JENKINS & SONAR

LAB ASSIGNMENTS

Pre-Requisitions:

1. Create one account with GitHub (https://github.com/)
2. JDK 1.6 or higher
3. Download Jenkins.war file. You can use below links to download Jenkins.
    a. https://jenkins.io/download/
4. Install Maven in your Machine. Please refer below link for to download Maven.

    a. http://maven.apache.org/download.cgi

5. Install Gradle in your Machine. Please refer below link to download Gradle, use any one of the link to download gradle.
    a. https://services.gradle.org/distributions
    b. https://gradle.org/gradle-download/

6. Download SonarQube from the below link
    a. http://www.sonarqube.org/downloads/

Lab 1:

Create one java application with supporting test cases; take Maven as your build script. Use Jenkins to build the maven project. Distribute the project with Jenkins from GitHub Repository.

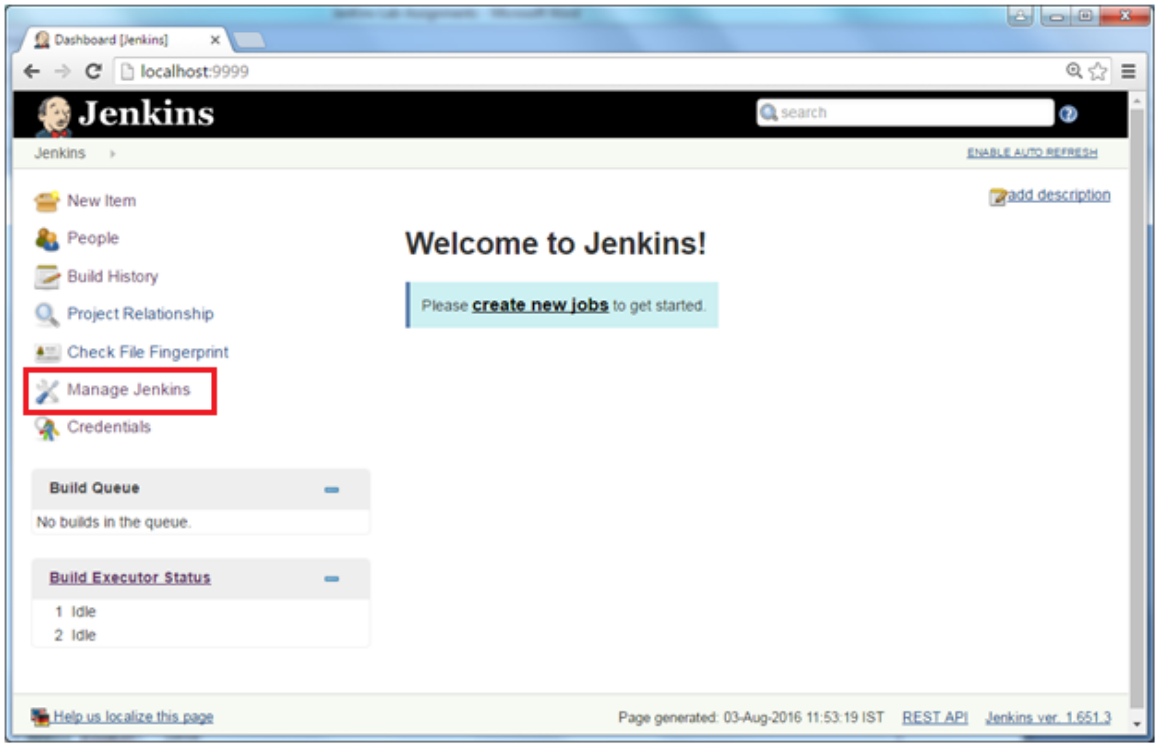
Steps:

1. Open GitHub URL (https://github.com/caprepo/softwareRepo)
2. Click **BankApp.zip folder,** next page click download link. The zip file will be downloaded.
3. Unzip the folder
4. Go to Eclipse IDE, choose import to import the Project into Workspace
5. Under select root directory click browse button.
6. Go to the unzip folder location and choose the folder in the name of Day1-BankApp (root directory of your project). Click next and Finish.
7. Now you got one Java Application.
8. Distribute the application in GitHub repository. (In case if you face any challenges, you can refer our GitHub Lab to check-in the project).
9. Once you check-in the project with you remote URL in GitHub, you could see the application in GitHub URL.
10. Now start your Jenkins server. Before you start Jenkins server JDK PATH and CLASSPATH need to be defined.
11. Open you command prompt, check the JDK path. If JDK path has been defined properly. Execute the below command to start **JENKINS** server.

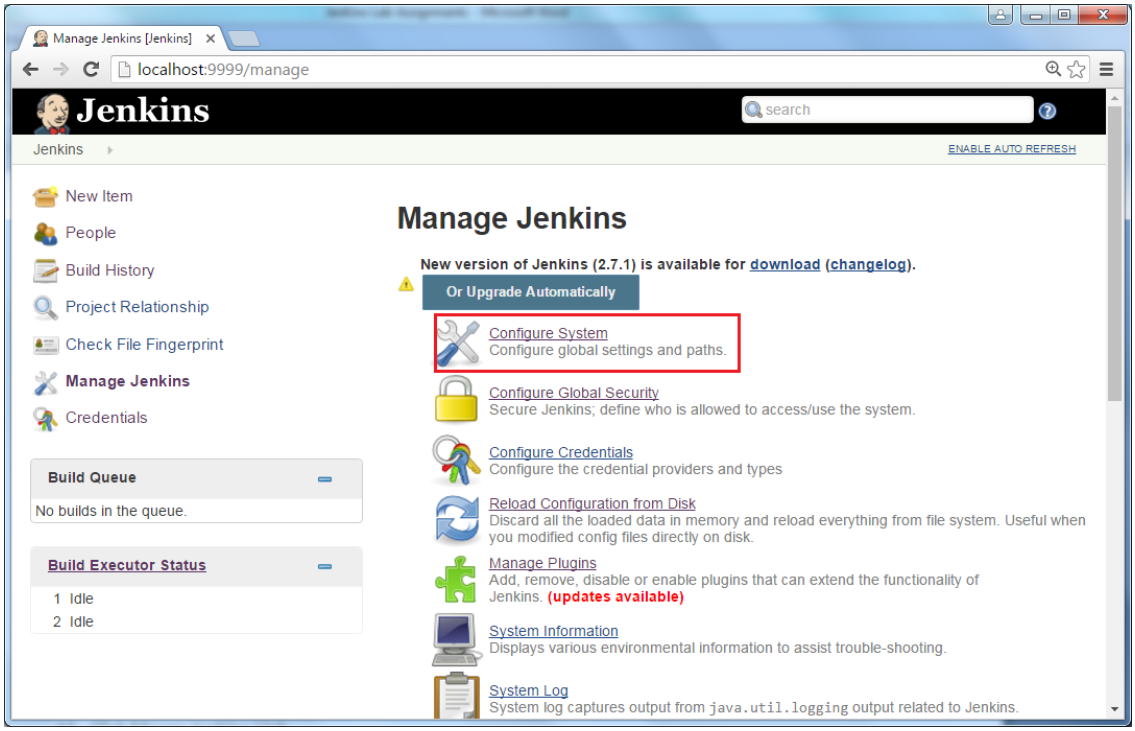
    java –jar jenkins.war

    a. The default port taken by Jenkins is 8080. In case if you get any error related to port, start the jenkin with appropriate port number.

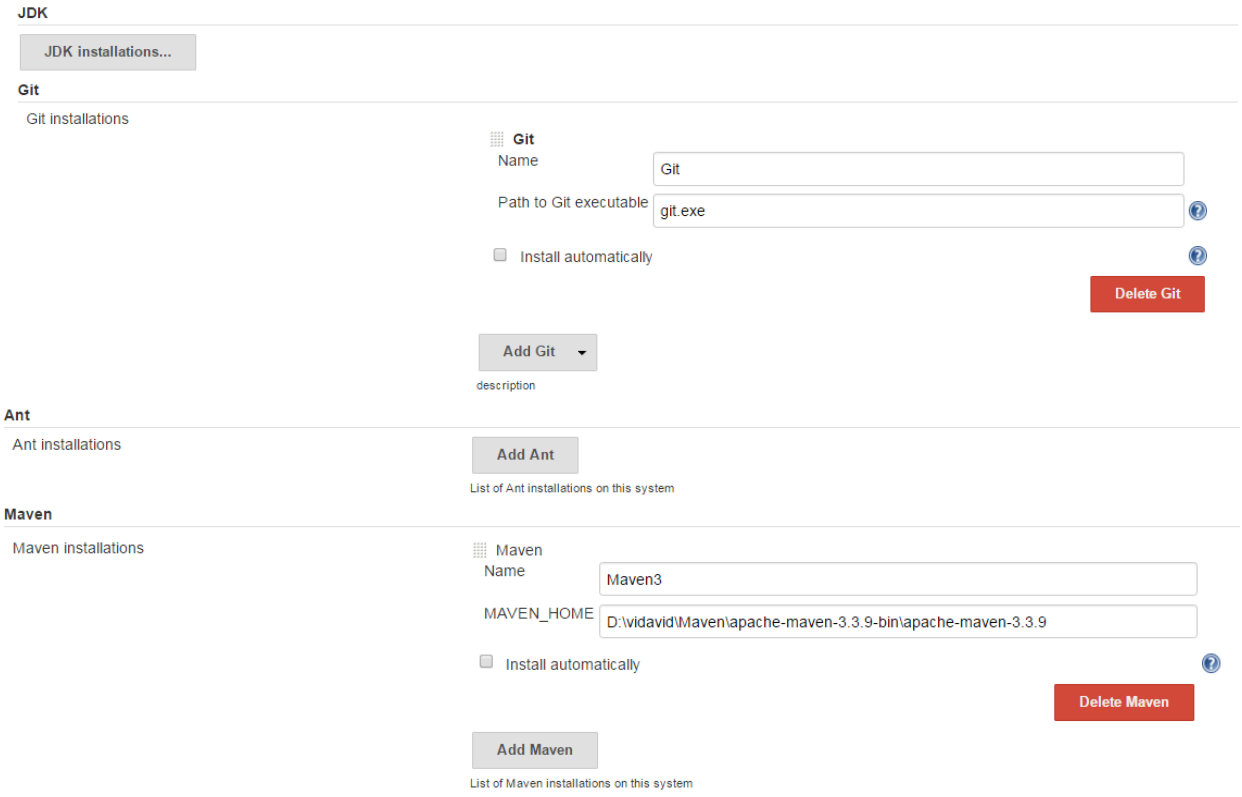
    java –jar jenkins.war --httpPort = 9999
12. If Jenkins server started successfully, you can see "**Jenkins is fully up and running"** command in the command prompt.
13. Now open browser enter the below URL
    a. http://localhost:8080 (by default) (or) http://localhost:9999 (if you change port)
14. You can see the below screen



15. Click Mange Jenkins Link, you will be redirected to the below screen

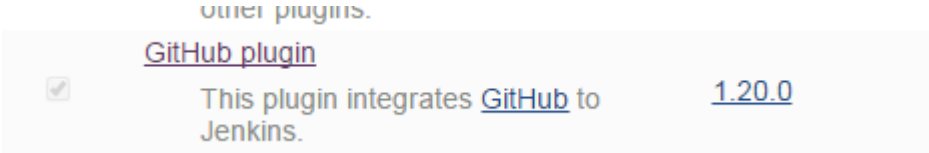


16. Click Configure System link, it will give the Jenkins configuration window.
17. In this window, you will be getting a different titles in the name of JDK, GIT, Maven , etc. That was displayed in the below screen shot.



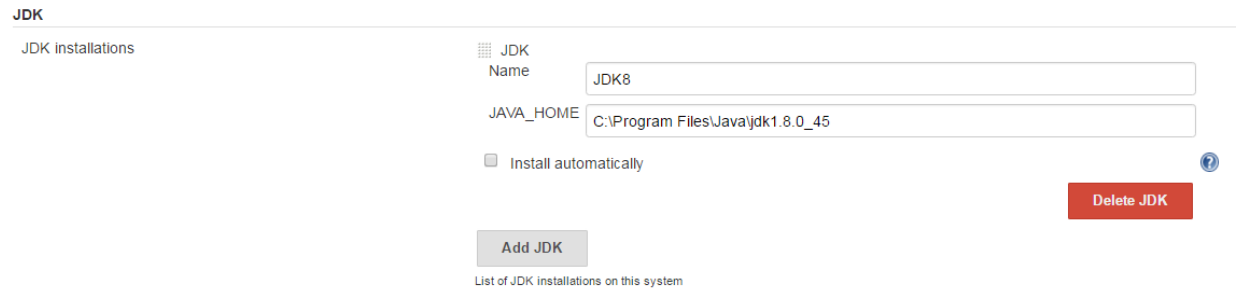
18. If you have not seen GIT as one of the title here. Click **Manage Jenkins** link. And then click **Manage Plugins** as highlighted below.

19. Under Available tab, search GitHub Plugin, it wil list GitHub is one of the plug in as below



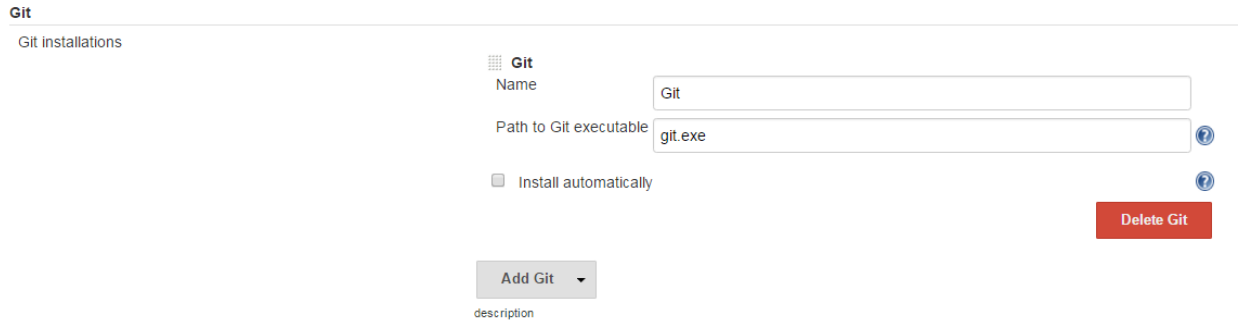
20. Select the plug-in and click **install without restart**, if you don't want to restart the Jenkins.
21. Follow the same to install the other plug-ins also.
22. And then click **Back to DashBoard → Manage Jenkins → Configure System**
23. Specify JDK Home under JDK. Please refer the below screen shot

JDK:

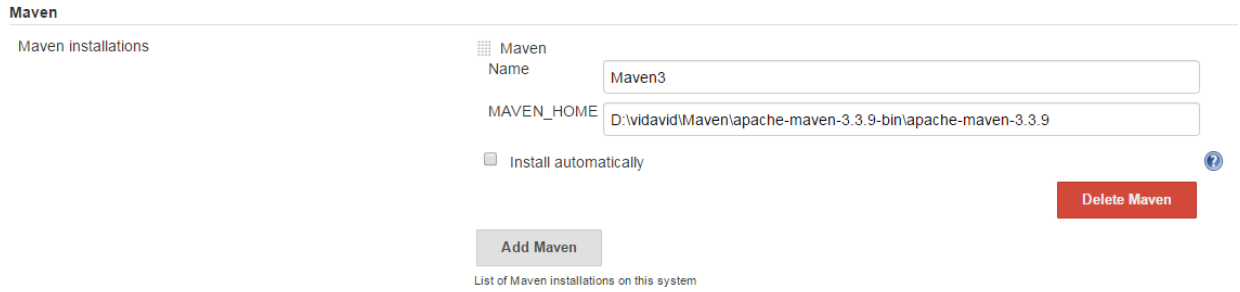


24. Do the followings for GIT and Maven too.

GIT:

