

Exercise 1: Testing lab set up

Step 1: Open the System Environment Variables by right-click on This PC -> Properties -> Advanced System Settings -> Environment Variables...

Add the following as System Variables if not added already:

- o **JAVA_HOME** = path to jdk folder (C:\Program Files\Java)
- o **M2_HOME** = path to maven folder (C:\Program Files\Maven)
- o **PATH** = add "%JAVA_HOME%/bin; %M2_HOME%/bin; <path to sonar-runner>\sonar-runner-2.4\bin;" to the existing path variables

Step 2: Start all the installed tools by executing the appropriate batch file.

Step 3: you can ensure all are working by accessing tools user interface with below mentioned URL's and credentials




1. <http://localhost:9000> – SonarQube [admin/admin]
2. <http://localhost:8080> – Tomcat [tomcat/s3cret]
3. <http://localhost:8064/jenkins> - Jenkins

Summary: You have tested lab set up for the forthcoming exercises for Jenkins.

Exercise 2: Git operations

Objective: Perform the basic operations on Git repositories using EGit (Eclipse plugin)

Pulling code from Git repository

Step 1: Go to Eclipse-> file->import->  Git _->  Projects from Git _->  Clone URI _-> enter **central repository details** -> use credentials.

Source Git Repository

Enter the location of the source repository.



Location

URL:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

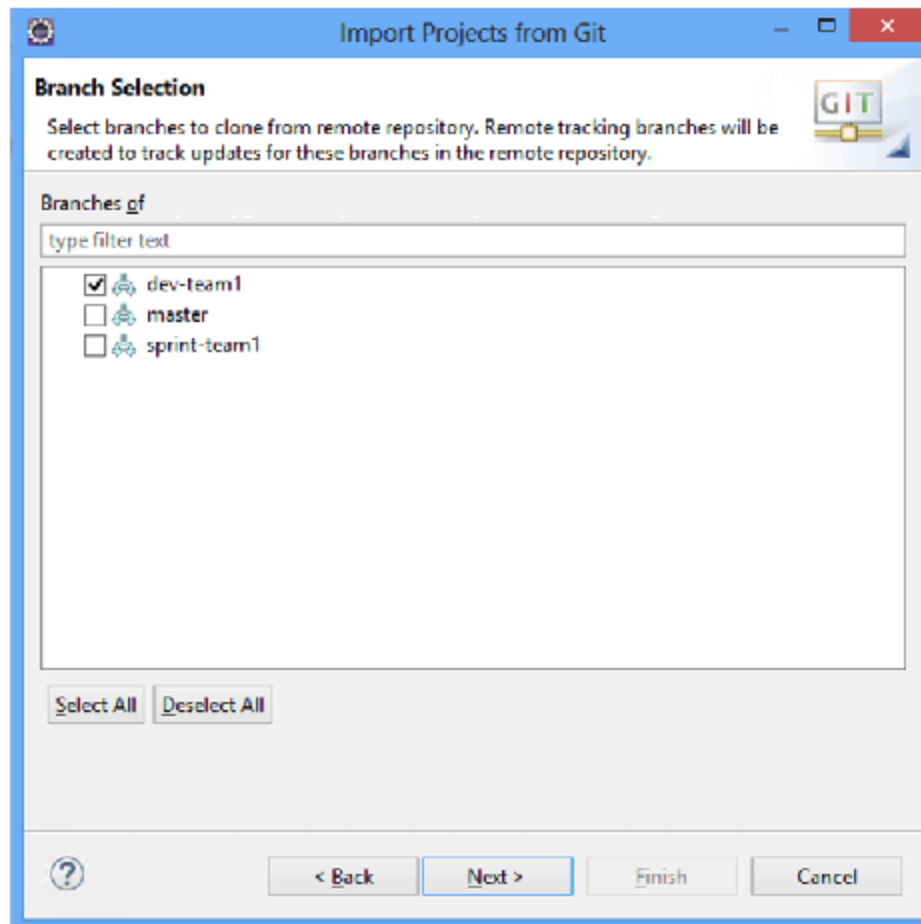
User:

Password:

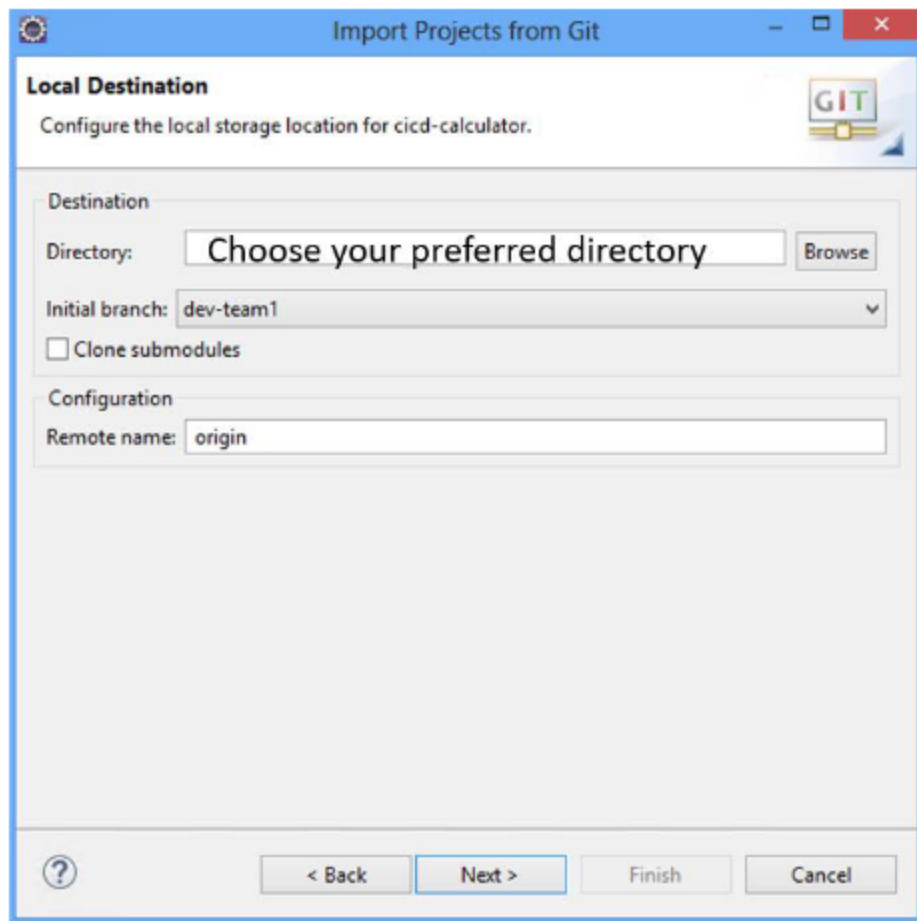
☐ Store in Secure Store

Click Next -> and choose branch (sample shown below)

