NAAN MUDHALVAN PROJECT REPORT

ADHIPARASAKTHI COLLEGE OF ENGINEERING
(NAAC ACCREDITED)
G.B NAGAR, KALAVAI.

SB8050-DEVOPS(2024-2025)

Certified that this project report **"SB8050-DEVOPS CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY"** is the bonafide work of _____ who carried out the NAAN MUDHALVAN project.

Submitted for the project and viva-voce held on_29-11-2024.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project focuses on the integration of SonarQube, Jenkins, and Maven within a DevOps pipeline to enhance the software development lifecycle by automating build processes, ensuring code quality, and enabling continuous integration and delivery (CI/CD). SonarQube, an open-source platform, is used for static code analysis to detect bugs, vulnerabilities, and code smells, helping to maintain a high standard of code quality. Jenkins, an automation server, orchestrates the CI/CD pipeline by automating build, test, and deployment processes, ensuring efficient software delivery. Maven is employed as the build automation tool to manage dependencies, compile, test, and package the application. By combining these tools, the project aims to streamline development, improve code quality, and reduce manual errors, providing a scalable solution for modern software projects. The integration of these tools allows for automated code inspection, continuous feedback, and faster release cycles, ultimately leading to more reliable and efficient software.

Continuous Integration and Continuous Deployment (CI/CD) are pivotal practices in modern software development, enabling rapid and reliable software delivery through automation. This project focuses on the implementation of a CI/CD pipeline to streamline the development, testing, and deployment processes. **Continuous Integration (CI)** involves automatically integrating code changes into a shared repository, ensuring that the software is always in a buildable state. The CI process is complemented by automated unit testing and static code analysis, which detect errors early, improve code quality, and reduce the risk of defects. **Continuous Deployment (CD)** goes a step further, automatically deploying the integrated code to production or staging environments after successful testing, ensuring faster release cycles and immediate delivery of features and bug fixes. By integrating tools such as **Jenkins** for automation, **SonarQube** for code quality checks, and **Maven** for build management, the CI/CD pipeline is designed to improve collaboration, reduce manual intervention, and enhance software reliability. DevOps is a collaborative software development approach that integrates development (Dev) and operations (Ops) teams to streamline the software delivery process. This methodology emphasizes a culture of collaboration, communication, and continuous improvement among all stakeholders involved in the software development lifecycle. DevOps aims to achieve faster and more reliable software delivery.

TABLE OF CONTENTS

TOOLS THAT WOULD BE USED

- Eclipse – Integrated development environment
- JUnit – Unit testing of code
- Jenkins – Continuous integration server
- Git- Source code management
- Jenkins – Build Automation
- SonarQube- Source code quality management

EXERCISE:1 TESTING THE LAB ENVIRONMENT

OBJECTIVE

To test the lab environment by using the various bat files of the software.

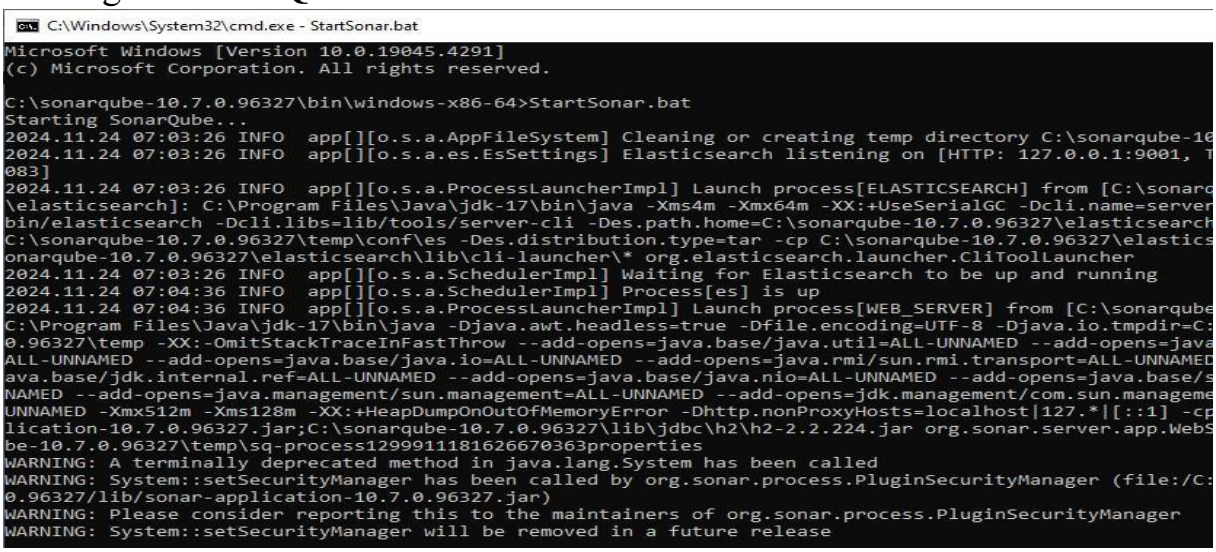
PROCEDURE

Step-(1)-open the edit environment variables menu from the windows search. Then add the following varibles in the %Path% of your environment variables.

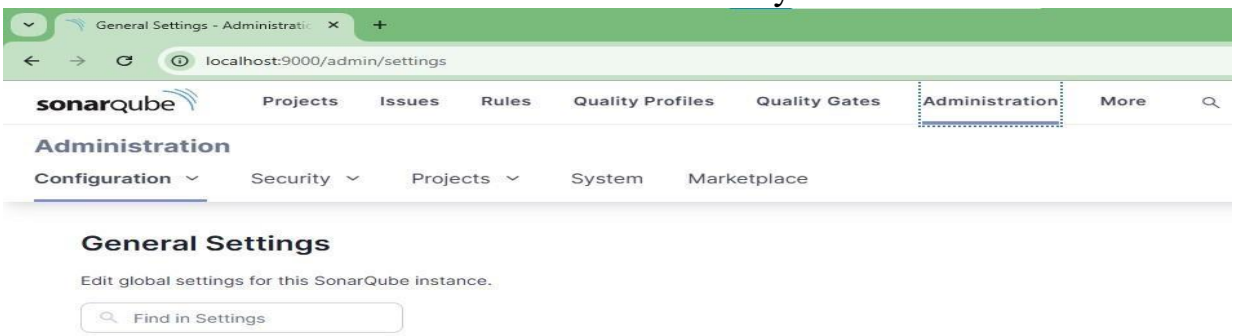
Software	Path to the bin directory
Java	C:\Program Files\Java\jdk-1.8\bin
SonarQube	C:\sonarqube-10.7.0.96327\bin\windows-x86-64
Jenkins	C:\Program Files\Java\jdk-1.8\bin
Tomcat	C:\Program Files\Apache Software Foundation\Tomcat 10.1\bin

Step-(2)-Start the all corresponding .bat files in the command line arguments.

Starting the **SonarQube.**

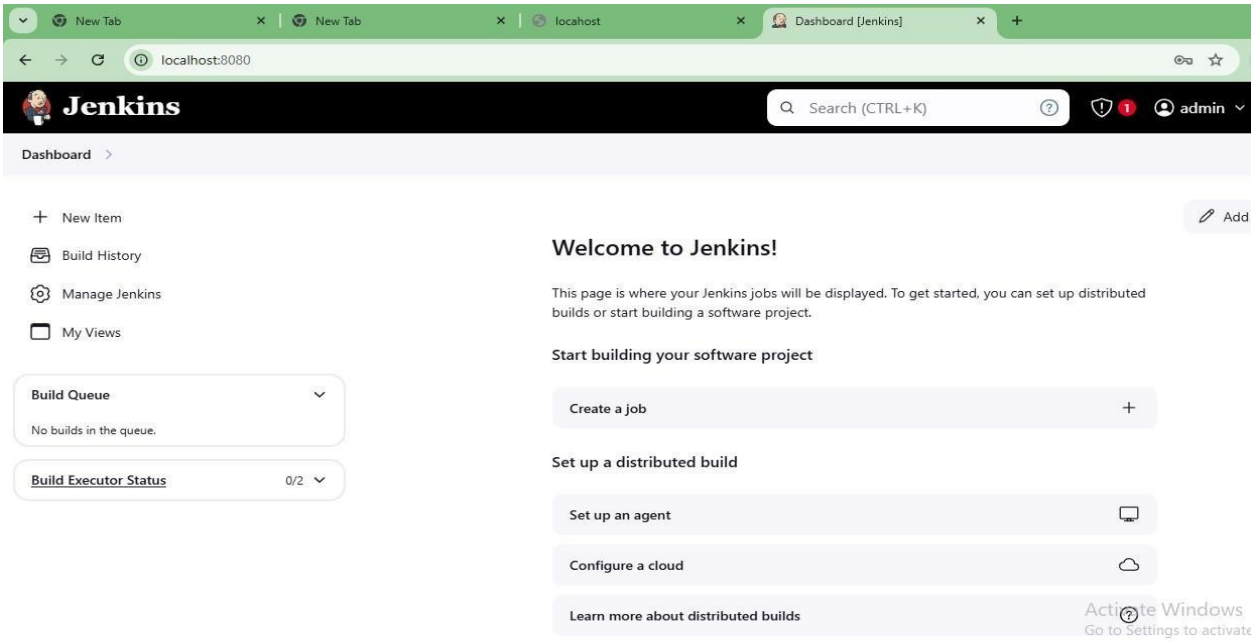


Check the Dashboard at the localhost:9000 in any browser.