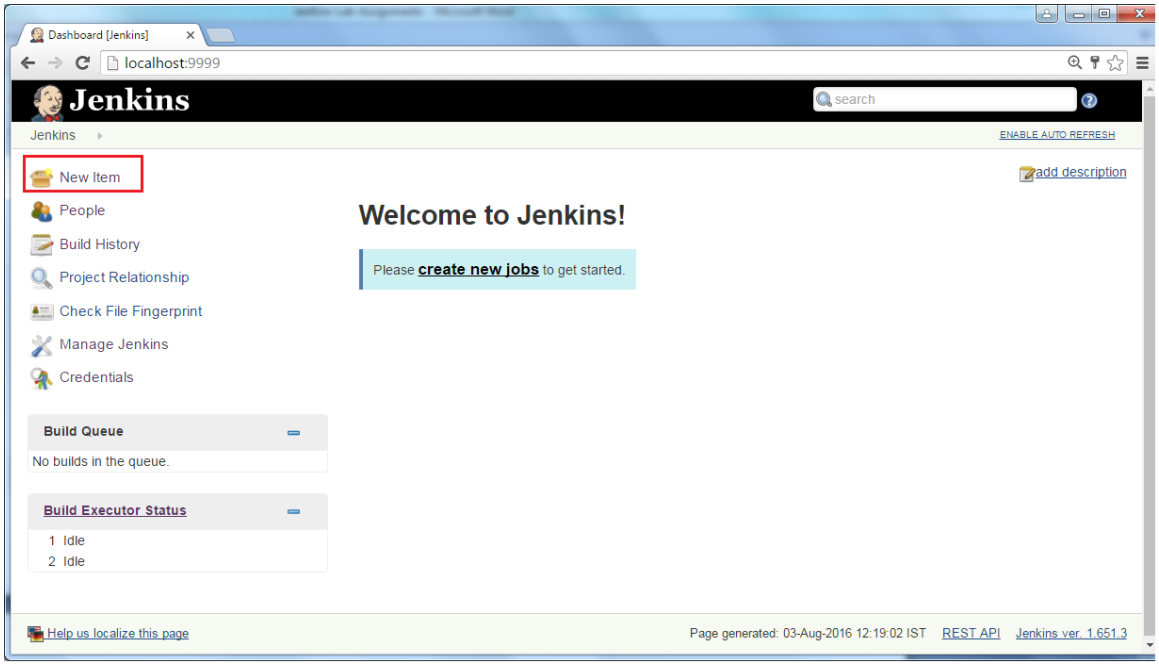


Maven:

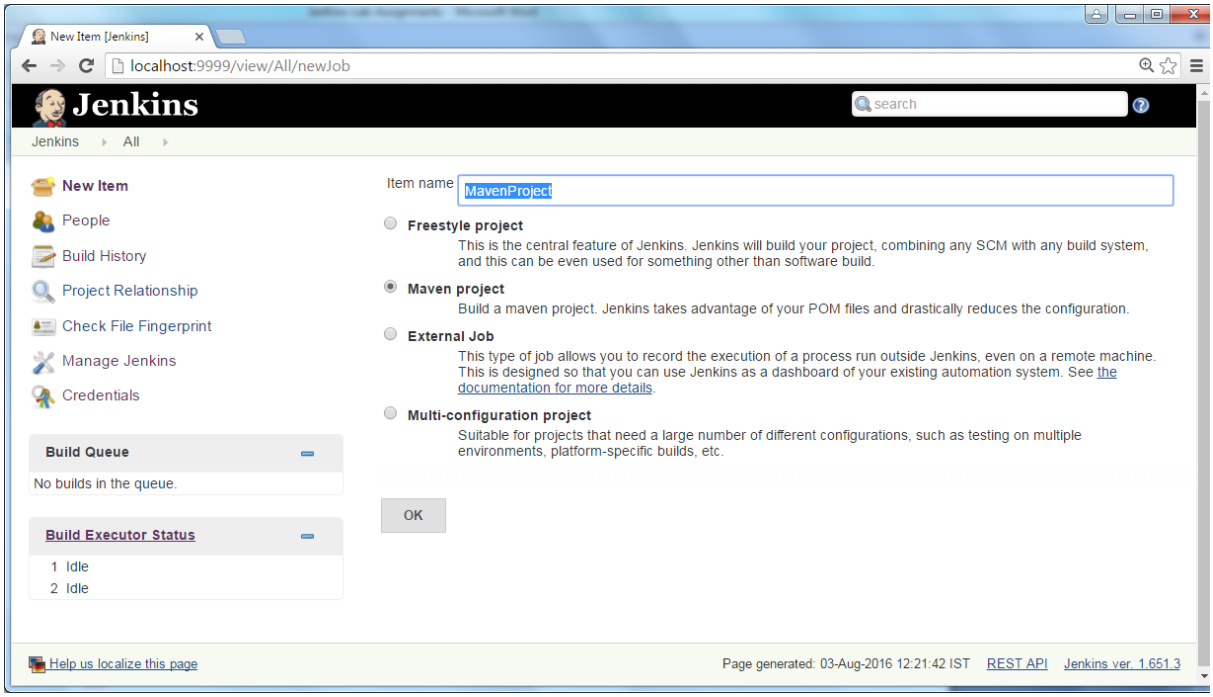


25. Click **Apply → Save** to save the configurations in Jenkins.

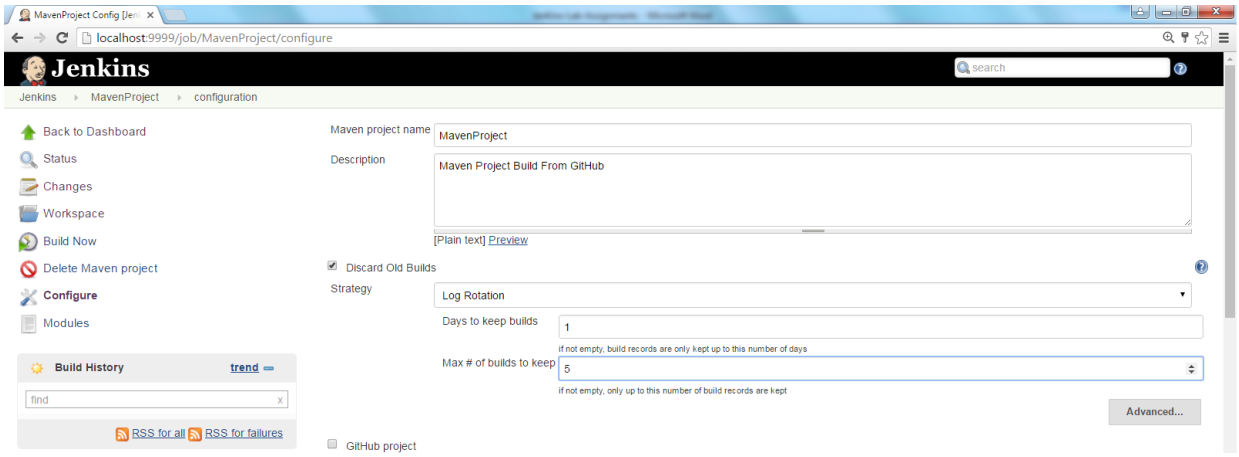
26. Click **New Items** Link as showed below.



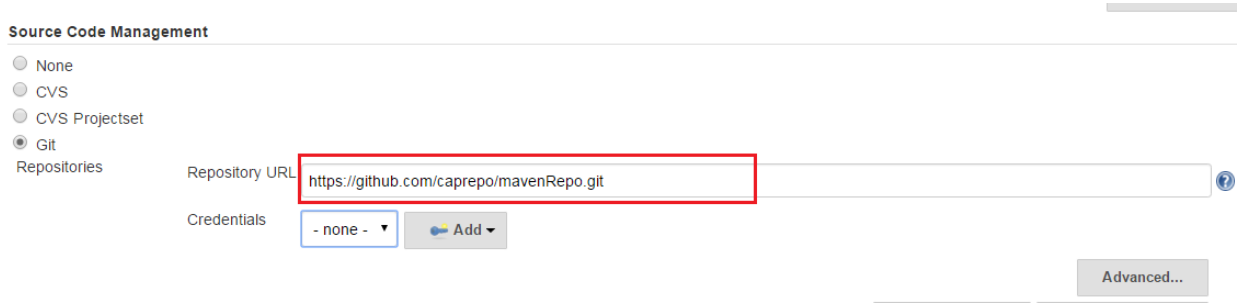
27. In **Item Name** text box mention your job Name example "**MavenProject**" and then select **maven Project** and then click **OK**.



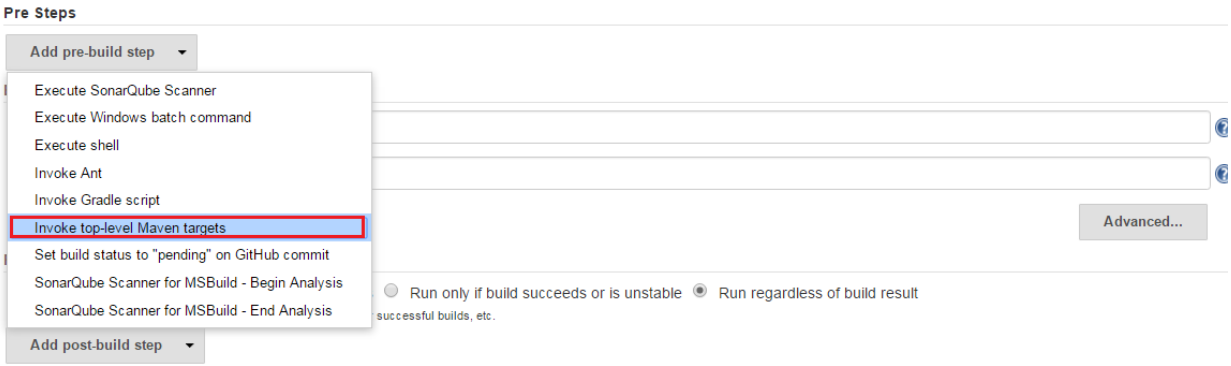
28. Mention project description and select Discard Old Build. Look at the below screen shot for further details



