

Table of Contents

[1](#_Toc532565761)

[Table of Contents 2](#_Toc532565762)

[1 Lab Details: 2](#_Toc532565763)

[2 DevOps Tools: Guided Exercises 3](#_Toc532565764)

[1.1.1 Jenkins –Problem Statement 3](#_Toc532565765)

[Guided Exercise 1 4](#_Toc532565766)

[Guided Exercise 2 15](#_Toc532565767)

# Lab Details:

You will be provided with the below DevOps Lab for practicing the guided exercise in this document.

Enrol for DevOps Tools - Practice Lab to practice these guided exercises.

# DevOps Tools: Guided Exercises

### Jenkins –Problem Statement

A leading training institute in India is planning to develop their Self Learning Portal. The proposed solution has been planned to develop in a DevOps environment. The project is under development and there is a module which allows learners to maintain their skill development plan. The functionality allows the learners to login and manage their learning schedule.

The application is being developed in DevOps environment. There are many developers working on this project and their daily builds shall be updated to the SCM repository. As per the requirement, a CI pipeline needs to be created using Jenkins to build, and check the code quality of the daily code updates.

**Scope:**

You have been assigned the task of managing the Jenkins Pipeline to enable automated build on the projects uploaded to SCM repository. You need to perform the following tasks.

**Tasks:**

* Get the project builds from the SCM repository
* Create a pipeline to automate the build process
* Automate Test cases execution
* Check the code quality using Sonar

**Steps*:***

1. Install and configure Jenkins
2. Install required plugins in Jenkins
3. Jenkins configuration with Maven ,Git and Sonar
4. Create a pipeline in Jenkins
5. Build and verify the pipeline
6. Generate Code quality reports.

**Pre- Requisites**

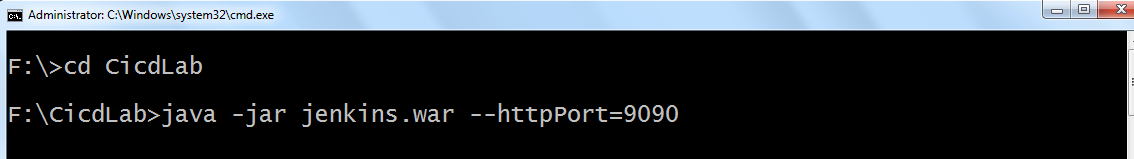
* Maven 3.5
* Jenkins
* Git
* SonarQube

Guided Exercise 1**:** **Setup Jenkins**

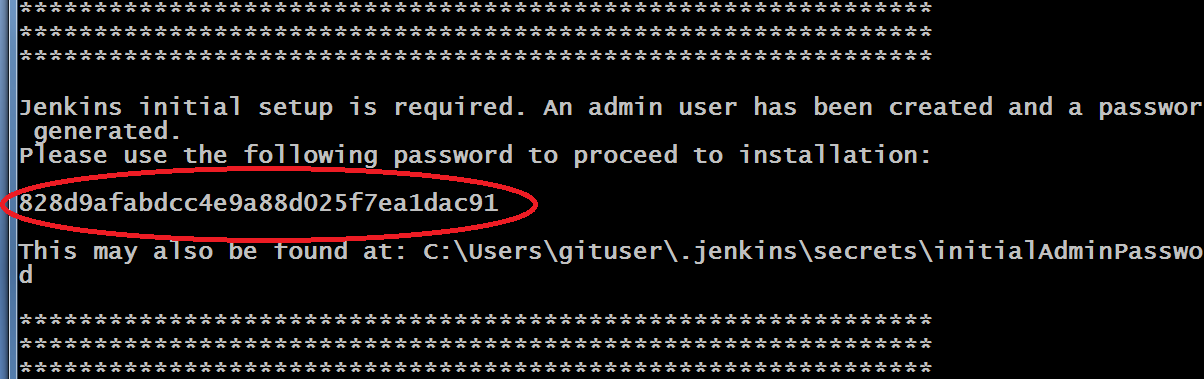
* + Verify the jenkins.war file is present in the path C:\Program Files (x86)\Jenkins
  + Open command prompt and navigate to the Jenkins folder mentioned above.
  + Start Jenkins on port 9090 by executing the below java command in the command prompt (default port is 8080) Result is shown in [Fig 1.9-2.1]

java –jar jenkins.war --httpPort=9090

* As part of the initial setup, observe the logs to get the admin password.

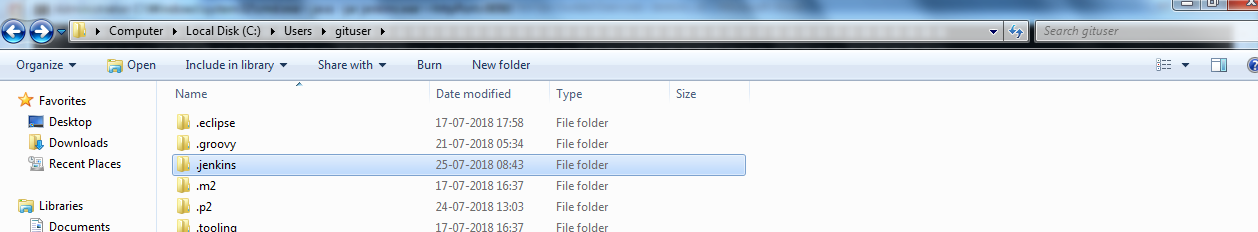


[Fig 1.9]



[Fig 2.1]

* Jenkins creates folder (.jenkins) in users profile (c:\users\username\.jenkins) Ref[Fig 2.2]



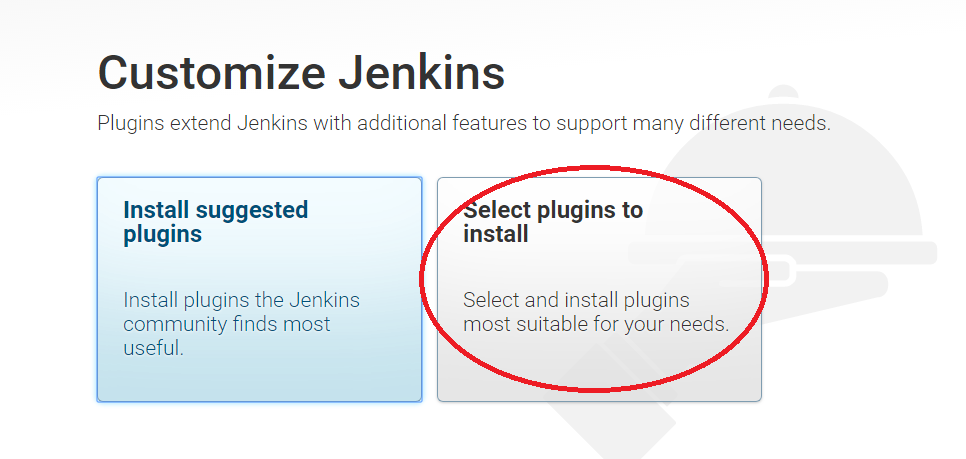
[Fig 2.2]

* Admin password is stored in initialAdminPassword file in users profile

C:\Users\username\.jenkins\secrets\initialAdminPassword

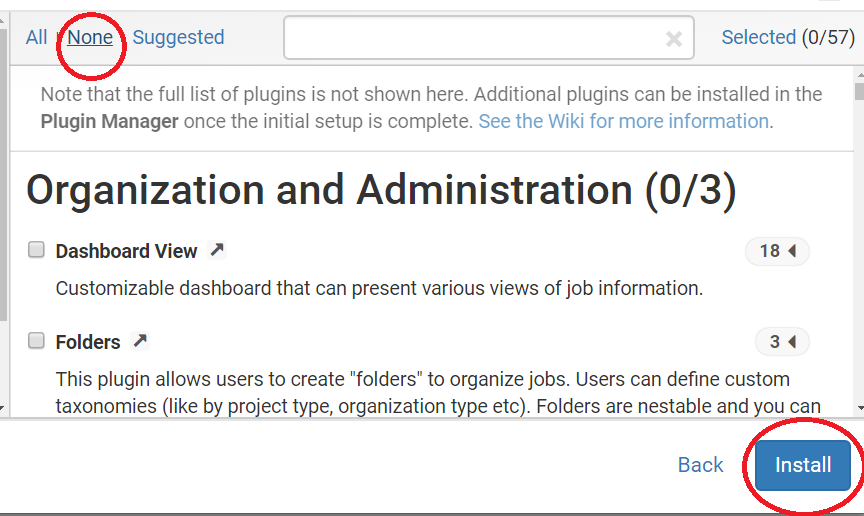
**Step 9: Set up the Jenkins Environment**