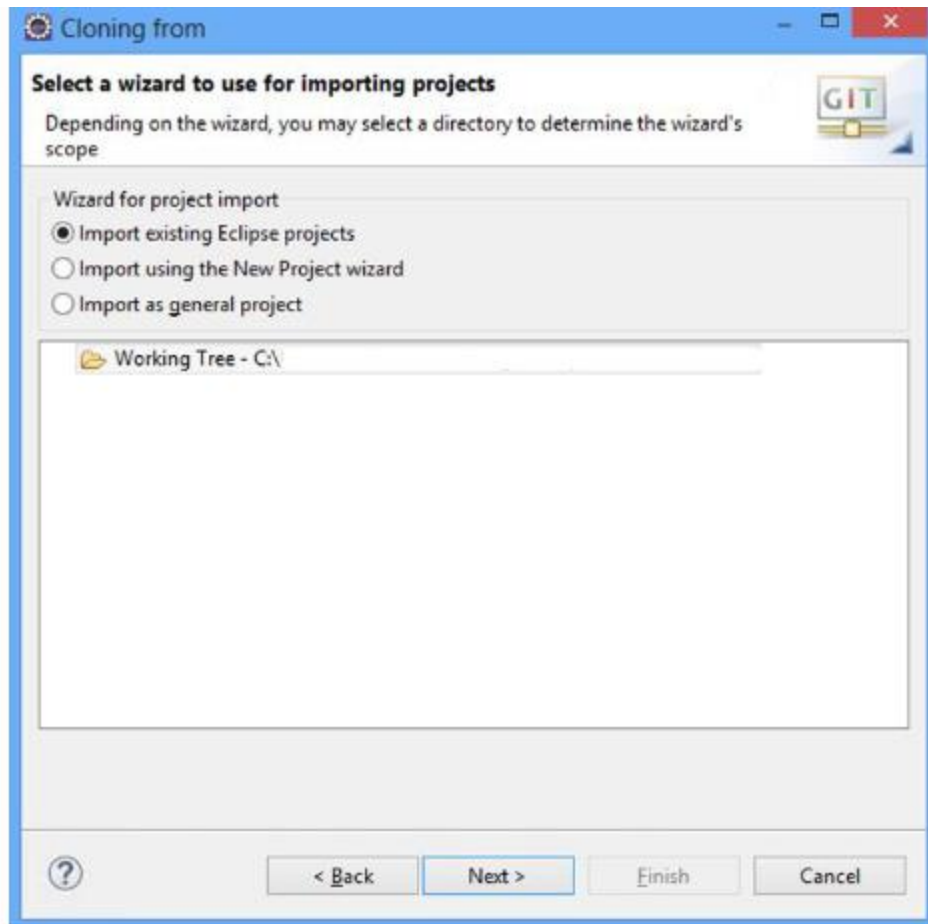
5



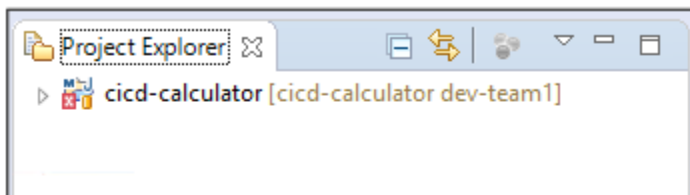
->Click next->browse directory to keep local repository as shown below->



->Next ->Choose first option as shown below->click Next->

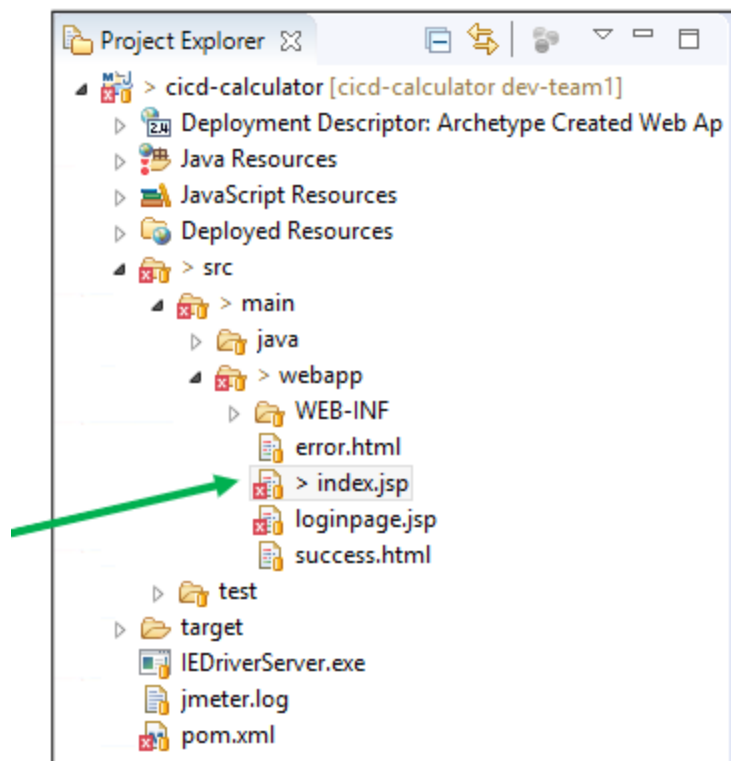


->Click Next -> click Finish->you can observe project explorer view with project cloned as shown below->. The name of the project is JNTU_Calc_Application in your case.

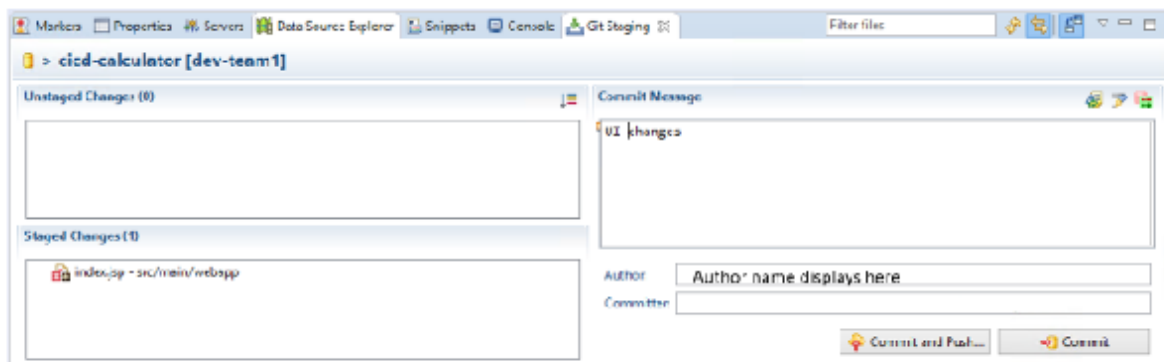


Commit Changes from Eclipse to Local Git Repository

Step 1: Expand the project cicd-calculator-> go to src\main\webapp\index.jsp -> make some changes to the code and save the changes->you can observe ">" symbol on project and edited source file as shown below.