29. Under **Source Management** select **Git** and paste the URL where your application has been uploaded.

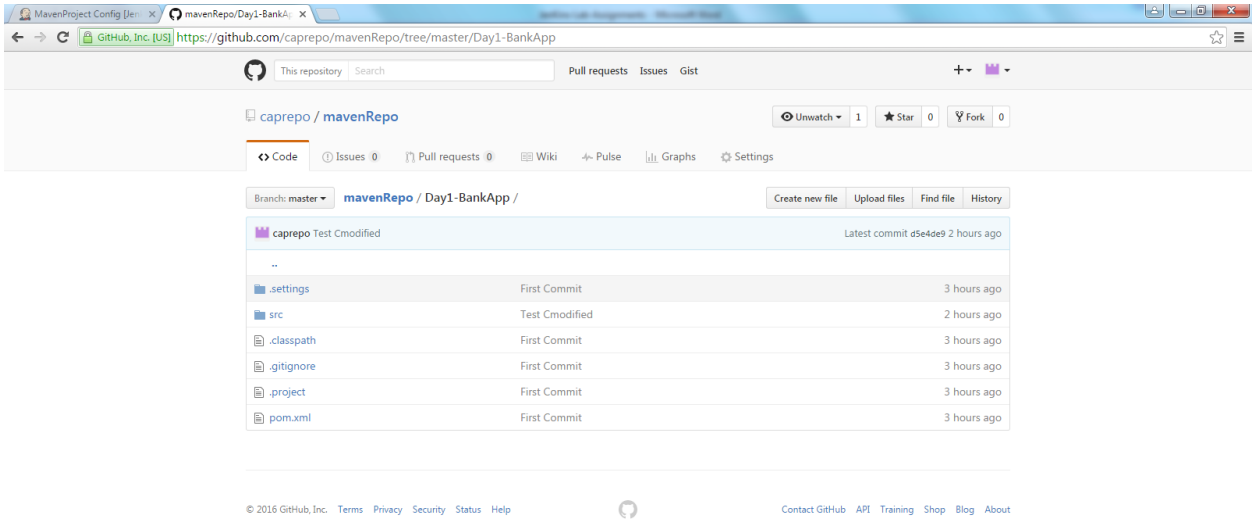


30. Under **Pre steps** click **Add pre-build step**→ choose **invoke top-level Maven targets**.

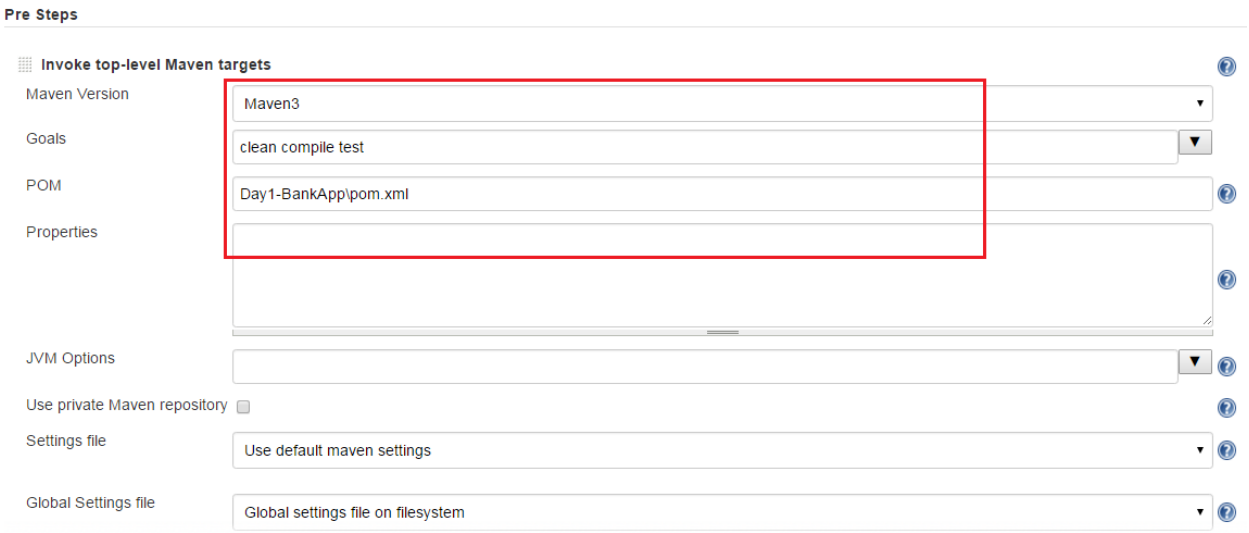


31. Now enter maven version and Goal details

32. Open git repository, check your project structure. As example if your project structure is mentioned below means:

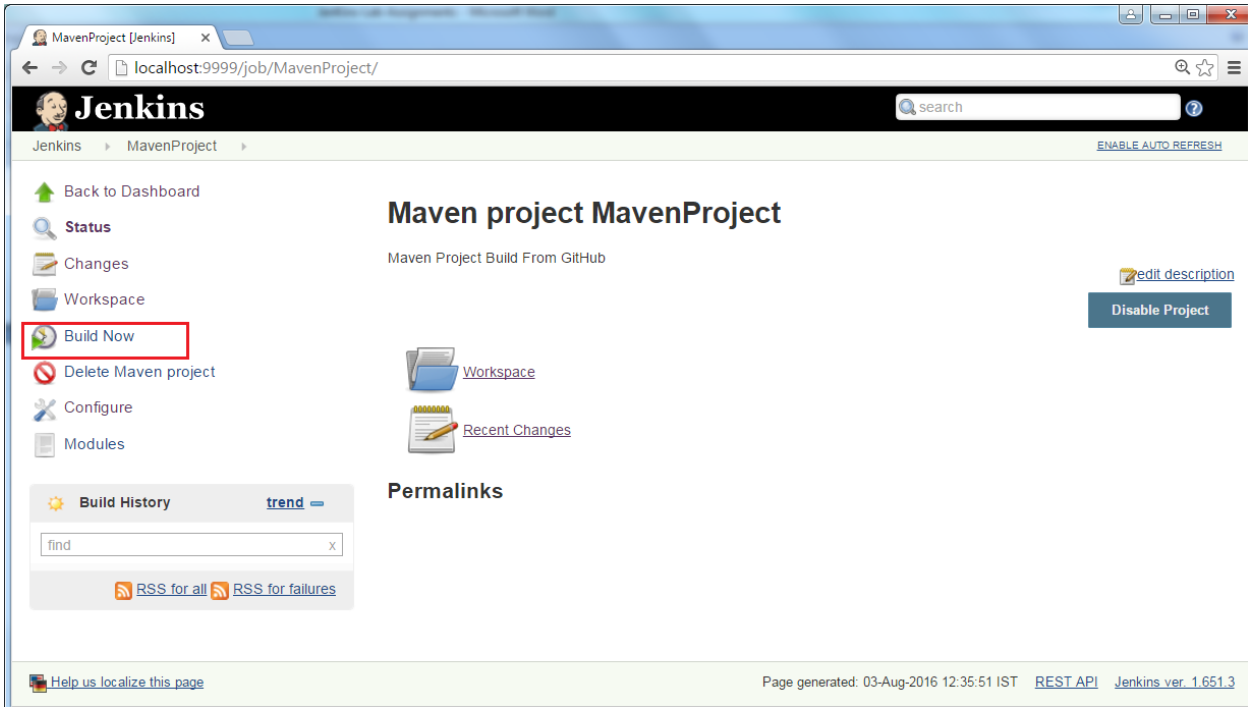


33. Under Pre Steps click Advanced Button, and mention the below configurations.

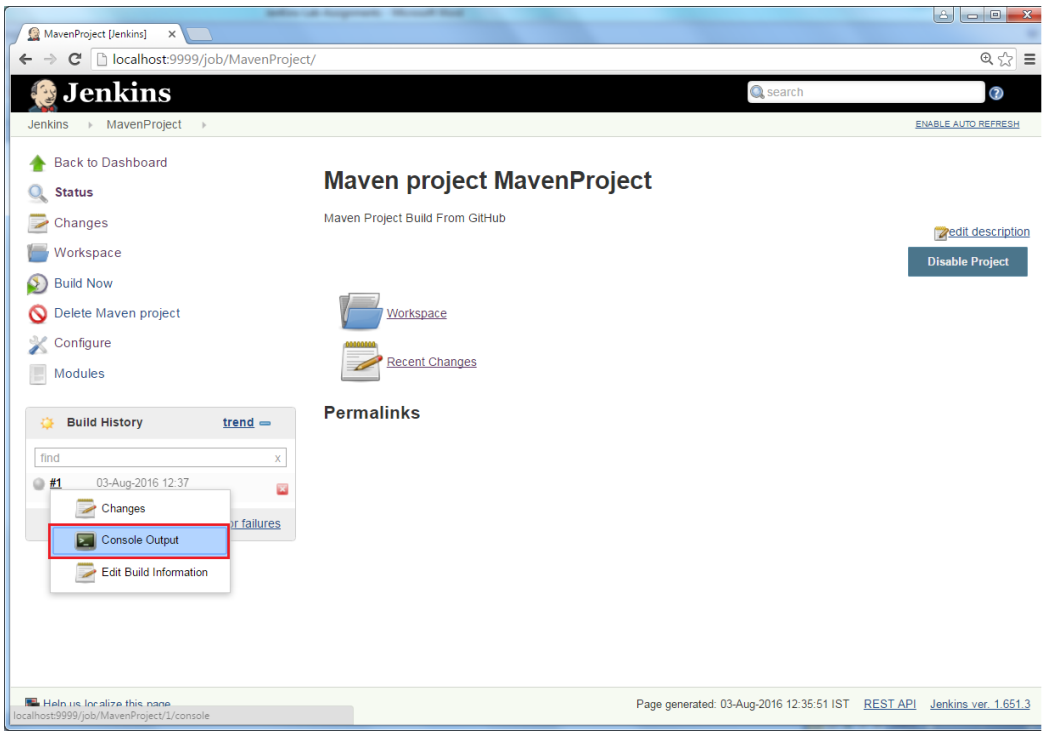


34. Under **Build,** Click **Advanced** and then check the option called **Resolve Dependencies during Pom parsing.**
35. Now click Apply → Save to save your project configurations.

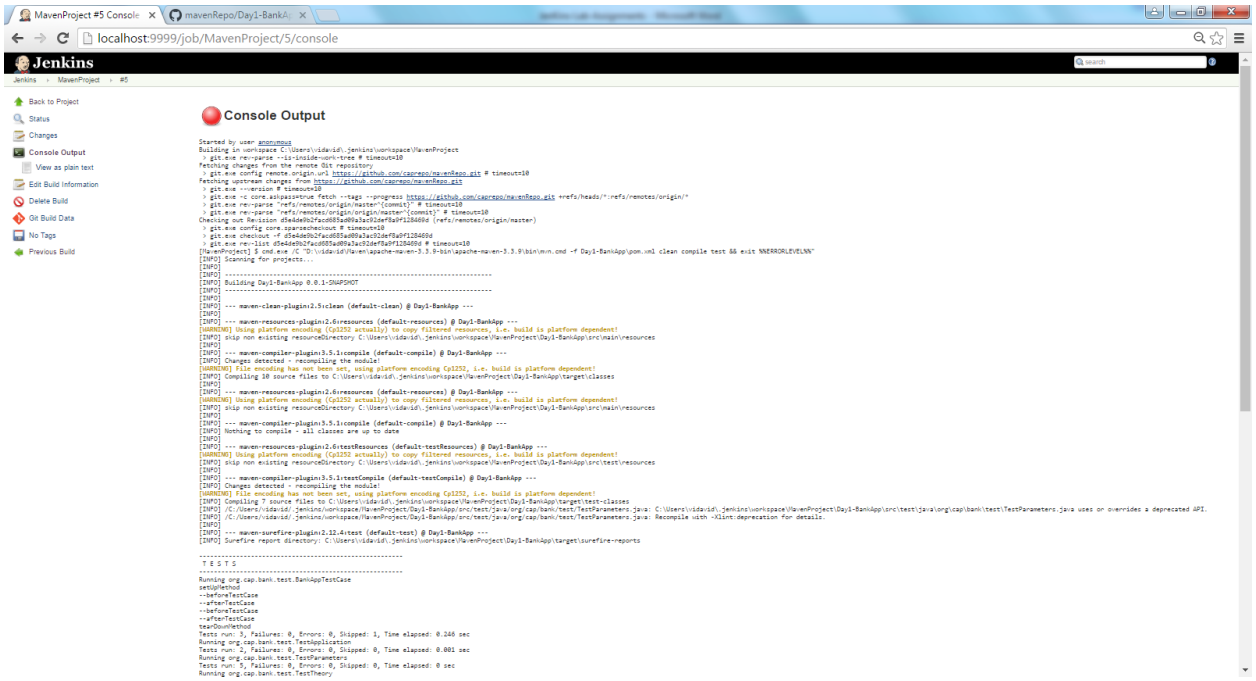
36. Click **Build Now**



37. Under Build History you can see the build count. Click the count and select console output.



38. In the console output, you can see the complete report contains clean, compile and test case execution status.



39. In case any one of the test case fails, the build will fail.
40. Do the changes in the project and commit your project with GIT again. And then build your application again. This process should be repeated until you get the success build.
41. For success build blue color balloon will be generated in Jenkins, Failure case Red color Balloon.

Conclusion:

From the above example, we learnt how continuously integrate maven project with Jenkins Server. We understand the analyses reports generated by Jenkins.

Lab 2:

Build one Gradle project in Jenkins. Take your application from GitHub repository which should be uploaded before. Once Jenkins builds the application analyze the complete report generated by Jenkins.

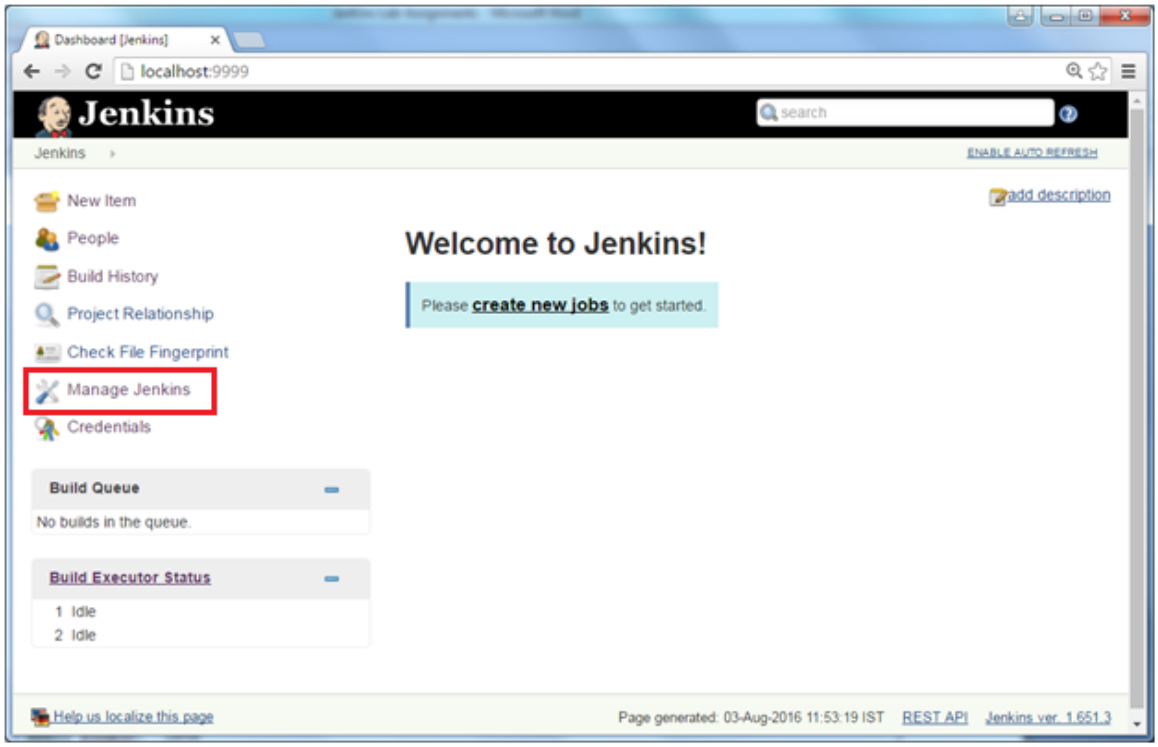
Steps:

1. Open GitHub URL (https://github.com/caprepo/softwareRepo)
2. Click **BankApp_Gradle.zip folder,** next page click download link. The zip file will be downloaded.
3. Unzip the folder
4. Go to Eclipse IDE, choose import ->Gradle -> Gradle Project into Workspace
5. Under select root directory click browse button.
6. Go to the unzip folder location and choose the folder in the name of BankApp_Gradle (root directory of your project). Click next.
7. Select local installation directory, click browse button to select Gradle HOME directory.
8. Click Next →Finish
9. Now the BankApp_Gradle Application will be imported in your local machine.
10. Open you command prompt, check the JDK path. If JDK path has been defined properly. Execute the below command to start **JENKINS** server.

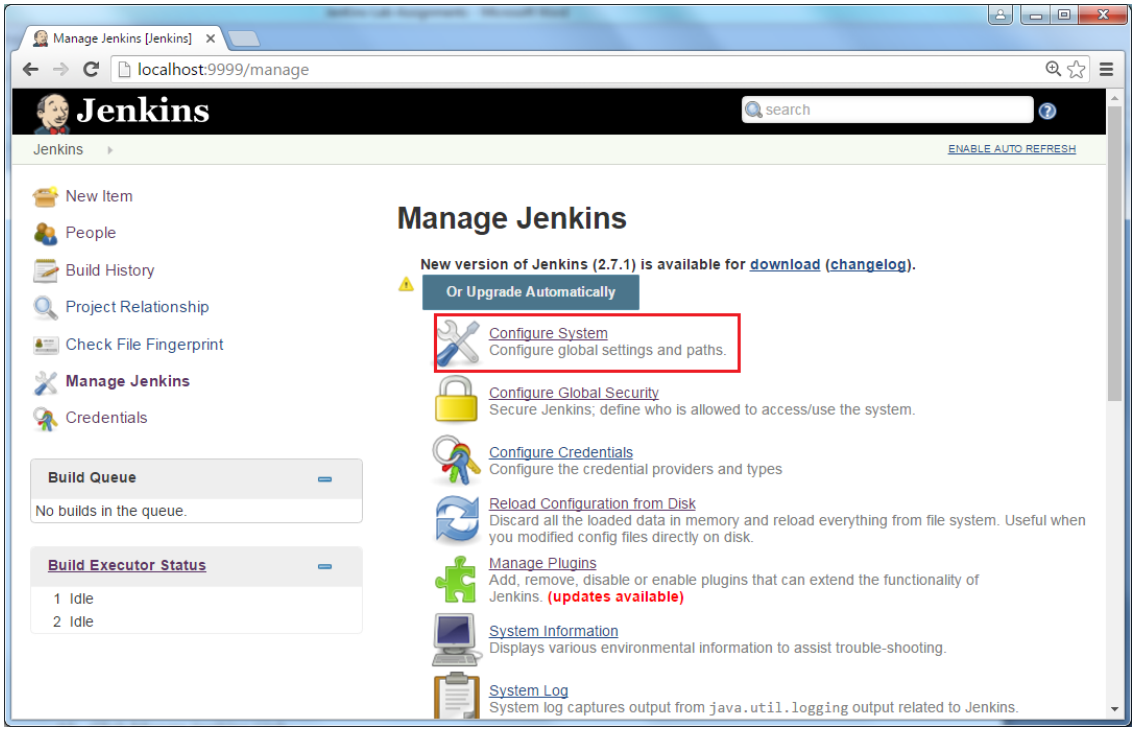
    java –jar jenkins.war

    a. The default port taken by Jenkins is 8080. In case if you get any error related to port, start the jenkin with appropriate port number.