* Open <http://localhost:9090> in a web browser.
* Enter the admin password received as part of the initial setup, to Unlock Jenkins.
* Next, click on Select plugins to install [Reference [Fig 2.3 ]



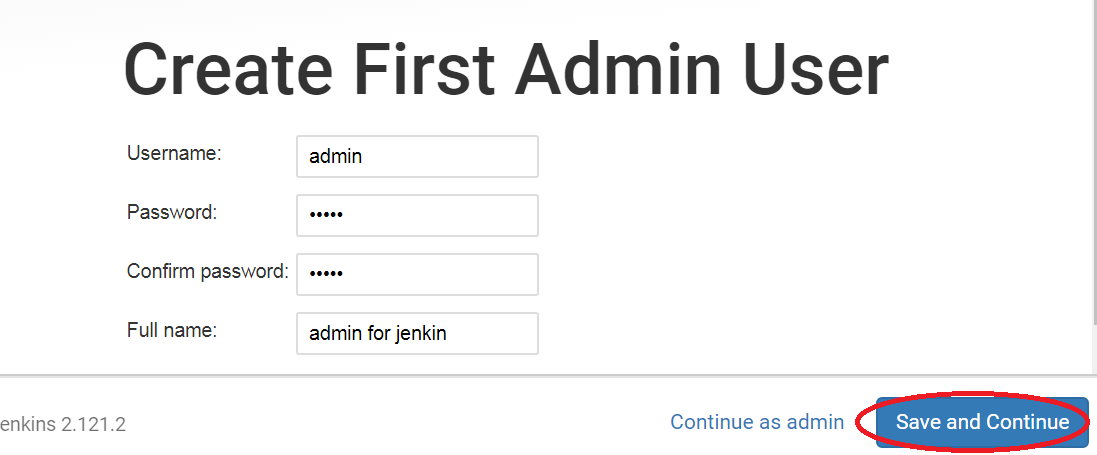
[Fig 2.3]

* Select None and then click on Install [ Reference [Fig 2.4 ]

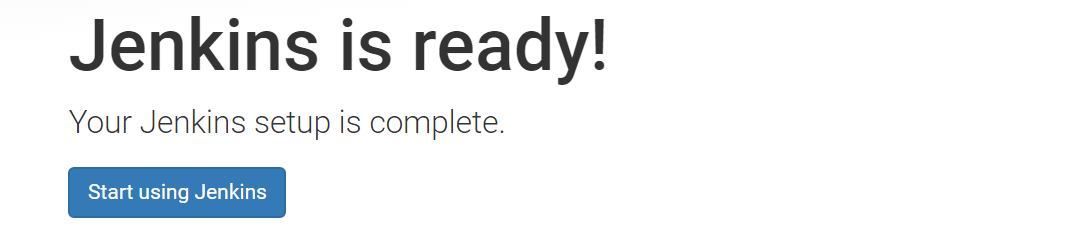


[Fig 2.4]

* Create the First Admin User account by specifying a username and password (Example : username – admin, password – admin)
* Click on Save and Continue [ Reference [Fig 2.5 - 2.6] ]

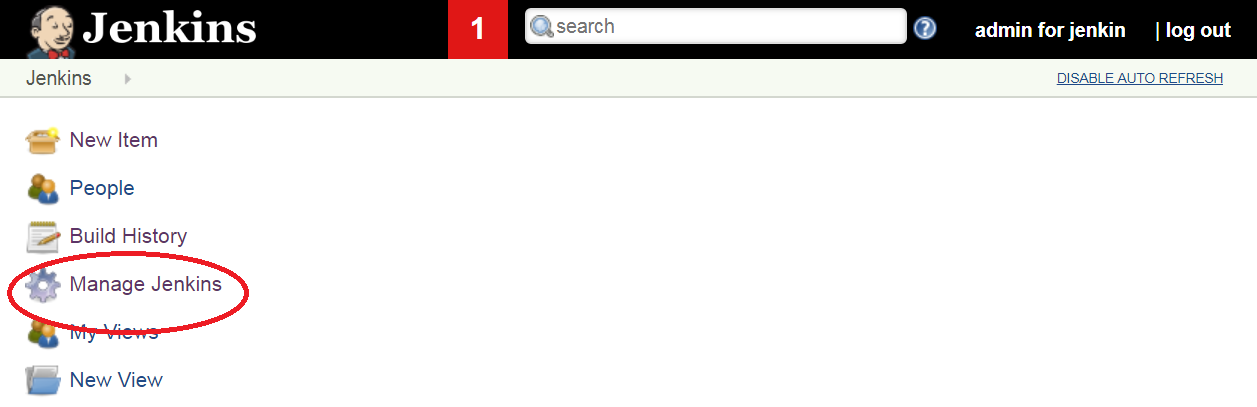


[Fig 2.5]



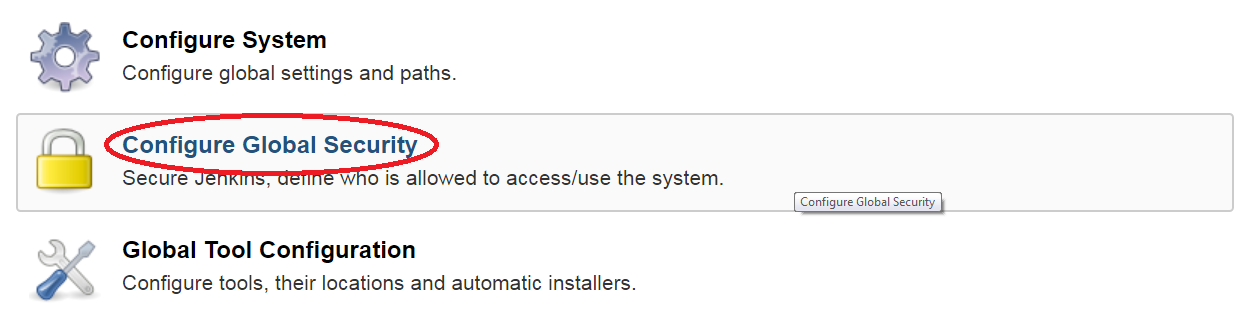
[Fig 2.6]

The following figure shows the Jenkins Homes Screen. Now the security permissions can be changed in Jenkins by using the **Manage Jenkins -> Configure Global Security** option. Refer Figure 1.11 and 1.12



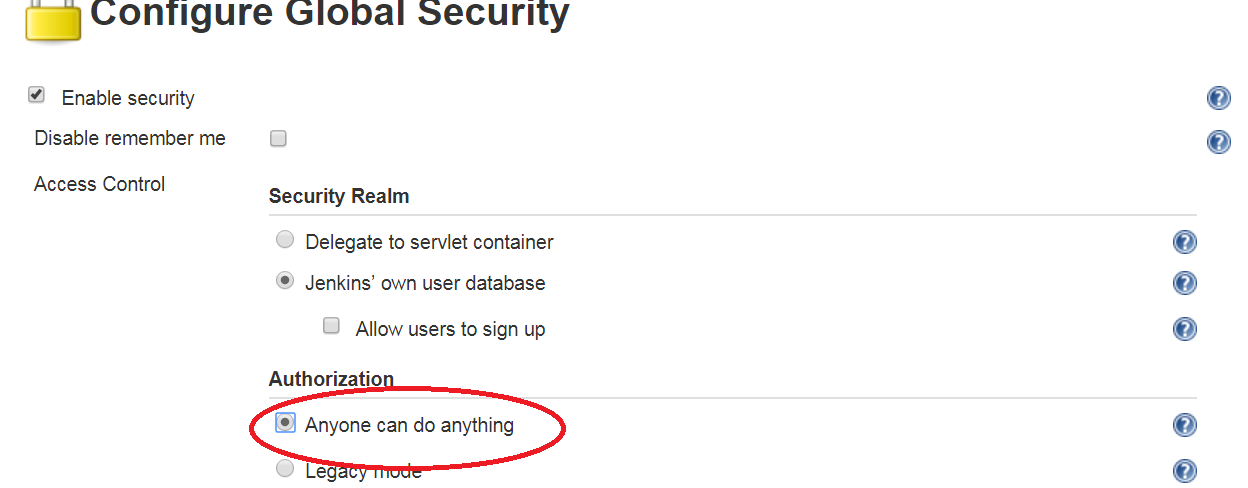
[Fig 1.11]

* Select Configure Global Security

****

[Fig 1.12]

* In Authorization, select an option **Anyone can do anything** (apply and save the setting). Refer Fig 1.13

****

[Fig 1.13]