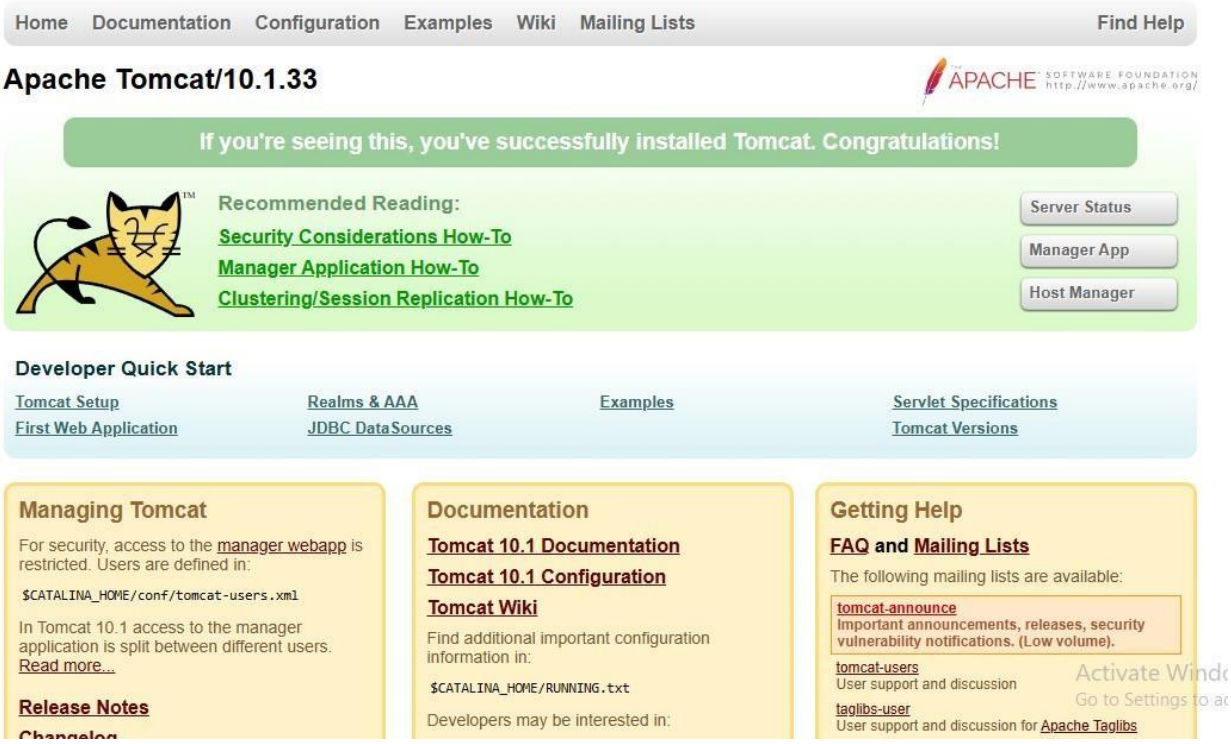


Starting the **Jenkins.**

After the Installation of the jenkins.Check the jenkins dashboard at localhost:8064



Starting the **Tomcat.** after the Installation of the tomcat.check the Tomcat dashboard at localhost:8080



CONCLUSION:

The lab environment for the SonarQube,Jenkins and Tomcat has been Tested.

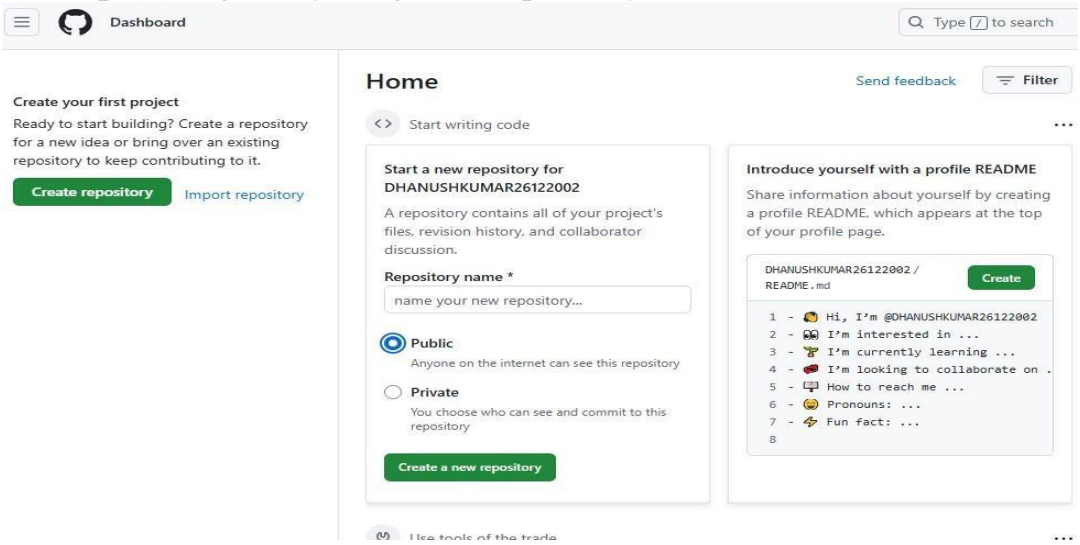
EXERCISE:2 **GIT OPERATIONS**

OBJECTIVE

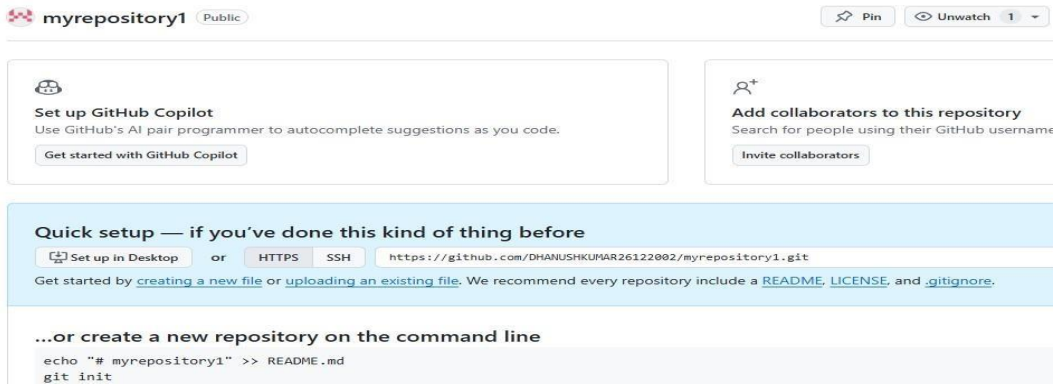
To perform the Git operations from the eclipse using the eclipse Git plugin(Egit).

PROCEDURE

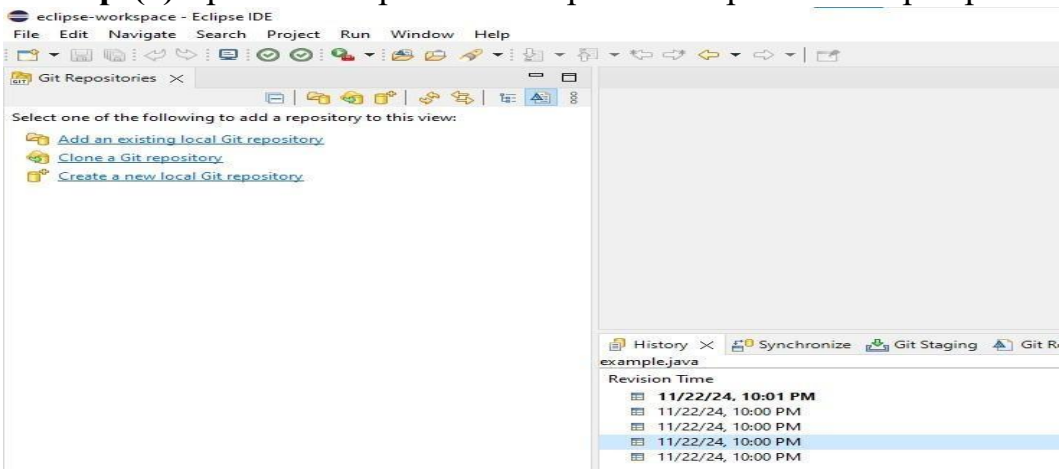
Step-(1)-login to your github repository.



Step-(2)-create the new repository for performing the Git operations.

