

PROJECT-1

Jenkins Free-Style Job & Pipeline Job

JENKINS FREE-STYLE JOB:

➤ Overview

This Jenkins Free-style job is designed to automate the task need to perform for *project-1*. It utilizes the master-slave architecture and integrates tools like Git, Maven, SonarQube, Nexus, and Apache Tomcat to achieve the following tasks:

➤ Tasks Overview

1. Assign Job to a Node using Label
2. Clone the GitHub repository.
3. Build the project and create Maven artifacts.
4. Perform code analysis using SonarQube.
5. Upload the artifact to a Nexus repository.
6. Deploy the WAR file to a Tomcat server.

➤ Pre-requisites

1. Create a Node using an AWS Instance, using SSH connection with Private key and make agent online.
2. **Jenkins Setup:** A running Jenkins instance with the following plugins:

1. Git
2. Maven Integration
3. SonarQube Scanner
4. Nexus Artifact Uploader
5. Deploy to Container

The screenshot shows a web browser window with multiple tabs open at the top, including Jenkins, SonarQube, and several EC2 instances. The main content area displays the Jenkins SonarQube Scanner plugin page. The page title is "SonarQube Scanner". It features a navigation bar with links to Documentation, Releases, Issues, Dependencies, and Health Score. Below the navigation, there's a status bar showing "quality gate passed", "plugin v2.17.3", and "installs 52k". A brief description states: "This plugin allows easy integration in Jenkins projects of SonarQube, the open-source solution for helping developers write Clean Code." It lists two bullet points: "Documentation and changelog" and "Issue tracking". A note says, "If you want to make changes, please clone the [Git repository](#)". Another note says, "With this plugin, you can configure SonarQube instances and run a Sonar Scanner analysis in several ways:". A bulleted list follows: "By injecting the SonarQube configuration as environment variables and using them in any job step (such as Maven, Ant, Gradle, ...)", "Using the SonarQube Scanner build step", and "Using SonarScanner for MSBuild analysis steps". A note at the bottom left says, "'SonarQube Scanner' and 'SonarScanner for MSBuild' are managed as installable tools. List of available versions is retrieved automatically by Jenkins from a json file hosted on the update site:". Two URLs are listed: "<https://updates.jenkins.io/updates/hudson.plugins.sonar.SonarRunnerInstaller.json>" and "<https://updates.jenkins.io/updates/hudson.plugins.sonar.MsBuildSonarQubeRunnerInstaller.json>". A note at the bottom right says, "The files are automatically updated when a new version of SonarScanner or SonarScanner for MSBuild is published, thanks to crawlers written in groovy:". To the right of the main content, there's a sidebar with sections for "Version: 2.17.3", "Installed on 18.8% of controllers", "Links" (GitHub, Open issues (Jira), Report an issue (Jira), Pipeline Step Reference, Javadoc), and "Labels" (External Site/Tool Integrations, Build Reports). The bottom of the browser window shows the Windows taskbar with various pinned icons.

plugins.jenkins.io/nexus-artifact-uploader/

Nexus Artifact Uploader

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

[Documentation](#) [Releases](#) [Issues](#) [Dependencies](#) [Health Score](#)

Nexus Artifact Uploader

This plugin goal is to upload artifacts generated from non-maven projects to Nexus

This plugin now supports Nexus-2.x & Nexus-3.x.

Uploading snapshots is not supported by this plugin.

Job DSL example

```
freeStyleJob('NexusArtifactUploaderJob') {
    steps {
        nexusArtifactUploader {
            nexusVersion('nexus2')
            protocol('http')
            nexusUrl('localhost:8080/nexus')
            groupId('sp.sd')
            version('2.4')
            repository('NexusArtifactUploader')
            credentialsId('44620c50-1589-4617-a677-7563985e46e1')
        }
    }
}
```

How to install

Version: 2.14

Released: 2 years ago
Requires Jenkins 2.319.3
ID: nexus-artifact-uploader

Installed on 2.87% of controllers

[View detailed version information](#)

Links

[GitHub](#) [Open issues \(Jira\)](#) [Report an issue \(Jira\)](#) [Pipeline Step Reference](#) [Javadoc](#)

Labels

[adopt-this-plugin](#) [Artifact Uploaders](#)

ENG IN 15-12-2024

plugins.jenkins.io/deploy/

Deploy to container

[Documentation](#) [Releases](#) [Issues](#) [Dependencies](#) [Health Score](#)

This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build. The implementation is based on [Cargo](#). The list of currently supported containers include:

- Tomcat 4.x/5.x/6.x/7.x/8.x/9.x
- JBoss 3.x/4.x/5.x/6.x/7.x
- Glassfish 2.x/3.x/4.x

Refer to the [Deploy WebSphere Plugin](#) to deploy to a running remote WebSphere Application Server.
 Refer to the [WebLogic Deployer Plugin](#) to deploy to a running remote WebLogic Application Server.

How to rollback or redeploy a previous build

There may be several ways to accomplish this, but here is one suggested method:

- Install the [Copy Artifact Plugin](#)
- Create a new job that you will trigger manually only when needed
- Configure this job with a build parameter of type "Build selector for Copy Artifact", and a copy artifact build step using "Specified by build parameter" to select the build.
- Add a post-build action to deploy the artifact that was copied from the other job

Now when you trigger this job you can enter the build number (or use any other available selector) to select which build to redeploy.
 Thanks to Helge Taubert for this idea.