Step 2: Right click on project->team->commit->drag the files from Unstaged Changes to Staged Changes ->Enter appropriate comments for the commit as shown below->



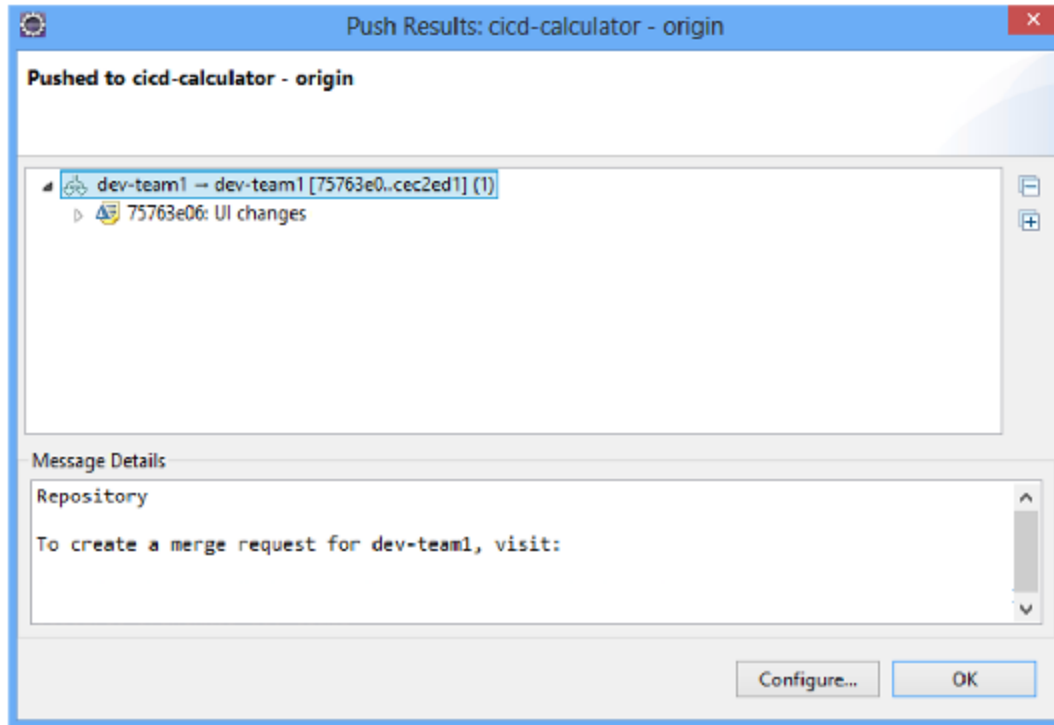
->click on Commit->you can observe the ">" symbol disappearing.

Note: similar way, we can commit multiple individual changes to local repository using commit option.

Push Changes from Local Git Repository (associated with Eclipse) to Remote Git Repository

Step 1: Right click on project->team-> click on Push to upstream -> you can see changes successfully pushed from local repository to central repository as shown below.

Note: Your repository name and number may be different, this is an illustration.



Note: if push results to “non fast forward” rejection, perform below mentioned operations in sequence.(This is due to the fact that there has been more changes made possibly by other developers to the central Git repository and hence those changes need to be merged before committing the changes)

Fetch from upstream->merge with local branch ->resolve if there are any merge conflicts->commit-> push.

You can observe the revision history using project->team->“Show In History” as shown below.

Project: cicd-calculator [cicd-calculator]

Id	Message	Author	Authored Date	Committer	Committed Date
75763e0	dev-team1 : origin/dev-team1 [HEAD] UI changes				
cac2ed1	removed extra html tag				
7809083	Initial commit				

commit 75763e0638c622fcc6560f9cc11c09871ff84535
Author:
Committer:
Parent: cac2ed1c18f8e2810519616e1327eb1f93ae5f14 (removed extra html tag)
Branches: dev-team1, origin/dev-team1

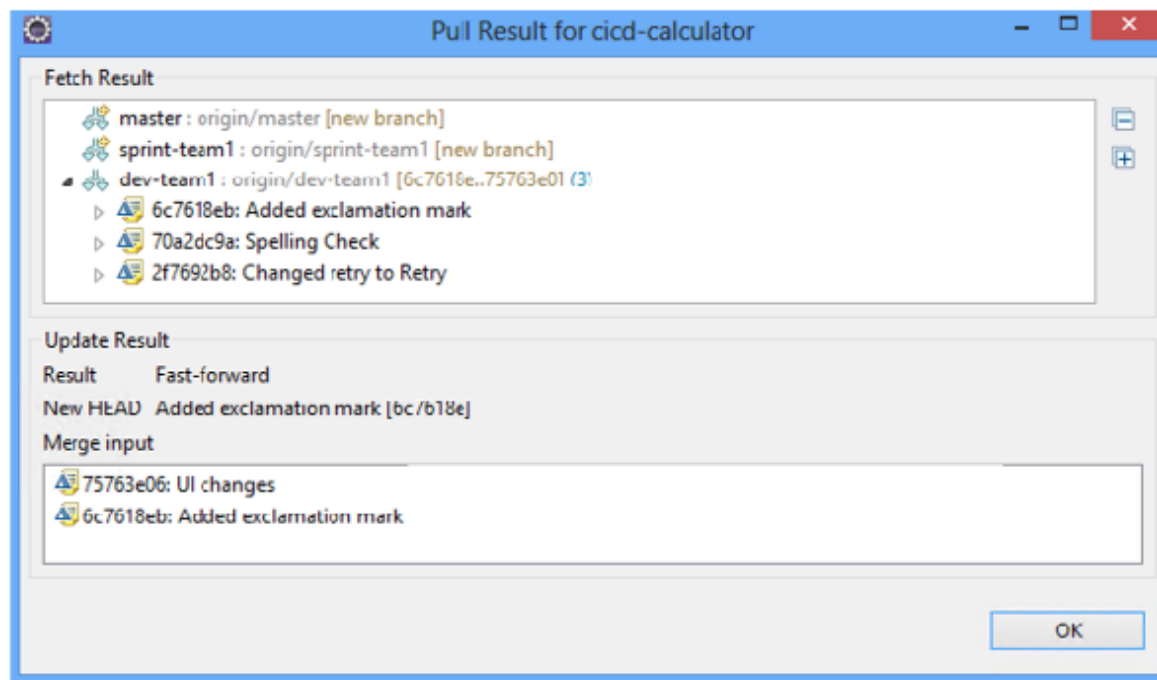
src/main/webapp/index.jsp

UI changes

Pull Changes from Remote Git Repository to Local Git Repository (associated with Eclipse)

Assumption: A team member has committed three changes to remote repository.

Step 1: Right click on project->team->Pull->enter credentials if required->you can observe fetch result and merge input see as shown below



Note: Pull perform two actions in sequence, Fetch and Merge. So if there are any merge conflicts after pull operation, you have to resolve and commit again.

You can observe the updated revision history using project->team->"Show in History".

Pull the project from remote Git repository to local Eclipse work space using EGit. Practice the following commands-

- a. Commit and push operations
- b. Fetch operation
- c. Merge operation with and without conflicts. When there is a conflict, resolve the conflict and push the code back to the branch provided.