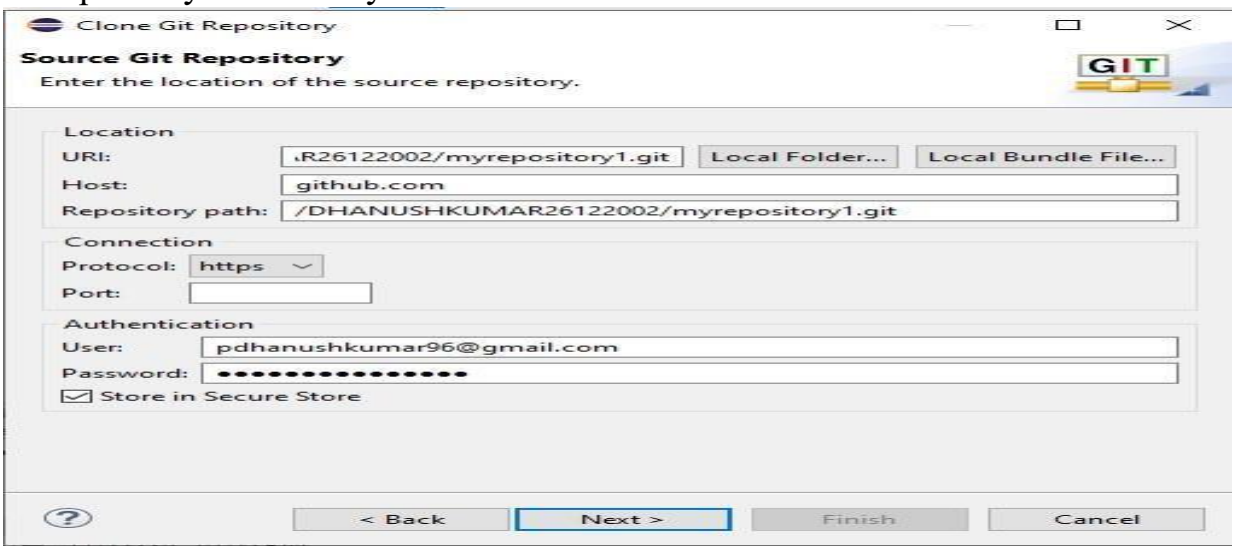


Step-(3)-open the eclipse in workspace and open Github prespective.



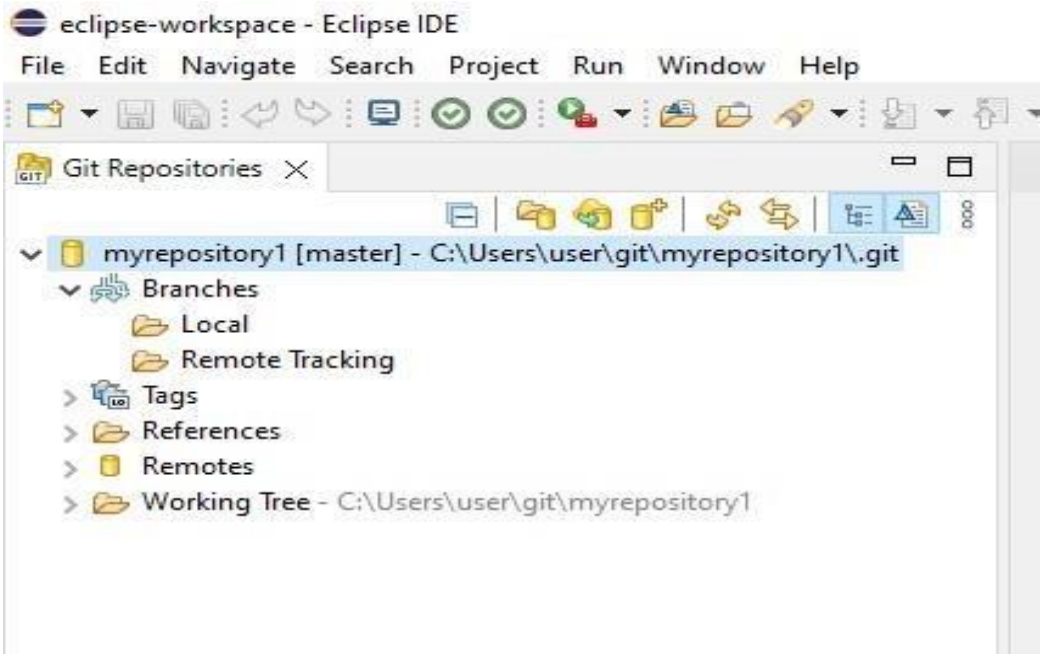
CLONING OPERATION:

click on the clone a Git respository and copy the URL of your repository.then enter your authentication details.



Click on next.

And the repository is cloned successfully in the eclipse.

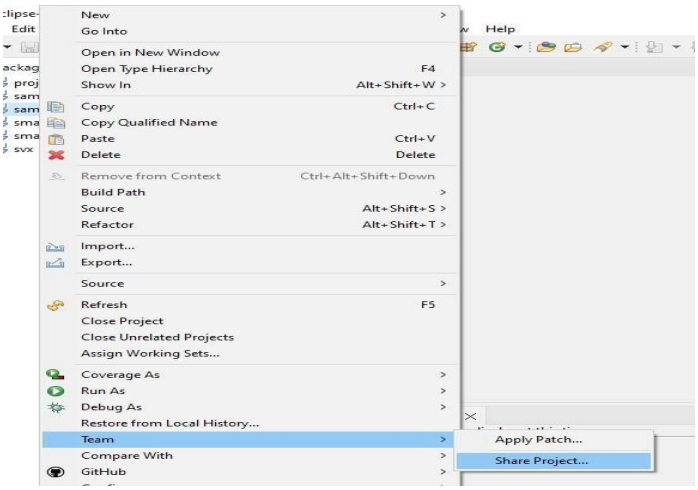


Thus the cloning operation was successfully executed in the eclipse idle through EGit.

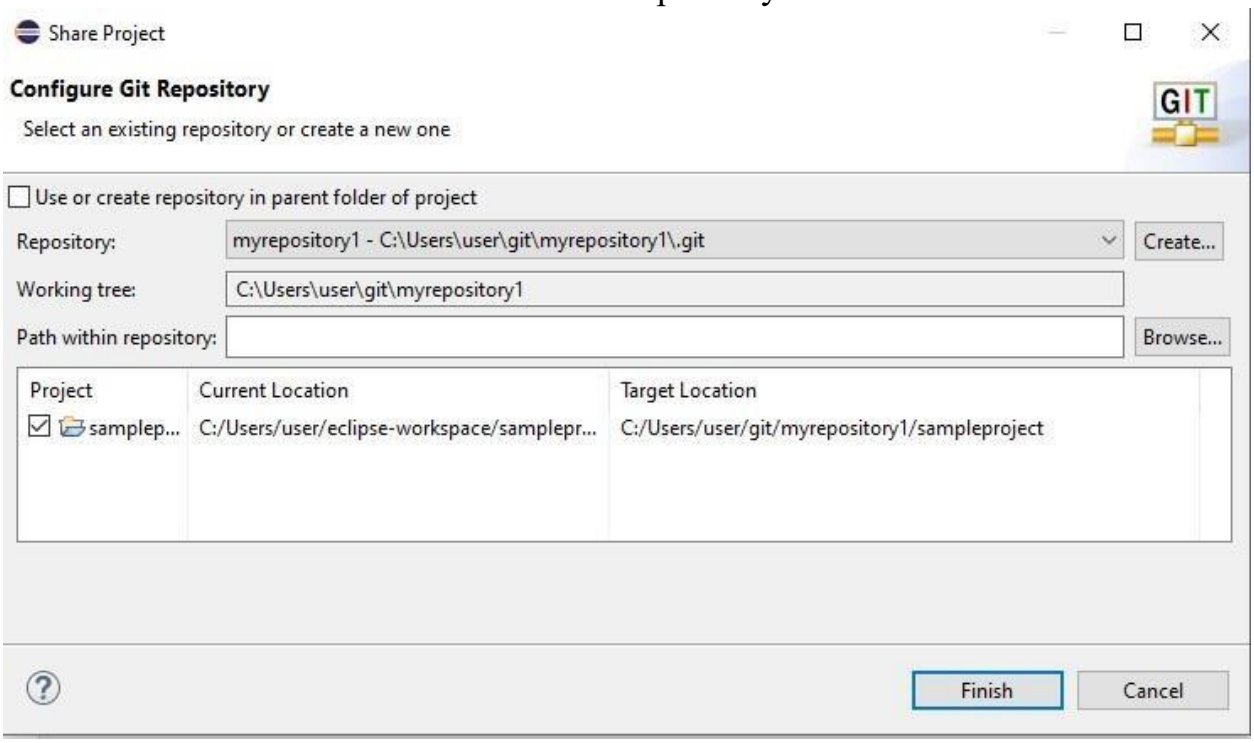
PUSH OPERATION:

The push operation involves the pushing of the local project into the github repo.

Right click on the project that you need to push into repository and then click on the team option in that menu.
Under click on the share.
team→share



Hit on the enter button. then select the repository.

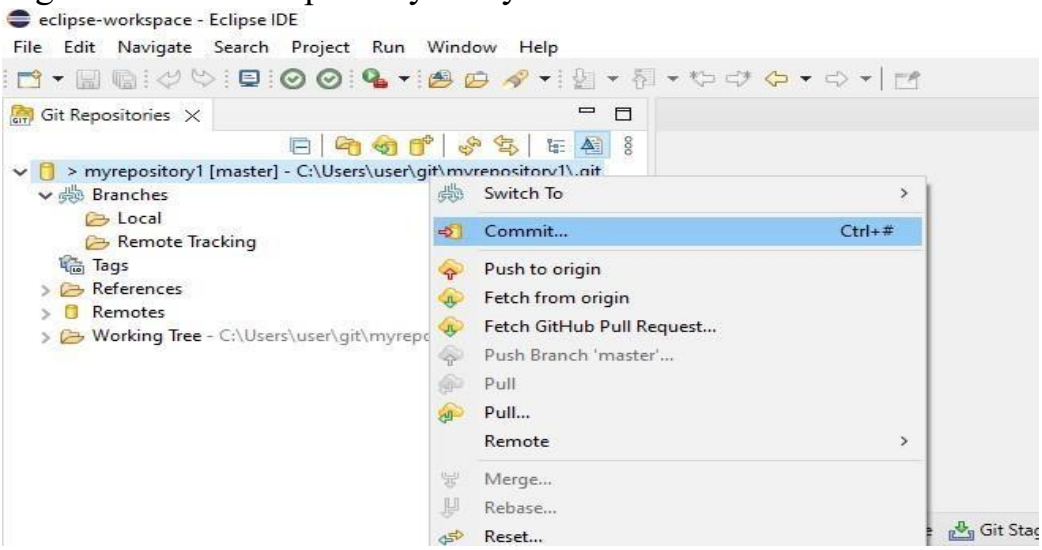


Click on finish.now the project state was changed into the staged state from the master state.