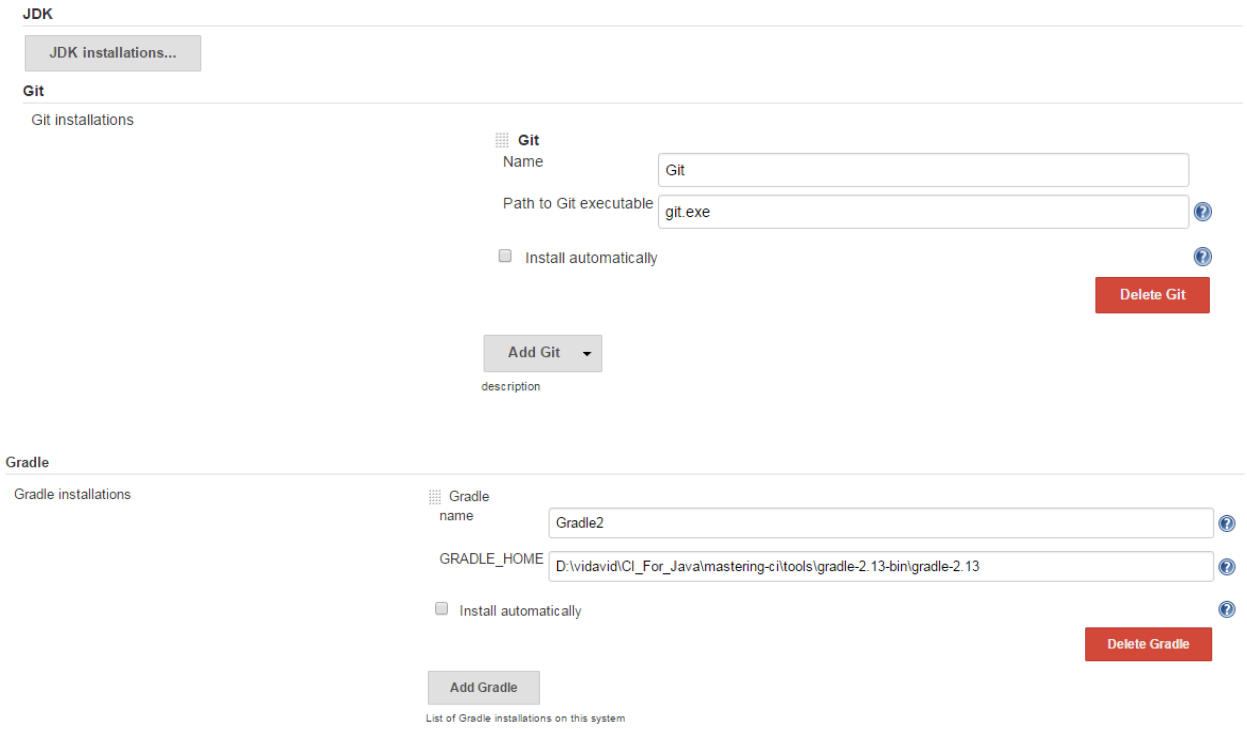
    java –jar jenkins.war --httpPort = 9999
11. If Jenkins server started successfully, you can see "**Jenkins is fully up and running"** command in the command prompt.
12. Now open browser enter the below URL
    a. http://localhost:8080 (by default) (or) http://localhost:9999 (if you change port)
13. You can see the below screen



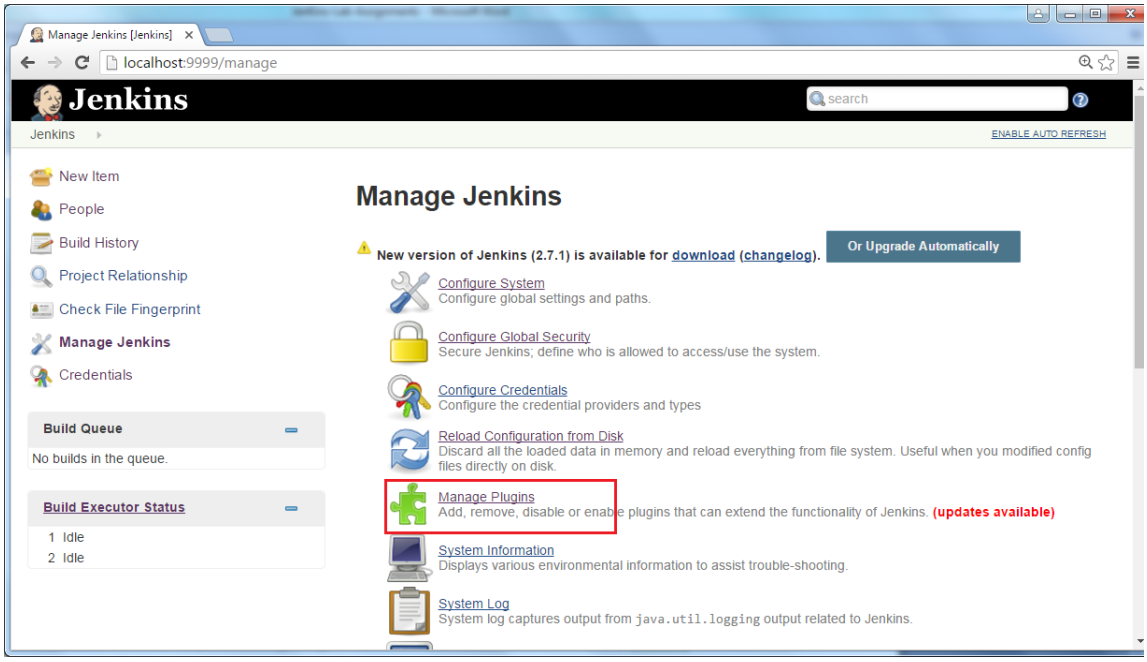
14. Click Mange Jenkins Link, you will be redirected to the below screen



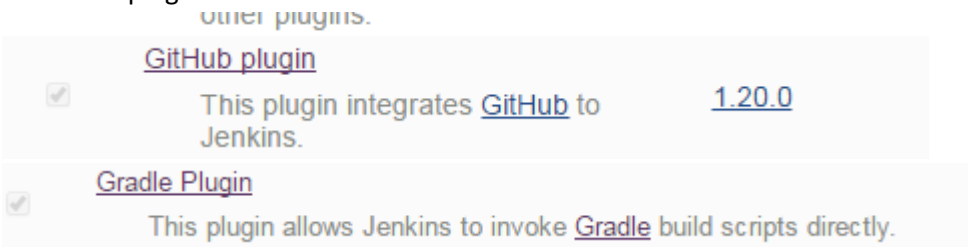
15. Click Configure System link, it will give the Jenkins configuration window.
16. In this window, you will be getting a different titles in the name of JDK, GIT, Gradle , etc. That was displayed in the below screen shot.



17. If you have not seen GIT as one of the title here. Click **Manage Jenkins** link. And then click **Manage Plug-ins** as highlighted below.

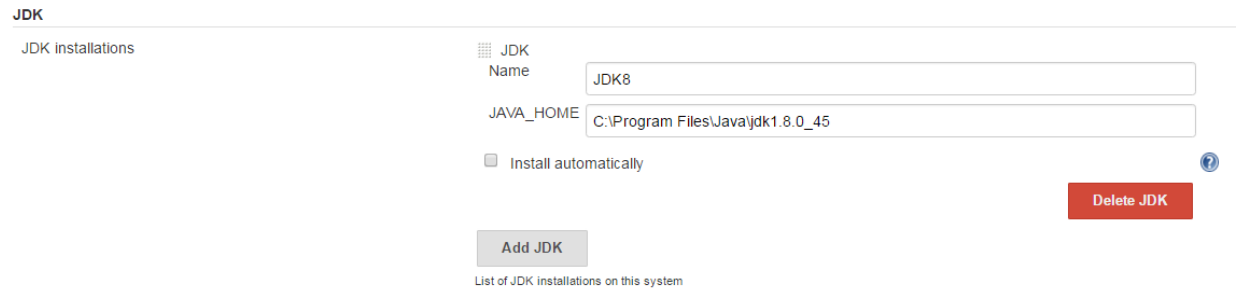


18. Under Available tab, search GitHub Plug-in and Gradle Plug-in, it will list GitHub and Gradle as one of the plug in shown below



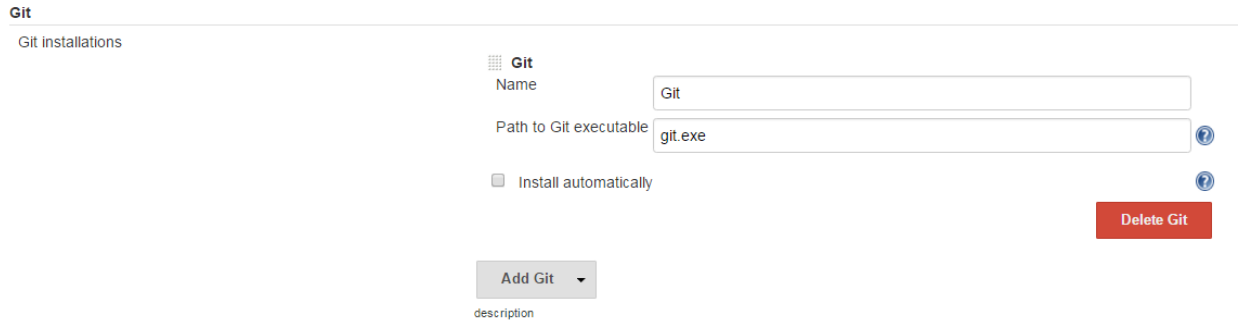
19. Select the plug-in and click **install without restart**, if you don't want to restart the Jenkins.
20. Follow the same to install the other plug-ins if it does not available.
21. And then click **Back to DashBoard → Manage Jenkins → Configure System**
22. Specify JDK Home under JDK. Please refer the below screen shot

JDK:

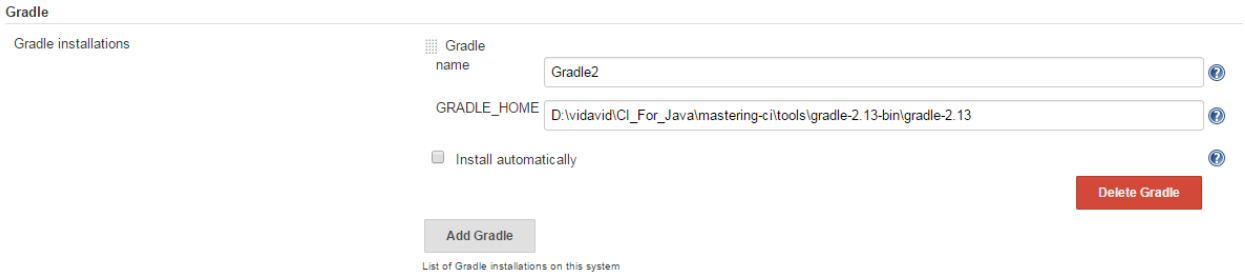


23. Do the followings for GIT and Maven too.

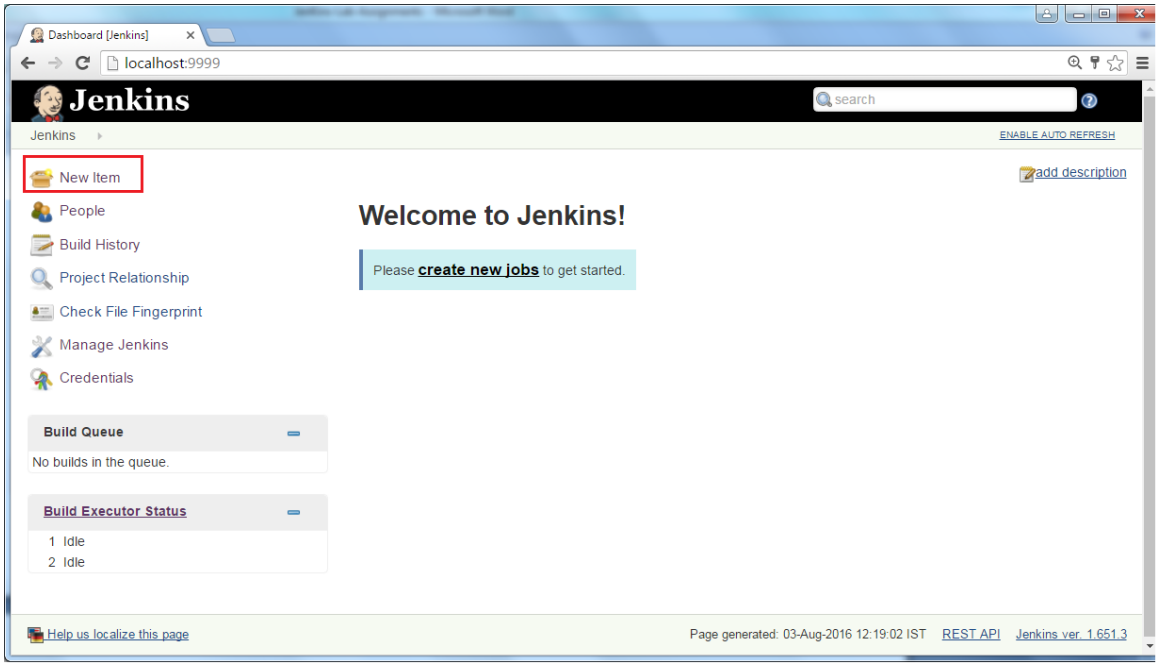
GIT:



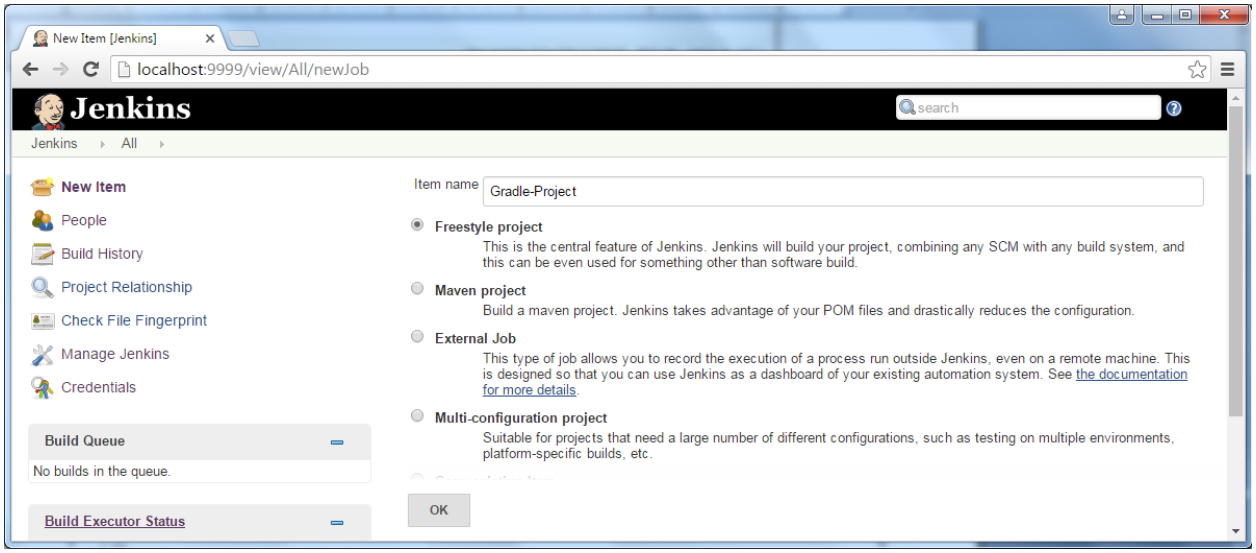
Gradle:



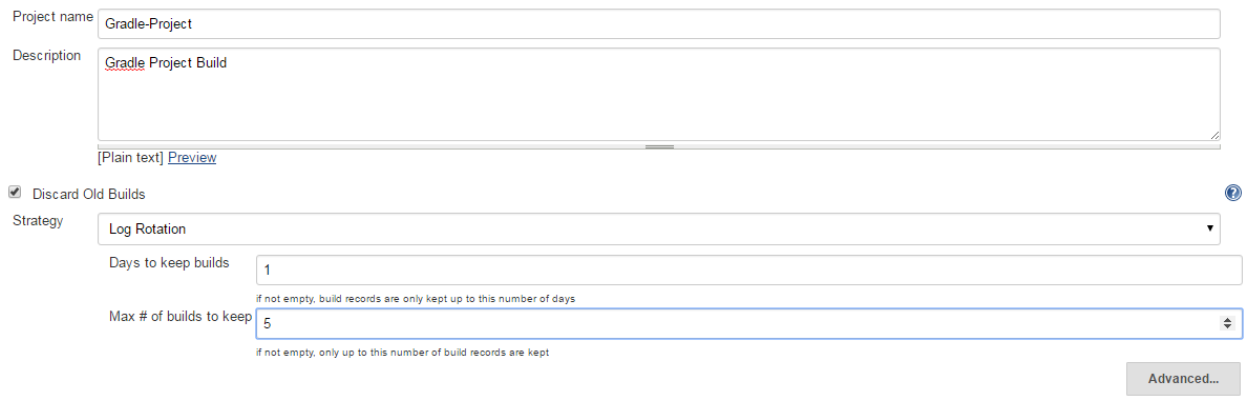
24. Click **Apply → Save** to save the configurations in Jenkins.
25. Click **New Items** Link as showed below.