Change Log

Version 1.14 (Jul 24, 2019)

How to install

Version: 1.16

Released: 4 years ago
Requires Jenkins 2.138.4
ID: deploy

Installed on 4.15% of controllers

[View detailed version information](#)

Links

[GitHub](#) [Open issues \(Jira\)](#) [Report an issue \(Jira\)](#) [Pipeline Step Reference](#) [Extension Points](#) [Javadoc](#)

Labels

[Artifact Uploaders](#)

ENG IN 15-12-2024

3 Tools and Configurations:

1. Git & Maven installed and configured globally or as a Jenkins tool.
2. SonarQube server configured in Jenkins.
3. Nexus repository with proper permissions and credentials stored in Jenkins.
4. Tomcat server configured with user roles for deployment.

The screenshot shows the Jenkins 'Manage Jenkins > Tools' configuration page. A 'Git' tool is being edited. The 'Name' field is set to 'Default'. The 'Path to Git executable' field contains 'git'. The 'Install automatically' checkbox is checked. Below the configuration form, there is a 'Save' button and an 'Apply' button. The browser's address bar shows the URL `localhost:8081/manage/configureTools/`. The system tray at the bottom indicates it's 28°C and sunny, with the date 15-12-2024.

The screenshot shows the Jenkins 'Manage Jenkins > Tools' configuration page. A 'Maven' tool is being edited. The 'Name' field is set to 'maven'. The 'Install automatically' checkbox is checked. Under the 'Install from Apache' section, the 'Version' dropdown is set to '3.9.9'. Below the configuration form, there is a 'Save' button and an 'Apply' button. The browser's address bar shows the URL `localhost:8081/manage/configureTools/`. The system tray at the bottom indicates it's 28°C and sunny, with the date 15-12-2024.

Screenshot of the AWS CloudShell interface showing the EC2 Instances page. The sidebar includes links for Dashboard, EC2 Global View, Events, Instances (5), Images, Elastic Block Store, and Network & Security. The main content shows a table of 5 instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. The instances are labeled slave, sonarqube, nexus, tomcat, and Docker.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
	slave	i-0a6c352baca6a8c4b	Running	t2.medium	Initializing	View alarms +	ap-south-1b	ec2-65-1
	sonarqube	i-0f75df54b4c73672f	Running	t2.medium	Initializing	View alarms +	ap-south-1b	ec2-3-11
	nexus	i-0ac6af65ebd0ac325	Running	t2.medium	Initializing	View alarms +	ap-south-1b	ec2-3-11
	tomcat	i-0d23b457cff4807af	Running	t2.medium	Initializing	View alarms +	ap-south-1b	ec2-43-2
	Docker	i-0b3090d97a9a622a7	Stopped	t2.micro	-	View alarms +	ap-south-1b	-

Select an instance

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS CloudShell interface showing the EC2 instance connect session for instance i-0f75df54b4c73672f. The session shows a terminal window with the following output:

```
'`#`_###`_          Amazon Linux 2023
~~`_###`_`_
~~`_###`_
~~`_`#
~~`_`/
~~`_`v~`_`->
~~`_`/`_
~~`_`/`_
~~`_`/`_
~~`_`/`_
~~`_`/`_
~~`_`/`_
Last login: Sun Dec 15 09:49:17 2024 from 13.233.177.5
[ec2-user@ip-172-31-5-239 ~]$ sudo -i
[root@ip-172-31-5-239 ~]# cd /opt/sonarqube/bin/linux-x86-64/
[root@ip-172-31-5-239 linux-x86-64]# ls
SonarQube.pid  sonar.sh
[root@ip-172-31-5-239 linux-x86-64]# su owner
[owner@ip-172-31-5-239 linux-x86-64]$ ./sonar.sh start
/usr/bin/java
Starting SonarQube...
SonarQube is already running.
[owner@ip-172-31-5-239 linux-x86-64]$ ./sonar.sh status
/usr/bin/java
SonarQube is running (2895).
[owner@ip-172-31-5-239 linux-x86-64]$ '
```

i-0f75df54b4c73672f (sonarqube)

PublicIPs: 3.110.135.223 PrivateIPs: 172.31.5.239

Screenshot of the AWS CloudShell interface showing the EC2 instance connect session for instance i-0f75df54b4c73672f. The session shows a terminal window with the following output:

```
1 BSE midcap +0.18%
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

SonarQube Scanner installations

SonarQube Scanner installations ^ Edited

Add SonarQube Scanner

SonarQube Scanner

Name: sonarqube

Install automatically ?

Install from Maven Central

Version: SonarQube Scanner 6.2.1.4610

Add Installer ▾

Save Apply

This screenshot shows the Jenkins 'Manage Jenkins > Tools' section. Under 'SonarQube Scanner installations', a new configuration is being added. The 'Name' field is set to 'sonarqube'. The 'Install automatically' checkbox is checked. Under the 'Install from Maven Central' section, the 'Version' dropdown is set to 'SonarQube Scanner 6.2.1.4610'. There is also an 'Add Installer' dropdown menu. At the bottom are 'Save' and 'Apply' buttons.

SonarQube installations

List of SonarQube installations

Name: sonarqube

Server URL: Default is http://localhost:9000
http://3.110.135.223:9000

Server authentication token: SonarQube authentication token. Mandatory when anonymous access is disabled.
sonarqube analysis

+ Add Advanced ▾

Add SonarQube

Save Apply

This screenshot shows the Jenkins 'Manage Jenkins > System' section. Under 'SonarQube installations', a new configuration is being added. The 'Name' field is set to 'sonarqube'. The 'Server URL' is set to 'http://3.110.135.223:9000'. The 'Server authentication token' is set to 'sonarqube analysis'. There are '+ Add' and 'Advanced' buttons. At the bottom are 'Save' and 'Apply' buttons.