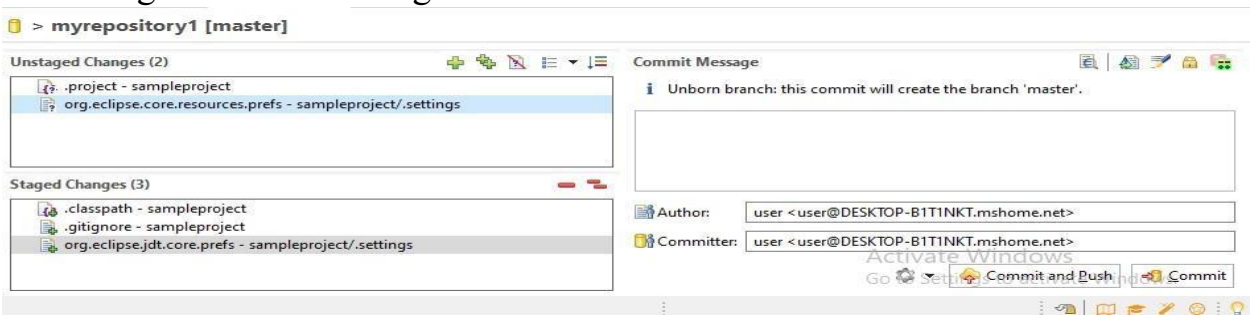
<u>COMMIT OPERATION:</u>

The commit operation is used to commit changes in the repository.

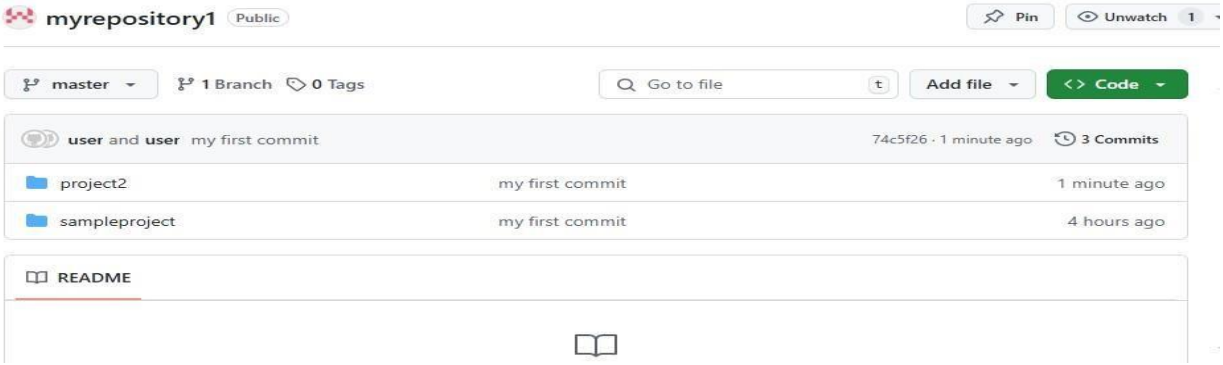
Right click on the repository that you need to commit.



Click on the commit option. before move changes from the unstaged state to the staged state.



then click on the commit button.now it asks the username and password enter your username and password ,then click on the login. the project is now pushed into the repository and reflection in the github is shown below.



CONCLUSION:

Thus the various Git commands are executed from the eclipse using EGit plugin and output was verified.

EXERCISE:3 CREATING THE PROJECT IN SONARQUBE

OBJECTIVE

To create the simple project in the sonarqube.

PROCEDURE

Step-(1)-Download the sonarqube from the official website www.sonarsource.com.

Step-(2)-Extract the zip folder into destination directory.

Step-(3)-set the path for the sonarqube as C:\sonarqube-10.7.0.96327\bin\windows-x86-64 in your environmental varibles.

Step-(4)-To start the sonarqube open the cmd and type command **StartSonar.bat**.

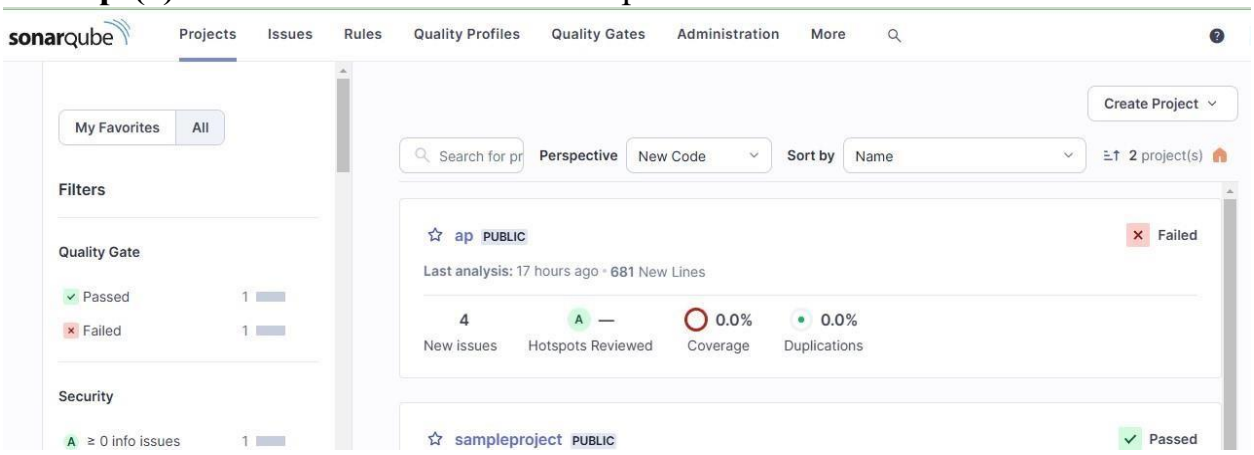
```
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\sonarqube-10.7.0.96327\bin\windows-x86-64>StartSonar.bat
Starting SonarQube...
2024.11.24 07:03:26 INFO  app[][o.s.a.AppFileSystem] Cleaning or creating temp di
2024.11.24 07:03:26 INFO  app[][o.s.a.es.EsSettings] Elasticsearch listening on [
2024.11.24 07:03:26 INFO  app[][o.s.a.ProcessLauncherImpl] Launch process[ELASTIC
n\java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=./bin/elast
icsearch -Des.path.conf=C:\sonarqube-10.7.0.96327\temp\conf\es -Des.distribution.
asticsearch\lib\cli-launcher\* org.elasticsearch.launcher.CliToolLauncher
2024.11.24 07:03:26 INFO  app[][o.s.a.SchedulerImpl] Waiting for Elasticsearch to
2024.11.24 07:04:36 INFO  app[][o.s.a.SchedulerImpl] Process[es] is up
2024.11.24 07:04:36 INFO  app[][o.s.a.ProcessLauncherImpl] Launch process[WEB_SER
.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\sonarqube-10.7.0.96327\t
ens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --a
f=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/su
=jdk.management/com.sun.management.internal=ALL-UNNAMED -Xmx512m -Xms128m -XX:+He
lication-10.7.0.96327.jar;C:\sonarqube-10.7.0.96327\lib\jdbc\h2\h2-2.2.224.jar or
63properties
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by org.sonar.process.PluginSe
WARNING: Please consider reporting this to the maintainers of org.sonar.process.P
WARNING: System::setSecurityManager will be removed in a future release
2024.11.24 07:07:20 INFO  app[][o.s.a.SchedulerImpl] Process[web] is up
```

Step-(5)-visit the sonarQube dash running in the port:localhost:9000.open the http://localhost:9000 in any browser.

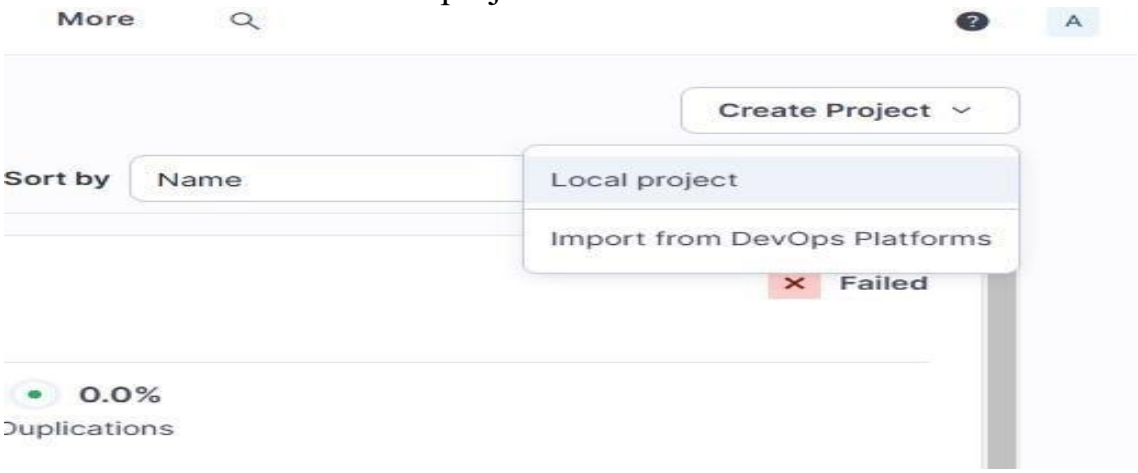
The admin login page is opened as shown below.enter the user name : admin and password : admin.