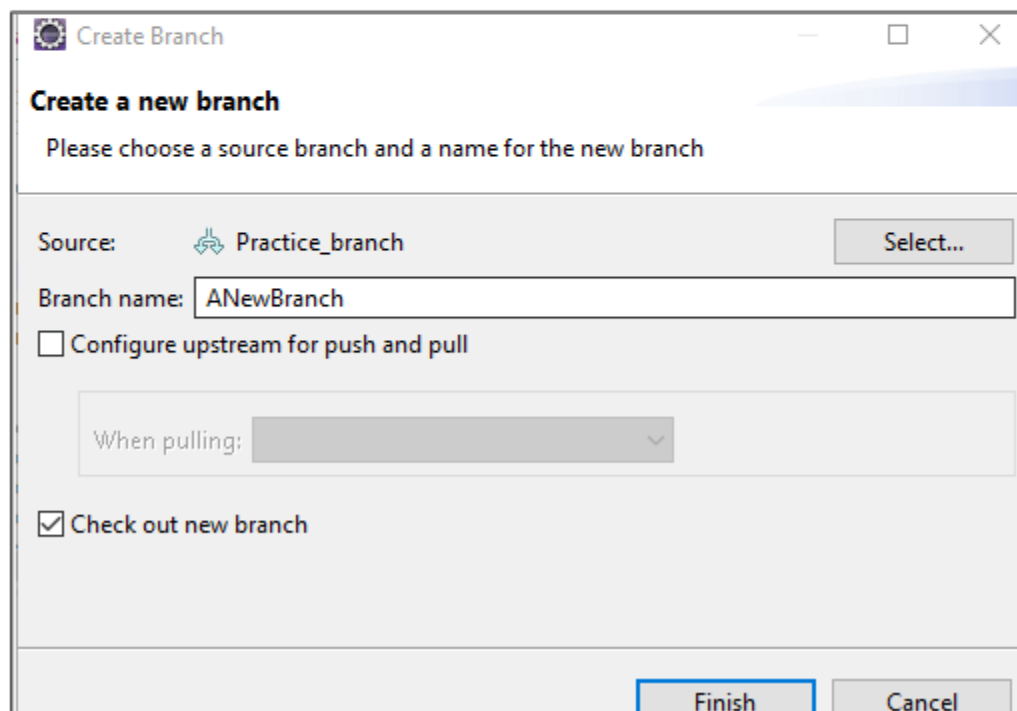
Note: Perform commit, and push operations on remote Git repository by making some changes to source code. You can use the repository provided in the earlier demo for completing this exercise.

Creating branches on local repository from eclipse via e-git plugin:

Step 1: Right click on the Project in the Project Explorer and go to Teams -> Switch to -> New Branch...

Step 2: The parent branch will by default be that branch that is currently open in Eclipse, you can also change it by clicking on the **Select** option.

Step 3: Name the New Branch and check on the **Check out as new branch** if you would like to switch to the new branch as well. Click on **Finish**.



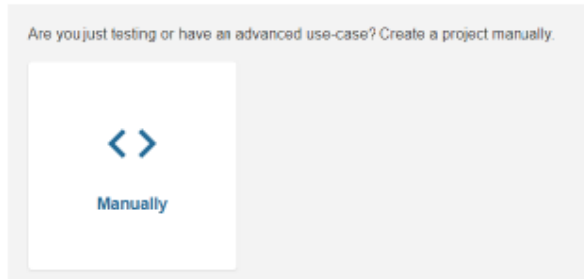
Summary: You have learnt to do basic operations with Git related to version control in this exercise.

Exercise 3: Creating a project in SonarQube

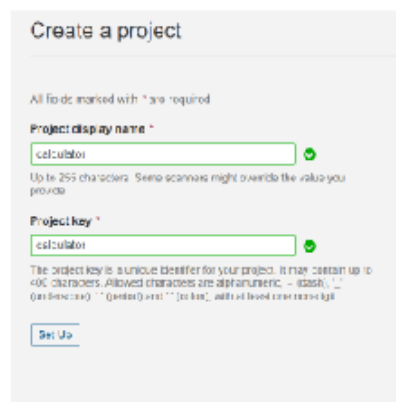
Objective: Understand creation of project in Sonarqube

Step 1: Go to SonarQube URL and login with credentials: admin: password

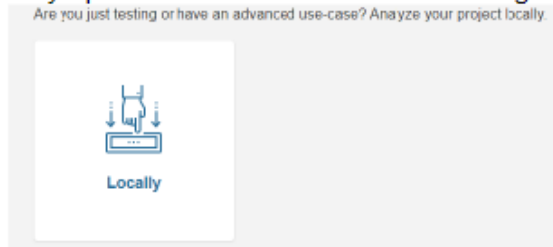
Step 2: Click on the manually option on the SonarQube dashboard as shown in the screenshot given below.



Step 3: Enter the project display name and project key as calculator as shown in the screenshot given below and click Set Up.

A screenshot of the "Create a project" form in SonarQube. The form has a title "Create a project" and a note "All fields marked with * are required". It contains two input fields: "Project display name *" with the value "calculator" and a green checkmark, and "Project key *" with the value "calculator" and a green checkmark. Below the second field, there is a small text block explaining that the project key is a unique identifier and listing allowed characters. At the bottom of the form is a "Set Up" button.

Step 4: Click on Locally option as shown in the screenshot given below.




Step 5: Click on the Generate option as shown in the screenshot given below.

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Generate a project token

Token name 

Analyze "calculator" [Generate](#)

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.

2 Run analysis on your project


Step 6: You can see the token generated as shown in the screenshot given below. Click on continue.

Note: Copy the token and save it somewhere on your system. It cannot be retrieved later.

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "calculator" **Generated token displays here** 

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](#)

2 Run analysis on your project

Step 7: Select Maven under the run analysis on your project as shown in the screenshot given below.

2 Run analysis on your project

What option best describes your build?

[Maven](#) [Gradle](#) [.NET](#) [Other \(for JS, TS, Go, Python, PHP, ...\)](#)

Step 8: Paste the copied token in the pom.xml within the <sonar.login> tag as shown in the screenshot given below.

```
<sonar.login>Place the token here</sonar.login>
```

Summary: You have learnt to create a project in sonarqube in this exercise.