**Step 3: Add the required plugins in Jenkins and configure the global tools and system.**

**Estimated Completion Time:** 10 Minutes

**Objective**: Install and configure the required plugins in Jenkins.

**Steps to follow:**

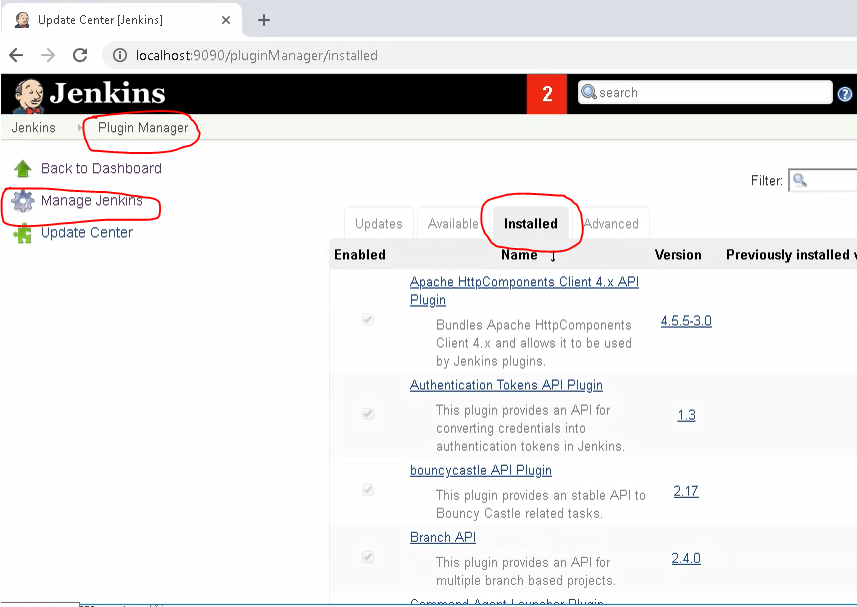
**Step 1: Install the required plugins.**

*Here, we will learn how to manually install plugins in Jenkins without an active internet connection*

1. If your Jenkins instance is already running, please stop the process first. You can do this by hitting Ctrl+C in the Windows Command Prompt window where Jenkins was started.
2. Copy all the contents from C:\Program Files (x86)\Jenkins\plugins into C:\Users\<your-id>\**.**jenkins\plugins
3. Restart Jenkins, by executing in the Windows Command Prompt, the same java command as before:

**java –jar Jenkins.war –httpPort=9090**

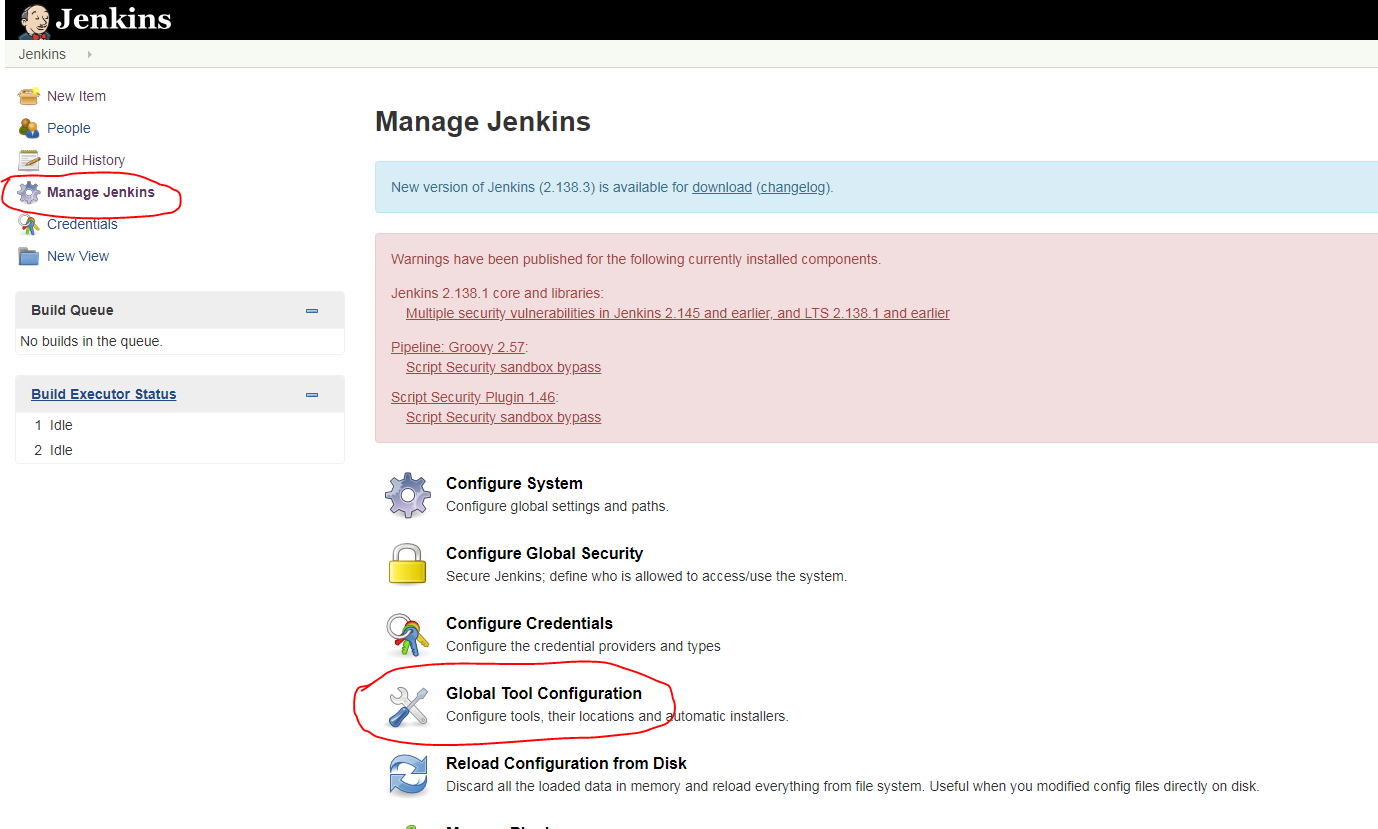
1. In a browser, navigate to the URL, <http://localhost:9090>
2. Click on Manage Jenkins -> Manage Plugins -> Installed
3. Here you will see a list of all the installed plugins.



**Step 4: Configure the installations inside Jenkins.**

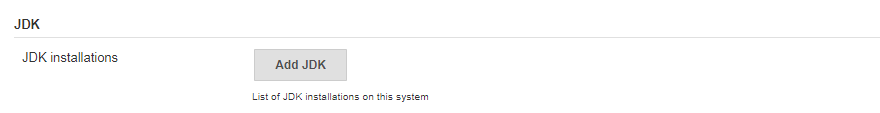
The installed plugins and build environment configurations are as follows.

1. Navigate to Manage Jenkins - > Global Tool Configuration. Refer Fig 3.9



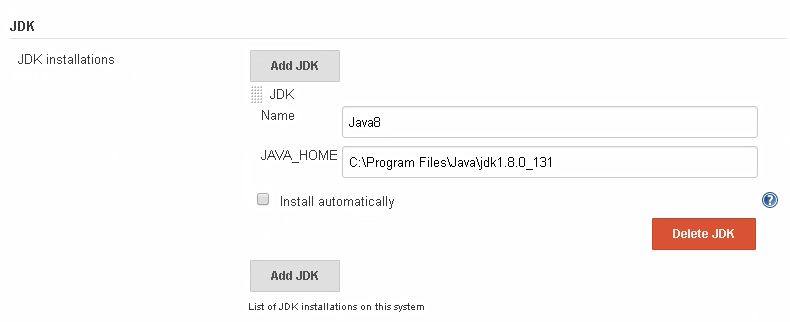
[Fig 3.9]

2. Set up JDK. Click on JDK installations -> Add JDK. Refer Fig 3.10



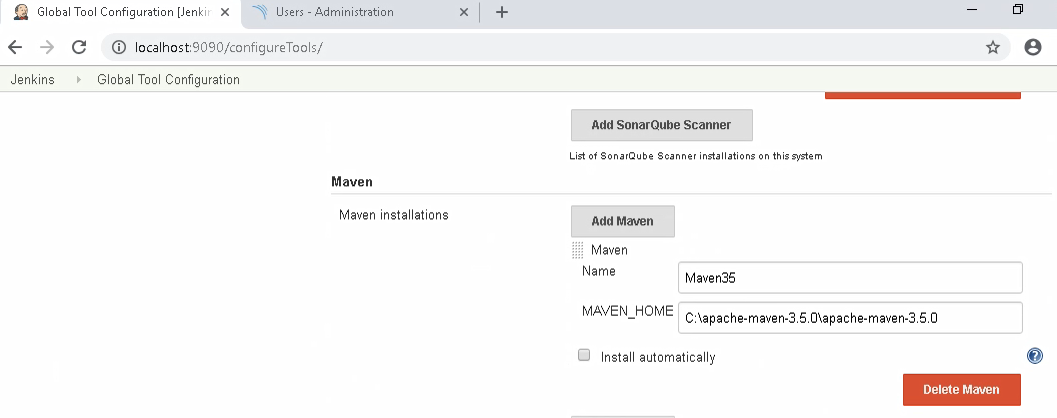
[Fig 3.10]