Then click on the log in button.

Step-(6)-Then the dashboard will be opened as follows.

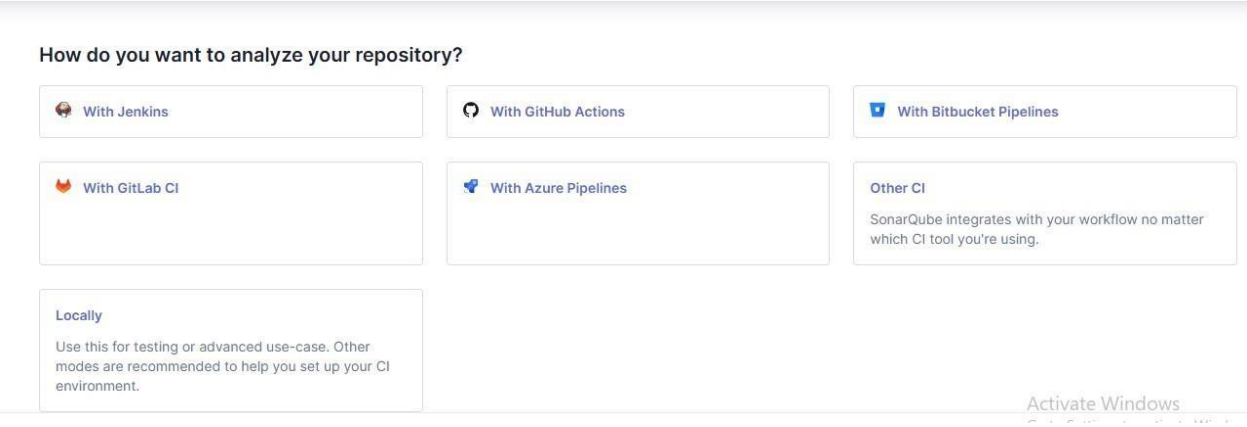


Step-(7)-Then in the dashboard click on the Create project button.under this click on the localproject.



Step-(8)-Enter the name of the project and the branch of the project.then click on the next.

Step-(9)-Then choose your analyze method has locally



Now the project was created successfully.

CONCLUSION

The project was created locally on the sonarqube and the corresponding token was generated successfully.

EXERCISE:4 USING SONAR-RUNNER WITH SONARQUBE

Step-(1)-open the project that you have created on the previous experiment and click on the configure analysis.

	Name	Visibility	Key	Last Analysis	Actions
☐	ap	Public	ap	Nov 23, 2024	⋮
☐	sampleproject	Public	sampleproject	Nov 23, 2024	⋮
☐	sampleproject1	Public	sampleproject1	—	⋮

Step-(2)- generate the token of your project.

Analyze "sampleproject1": **sqp_66f6b6e8097b4ef298b7334f29fdc8cf9c51bf3a**

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

Continue

Click on continue.

Step-(3)-choose the option for analyze,Os and then copy the command which is used to run the static analysis in the Sonar-Scanner.

2 **Run analysis on your project**

What option best describes your project?

Maven | Gradle | .NET | Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux | Windows | macOS

Download and unzip the Scanner for Windows

Visit the official documentation of the Scanner to download the latest version, and add the `bin` directory to the `%PATH%` environment variable

Execute the Scanner

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner.bat -D"sonar.projectKey=sampleproject1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -
D"sonar.token=sqp_9a01e5093725a3b0da83b465f83b540d0eb70682"
```

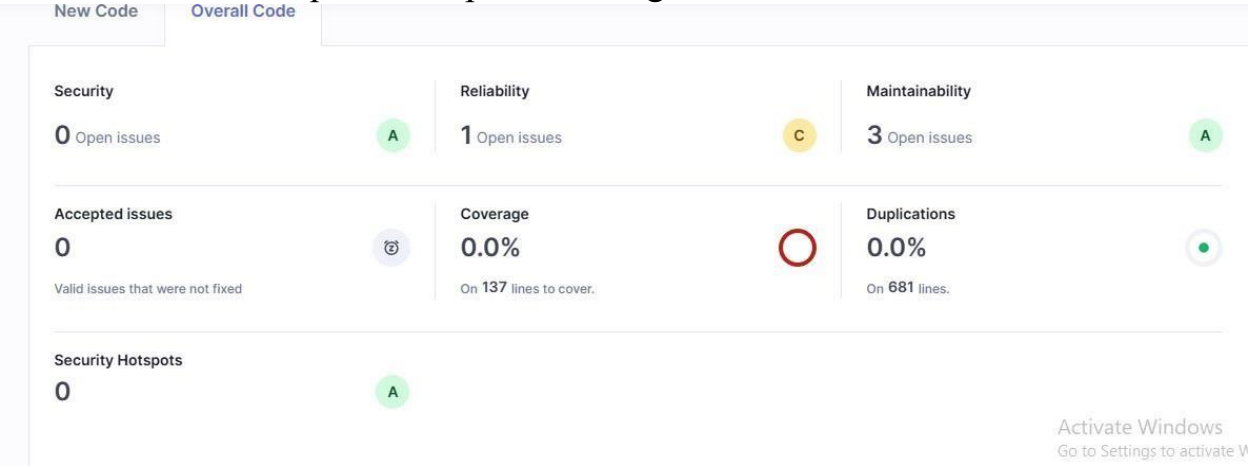
Copy

Step-(4)-start the sonar-scanner in the root directory of the project. Open the cmd in the project root directory,then run this command in cmd.

sonar-scanner.bat -D"sonar.projectKey=sampleproject1" -D"sonar.sources=." D"sonar.host.url=http://localhost:9000" -D"sonar.token=sqp_9a01e5093725a3b0da83b465f83b540d0eb70682"

```
C:\web-scraping-with-python-master>sonar-scanner.bat -D"sonar.projectKey=sampleproject1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.token=sqp
_9a01e5093725a3b0da83b465f83b540d0eb70682"
15:32:52.355 INFO  Scanner configuration file: C:\sonar-scanner-6.2.1.4610-windows-x64\bin\..\conf\sonar-scanner.properties
15:32:52.370 INFO  Project root configuration file: NONE
15:32:52.427 INFO  SonarScanner CLI 6.2.1.4610
15:32:52.439 INFO  Java 17.0.12 Eclipse Adoptium (64-bit)
15:32:52.443 INFO  Windows 10 10.0 amd64
15:32:52.510 INFO  User cache: C:\Users\user\.sonar\cache
15:32:53.920 INFO  JRE provisioning: os[windows], arch[amd64]
15:32:54.305 INFO  Communicating with SonarQube Server 10.7.0.96327
15:32:55.455 INFO  Starting SonarScanner Engine...
15:32:55.457 INFO  Java 17.0.11 Eclipse Adoptium (64-bit)
15:32:57.768 INFO  Load global settings
15:32:58.020 INFO  Load global settings (done) | time=259ms
15:32:58.032 INFO  Server id: 147B411E-AZNZoWGX8va5qaIXy7gK
15:32:58.069 INFO  Loading required plugins
15:32:58.124 INFO  Load plugins index
15:32:58.126 INFO  Load plugins index (done) | time=44ms
15:32:58.128 INFO  Load/download plugins
15:32:58.255 INFO  Load/download plugins (done) | time=139ms
15:32:59.268 INFO  Process project properties
15:32:59.299 INFO  Process project properties (done) | time=33ms
15:32:59.334 INFO  Project key: sampleproject1
15:32:59.385 INFO  Base dir: C:\web-scraping-with-python-master
15:32:59.638 INFO  Working dir: C:\web-scraping-with-python-master\.scannerwork
15:32:59.643 INFO  Load project settings for component key: 'sampleproject1'
15:32:59.644 INFO  Load project settings for component key: 'sampleproject1' (done) | time=189ms
15:32:59.744 INFO  Load quality profiles
15:33:00.559 INFO  Load quality profiles (done) | time=815ms
15:33:00.746 WARN  SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
15:33:00.779 INFO  Load active rules
15:33:21.352 INFO  Load active rules (done) | time=20571ms
15:33:21.411 INFO  Load analysis cache
15:33:21.417 INFO  Load analysis cache (404) | time=42ms
15:33:21.692 WARN  The property 'sonar.login' is deprecated and will be removed in the future. Please use the 'sonar.token' property instead when passing a token.
```

Step-(5)-once the execution is successful then open the dashboard of the sonarqube the report will be generated like this as shown below.



CONCLUSION

The project was created locally on the sonarqube and it is executed generated successfully.

EXERCISE:5 CREATING A LOCAL REPOSITORY IN ARTIFACTORY

OBJECTIVE:
Understand creation of local repository in Artifactory

Step 1: Go to Artifactory URL and login with credentials: admin: Password!!

Step 2: Go to Administration Repositories Repositories> Add repositories ->

Local Repository

Step 2: Select the package type as Maven

Step 4: To add the repository key, go to pom.xml and copy the <name> tag value (Calc Dev Snapshot) as shown in the screenshot given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>

   <!-- Basic project information -->
   <groupId>com.example</groupId>
   <artifactId>my-app</artifactId>
   <version>1.0.0</version>
   <packaging>jar</packaging> <!-- or 'war' depending on the type of project
-->

   <name>My Application</name>
   <url>http://www.example.com</url>

   <!-- Parent POM reference (optional) -->
   <parent>
      <groupId>org.springframework</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>2.7.0</version>
  </parent>
  <!-- Dependencies -->
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
      <ve
```

Step 5: Click on Create Local Repository.