Exercise 4: Using Sonarqube with Sonar-Runner

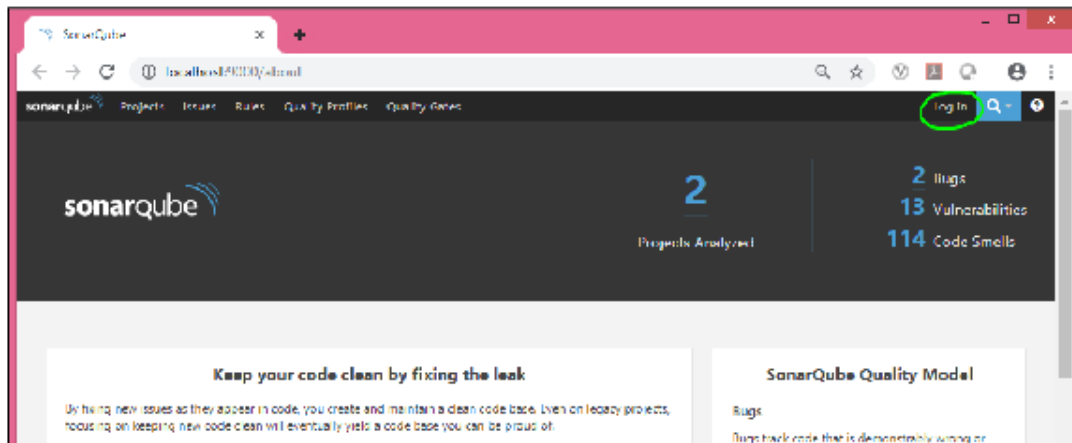
Objective: Understand running of Sonarqube using Sonar-Runner command-line tool

Step 1: Start Sonar server by using the appropriate bat file.

Once started, you should see success message in console window:

```
SonarQube
jvm 1 | 2019.01.20 00:02:57 INFO app[] [o.s.p.m.Monitor] Process[web] is up
jvm 1 | 2019.01.20 00:02:57 INFO app[] [o.s.p.m.JavaProcessLauncher] Launch process[ce]
e:\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Xmx512m -Xms128m -XX:+HeapDumpOn
sonarqube-6.2\temp -javaagent:C:\Program Files\Java\jdk1.8.0_141\jre\lib\manag
b/server/";./lib/ce/"C:\sonarqube-6.2\lib\jdbc\h2\h2-1.3.176.jar org.sonar.c
e-6.2\temp\sq-process822334252546262056properties
jvm 1 | 2019.01.20 00:03:12 INFO app[] [o.s.p.m.Monitor] Process[ce] is up
jvm 1 | 2019.01.20 00:03:12 INFO app[] [o.s.application.App] SonarQube is up
```

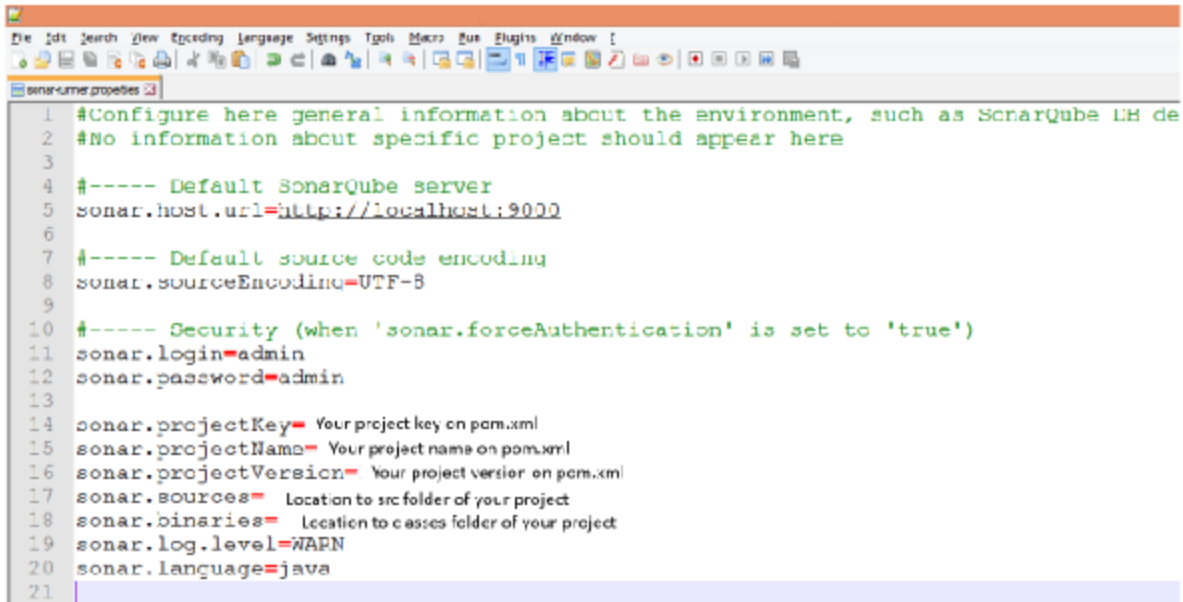
Go to SonarQube server dashboard at <http://localhost:9000> and login using admin/admin



Log In to SonarQube

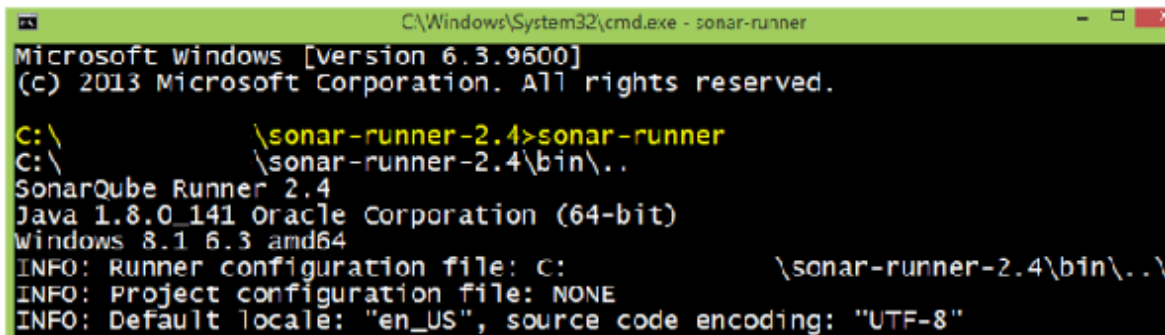
[Log In](#) [Cancel](#)

Step 2: Go to conf folder of Sonar Runner. Make the following changes based on the name of the Project provided to you and the path where it exists in your system:

A screenshot of an IDE window titled 'sonar-runner.properties'. The window shows a list of configuration properties for SonarRunner. The properties are as follows:

```
1 #Configure here general information about the environment, such as SonarQube LH de
2 #No information about specific project should appear here
3
4 #----- Default SonarQube server
5 sonar.host.url=http://localhost:9000
6
7 #----- Default source code encoding
8 sonar.sourceEncoding=UTF-8
9
10 #----- Security (when 'sonar.forceAuthentication' is set to 'true')
11 sonar.login=admin
12 sonar.password=admin
13
14 sonar.projectKey= Your project key on pom.xml
15 sonar.projectName= Your project name on pom.xml
16 sonar.projectVersion= Your project version on pom.xml
17 sonar.sources= Location to src folder of your project
18 sonar.binaries= Location to classes folder of your project
19 sonar.log.level=WARN
20 sonar.language=java
21
```

Open command prompt inside the project src folder in File Explorer and run the following command:

A screenshot of a Windows command prompt window titled 'C:\Windows\System32\cmd.exe - sonar-runner'. The prompt shows the execution of the 'sonar-runner' command. The output is as follows:

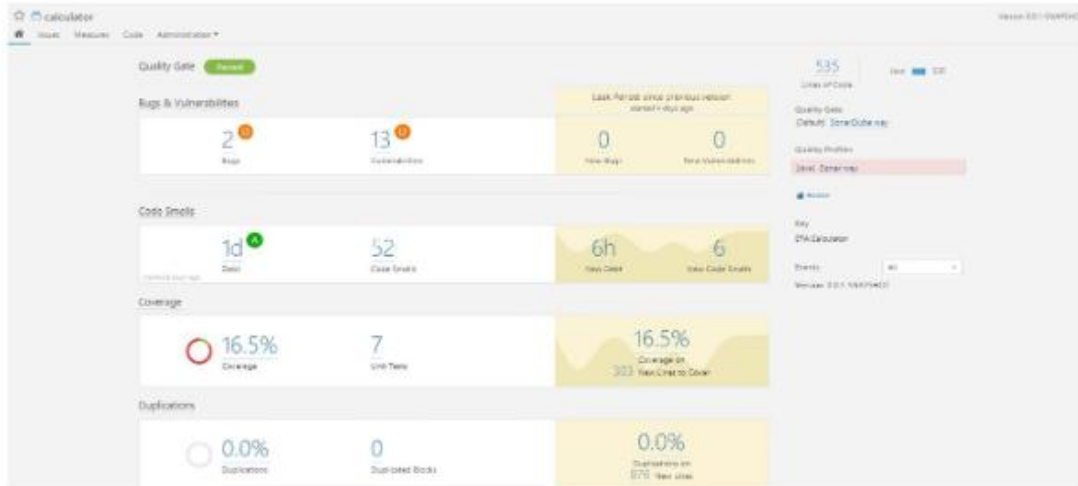
```
C:\Windows\System32\cmd.exe - sonar-runner
Microsoft Windows [version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\>\sonar-runner-2.4>sonar-runner
C:\>\sonar-runner-2.4\bin\..
SonarQube Runner 2.4
Java 1.8.0_141 Oracle Corporation (64-bit)
Windows 8.1 6.3 amd64
INFO: Runner configuration file: C:\>\sonar-runner-2.4\bin\..\
INFO: Project configuration file: NONE
INFO: Default locale: "en_US", source code encoding: "UTF-8"
```

Once execution is successful:

- Observe the static code analysis report and critical issues on SonarQube dashboard. Select your project name from list of projects to view the latest report.
- Resolve the technical issues in code and rerun the Maven build. Notice the changes to the technical debt value.

Run the build file and observe the results.



Summary of this Exercise:

You have learnt to observe the results of static code analysis using Sonarqube

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: Password1!

Step 2: Go to Administration -> Repositories -> Repositories -> Add repositories -> Local Repository.

Step 3: 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.

```
<distributionManagement>
  <repository>
    <id>artifactory</id>
    <name>Calc_Dev_Snapshot</name>
    <url>http://localhost:8081/artifactory/Calc_Dev_Snapshot</url>
  </repository>
</distributionManagement>
```

Step 5: Click on Create Local Repository.

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

Summary: You have learnt to create a local repository in artifactory in this exercise.

Exercise 6: Build automation using Maven

Objective: Understand build automation by writing a script in Maven with goals to invoke activities in a CI pipeline.

```
<?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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>ETA</groupId>
  <artifactId>Calculator</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>calculator</name>
  <url>http://calculator</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
    </dependency>
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.6.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>calculator</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.1.1</version>
        <configuration>
          <archive>
            <manifestEntries>
              <version>${project.version}</version>
            </manifestEntries>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```



```

        </manifestEntries>
    </archive>
</configuration>
</plugin>
<plugin>
    <groupId>org.sonarsource.scanner.maven</groupId>
    <artifactId>sonar-maven-plugin</artifactId>
    <version>3.2</version>
</plugin>
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.9</version>
    <executions>
        <execution>
            <id>default-prepare-agent</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>default-report</id>
            <phase>prepare-package</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
        <execution>
            <id>default-check</id>
            <goals>
                <goal>check</goal>
            </goals>
            <configuration>
                <rules>
                    <!-- implementation is
needed only for Maven 2 -->
                    <rule
implementation="org.jacoco.maven.RuleConfiguration">
                        <element>BUNDLE</element>
                        <limits>
                            <!--
implementation is needed only for Maven 2 -->
                            <limit
implementation="org.jacoco.report.check.Limit">
                                <counter>COMPLEXITY</counter>
                                <value>COVEREDRATIO</value>
                                <minimum>0.10</minimum>
                                </limit>
                            </limits>
                        </rule>
                    </rules>

```

```

        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>

<profiles>
  <profile>
    <id>ut</id>
    <build>
      <plugins>
        <plugin>

          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-surefire-
plugin</artifactId>
          <configuration>
            <includes>

<include>*/Calclaterut.java</include>

          </includes>
          </configuration>
        </plugin>
      </plugins>
    </build>
  </profile>
  <profile>
    <id>it</id>
    <build>
      <plugins>
        <plugin>

          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-surefire-
plugin</artifactId>
          <configuration>
            <includes>

<include>*/CalculatorIT.java</include>

          </includes>
          </configuration>
        </plugin>
      </plugins>
    </build>
  </profile>
  <profile>
    <id>pt</id>
    <build>
      <plugins>
        <plugin>

```

```

        <groupId>com.lazerycode.jmeter</groupId>
        <artifactId>jmeter-maven-
plugin</artifactId>
        <version>2.4.0</version>
        <executions>
            <execution>
                <id>jmeter-tests</id>
                <phase>test</phase>
                <goals>
                    <goal>jmeter</goal>
                </goals>
            </execution>
        </executions>
    </plugin>
</plugins>
</build>
</profile>
</profiles>