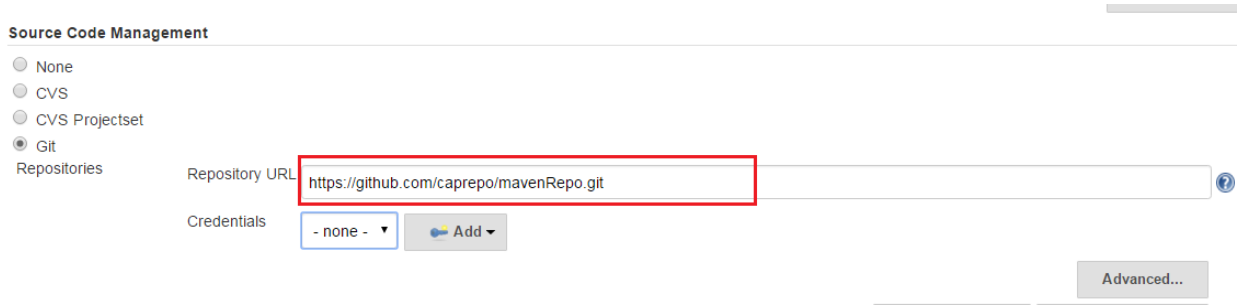
26. In **Item Name** text box mention your job Name example "**Gradle-Project**" and then select **Freestyle Project** and then click **OK**.



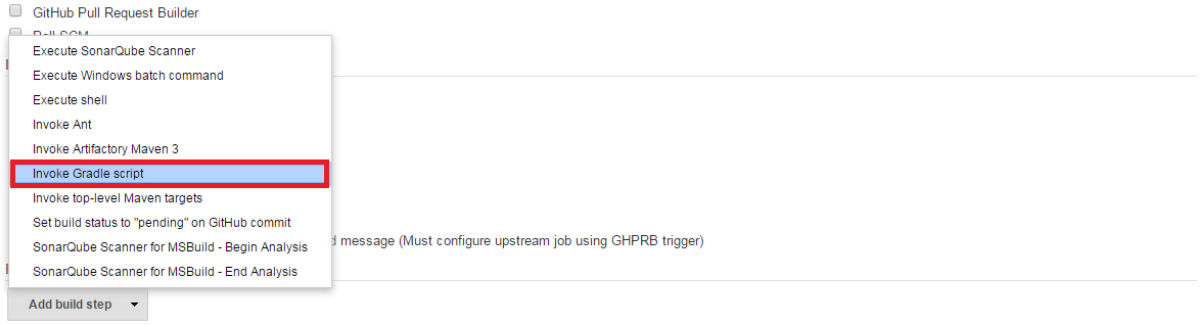
27. Mention project description and select Discard Old Build. Look at the below screen shot for further details



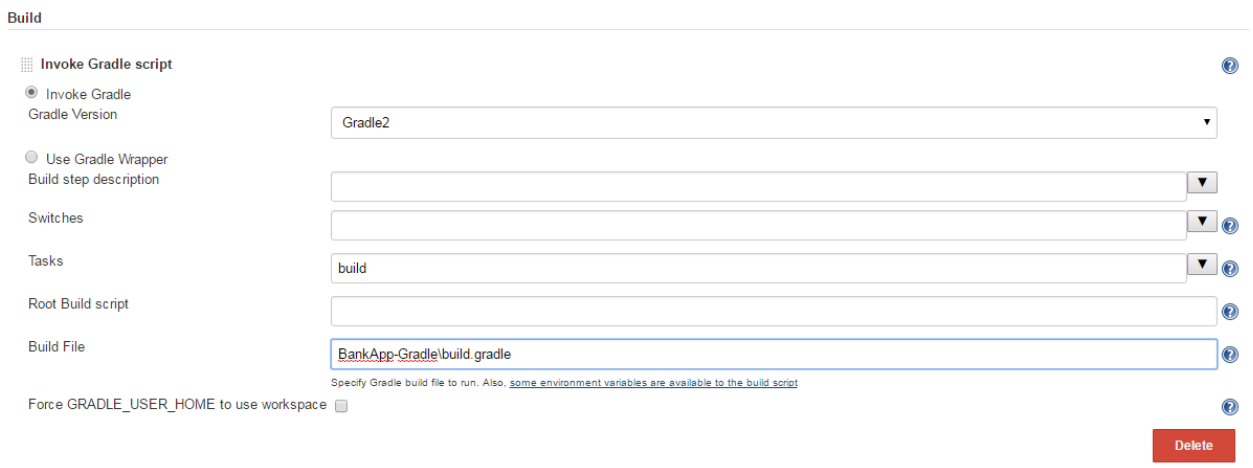
28. Under **Source Management** select **Git** and paste the URL where your application has been uploaded.



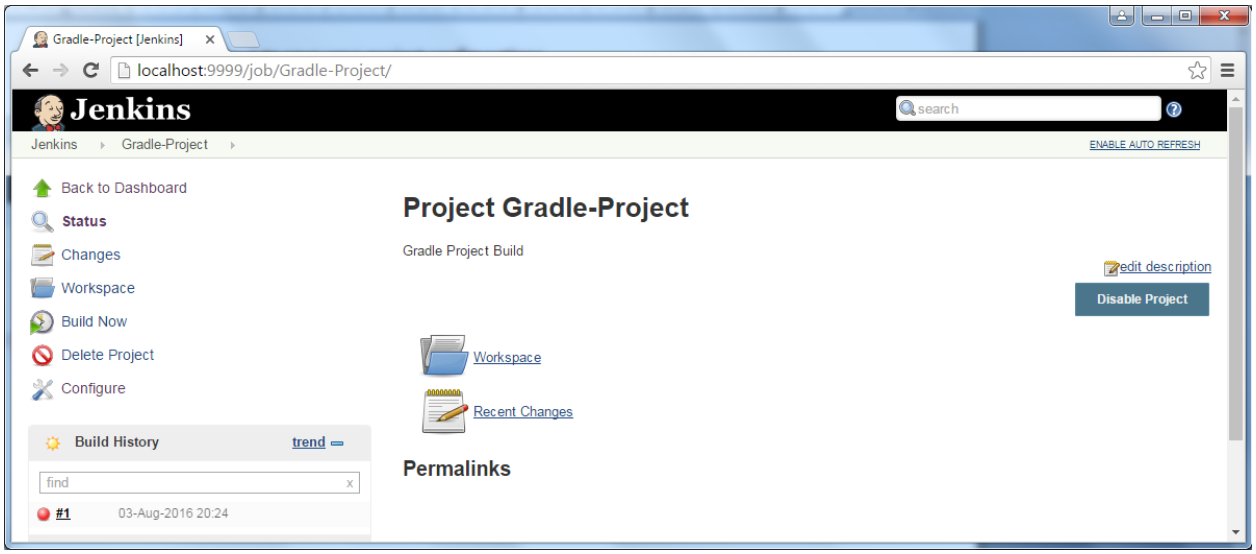
29. Under **Pre steps** click **Add pre-build step**→ choose **invoke Gradle Script**.



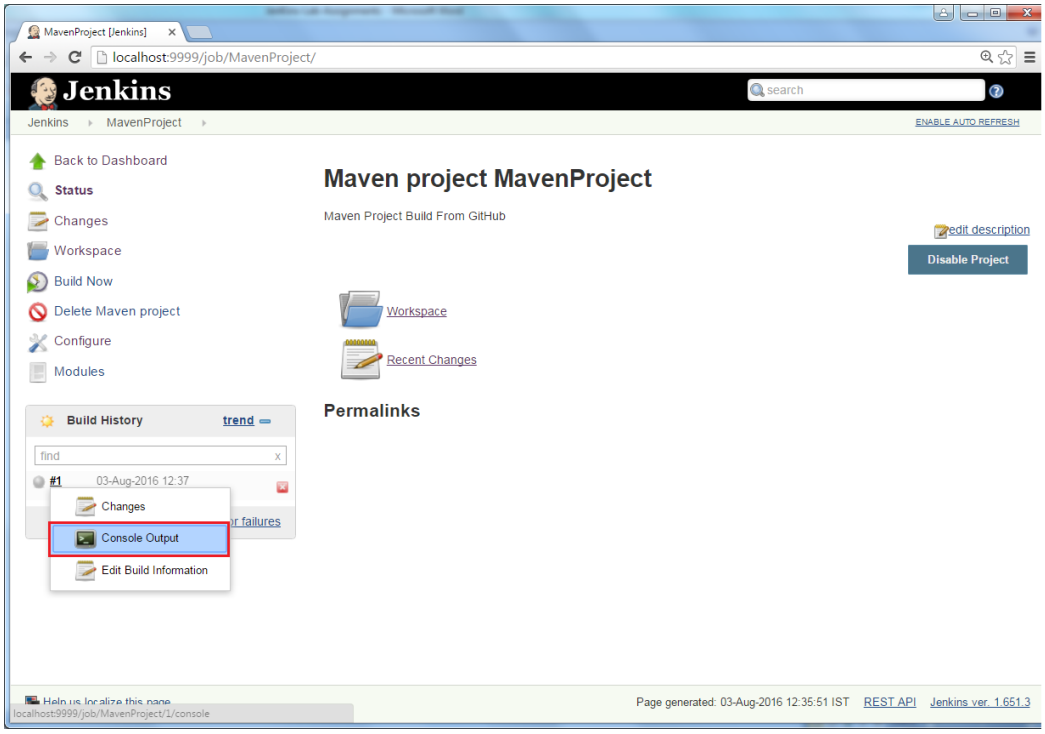
30. Now under **Build**, enter **Gradle version** and **Task** details, mention where is your **build.gradle file** under GitHub project repository (Usually its under your project name, so enter projectname/build.gradle)



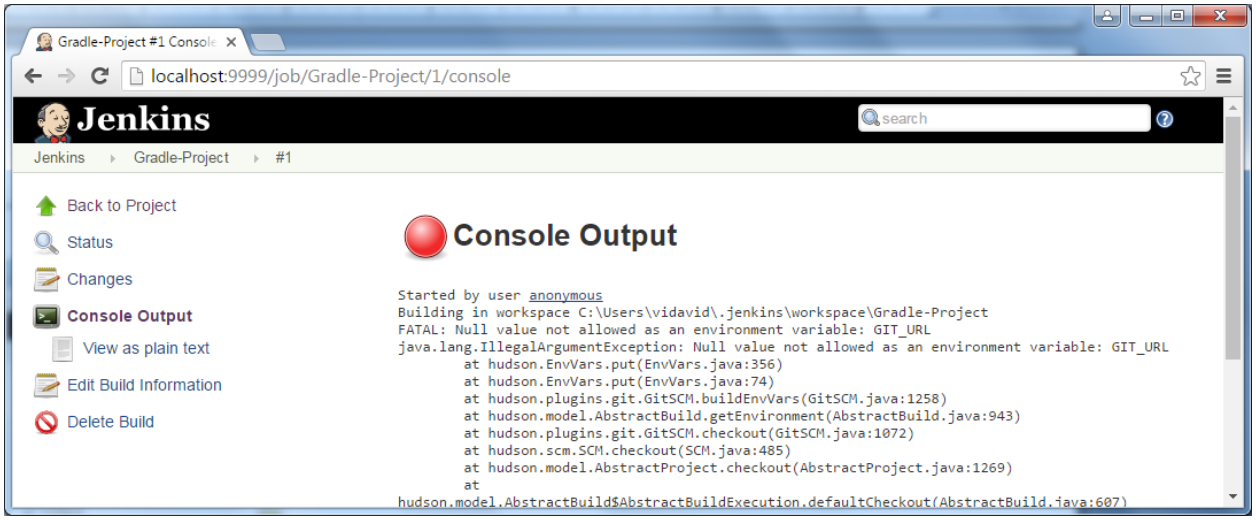
31. Now click Apply → Save to save your project configurations.
32. Click **Build Now**



33. Under Build History you can see the build count. Click the count and select console output.



34. In the console output, you can see the complete report for build task.



35. In case any one of the test case fails, the build will fail.
36. Do the changes in the project and commit your project with GIT again. And then build your application again. This process should be repeated until you get the success build.
37. For success build blue color balloon will be generated in Jenkins, Failure case Red color Balloon.

Conclusion:

From the above example, we learnt how continuously integrate gradle project with Jenkins Server. We analyses reports generated by Jenkins.

Lab 3:

Configure SonarRunner in your project (use Lab2 for updating). Add SonarRunner Task, call the SonarRunner Task in Jenkins to test your code Quality.