Step 6: You can view the binaries stored in the artifactory under Application-> Artifactory Packages

CONCLUSION:
local repository is created in artifactory in this exercise.

EXERCISE:6 BUILD AUTOMATION USING MAVEN

OBJECTIVE:

To build the automation of the projects using the maven.

PROCEDURE:

Step-(1)- Install Apache Maven

1. Download the latest version of **Apache Maven** from the [official website](#).
2. Extract the archive to a directory on your computer (e.g., C:\apache-maven).
3. Set the MAVEN_HOME environment variable:
   - Open **System Properties** > **Environment Variables**.
   - Under **System variables**, click **New** and add the variable:
     - **Variable name**: MAVEN_HOME
     - **Variable value**: The path to your Maven installation (e.g., C:\apache-maven).
   - Edit the Path variable in **System variables** and add %MAVEN_HOME%\bin to the list.
4. Verify Maven installation by opening **Command Prompt** and typing. **mvn –version**

```
C:\Users\user>mvn -version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\apache-maven-3.9.9
Java version: 17.0.13, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Common Maven Commands: The commands are executed in the root directory. **Compile the project**: mvn compile

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.15.0/plexu
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler/2.15.0/plexus-c
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler/2.15.0/plexus-co
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/13/plexus-13.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/13/plexus-13.pom (27 kB a
Downloading from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.9.3/junit-bom-5.9.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.9.3/junit-bom-5.9.3.pom (5.6 k
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.15.0/
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.15.0/p
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-xml/3.0.0/plexus-xml-3.0
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-xml/3.0.0/plexus-xml-3.0.
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.15.0/pl
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.15.0/ple
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compilers/2.15.0/plexus-
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compilers/2.15.0/plexus-c
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-java/1.2.0/plexus-java-1
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-java/1.2.0/plexus-java-1.
Downloading from central: https://repo.mavpn.apache.org/maven2/org/ow2/asm/asm/9.6/asm-9.0.jar
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.15.0/plex
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.15.0/
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-xml/3.0.0/plexus-xml-3.0
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.15.0/pl
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.15.0/p
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.15.0/plexu
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.15.0/ple
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-xml/3.0.0/plexus-xml-3.0.
Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/9.6/asm-9.6.jar (124 kB at 172 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding windows-1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target\classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  37.068 s
[INFO] Finished at: 2024-11-24T19:08:22+05:30
[INFO] ------------------------------------------------------------------------
```

Run unit tests:mvn test

```
C:\Users\user\my-app>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.example:my-app >------------------------
[INFO] Building my-app 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ my-app ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\user\my-app\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ my-app ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\user\my-app\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ my-app ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.110 s -- in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.631 s
[INFO] Finished at: 2024-11-24T19:11:05+05:30
[INFO] ------------------------------------------------------------------------
```

Package the project:mvn package

```
[INFO] Building jar: C:\Users\user\my-app\target\my-app-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  56.186 s
[INFO] Finished at: 2024-11-24T19:15:36+05:30
[INFO] ------------------------------------------------------------------------
```

CONCLUSION:

Thus the sample project was automated using the maven and commands were executed.

EXERCISE:7 INSTALLATION & CONFIGURATION OF JENKINS

OBJECTIVE

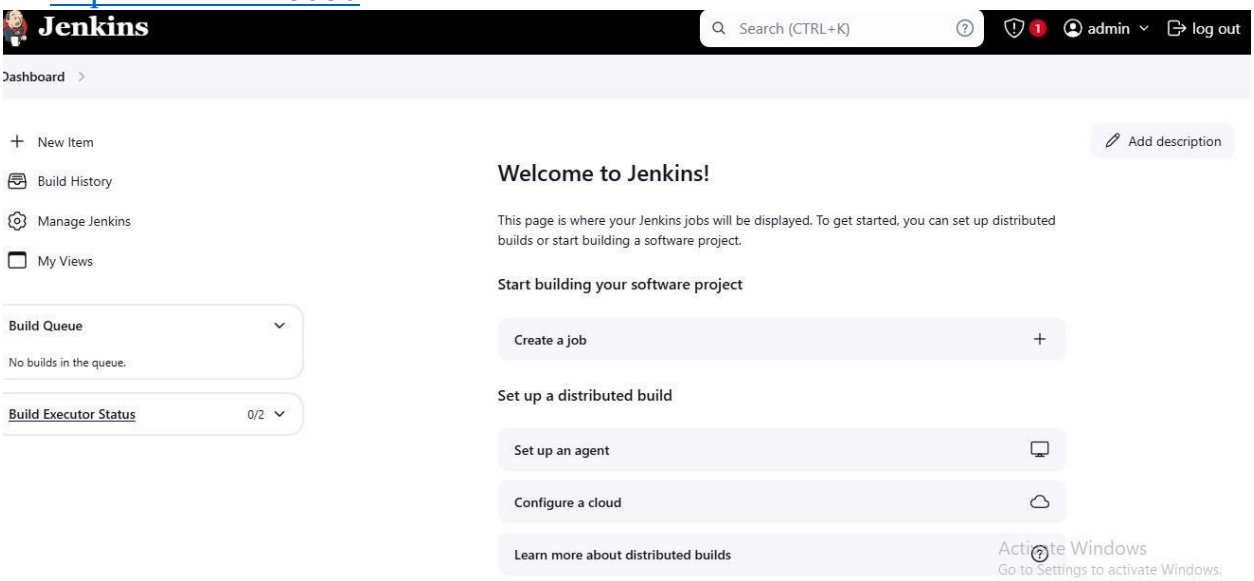
To install and configure the jenkins.

PROCEDURE

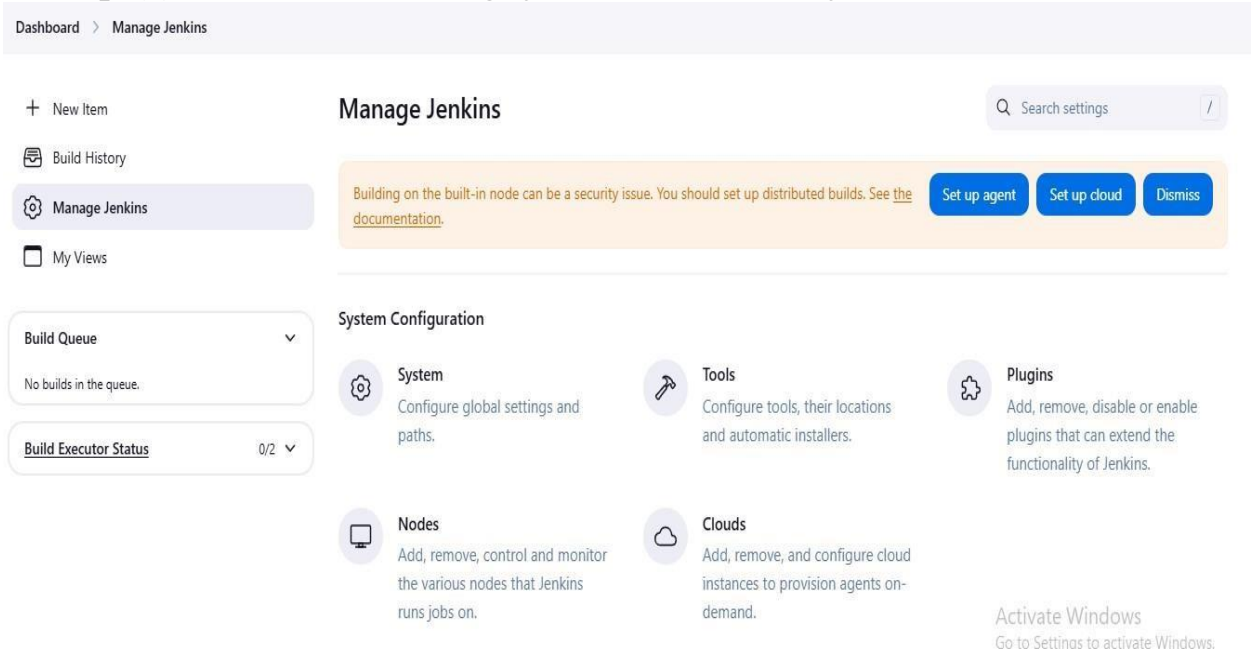
Step-(1)-Download the jenkins.exe from the official website jenkins.in

Step-(2)-Install the jenkins and set the port of the jenkins as 8064(TCP).

Step-(3)-login to the dashboard of the jenkins by using http://localhost:8080.