```

~~~ V~' '->
~~~ /
~~~ .-' /-
~~~ /m/'

Last login: Thu Dec 12 10:42:48 2024 from 13.233.177.3
[ec2-user@ip-172-31-10-62 ~]$ sudo -i
[root@ip-172-31-10-62 ~]# ls
nexus-3.70.3-01-java8-unix.tar.gz sonatype-work
[root@ip-172-31-10-62 ~]# cd /opt/nexus
[root@ip-172-31-10-62 nexus]$ ls
NOTICE.txt OSS-LICENSE.txt PRO-LICENSE.txt bin deploy etc lib public replicator system
[root@ip-172-31-10-62 nexus]$ cd bin
[root@ip-172-31-10-62 bin]$ ls
contrib nexus nexus.rc nexus.vmoptions
[root@ip-172-31-10-62 bin]$ ./nexus start
WARNING: ****
WARNING: Detected execution as "root" user. This is NOT recommended!
WARNING: ****
Starting nexus
[root@ip-172-31-10-62 bin]$ ./nexus status
WARNING: ****
WARNING: Detected execution as "root" user. This is NOT recommended!
WARNING: ****
nexus is running.
[root@ip-172-31-10-62 bin]#

```

i-0ac6af65ebd0ac325 (nexus)

Public IPs: 3.110.164.242 Private IPs: 172.31.10.62



```

~~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~~ V~' '->
~~~ /
~~~ .-' /-
~~~ /m'

Last login: Thu Dec 12 10:43:04 2024 from 13.233.177.4
[ec2-user@ip-172-31-11-221 ~]$ sudo -i
[root@ip-172-31-11-221 ~]# ls
apache-tomcat-9.0.98.tar.gz
[root@ip-172-31-11-221 ~]# cd /opt/tomcat
[root@ip-172-31-11-221 tomcat]$ ls
BUILDING.txt CONTRIBUTING.md LICENSE NOTICE README.md RELEASE-NOTES RUNNING.txt bin conf lib logs temp webapps work
[root@ip-172-31-11-221 tomcat]$ cd bin
[root@ip-172-31-11-221 bin]$ ls
bootstrap.jar catalina.sh commons-daemon-native.tar.gz configtest.sh digest.sh setclasspath.bat shutdown.sh tomcat-juli.jar tool-wrapper.sh
catalina-tasks.xml ciphers.bat commons-daemon.jar daemon.sh makebase.bat setclasspath.sh startup.bat tomcat-native.tar.gz version.bat
catalina.bat ciphers.sh configtest.bat digest.sh makebase.sh shutdown.bat startup.sh tool-wrapper.bat version.sh
[root@ip-172-31-11-221 bin]$ ./startup.sh
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME: /usr
Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[root@ip-172-31-11-221 bin]#

```

i-0d23b457cff4807af (tomcat)

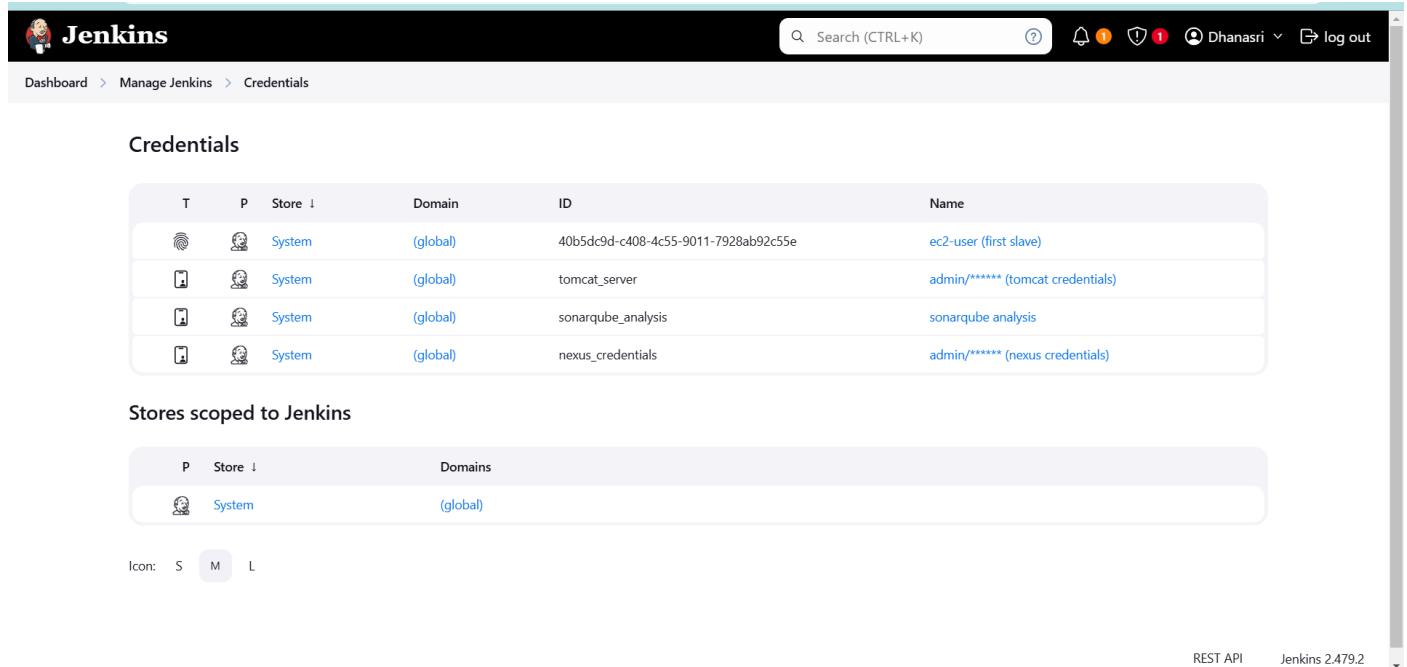
Public IPs: 43.204.130.230 Private IPs: 172.31.11.221