Steps:

1. Open GitHub URL (https://github.com/caprepo/softwareRepo)
2. Click **BankApp_Gradle.zip folder,** next page click download link. The zip file will be downloaded.
3. Unzip the folder
4. Go to Eclipse IDE, choose import ->Gradle -> Gradle Project into Workspace
5. Under select root directory click browse button.
6. Go to the unzip folder location and choose the folder in the name of BankApp_Gradle (root directory of your project). Click next.
7. Select local installation directory, click browse button to select Gradle HOME directory.
8. Click Next →Finish
9. Now the BankApp_Gradle Application will be imported in your local machine.
10. Open **build.gradle** file add the below **SonarRunner** task in your build file.

```
apply plugin: 'sonar-runner'

sonarRunner {

        sonarProperties {

          property "sonar.projectName", "My Project Name"

          property "sonar.projectKey", "org.sonarqube:java-gradle-simple"

        }

       }
```

11. Save your project, and commit the project changes with GIT repository.
12. Open Command prompt locate the directory where SonarQube Setup was unzipped.
13. Locate the bin directory, choose the file which is appropriate for your machine (Example if you are using windows 64 bit, then open **windows-x86-64**)
14. Then enter StartSonar command in your command prompt.
15. It will start SonarQube Server as mentioned below:

```
D:\vidavid\CI_For_Java\mastering-ci\tools\sonarqube-4.5.7\sonarqube-4.5.7\bin\wi
ndows-x86-64>StartSonar
wrapper  | --> Wrapper Started as Console
wrapper  | Launching a JVM...
jvm 1    | Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org
jvm 1    |   Copyright 1999-2006 Tanuki Software, Inc.  All Rights Reserved.
jvm 1    |
jvm 1    |
jvm 1    | WARNING - Unable to load the Wrapper's native library 'wrapper.dll'.
jvm 1    |           The file is located on the path at the following location b
ut
jvm 1    |           could not be loaded:
jvm 1    |             D:\vidavid\CI_For_Java\mastering-ci\tools\sonarqube-4.5.7
\sonarqube-4.5.7\bin\windows-x86-64\.\lib\wrapper.dll
jvm 1    |           Please verify that the file is readable by the current user
```

16. Bu default SonarQube will take 9000 port, If you wish to see SonarQube started properly you can open "http://localhost:9000"

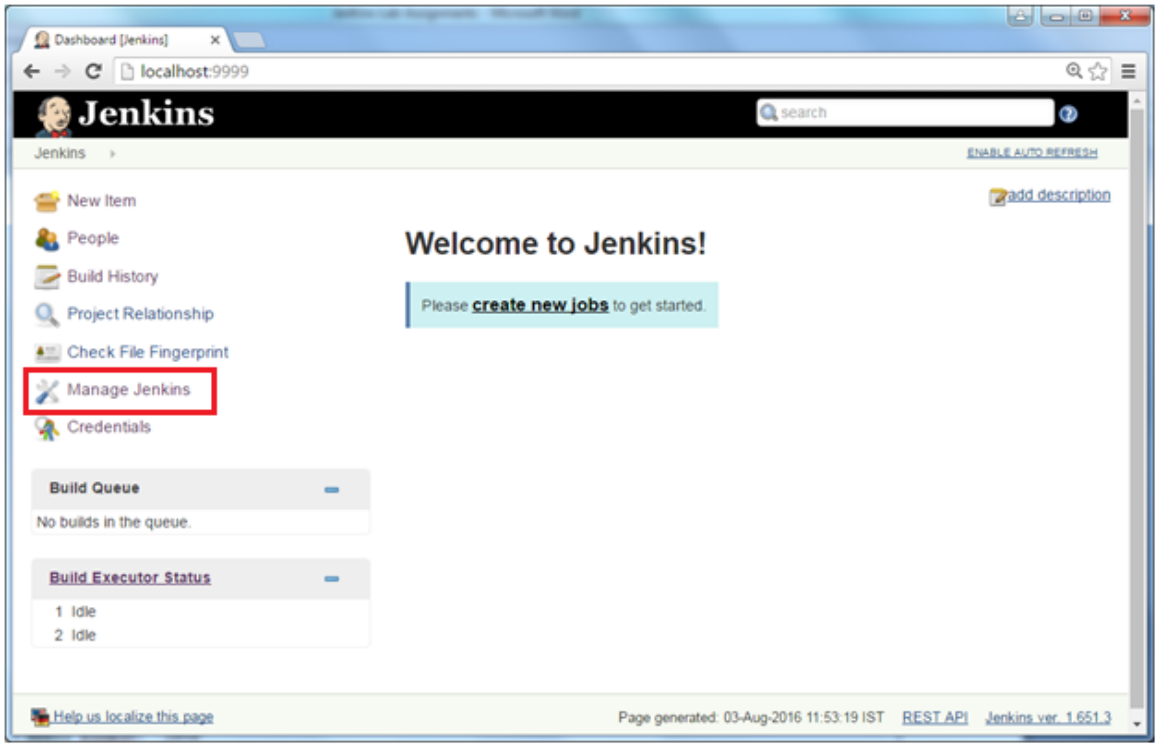
17. Open you command prompt, check the JDK path. If JDK path has been defined properly. Execute the below command to start **JENKINS** server.

    java –jar jenkins.war

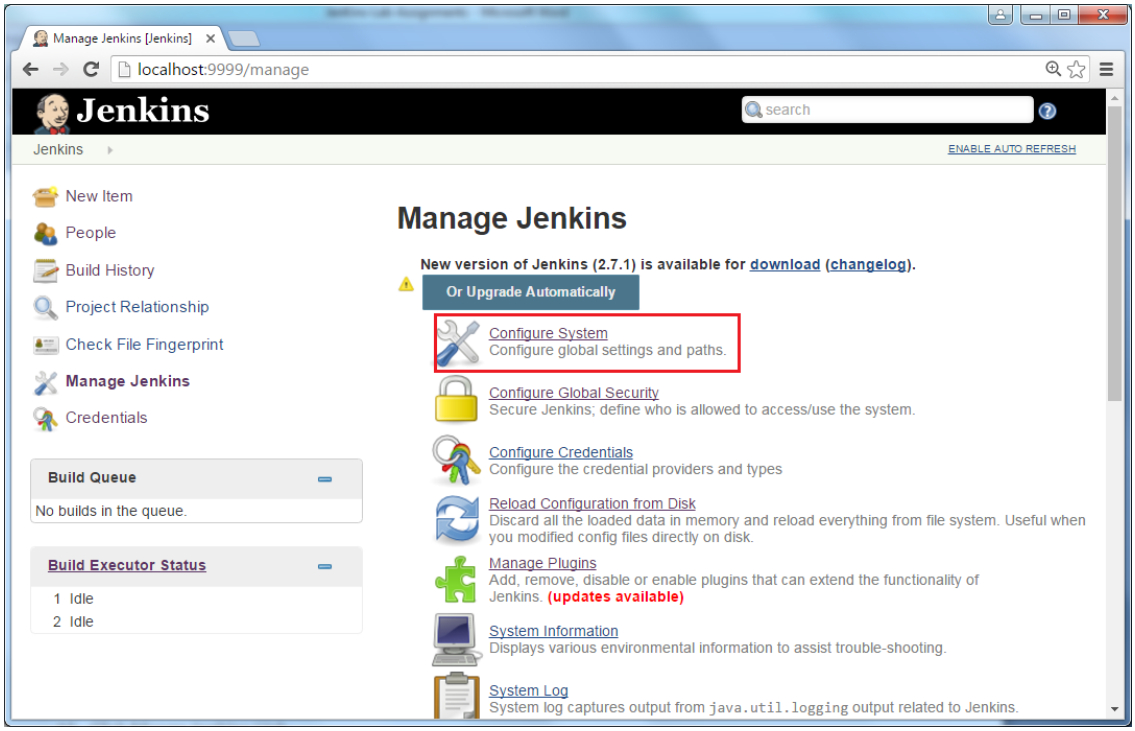
    a. The default port taken by Jenkins is 8080. In case if you get any error related to port, start the jenkin with appropriate port number.

    java –jar jenkins.war --httpPort = 9999

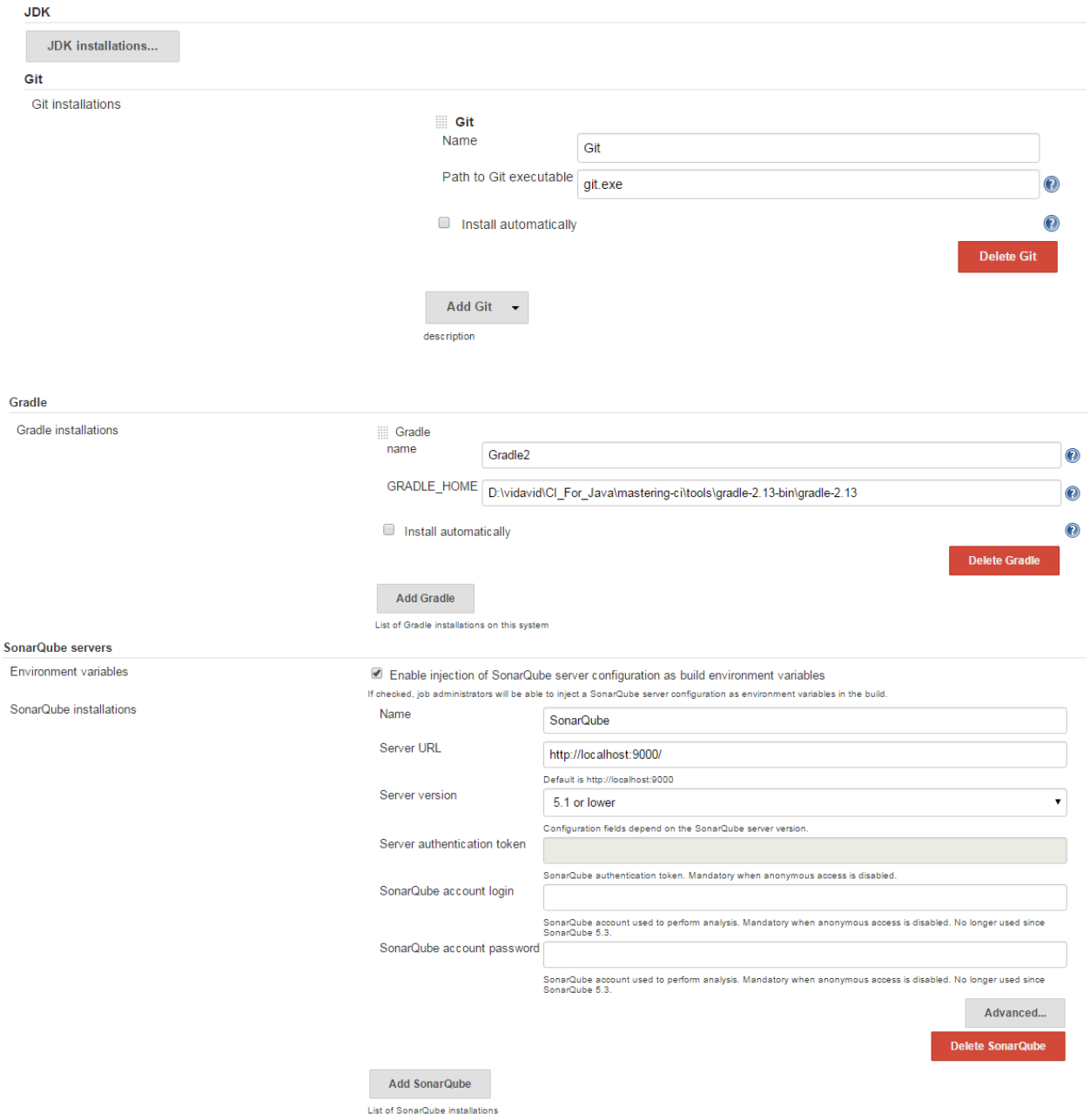
18. If Jenkins server started successfully, you can see "**Jenkins is fully up and running"** command in the command prompt.
19. Now open browser enter the below URL
    a. http://localhost:8080 (by default) (or) http://localhost:9999 (if you change port)
20. You can see the below screen



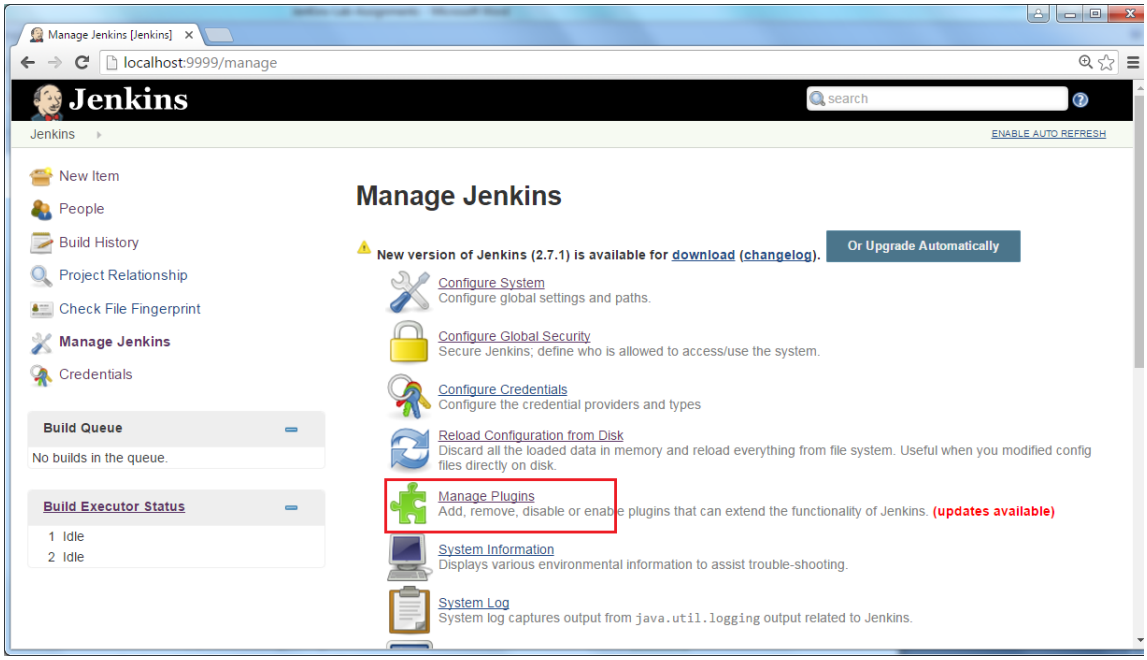
21. Click Mange Jenkins Link, you will be redirected to the below screen



22. Click Configure System link, it will give the Jenkins configuration window.
23. In this window, you will be getting different titles in the name of **JDK, GIT, Gradle, SonarQubeServers** etc. That was displayed in the below screen shot.



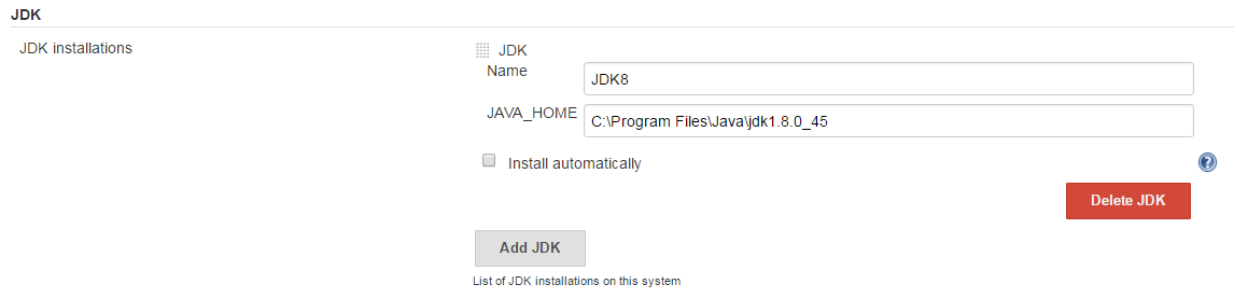
24. If you have not seen GIT as one of the title here. Click **Manage Jenkins** link. And then click **Manage Plug-ins** as highlighted below.