Specify the JDK Home path as per your JDK Installation. Give a name and uncheck “**install automatically**” .Refer Fig 4.0



[Fig 4.0]

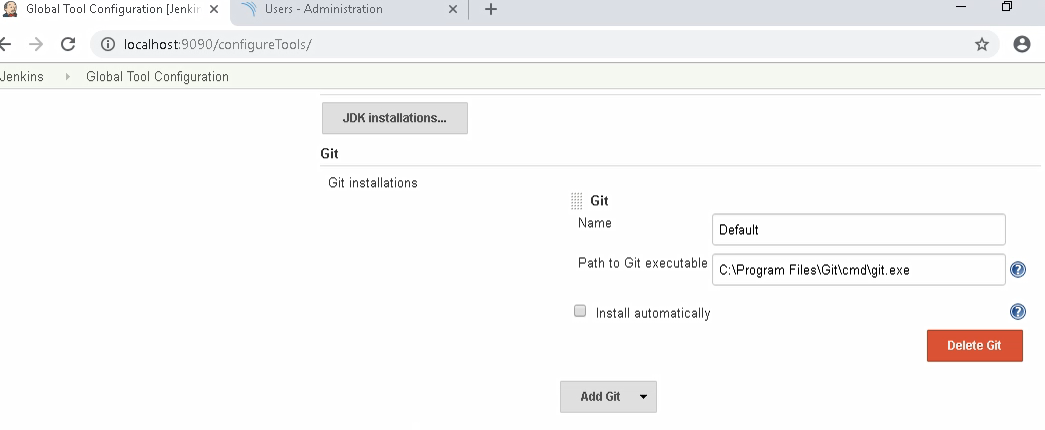
3. Set up Maven as same as done in the previous step. Refer Fig 4.1



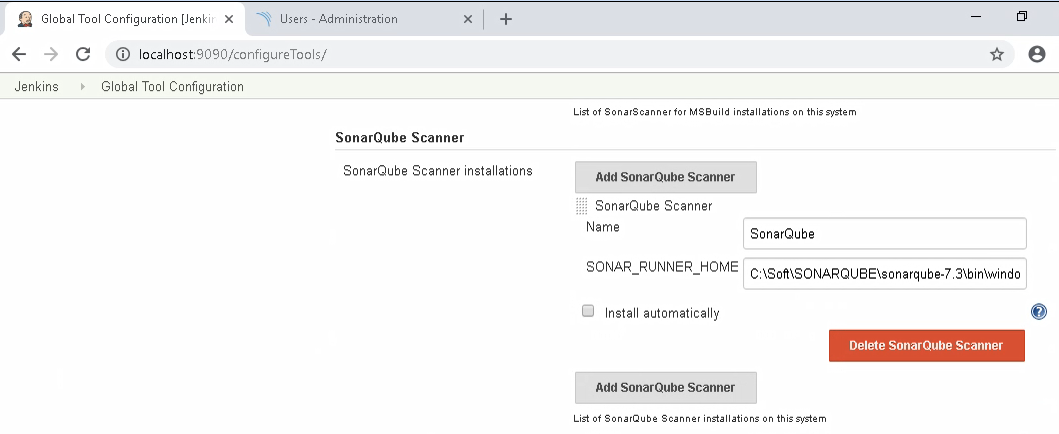
[Fig 4.1]

4. Set up GIT.

Configure the GIT installation path. Refer Fig 4.2



Specify the path to your SonarQube installation as done in the previous steps.

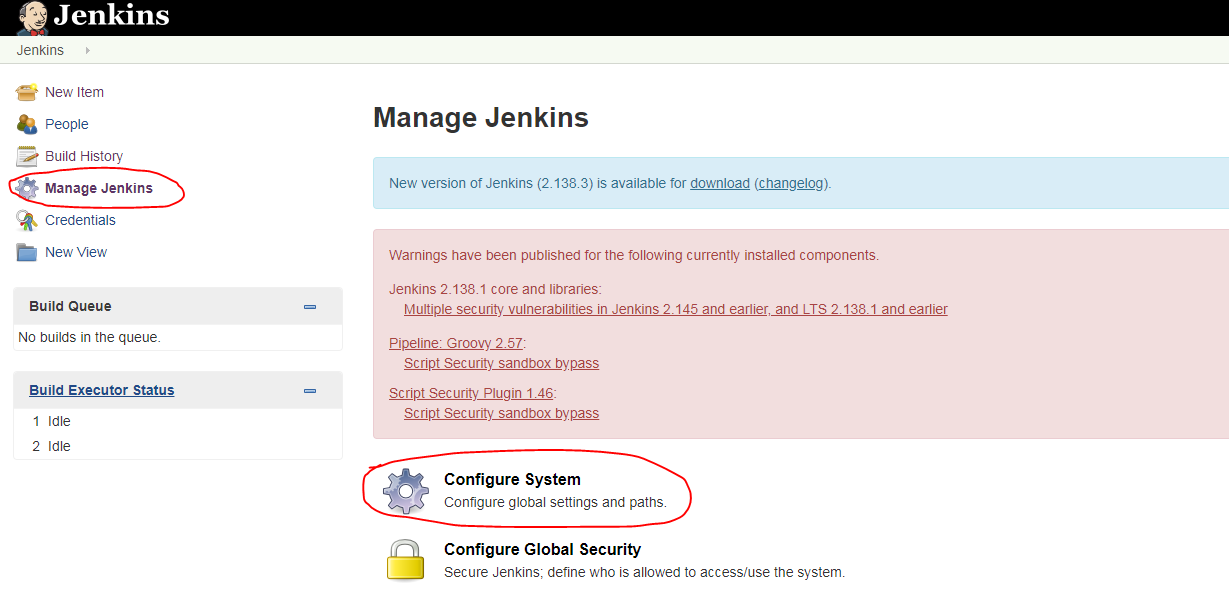


[Fig 4.4]

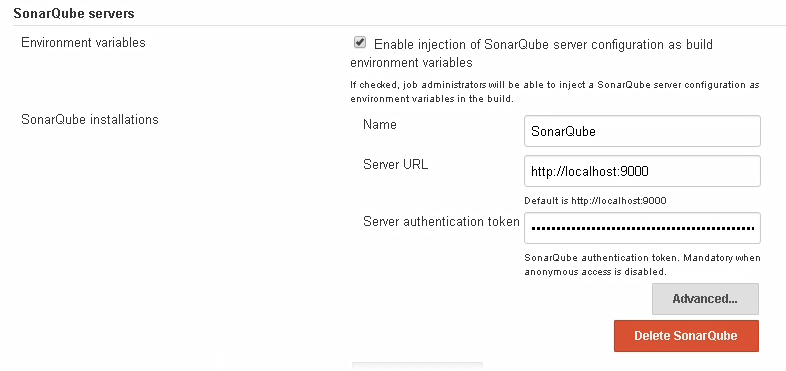
After all configurations are done, click on Save to make the changes permanent.

**Step 5: Configure Sonar Server on Jenkins.**

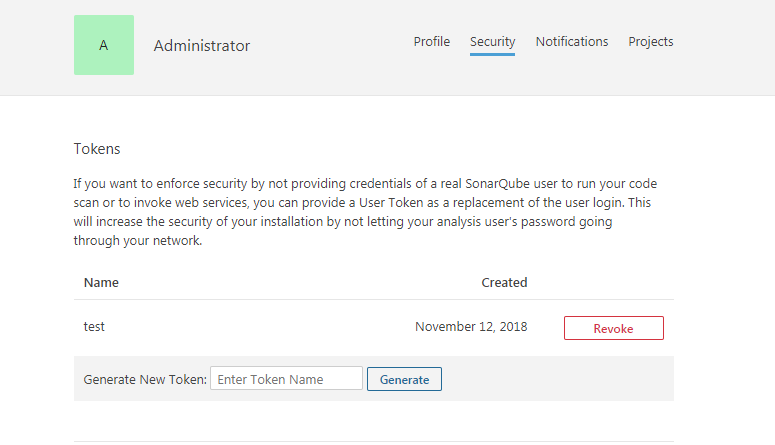
1. Navigate to Manage Jenkins -> Configure System. Refer Fig 4.5

[Fig 4.5]

Configuring Sonar as shown in the image below. Refer Fig 4.8

[Fig 4.8]

The server authentication key has to be generated from the SonarQube GUI. To generate a key do the following. Navigate to My Account -> Security -> Give a name and generate token. Refer Fig 4.9



[Fig 4.9]

**Step 5:** Create a Sonar profile in Maven. Follow the given steps.

1. Navigate to <<maven installation folder >>/ conf
2. Edit **settings.xml** and add the following lines under <settings> tag.