CONFIGURE SONARQUBE NEXUS TOMCAT with JENKINS by Adding Credentials respectively

SonarQube: Configure with Secret Text i.e. Go to SonarQube Admin > MyAccount > Security > Create a USER TOKEN

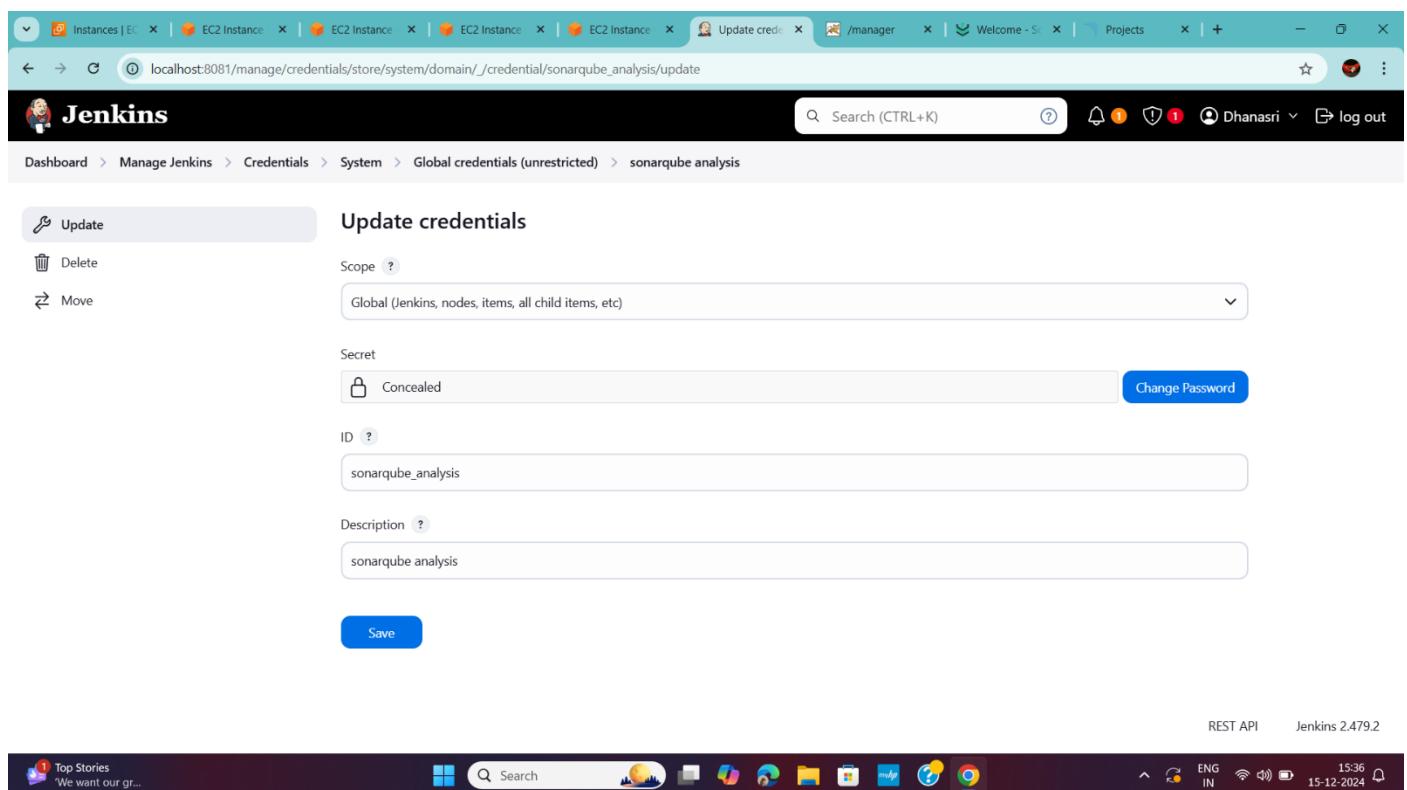
NEXUS & TOMCAT: Configure with Username and Password



The screenshot shows the Jenkins 'Credentials' management interface. It lists four global credentials:

T	P	Store	Domain	ID	Name
👤	🔑	System	(global)	40b5dc9d-c408-4c55-9011-7928ab92c55e	ec2-user (first slave)
💻	🔑	System	(global)	tomcat_server	admin/******** (tomcat credentials)
💻	🔑	System	(global)	sonarqube_analysis	sonarqube analysis
💻	🔑	System	(global)	nexus_credentials	admin/******** (nexus credentials)

Below the table, there is a section titled 'Stores scoped to Jenkins' which shows a single entry for 'System' with 'Domains' '(global)'. There are also 'Icon' and 'S' buttons.