25. Under Available tab, search GitHub Plug-in and Gradle Plug-in, it will list GitHub and Gradle as one of the plug in shown below

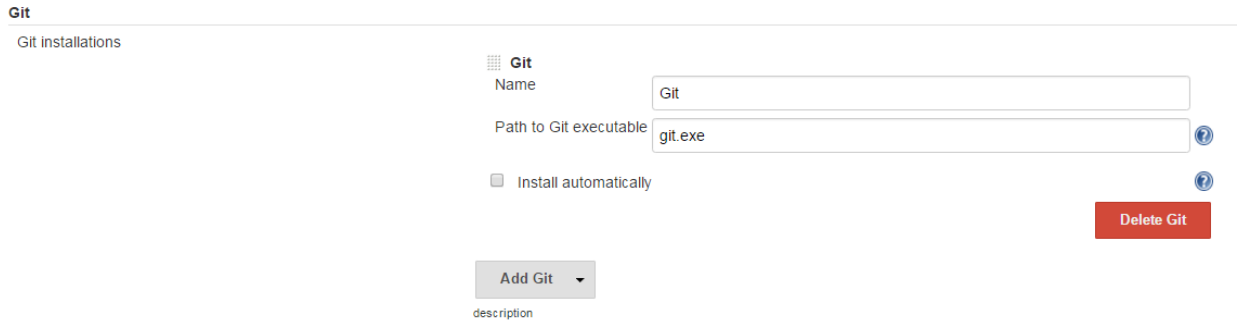
26. Select the plug-in and click **install without restart**, if you don't want to restart the Jenkins.
27. Follow the same to install the other plug-ins if it does not available.
28. And then click **Back to DashBoard → Manage Jenkins → Configure System**
29. Specify JDK Home under JDK. Please refer the below screen shot

JDK:

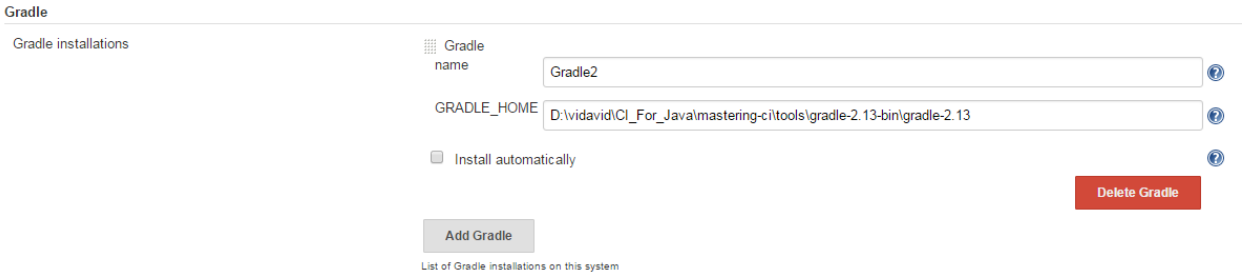


30. Do the followings for GIT and Maven too.

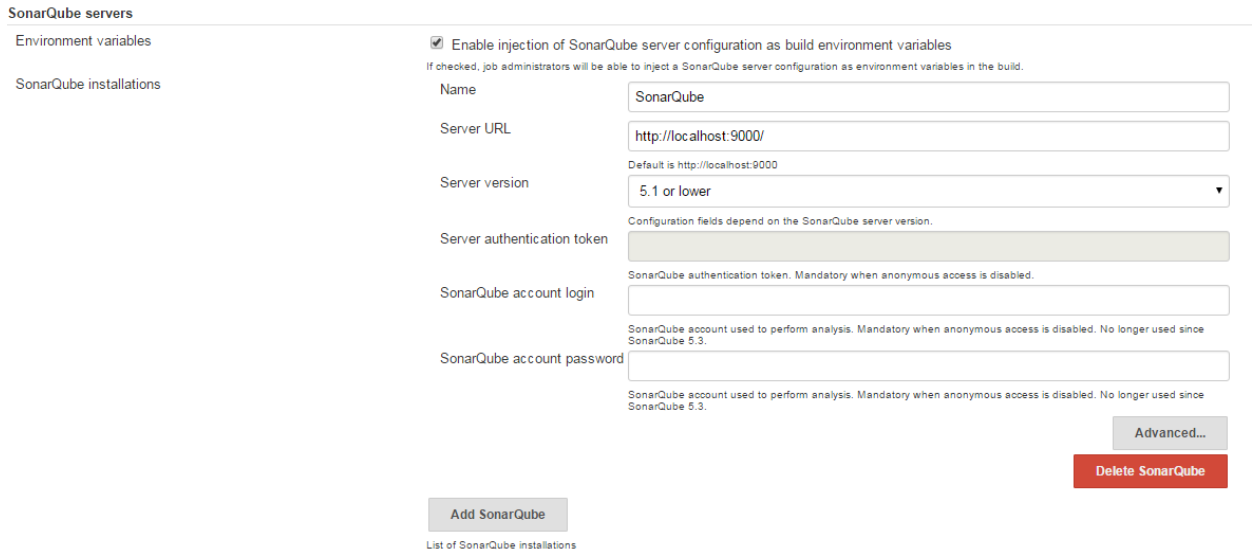
GIT:



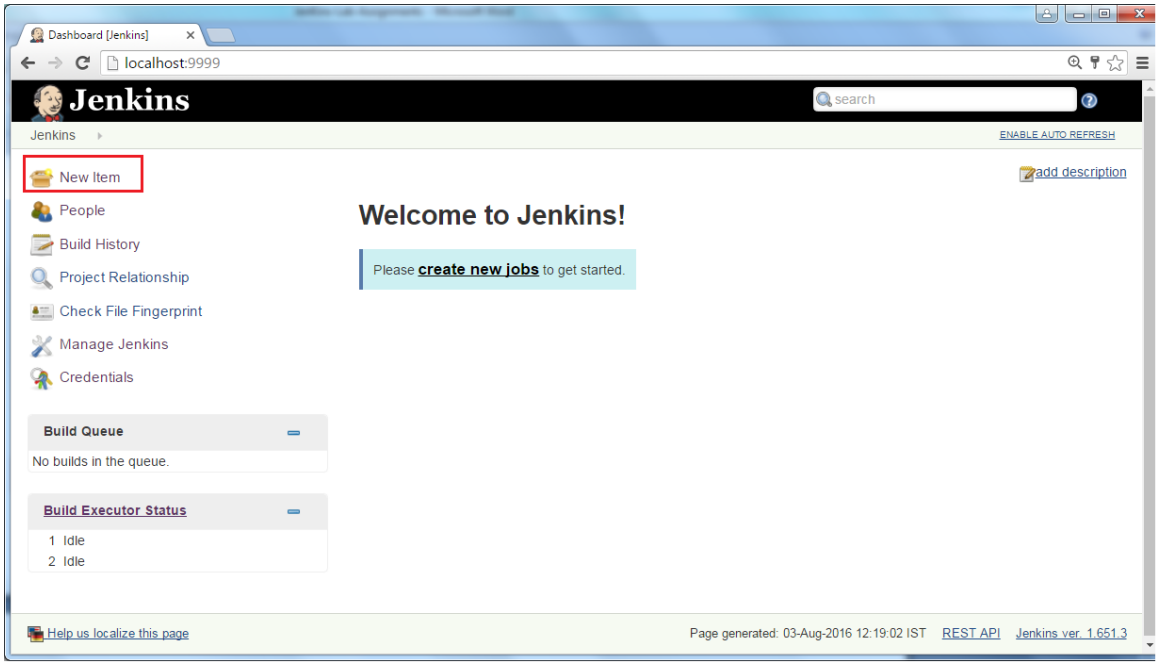
Gradle:



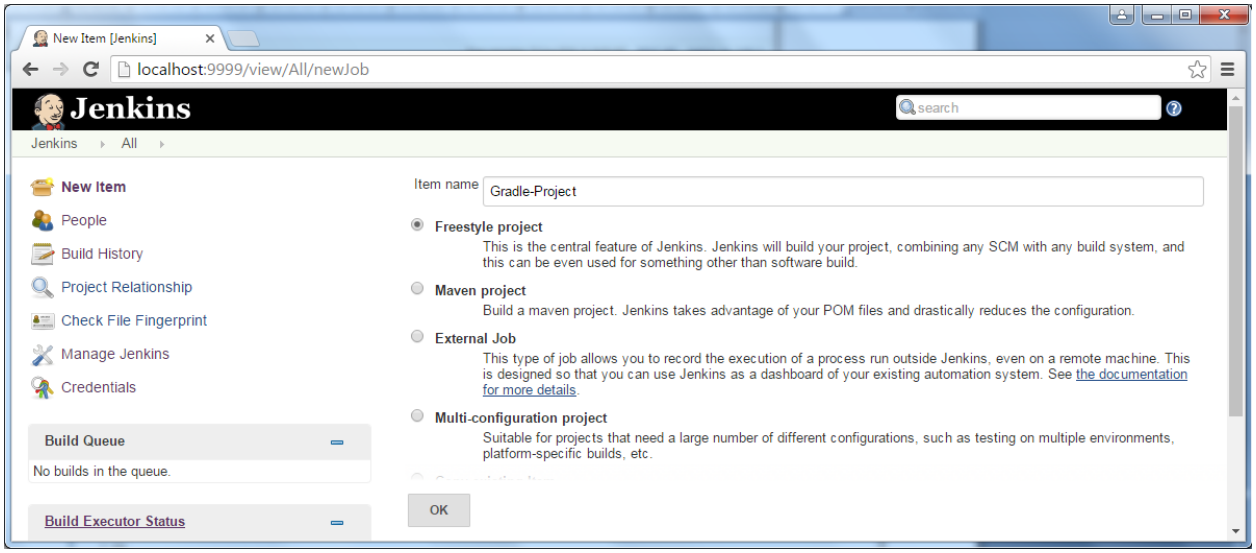
SonarQubeServers:



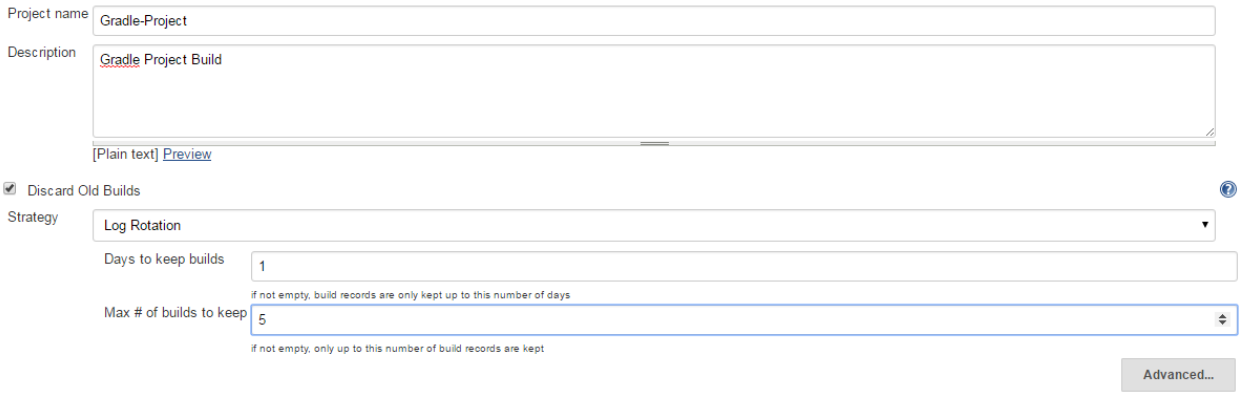
31. Click **Apply → Save** to save the configurations in Jenkins.
32. Click **New Items** Link as showed below.



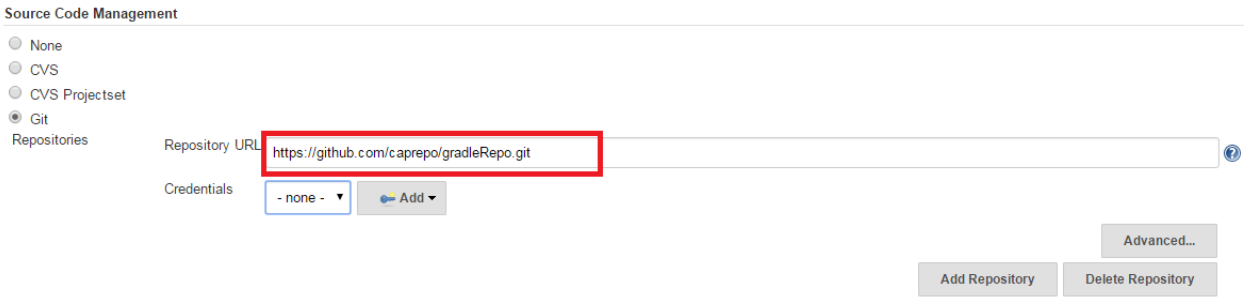
33. In **Item Name** text box mention your job Name example "**Gradle-Project**" and then select **Freestyle Project** and then click **OK**.



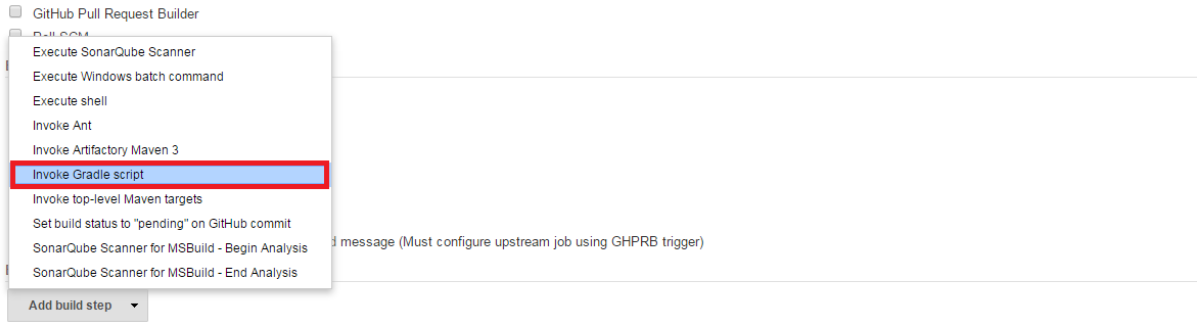
34. Mention project description and select Discard Old Build. Look at the below screen shot for further details



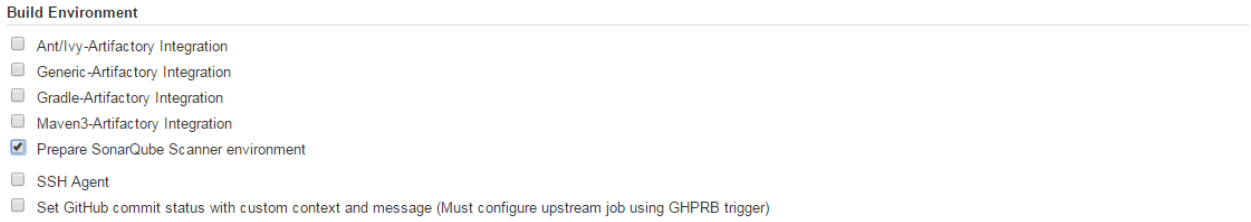
35. Under **Source Management** select **Git** and paste the URL where your application has been uploaded.