Step-(4)-Enter into the manage jenkins menu of the jenkins.



Step-(5)-Under click on the system configuration.click on the tools and configure the JDK as follows.

Step-(6)-also configure the maven in the same menu.

CONCLUSION:

Thus the jenkins was configured with the maven and JDK.

EXERCISE:8 INSTALLATION OF PLUGINS IN JENKINS

OBJECTIVE

To install the plugins in the jenkins.

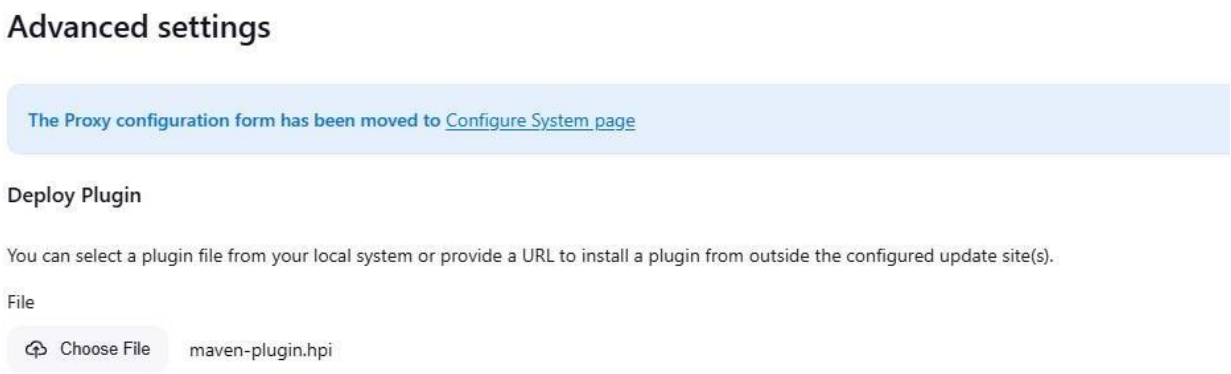
PROCEDURE Step-(1)- login to the dashboard of the jenkins by using http://localhost:8080.

Step-(2)-Download the plugin:

- Visit Jenkins Plugin Index and download the .hpi or .jpi file for the plugin.

Upload the plugin:

- Navigate to **Manage Jenkins** > **Manage Plugins** > **Advanced**.
- Click **Choose File** under **Upload Plugin**, select the .hpi file you downloaded, and click **Upload**.



Then click on deploy.

After the plugin was installed on your Jenkins.

By using the similar steps you can install any plugins.

CONCLUSION:

Thus the plugins are installed in the Jenkins.

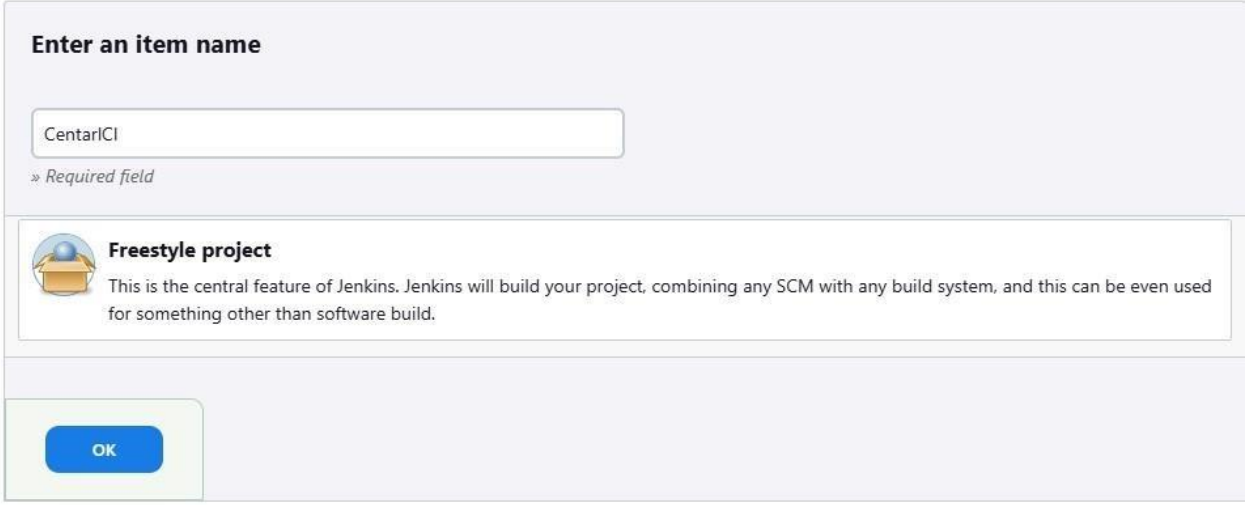
EXERCISE:9 CREATING CENTRAL PIPELINE

OBJECTIVE:

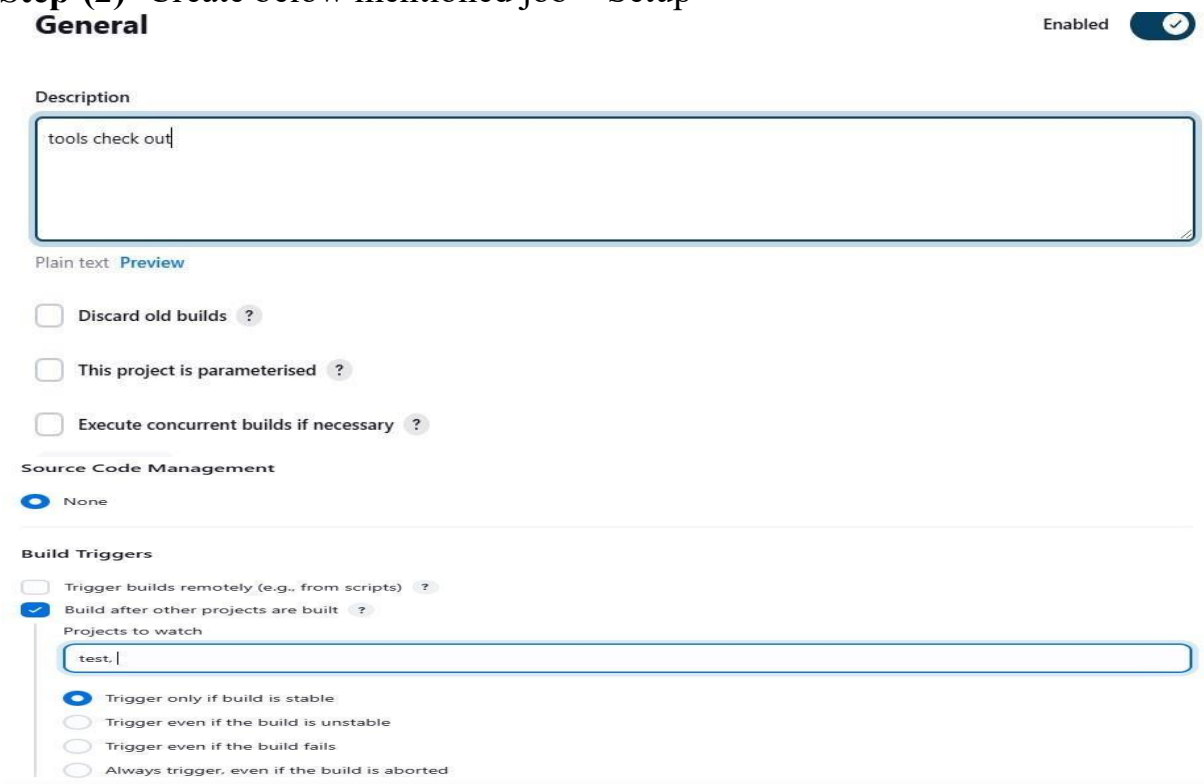
Creating main line CI pipeline .