The screenshot shows the 'Update credentials' page for the 'sonarqube analysis' credential. The page includes fields for 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'Secret' (set to 'Concealed'), 'ID' (set to 'sonarqube_analysis'), and 'Description' (set to 'sonarqube analysis'). A 'Save' button is at the bottom.

REST API Jenkins 2.479.2



localhost:8081/manage/credentials/store/system/domain/_/credential/nexus_credentials/update

Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: admin

Treat username as secret:

Password: Concealed

ID: nexus_credentials

Description: nexus credentials

Save



localhost:8081/manage/credentials/store/system/domain/_/credential/tomcat_server/update

Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: admin

Treat username as secret:

Password: Concealed

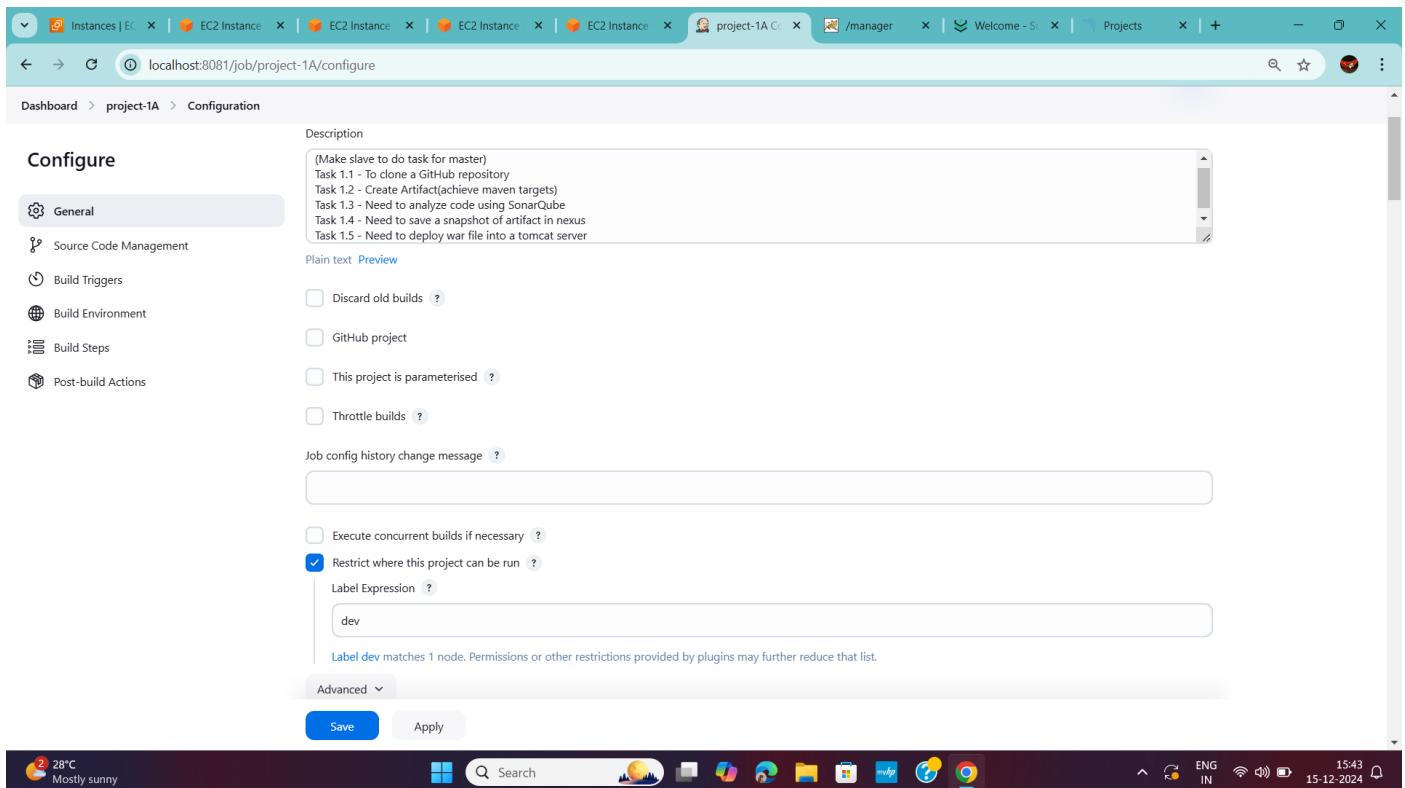
ID: tomcat_server

Description: tomcat credentials

Save



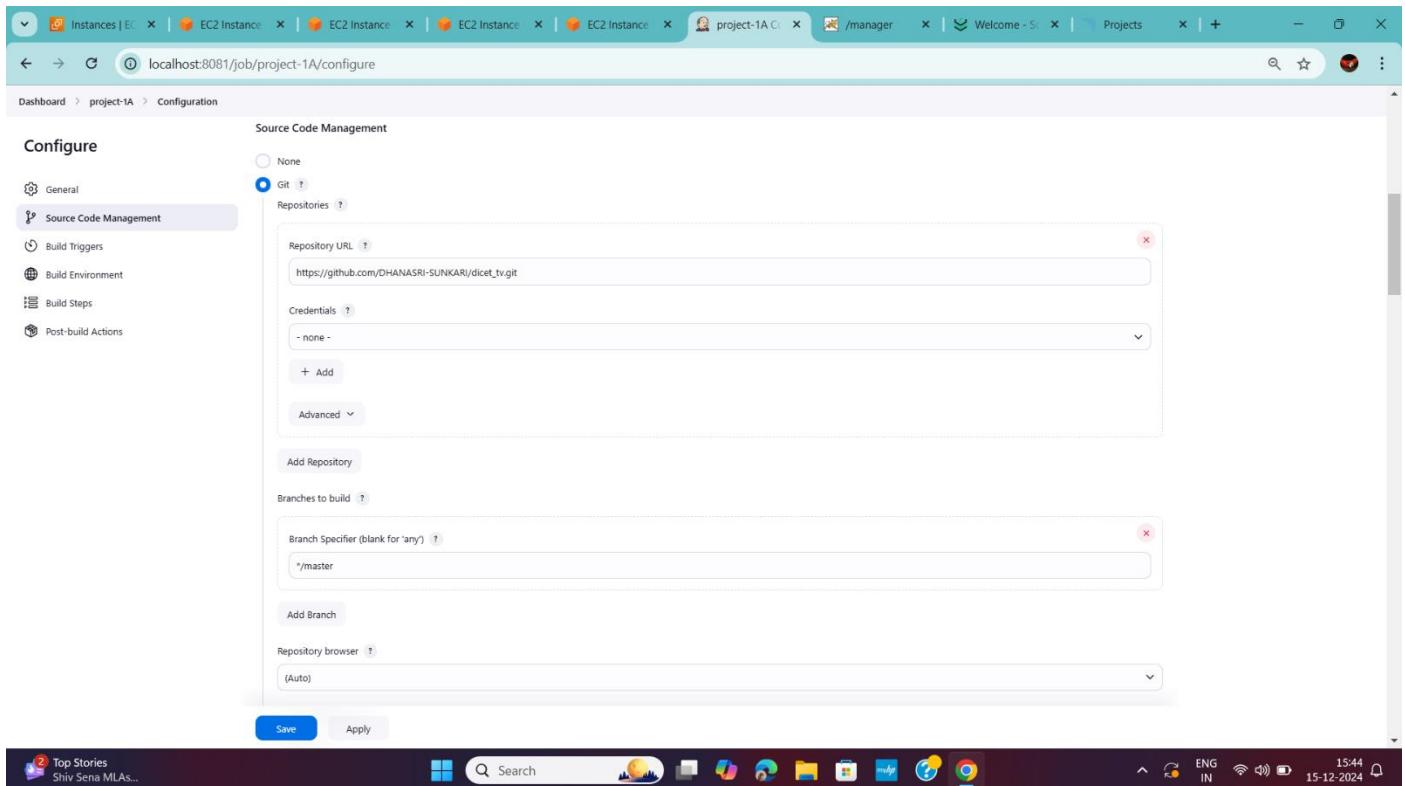
➤ Job Configuration



The screenshot shows the Jenkins job configuration interface for a job named 'project-1A'. The left sidebar has a 'Configure' section with several tabs: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'General' tab contains a 'Description' field with a rich text editor containing a list of tasks. The 'Source Code Management' tab shows a 'Git' repository configuration with 'Repository URL' set to 'https://github.com/DHANASRI-SUNKARI/dicet_tv.git' and 'Credentials' set to 'none'. Other tabs like 'Build Triggers' show options like 'Discard old builds', 'GitHub project', 'Parameterised', 'Throttle builds', and 'Execute concurrent builds if necessary'. A 'Label Expression' field is set to 'dev', which matches one node. The 'Advanced' tab is collapsed. At the bottom are 'Save' and 'Apply' buttons.

Step 1: Clone GitHub Repository

1. Navigate to **Source Code Management > Git**.
2. Enter the repository URL: <https://github.com/<username>/<repository>.git>.
3. Specify the credentials for GitHub if needed.
4. Configure the branch to build (e.g., main or master).