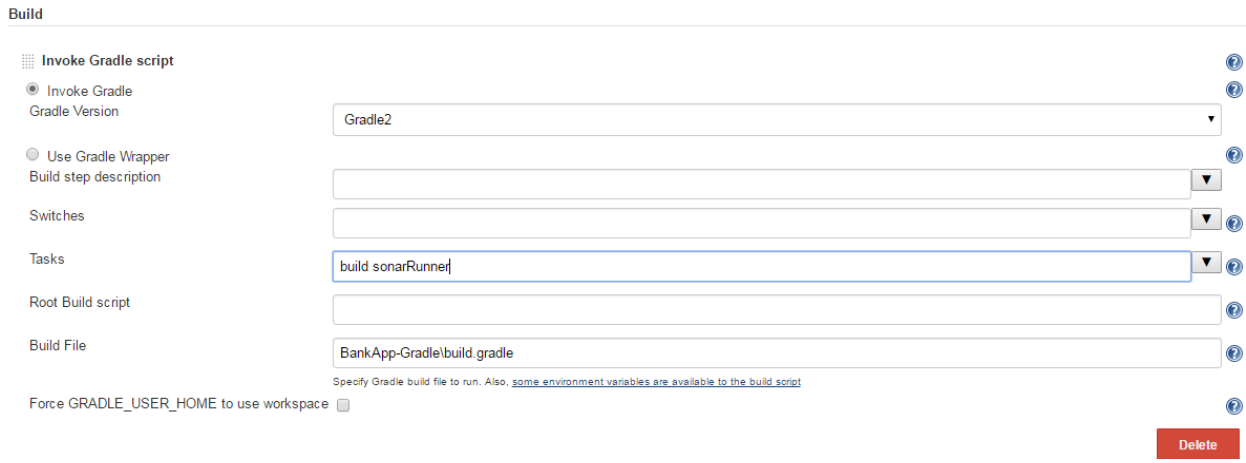
36. Under **Pre steps** click **Add pre-build step**→ choose **invoke Gradle Script**.



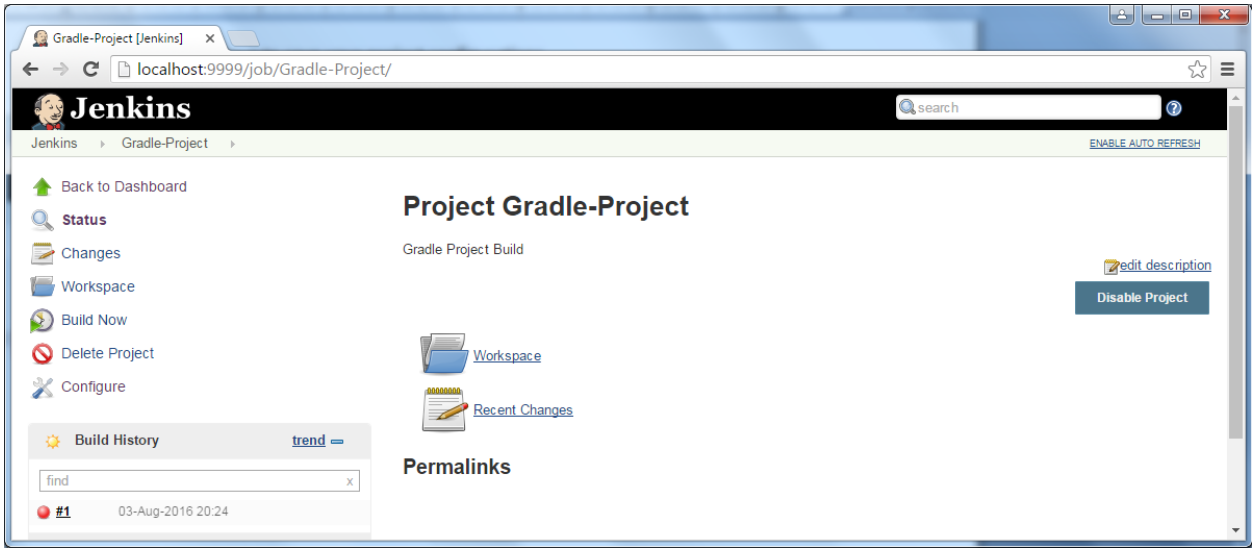
37. Under Build Environment check **Prepare SonarQube Scanner environment**



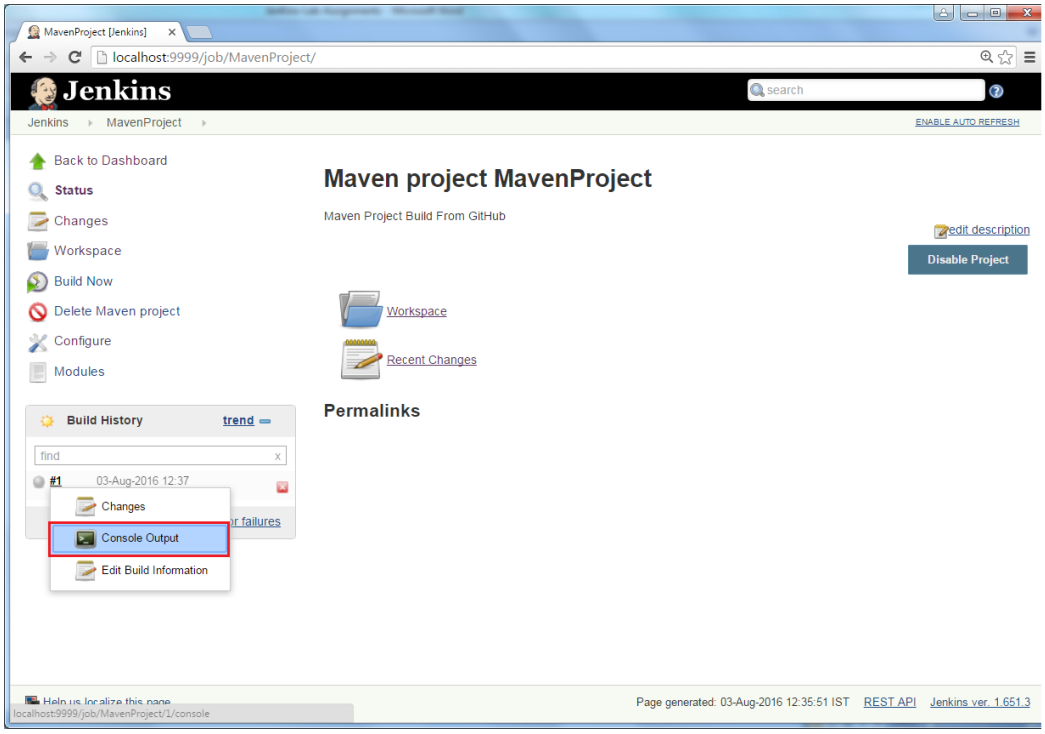
38. Now under **Build**, enter **Gradle version** and **Task** details, mention where is your **build.gradle file** under GitHub project repository (Usually its under your project name, so enter projectname/build.gradle)
39. Task should be **build sonarRunner**.



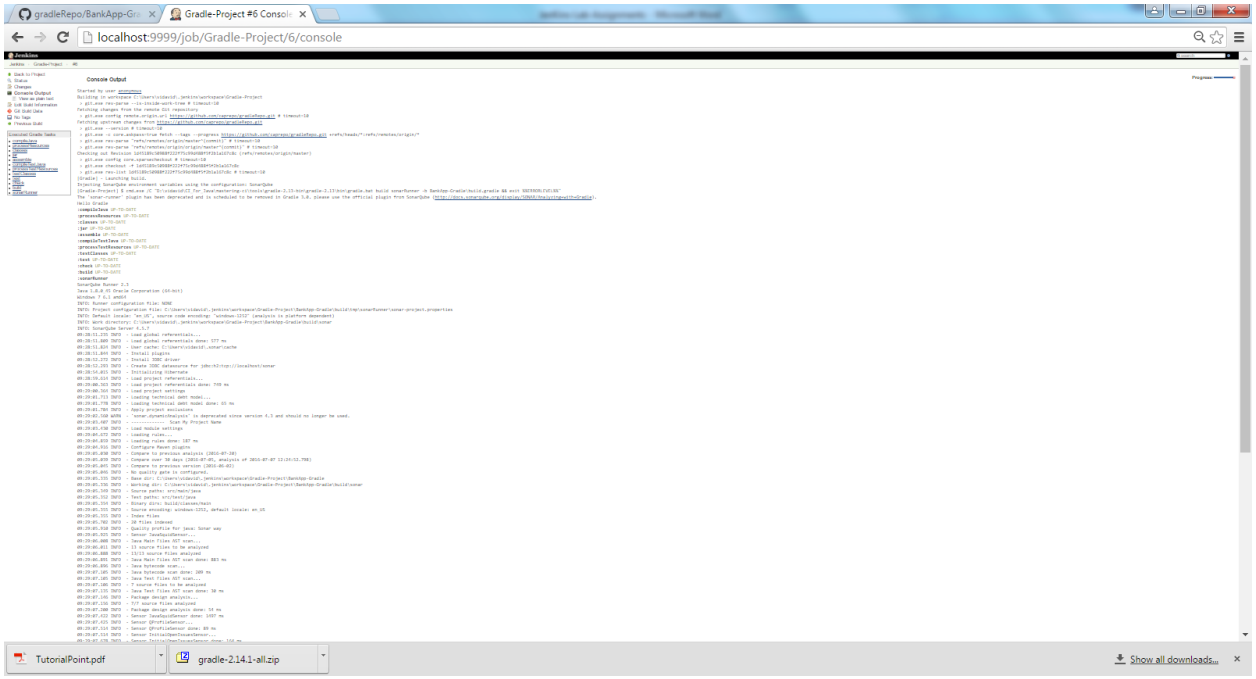
40. Now click Apply → Save to save your project configurations.
41. Click **Build Now**



42. Under Build History you can see the build count. Click the count and select console output.

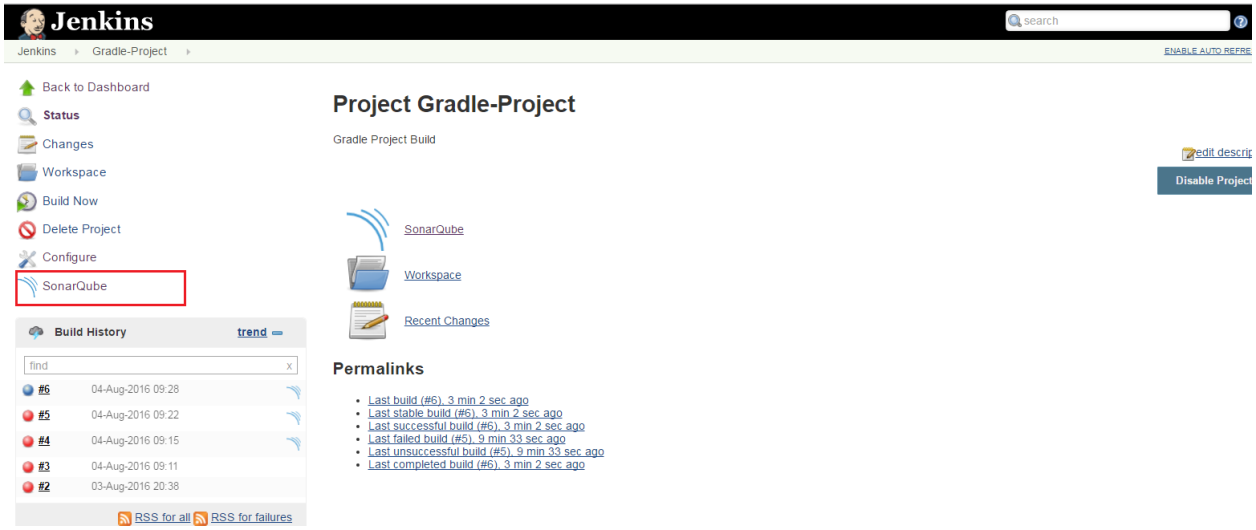


43. In the console output, you can see the complete report for build task.



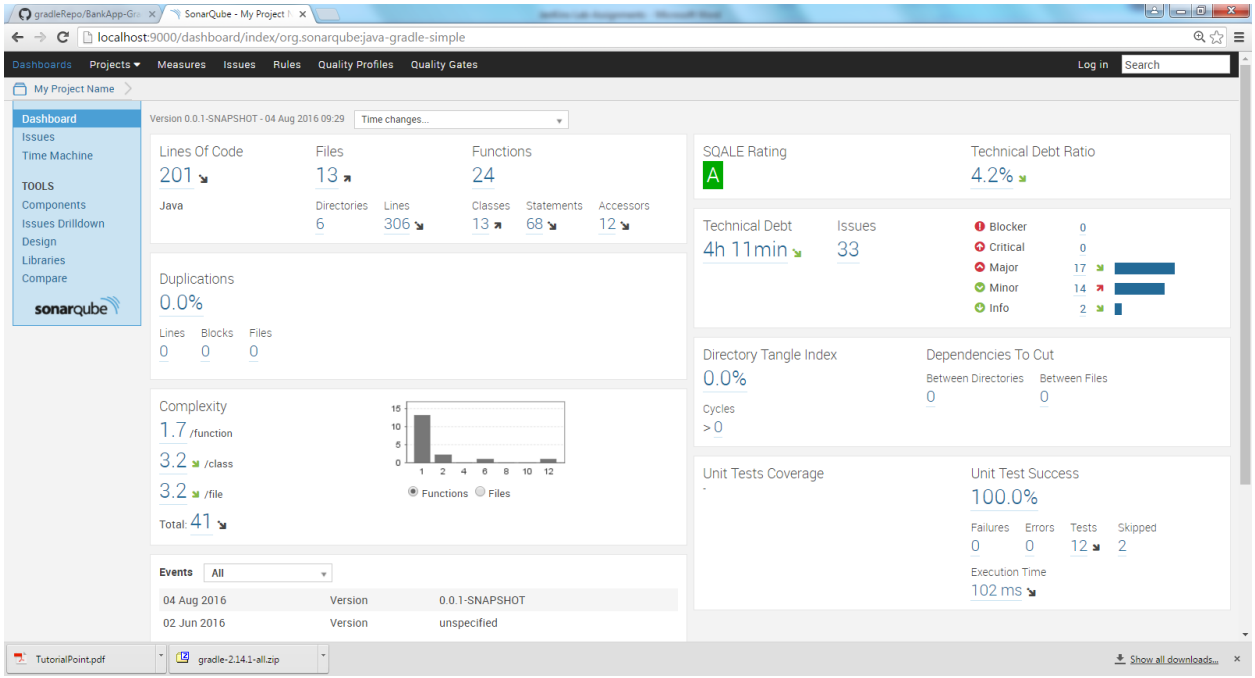
44. In case any one of the test case fails, the build will fail.

45. Do the changes in the project and commit your project with GIT again. And then build your application again. This process should be repeated until you get the success build.
46. Repeat previous steps until your build success. Once build success, Click **Back to Project**. Here Click SonarQube link which is highlighted below:



47. For success build blue color balloon will be generated in Jenkins, Failure case Red color Balloon.

48. You will be getting the complete code quality analysis report. You can discuss your report and do the changes. Report looks like the below:



Conclusion:

From the above example, we learnt how continuously integrate gradle project with Jenkins Server. We understand how to check code quality using SonarQube.