted in the below image. |
| 3 | 1. 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: 2. Click **OK** to create job     This action leads to the Project Configuration page, where you can configure settings for testing activity |
| 4 | To configure job with GitHub project, perform the following tasks:   1. In the **Project Configuration** page, under the **General** tab, type the details in the **Description** field |

1. Configure Source Code Management with Git

|  |  |
| --- | --- |
| 4 | To configure SCM with GitHub project, perform the following tasks:   1. In the **Project Configuration page**, under the **Source Code Management** section:    1. Select **Git**    2. Specify Repository URL    3. Select the credentials from dropdown    4. Click **Save** to save the configuration   C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Git123.png |

1. Configure build with Maven and test results with TestNG

|  |  |
| --- | --- |
| 1 | To build job using maven commands, do the following steps.   1. Navigate to **Build** section 2. Choose **Invoke top-level Maven targets** from **Add build step** drop-down list. 3. Specify the Maven version and type target name as shown below to execute clean and package goals in Maven |
| 2 | 1. In the Project Configuration page, navigate to the Post-build Actions section 2. From the Add post-build action drop-down items, select Publish TestNG Results 3. Specify the XML report pattern as shown, to save results during build execution |
| 3 | Once configurations are completed, execute build job manually by following the below steps:   1. Click **Save** 2. Schedule the build to be executed immediately by clicking the **Build Now** menu item |

1. View results using TestNG

|  |  |
| --- | --- |
| 1 | View the generated test results by following the below steps:   1. Select the execute build number 2. Click **TestNG Results** link     The test TestNG results are as shown below:  C:\Users\vishnu.k.kallimakula\AppData\Local\Microsoft\Windows\INetCache\Content.Word\TEstng.png |