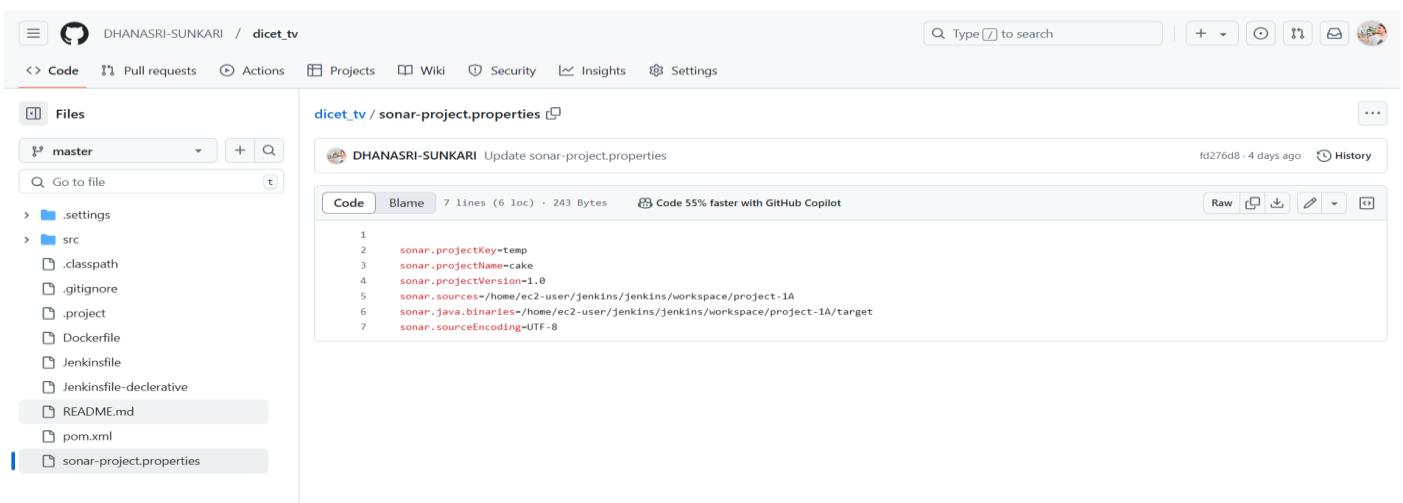
The screenshot shows the 'Source Code Management' configuration for the 'Git' provider. Under the 'General' tab, the 'Repository URL' is set to 'https://github.com/DHANASRI-SUNKARI/dicet_tv.git'. The 'Credentials' dropdown is currently empty ('none'). Below this, there's an 'Add' button for adding more repositories. The 'Advanced' tab is collapsed. In the 'Branches to build' section, the 'Branch Specifier' is set to '*/*master'. The 'Add Branch' and 'Repository browser' sections are also visible. At the bottom are 'Save' and 'Apply' buttons.

Step 2: Build the Project

1. Under **Build Environment**, check **Provide Node & Label Settings** to ensure the slave node is utilized.
2. Add a build step: **Invoke Top-Level Maven Targets**.
 1. Goals: clean package.
 2. POM: pom.xml.
 3. Select the Maven version configured in Jenkins.