*<pluginGroups>*

*<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>*

*</pluginGroups>*

1. Add the following lines under <profiles> tag.  
   *<profile>*

*<id>sonar</id>*

*<activation>*

*<activeByDefault>true</activeByDefault>*

*</activation>*

*<properties>*

*<!-- Optional URL to server. Default value is http://localhost:9000 -->*

*<sonar.host.url>*

*http://localhost:9000*

*</sonar.host.url>*

*<sonar.login>admin</sonar.login>*

*<sonar.password>admin</sonar.password>*

*<!—Use the same default username and password admin/admin -->*

*</properties>*

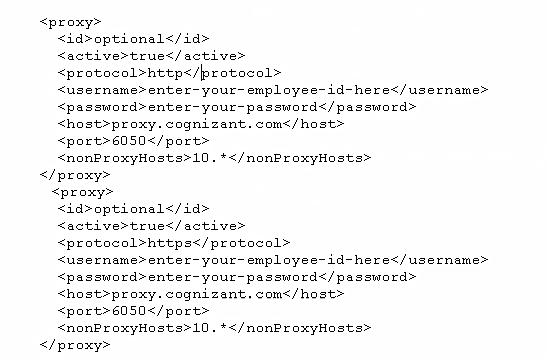
*</profile>*

*</profiles>*

1. Ensure that the same settings.xml file is present under YOUR\_HOME\_DIR\**.**m2\conf and maven\_installation\_dir\conf. This will ensure that you don’t encounter any errors when running Maven commands from the command prompt as well as from within Jenkins builds.

**Step 3: Configure proxy for Maven**

1. Open the settings.xml file in the maven\_installation\_dir\conf.
2. Under proxies, look for the section for the http protocol. Update this section with:  
   username – your CTS Employee ID  
   password – your CTS network password  
   host – proxy.cognizant.com  
   port – 6050  
   nonProxyHosts – 10.\*
3. Add another proxy section for the https protocol.

****

1. Ensure that the same settings.xml file is present under YOUR\_HOME\_DIR\**.**m2\conf and maven\_installation\_dir\conf. This will ensure that you don’t encounter any errors when running Maven commands from the command prompt as well as from within Jenkins builds.

Guided Exercise 2**:** **Create the CICD Pipeline using Jenkins**

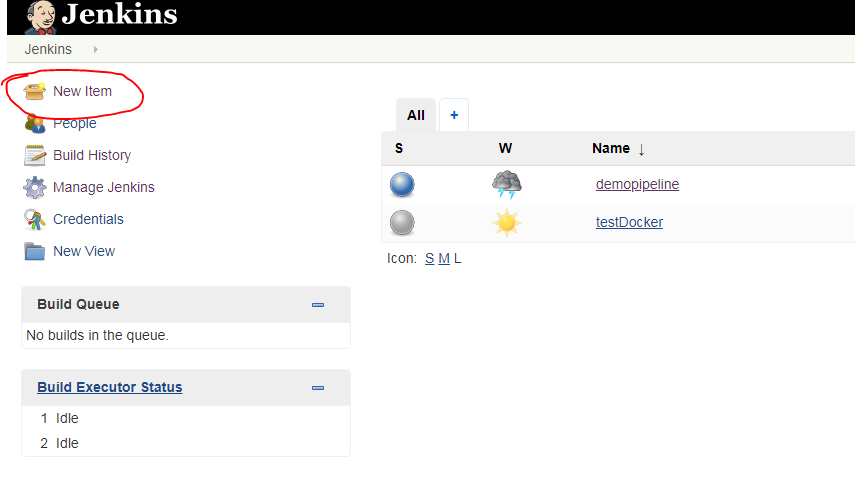
**Estimated Completion Time:** 10 Minutes