PROCEDURE

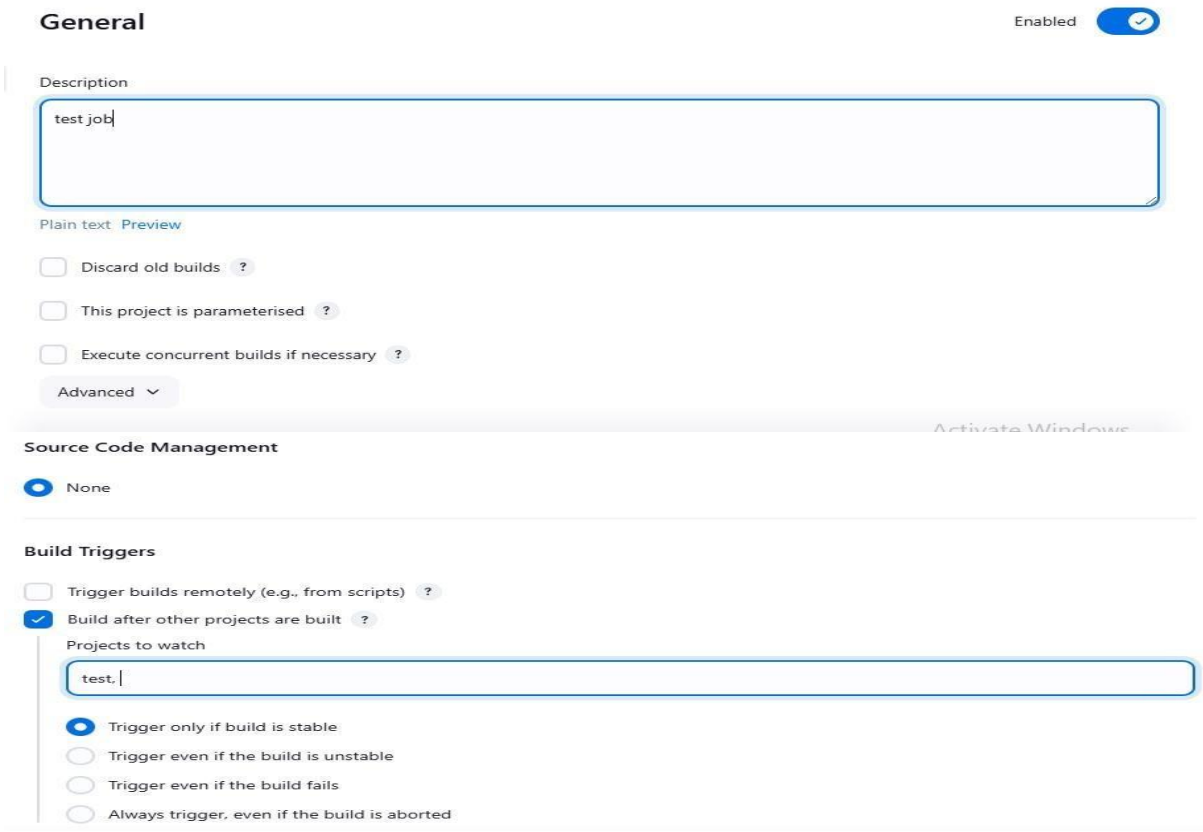
Step-(1)-Create a new Folder item with name "Central_CI" using "New Item" option as shown below. Add jobs of type Freestyle Project for each of tasks needed in the the continuous integration pipeline.



Step-(2)- Create below mentioned job – Setup



Step-(2)- Create below mentioned job – Setup



Thus the pipelines are created and executed as follows.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☼	CentarlCI	12 hr #1	N/A	0.55 sec
✓	☼	test	6.9 sec #21	N/A	0.12 sec

CONCLUSION

The central ci pipelines are created and executed successfully.

EXERCISE:10 COPYING AND MOVING JOBS IN JENKINS

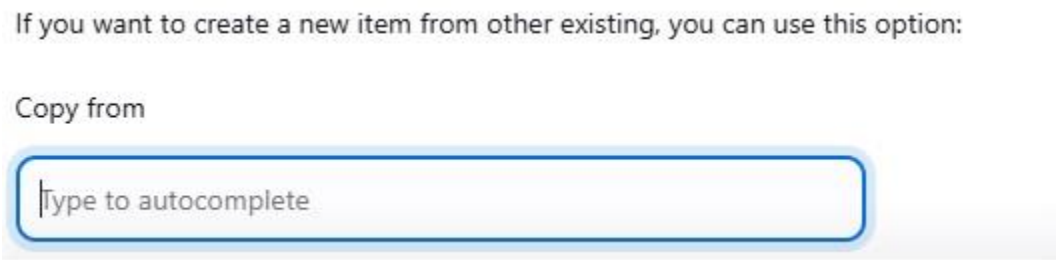
OBJECTIVE

To copy and move jobs in jenkins

I. Copying Jobs:

Step 1: Click on the option of **New Item** from the left panel.

Step 2: Name the new job and enter the name of the job you wish to copy below as shown:



Step 3: Click on **OK** and observe the newly copied job.

II. Moving Jobs:

Step 1: Enter the folder which has the jobs that need to be moved to a new location.

Step 2: Click on the option of **Move** from the left panel.

Step 3: Enter the location where you wish to move the jobs to, as shown below.

CONCLUSION

The jobs are copied and moved successfully.

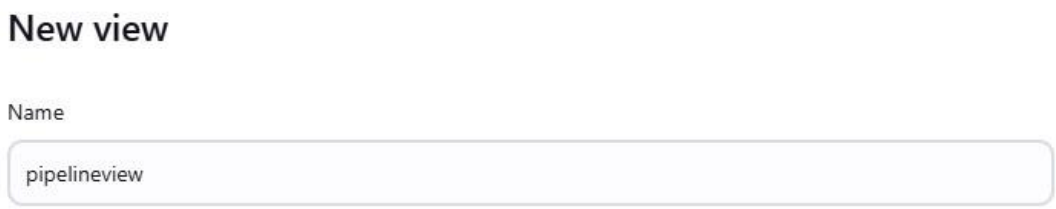
EXERCISE 11: CREATING PIPELINE VIEW IN JENKINS

OBJECTIVE:

Understand creation of pipeline view in Jenkins

PROCEDURE

Step 1: Click on the '+' symbol under the Jenkins project folder as shown in the screenshot given below



Step 2:select the jobs you need to view as pipepline.

Step 3:now execute the upstream job you got the pipeline as follows.

CONCLUSION

Thus jobs are viewed in the pipeline view.

EXERCISE:12 CONFIGURE GATING CONDITIONS IN JENKINS.

OBJECTIVE

Configure Gating Conditions In Jenkins.

PROCEDURE

1. Go to your Jenkinsfile.
2. Find the stage in the pipeline where you would like to raise a change request. For example, before the 'Production' deployment stage.

 We recommend adding 2 stages to your pipeline for deployment gating: One stage to raise a change request, and another stage to check the status of the change request.

3. Add the following snippet to your pipeline, replacing your **Jira site name, environment ID, environment type** and **service ID(s)** you copied from your Jira Service Management project.

```
   stage('Request approval') { // Raise change
request        steps {
          echo 'Raise change request...'
          jiraSendDeploymentInfo(site:'<YOUR-
SITE>.atlassian.net',                    environmentId:'us-prod-1',
environmentName:'us-prod-1',
environmentType:'production',
                 state:"pending",
enableGating:true,
serviceIds: [
                  '<YOUR-SERVICE-ID>'
                ]
            )
      }
   }
   stage("Approval gate") { // Check request
status        steps {
          retry(20) { // Poll every 30s
for 10min                waitUntil {
sleep 30
checkGatingStatus(
```

```
                    site:'<YOUR-
SITE>.atlassian.net',
environmentId:'us-prod-1'
                )
            }

}

}

}
```

Snippet example: Raise gated change request

```
stage('Request approval') { // Raise change
request            steps {
            echo 'Raise change request...'
            jiraSendDeploymentInfo(site:'<YOUR-
SITE>.atlassian.net',                    environmentId:'us-prod-
1',                    environmentName:'us-prod-1',
environmentType:'production',
state:"pending", // Deployment has not started yet
enableGating:true, // Notify Jira the pipeline is gated
serviceIds: [
                    '<YOUR-SERVICE-ID>'
                ]
            )
        }
    }
```

Snippet example: Manually check change request status

12

```
stage("Approval
gate") {            steps
{            waitUntil
{
                input message: "Check for approval?" // Manually trigger check
status                checkGatingStatus(
```

```
                    site:'<YOUR-
SITE>.atlassian.net',
environmentId:'us-prod-1'
                  )
              }
            }
          }
        }
```

Snippet example: Automatically check change request status after a delay

```
stage("Approval
gate") {            steps
{               waitUntil
{
                  sleep 30 // check status after 30s
checkGatingStatus(
site:'<YOUR-SITE>.atlassian.net',
environmentId:'us-prod-1'
                  )
              }
            }
          }
        }
```

Snippet example: Automatically check change request status (poll)

```
stage("Approval gate") {
          steps {
            retry(20) { // Retry every 30s
for 10min                  waitUntil {
sleep 30
checkGatingStatus(
                    site:'<YOUR-
SITE>.atlassian.net',
environmentId:'us-prod-1'
                  )
              }
            }
          }
```

Full Jenkinsfile example

Raise a change request and wait for approval before deploying to production. When the approval is complete, restart the pipeline automatically.

```
1 pipeline
{     agent
any
stages {
stage("Test
") {
steps {
            echo "Deploying to test"
          }
       }

stage("Stage") {
steps {
            echo "Deploying to staging"
          }
       }
       stage('Request approval') { // Raise
change request            steps {
            echo 'Raise change request...'
            jiraSendDeploymentInfo(site:'<YOUR-SITE>.atlassian.net',
                  environmentId:'us-prod-1',
environmentName:'us-prod-1',
environmentType:'production',
                  state:"pending",
enableGating:true,
serviceIds: [
                   '<YOUR-SERVICE-ID>'
                  ]
              )
          }
       }
       stage("Approval gate") { // Check change request
status            steps {
```

```
 retry(20) { // Poll every 30s for 10min
                waitUntil    {
sleep                        30
checkGatingStatus(
                    site:'<YOUR-
SITE>.atlassian.net',
environmentId:'us-prod-1'
                    )
                }
            }
        }
    }

stage("Production") {
steps {
            echo "Deploying to production!!"
        }
post {
always {
sh 'sleep 2'
            }
            // Notify Jira based on deployment
step result              success {
            jiraSendDeploymentInfo        (
site:         '<YOUR-SITE>.atlassian.net',
environmentId:                    'us-prod-1',
environmentName:                  'us-prod-1',
environmentType: 'production',
                    state: 'successful',
                    serviceIds: [
                      '<YOUR-SERVICE-ID>'
                    ]
                  )
            }
```

CONCLUSION

Thus the gating conditions are created in the jenkins.