</project>

<!-- -->

```

Once you run the pom.xml in the order "clean compile test jacoco:report sonar:sonar

Once you run the pom.xml in the order "clean compile test jacoco:report sonar:sonar war:war", you can see the output on console as shown below.

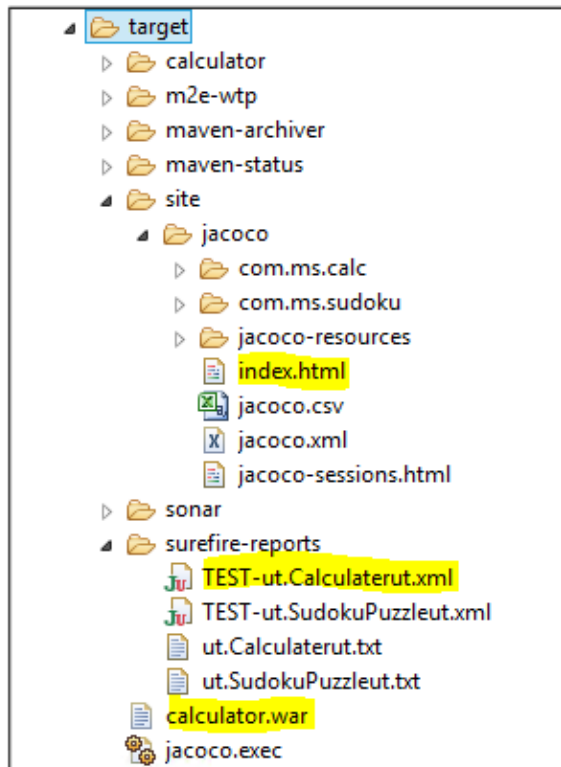
```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 48.728 s
[INFO] Finished at:
[INFO] Final Memory: 43M/351M
[INFO] -----

```

You can also find the results of the maven goals executed in the target folder of your project:

1. JUnit test reports in xml and txt: surefire-reports folder
2. Jacoco code coverage reports in html and xml: Under site/jacoco/
3. War file



Summary of this Exercise: You have learnt to use Maven tool for build automation.

Exercise 7: Jenkins Installation & System Configuration

Objective: Configure Jenkins for CI

Note:

You can install Jenkins in two ways on Windows –

- Install Jenkins as a Windows service
- Use webserver with a servlet container like GlassFish or Tomcat, and then deploy Jenkins.war to it.

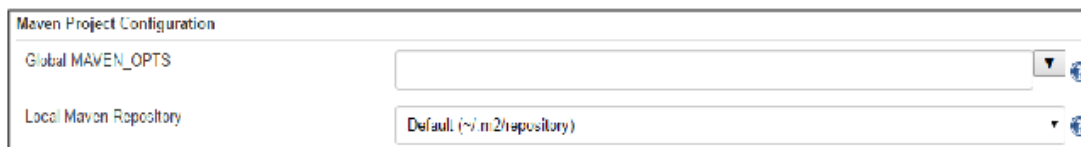
We have used the first approach in this exercise.

Configuring Jenkins

Step 1: Start the Jenkins server and enter URL `http://localhost:8064/jenkins` in browser

Step 2: Go to  [Manage Jenkins](#) ->  [Configure System](#)
Configure global settings and paths.

Step 3: You can observe Jenkins Maven integration as shown below:




Maven Project Configuration

Global MAVEN_OPTS

Local Maven Repository

Additional configurations

- Provide the tool configuration (JDK, Maven) in the Manage Jenkins -> Global Tool configuration tab



Maven Configuration

Default settings provider

Default global settings provider

JDK

JDK installations

Name	JAVA_HOME
JDK	C:\Program Files\Java\jdk-1.8.0_111

☐ Install automatically

List of JDK installations on the system

Git

Git installations

Git	
Name	Default
Path to Git executable	git.exe
<input type="checkbox"/> Install automatically	

Add Git

Delete Git

2. Go to global properties (Manage Jenkins->Configure system->Global properties->environment variables) and set JAVA_HOME and M2_HOME to the respective machine path

Summary of this Exercise:

You have learnt to configure Jenkins.

Exercise 8: Download the plugins in Jenkins

Objective: Download the plugins in Jenkins

Step 1: Go to Jenkins URL

Step 2: Go to Manage Jenkins -> Manage Plugins from the Jenkins dashboard

Step 3: Under the available tab of plugins manager search for the copy artifact plugin and check the check box as shown in the screenshot given below.

Plugin Manager

Updates Available Installed Advanced

Search: copy artifact

Install	Name	Released
<input checked="" type="checkbox"/>	Copy Artifact 1.45.4 build parameters add more Add a build step to copy artifacts from another project.	2 mo 2 days ago

[install without restart](#) [Uninstall and restart](#) Update Information obtained: 2 days 4 hr ago [Check now](#)

Step 4: Repeat the step – 3 and select the below mentioned plugins and click on install without restart.

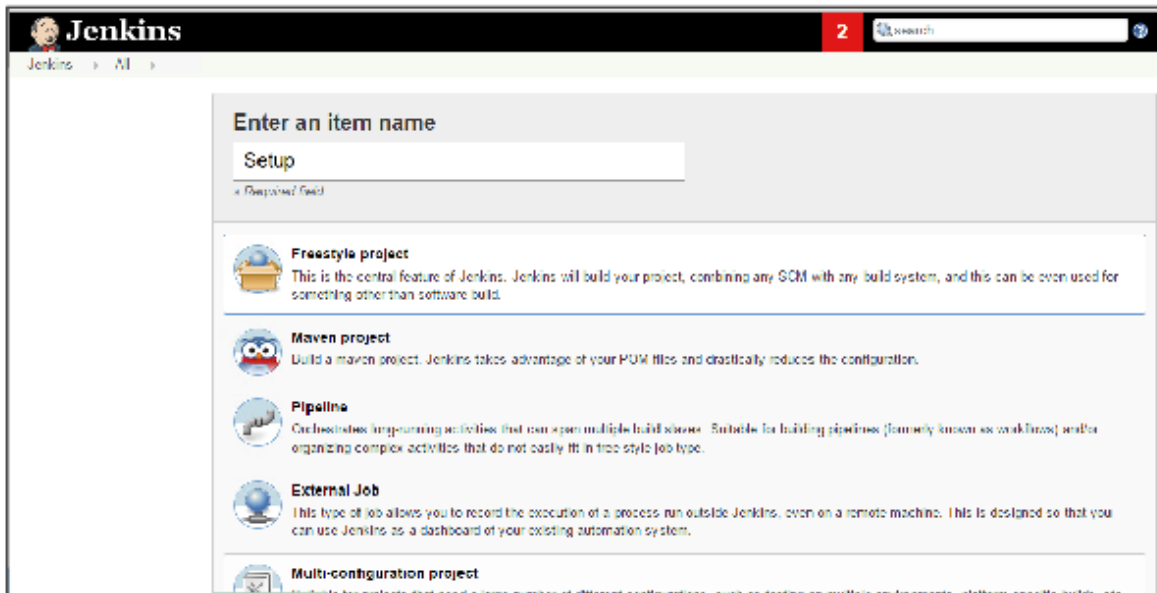
- JaCoCo
- Artifactory
- Build pipeline
- Deploy to container

Summary: You have learnt to download plugins in Jenkins in this exercise.

Exercise 9: Creating Central CI pipeline

Objective: Creating main line CI pipeline

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 1: Create below mentioned job - Setup

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

code check out

(Plain text) [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

If not empty, build records are only kept up to this number of days

Max # of builds to keep

3

If not empty, only up to this number of build records are kept

Advanced...

☐ Github project

Save

Apply

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Enter your repo URL

Credentials

Enter Credentials

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Enter your working branch

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

☐ Subversion

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when another project is promoted
- ☐ Git lab hook trigger for GIT3cm polling
- ☒ Poll SCM

Schedule

*12 * * * *

Ignore post-commit hooks ☐

Build Environment

- ☒ Delete workspace before build starts

Advanced...

- ☐ Abort the build if it's stuck

Save

Apply

console Output

- ☐ Use secret tool(s) or files

- ☐ With Ant

Build

Invoke top-level Maven targets

Maven Version

new

Goals

clean

Advanced...

Add build step

Post-build Actions

Save

Apply

Files to archive

**/*

Advanced...

Build other projects

Projects to build

Compile

- ☒ Trigger only if build is stable
- ☐ Trigger even if the build is unstable
- ☐ Trigger even if the build fails

Add post-build action

Save

Apply

Step 2: Create below mentioned job – Compile

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

compiling

[Main text] [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ Clobber project

Permissions to Copy Artifact

Save

Apply

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

save

ADDIV

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ Github hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use remote tool(s) or file(s)

Copy artifacts from another project

Project name

Setup

Which build

Latest successful build

☐ Stable build only

Artifacts to copy

**/*

Artifacts not to copy

Target directory

Parameter filters

☐ Flatten directories ☐ Optional ☒ Fingerprint Artifacts

Advanced...

Save

Apply

Invoke top-level Maven targets

Maven Version

new

Goals

compile

Advanced...

Add build step

Post-build Actions

Archive the artifacts

Files to archive

**/*

Advanced...

Build other projects

Projects to build

StaticAnalysis

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

Add post-build action

Save

Apply

Step 3: Create below mentioned job – Unit test

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

unitTest

[Plain text] [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Create concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if its stuck

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ Wtihout

Build

Copy artifacts from another project

Project name:

Which build:

☐ Stable build only

Artifacts to copy:

Artifacts not to copy:

Target directory:

Preserver filters:

☐ Flatten directories ☐ Optional ☒ Fingerprint Artifacts

Advanced...

Invoke top-level Maven targets

Maven Version:

Goals:

Advanced...

See build step

Post-build Actions

Archive the artifacts