**Objective**: Create a pipeline to manage continuous integration and delivery.

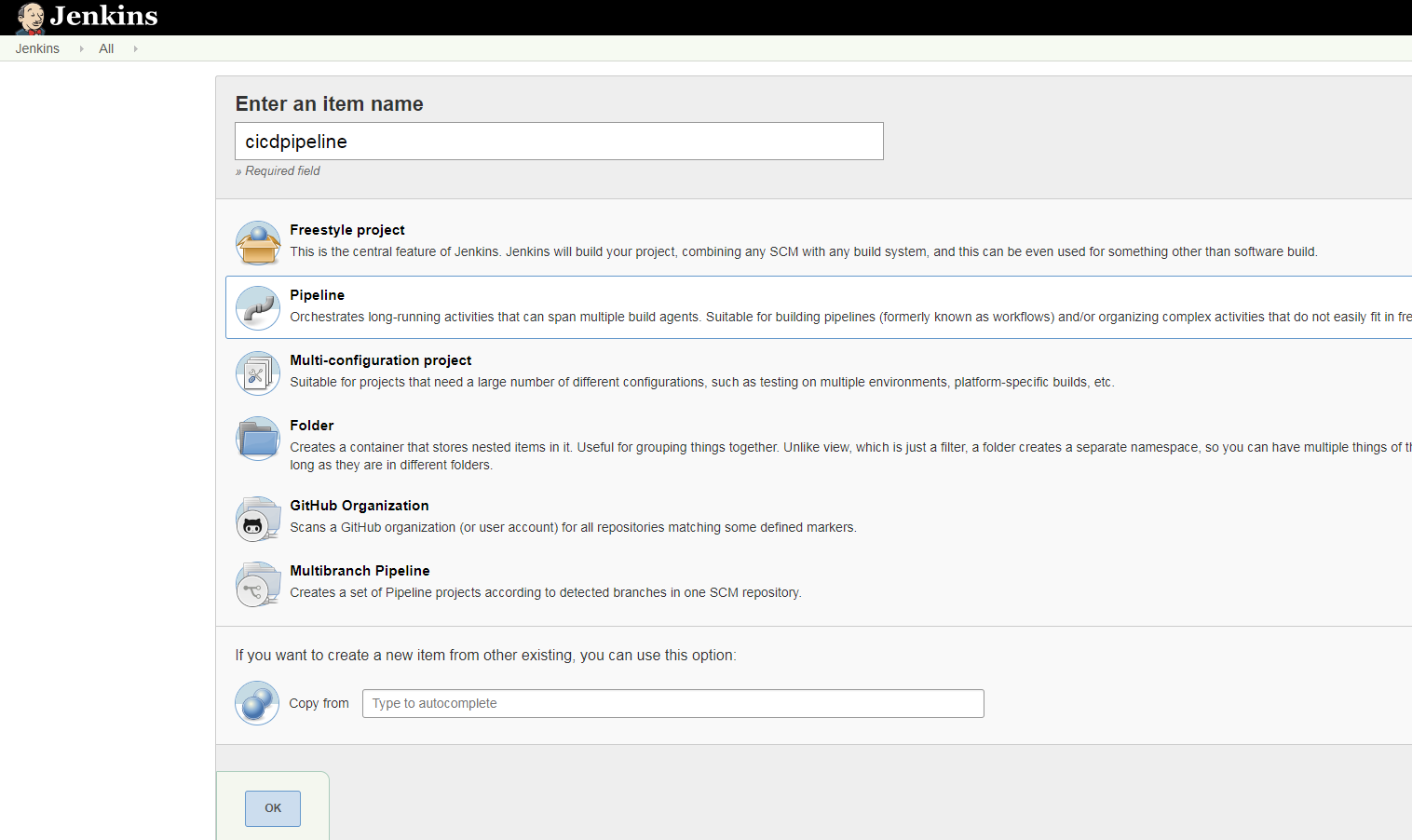
**Steps to follow:**

**Step 1: Create a Pipeline project in Jenkins.**

1. On Jenkins home page, select **“New Item”.** Refer Fig 5.0

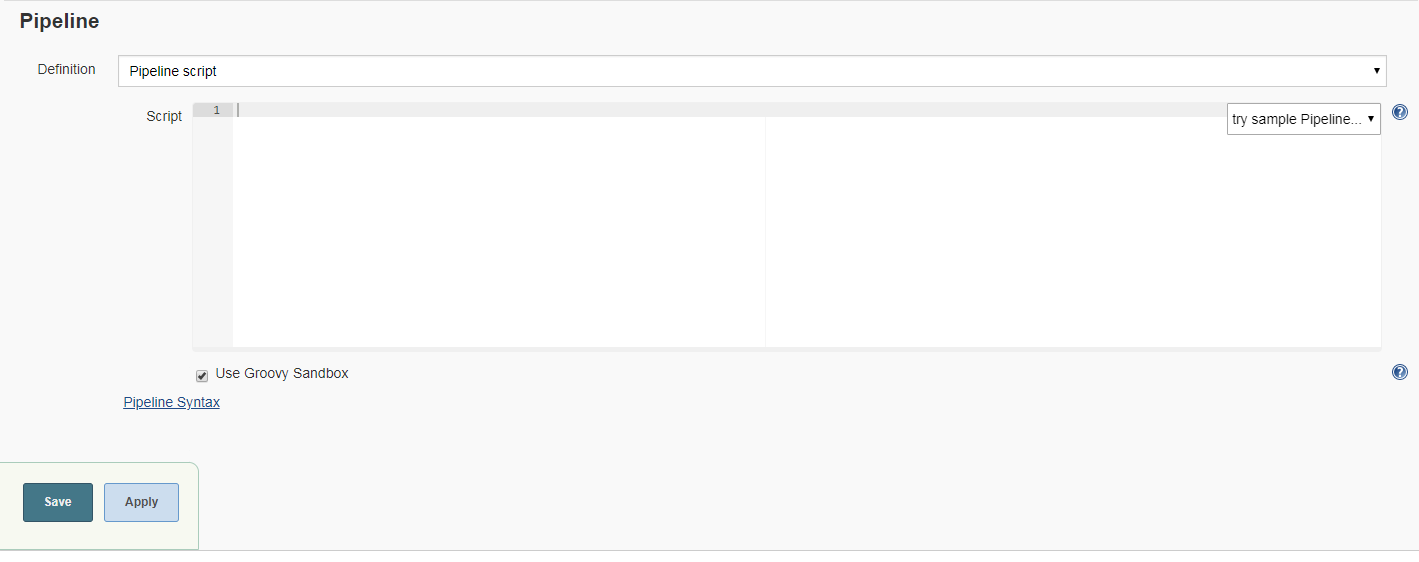
[Fig 5.0]

2. Enter the Project Name, Select Pipeline as the category and create the project. Refer Fig 5.1



[Fig 5.1]

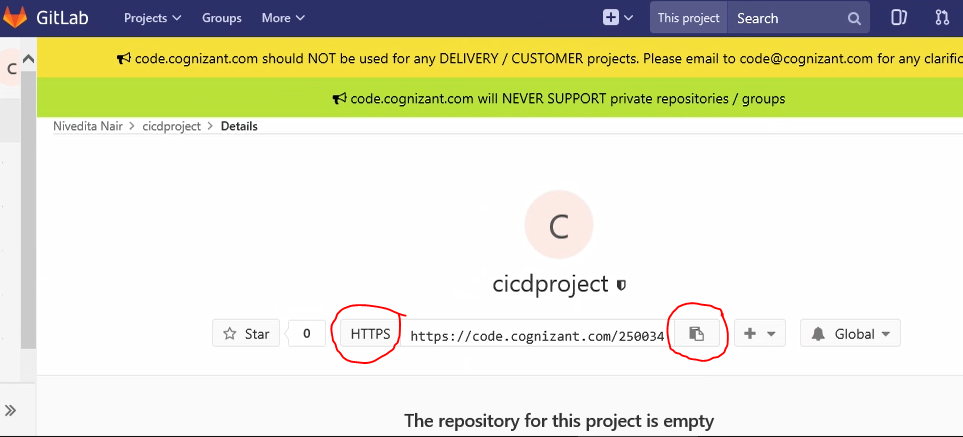
3. Navigate to pipeline script. Refer fig 5.2



[Fig 5.2]

Where the Pipeline syntax link is used for generating the sample script template.

4. As the first step the project has to be checked out from the SCM.ie the files submitted to the GIT repository has to be checked out first. Get the git repository path from GitLab as shown in Fig 5.3



[Fig 5.3]

The git path is used in this guided exercise pipeline script is **https://github.com/dsrcgithubtest/cicdproject.git**

**Step 2: Create a Pipeline project in Jenkins.**

1. Add the following pipeline script to check out the project from Git. Refer Fig 5.4

pipeline {

agent any

tools {

maven 'Maven35'

}

stages {

stage('Git checkout') {

steps {

git 'https://code.cognizant.com/250034/testing1.git'

}

}

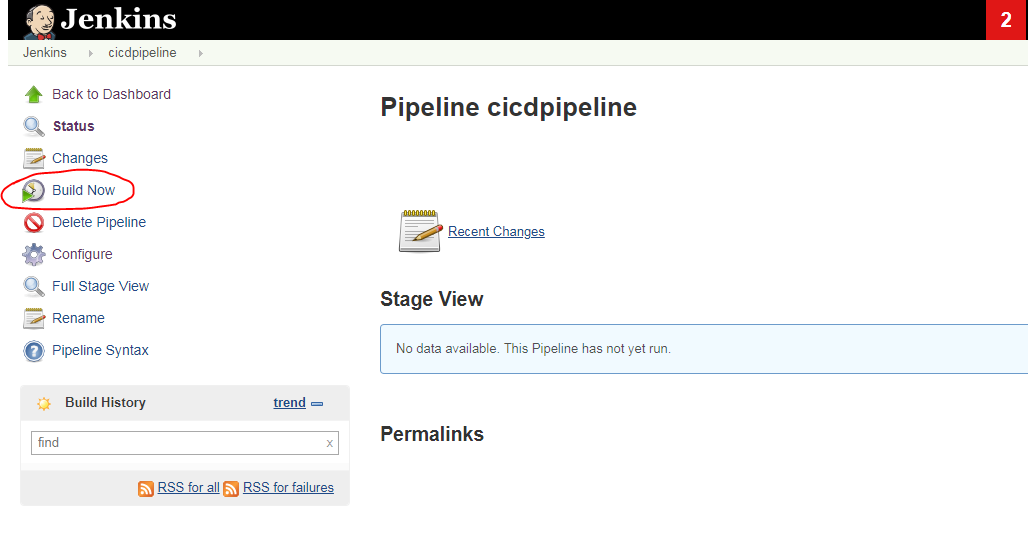
}

}



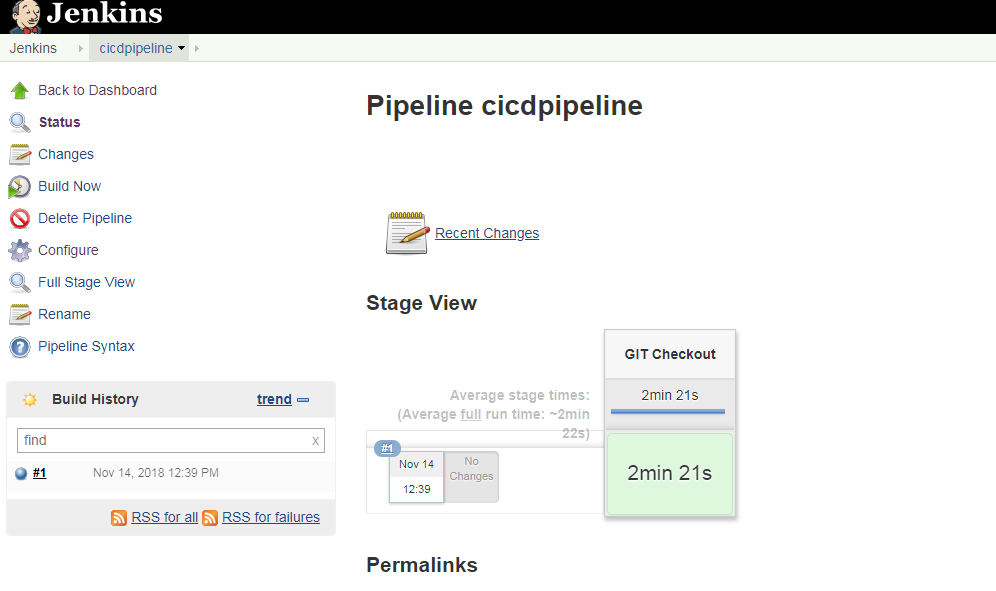
[Fig 5.4]

Apply and Save the script and build the stage as follows. Refer Fig 5.5



[Fig 5.5]

Jenkins will build the stage and the status can be viewed on the stage view .Ref Fig 5.6



[Fig 5.6]

2. Add the following two stages to the pipeline script to build and test the project. Refer Fig 5.7. Click on **Configure** to edit the pipeline script.

pipeline {

agent any

tools {

maven 'Maven35'

}

stages {

stage('Git checkout') {

steps {

git 'https://code.cognizant.com/250034/testing1.git'

}

}

stage('Maven package') {

steps {

bat 'mvn clean install'

}

}

stage('Test') {

steps {

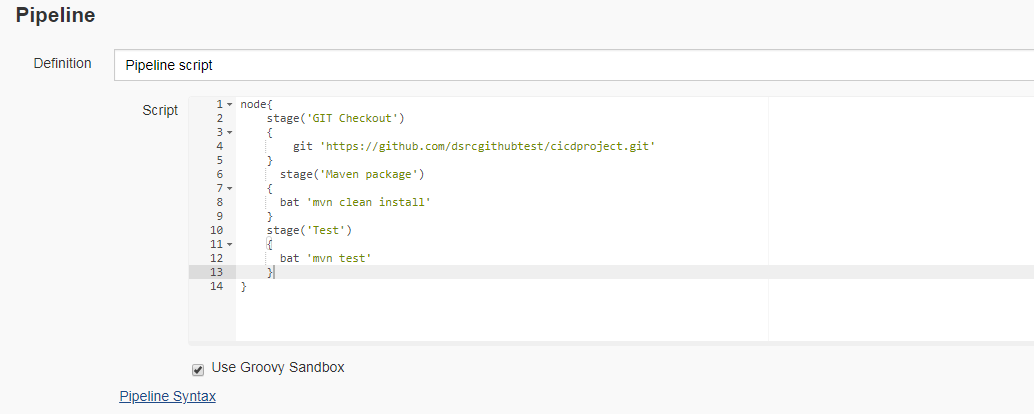
bat 'mvn test'

}

}

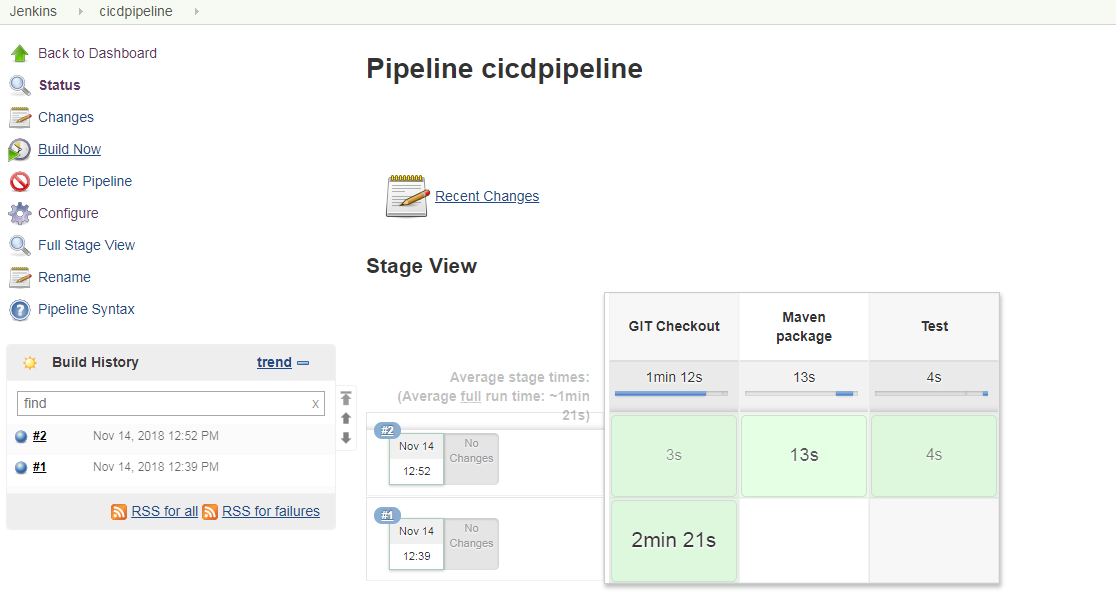
}

}



[Fig 5.7]

Build the pipeline and verify the stage view. Refer Fig 5.8



[Fig 5.8]

3. Add the stage to do a SonarQube analysis on the code. Use the following script to add sonar stage. Refer Fig 5.9

pipeline {

agent any

tools {

maven 'Maven35'

}

stages {

stage('Git checkout') {

steps {

git 'https://code.cognizant.com/250034/testing1.git'

}

}

stage('Maven package') {

steps {

bat 'mvn clean install'

}

}

stage('Test') {

steps {

bat 'mvn test'

}

}

stage('Sonar Analysis') {

steps {

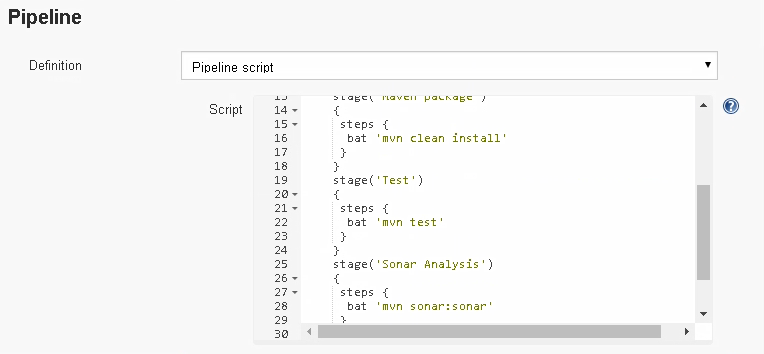
bat 'mvn sonar:sonar'