Files to archive:

Advanced...

Build other projects

Projects to build:

- ☒ Trigger only if build is stable
☐ Trigger even if the build is unstable
☐ Trigger even if the build fails

See post build action

Save

Apply

Step 4: Create below mentioned job – Code coverage

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

codeCoverage

[Plan test] [Preview]

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

If not empty, build records are only kept up to this number of days

Max # of builds to keep

3

If not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Thin the builds

☐ Unarchive this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use console text(x) or file(y)

☐ With Ant

Build

Copy artifacts from another project

Project name

UnitTest

Which build

Latest successful build

☐ Stable build only

Artifacts to copy

**/*

Artifacts not to copy

Target directory

Parameter filters

☐ Flatten directories

☐ Optional

☒ Fingerprint Artifacts

Advanced...

Invoke top-level Maven targets

Maven Version

3.5.0

Goals

jacoco:report

Advanced...

Add build step

Post-build Actions

Archive the artifacts

Files to archive

**/*

Advanced...

Build other projects

Projects to build

/*

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post build action

Save

Apply

Step 5: Create below mentioned job – Static analysis

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

static analysis conan

[Plain text] Preview

☒ Discard old builds

Strategy

Log Retention

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ With Ant

Build

Copy artifacts from another project

Project name

Compile

Which build

Latest successful build

☐ Stable build only

Artifacts to copy

**/*

Artifacts not to copy

Target directory

Personalizer filters

☐ Flatten directories ☐ Optional ☒ Fingerprint Artifacts

Advanced...

Invoke top level Maven targets

Maven Version

new

Goal

sonar:sonar

Advanced...

Add build step

Post-build Actions

Archive the artifacts

Files to archive

**/*

Advanced...

Build other projects

Projects to build

UnitTest

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action

Save

Apply

Step 6: Create below mentioned job – war

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

war

[Plain text](#) [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

Build Triggers

Build Environment

☒ None

☐ Git

☐ Subversion

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ With Ant

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it sticks

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ With Art

Build

Copy artifact from another project

Project name: CodeCoverage

Which build: Latest successful build

☐ Stable build only

Artifacts to copy: **/*

Artifacts not to copy:

Target directory:

Parameter filters:

☐ Flatten directories ☐ Optional ☒ Fingerprint Artifacts

Advanced...

Invoke top-level Maven targets

Maven Version: new

Goals: war:war

Advanced...

Add build step

Post-build Actions

Archive the artifacts

Files to archive: **/*

Advanced...

Build other projects

Projects to build: ToBeFactory

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action

Save

Apply

Step 7: Create below mentioned job – ToArtifactory

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

war

[Plain text] [Preview]

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use server time(s) or file(s)

☐ With Ant

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck
☐ Add timestamps to the Console Output
☐ Use secret text(s) or file(s)
☐ With Ant

Build

☐ Copy artifacts from another project

Project name

War

Which build

Latest successful build

☐ Stable build only

Artifacts to copy

**/*

Artifacts not to copy

Target directory

Parameter filters

☐ Flatten directories
 ☐ Optional
 ☒ Fingerprint Artifacts

Advanced...

☐ Invoke top-level Maven targets

Maven Version

new

Goals

deploy

Advanced...

Add build step

Post-build Actions

☐ Archive the artifacts

Files to archive

**/*

Advanced...

☐ Build other projects

Projects to build

SmokeTest

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

Save

Apply

Step 8: Create below mentioned job – SmokeTest

General

Source Code ManagementBuild TriggersBuild EnvironmentBuildPost-build Actions

Description

smoke test

[Plain text] [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

3

if not empty, build records are only kept up to this number of days

Max # of builds to keep

3

if not empty, only up to this number of build records are kept

Advanced...

☐ GitHub project

☐ Permission to Copy Artifact

☐ Promote builds when...

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None

☐ Git

☐ Subversion

Build Triggers

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

☐ Build after other projects are built

☐ Build periodically

☐ Build when another project is promoted

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ With Ant

Build

Copy artifacts from another project

Project name

Yijir

Which build

Latest successful build

☐ Stable build only

Artifacts to copy

**/*

Artifacts not to copy

Target directory

Parameter filters

☐ Flatten directories☐ Optional☒ Fingerprint Artifacts

Advanced...

Add build step

Post-build Actions

☐ Deploy war/ear to a container ✕

WAR/EAR files ⓘ

Context path ⓘ

Containers

Tomcat 8.x ✕

Credentials Add

Tomcat URL

Add Container ▾

☐ Deploy on failure

Add post-build action ▾

Save

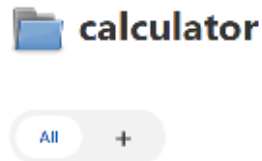
Apply

Summary: You learned main line CI pipeline creation.

Exercise 11: Creating pipeline view in Jenkins

Objective: Understand creation of pipeline view in Jenkins

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



Step 2: Provide the name for the pipeline in the viewname field and select the Build Pipeline View and click create as shown in the screenshot given below.

View name

Type

☒ Build Pipeline View
Shows the jobs in a build pipeline view

☐ Include a global view
Shows the content of a global view.

☐ List View
Shows items in a simple list format. You

Create

Step 3: Select calculator->Setup as the Select Initial Job under the Upstream/downstream config as shown in the screenshot given below and click ok.

Pipeline Flow

Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the up

Upstream / downstream config

Select Initial Job ⓘ

calculator - Setup

Step 4: Execute the job from setup and you can see the pipeline flow as shown in the screenshot given below.



Summary: You learned to create pipeline view in Jenkins.