}

}

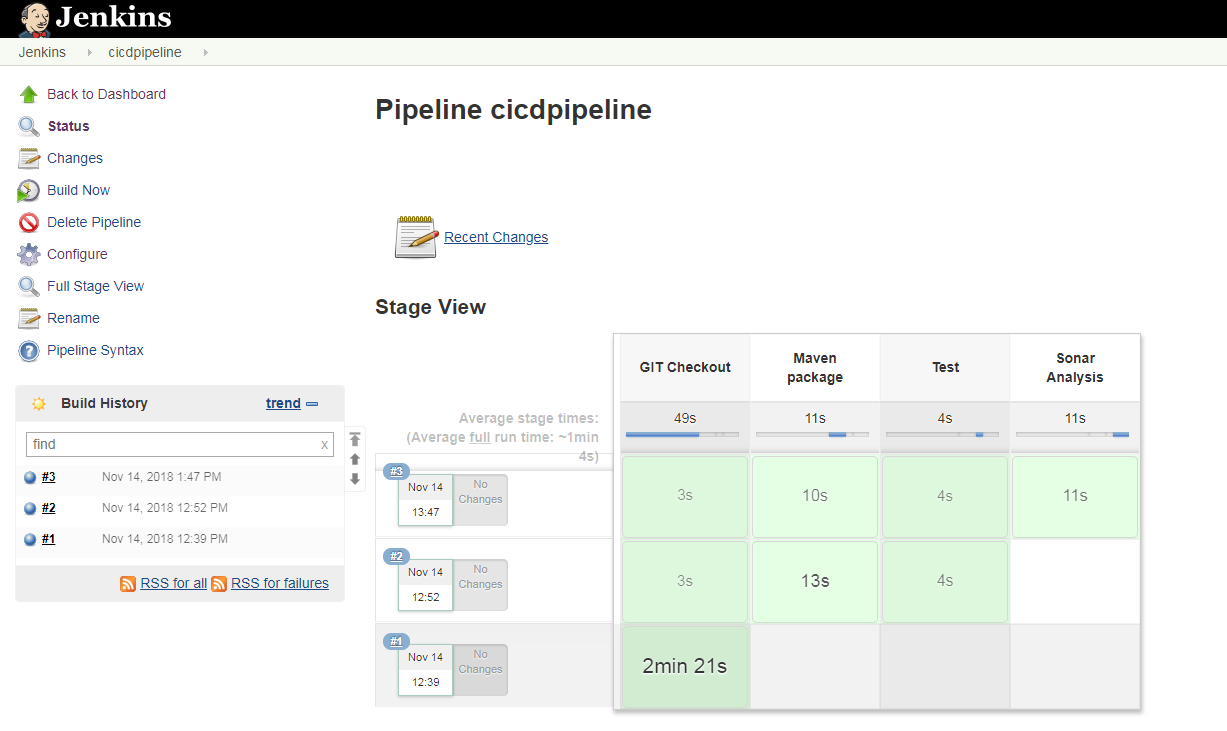
}

}



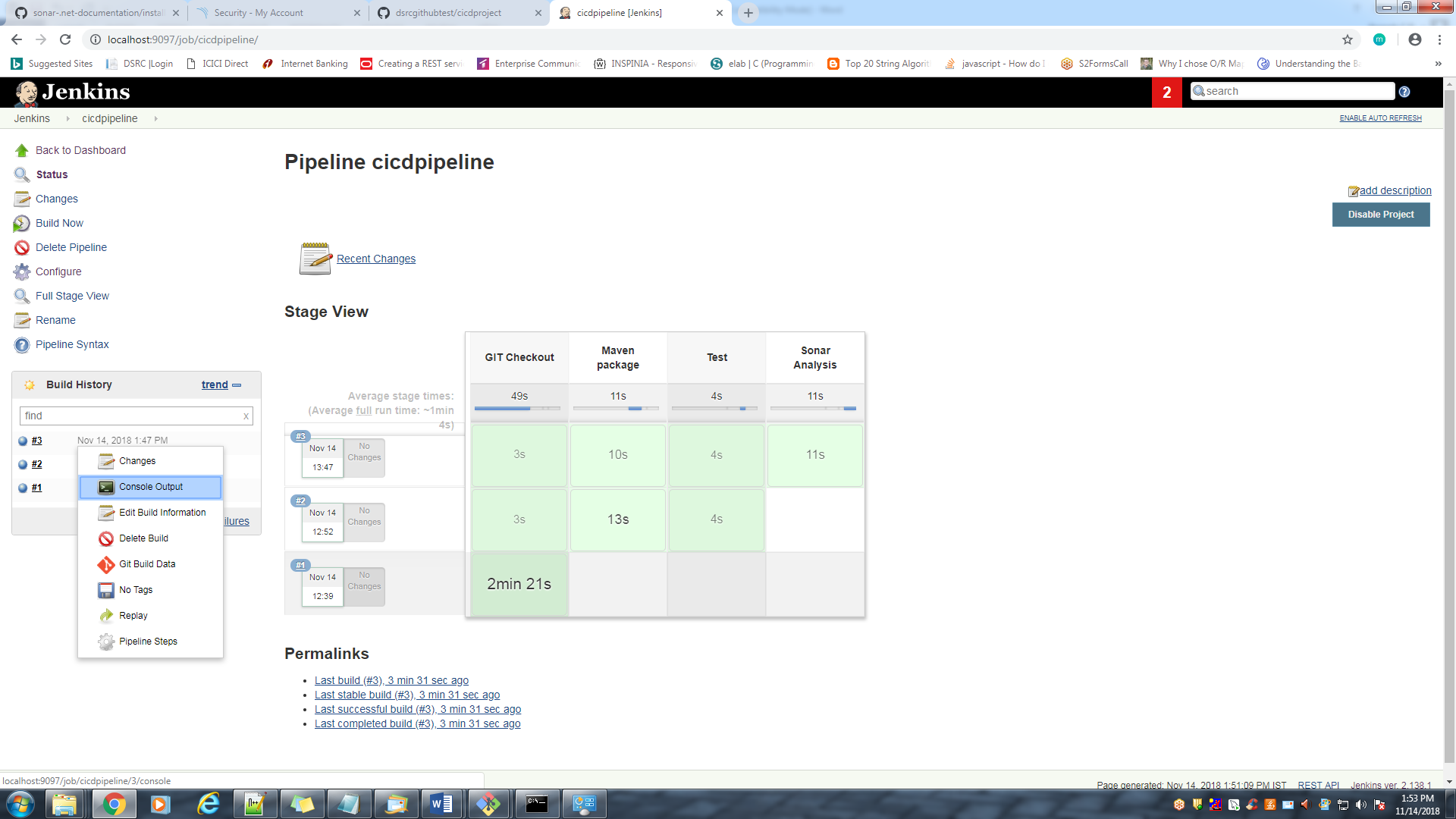
[Fig 5.9]

Build the pipeline and verify the stage view. Refer Fig 6.0



[Fig 6.0]

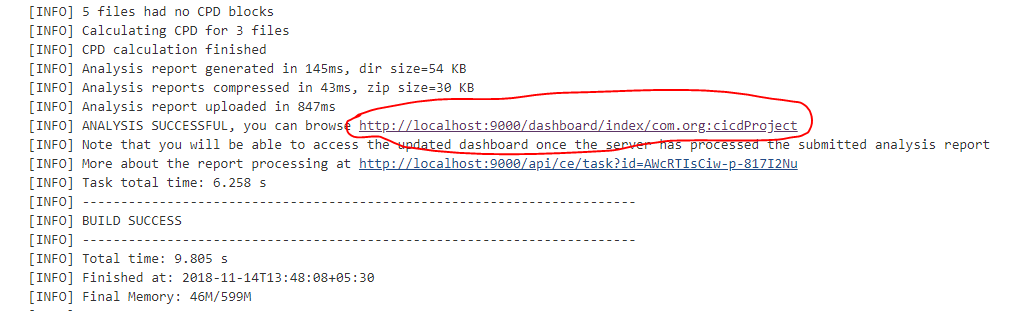
4. To view the sonar analysis report from Jenkins, select the latest build from build history, expand the down arrow and select console output. Refer Fig 6.1



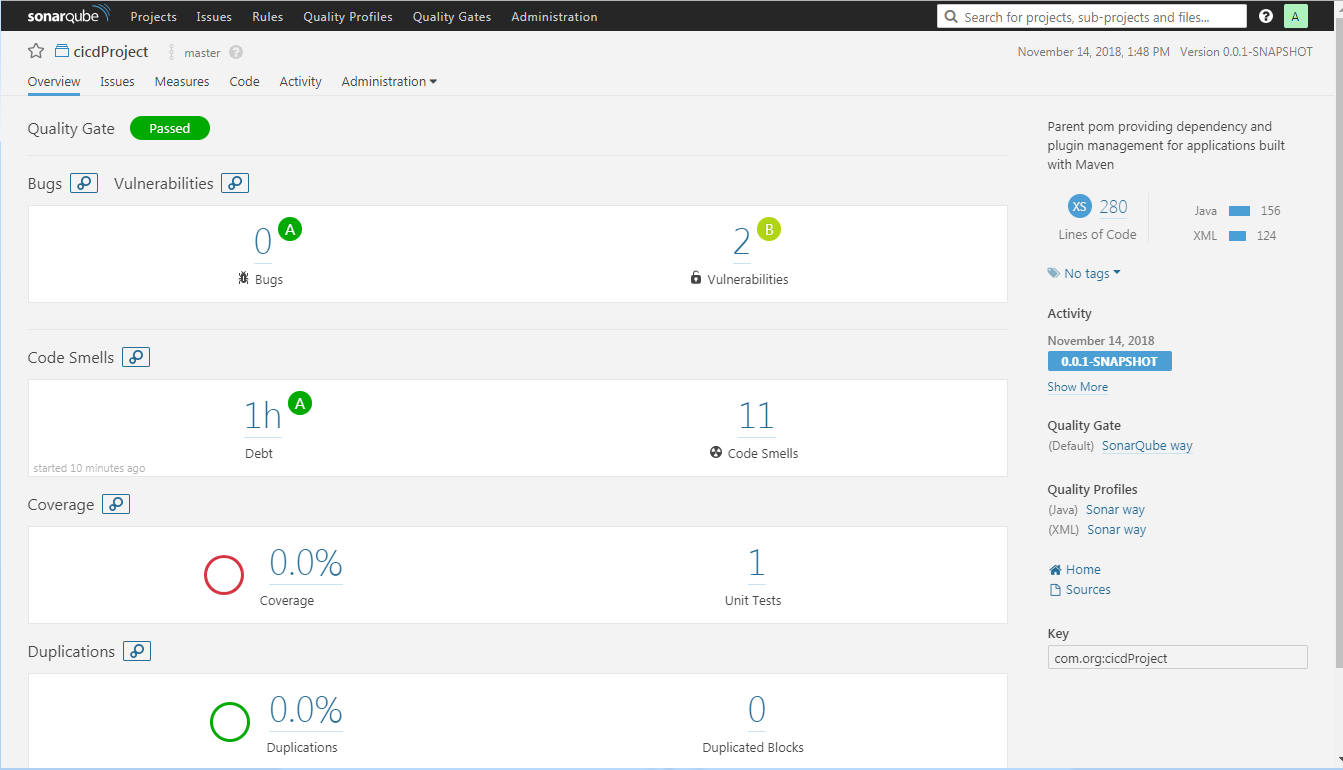
[Fig 6.1]

Navigate to Sonar analysis console view and select the sonar link to open the report.

Refer Fig 6.2 and 6.3



[Fig 6.2]



[Fig 6.3]

**Summary:**

You have learnt to create a Jenkins Pipeline in DevOps Environment.