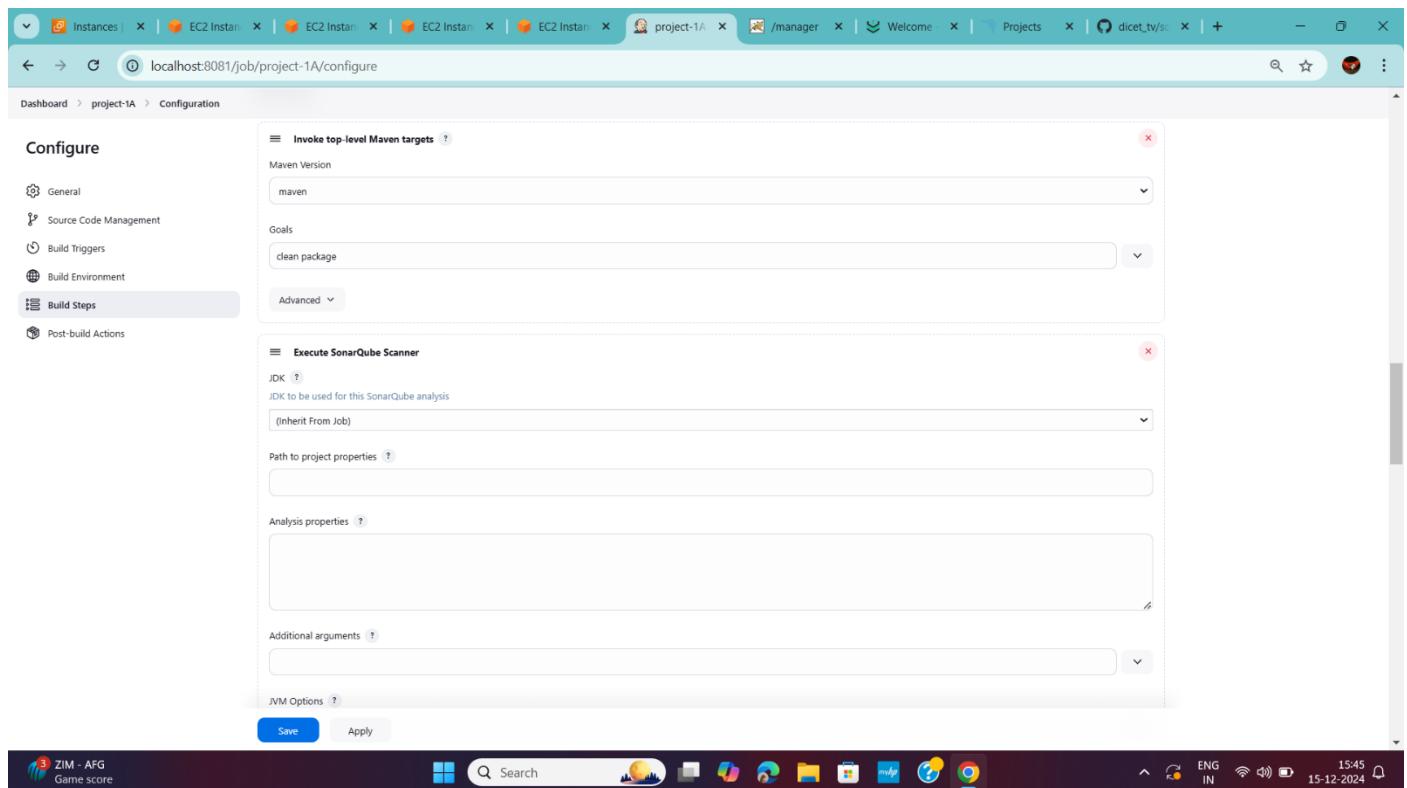
Step 3: Analyze Code with SonarQube

1. Add a build step: **Execute SonarQube Scanner**.
 1. Provide the **SonarQube Project Key, Branch**, and other required parameters.
 2. Ensure that the analysis properties are defined (e.g., sonar-project.properties, sonar.java.binaries).



The screenshot shows a GitHub repository named 'dicet_tv'. On the left, the file tree displays files like .settings, src, .classpath, .gitignore, .project, Dockerfile, Jenkinsfile, Jenkinsfile-declarative, README.md, pom.xml, and sonar-project.properties. The right pane shows the content of the sonar-project.properties file:

```
sonar.projectKey=temp
sonar.projectName=cake
sonar.projectVersion=1.0
sonar.sources=/home/ec2-user/jenkins/jenkins/workspace/project-1A
sonar.java.binaries=/home/ec2-user/jenkins/jenkins/workspace/project-1A/target
sonar.sourceEncoding=UTF-8
```



The screenshot shows the Jenkins configuration page for 'project-1A'. The 'Build Steps' section is selected. It contains two steps: 'Invoke top-level Maven targets' and 'Execute SonarQube Scanner'. The 'Invoke top-level Maven targets' step has 'Goals' set to 'clean package'. The 'Execute SonarQube Scanner' step has 'Path to project properties' set to '(Inherit From Job)' and 'Analysis properties' set to an empty field. At the bottom, there are 'Save' and 'Apply' buttons.

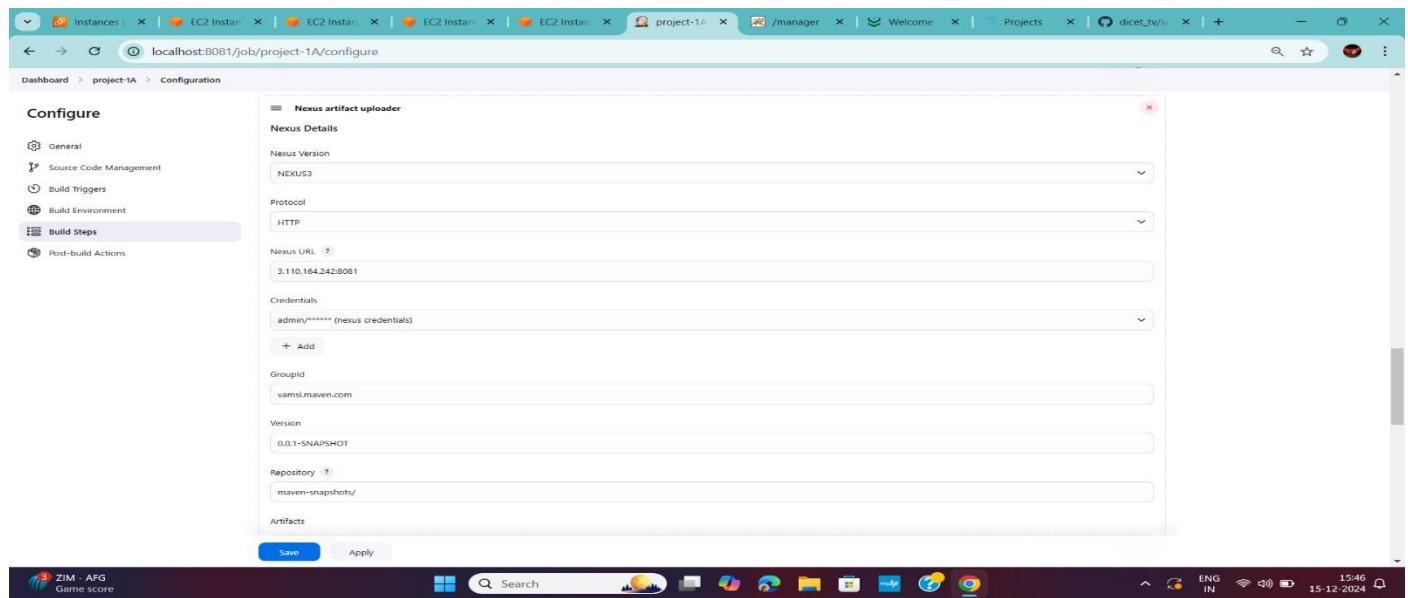
Step 4: Upload Artifact to Nexus

1. Add a post-build action: **Nexus Artifact Uploader**.

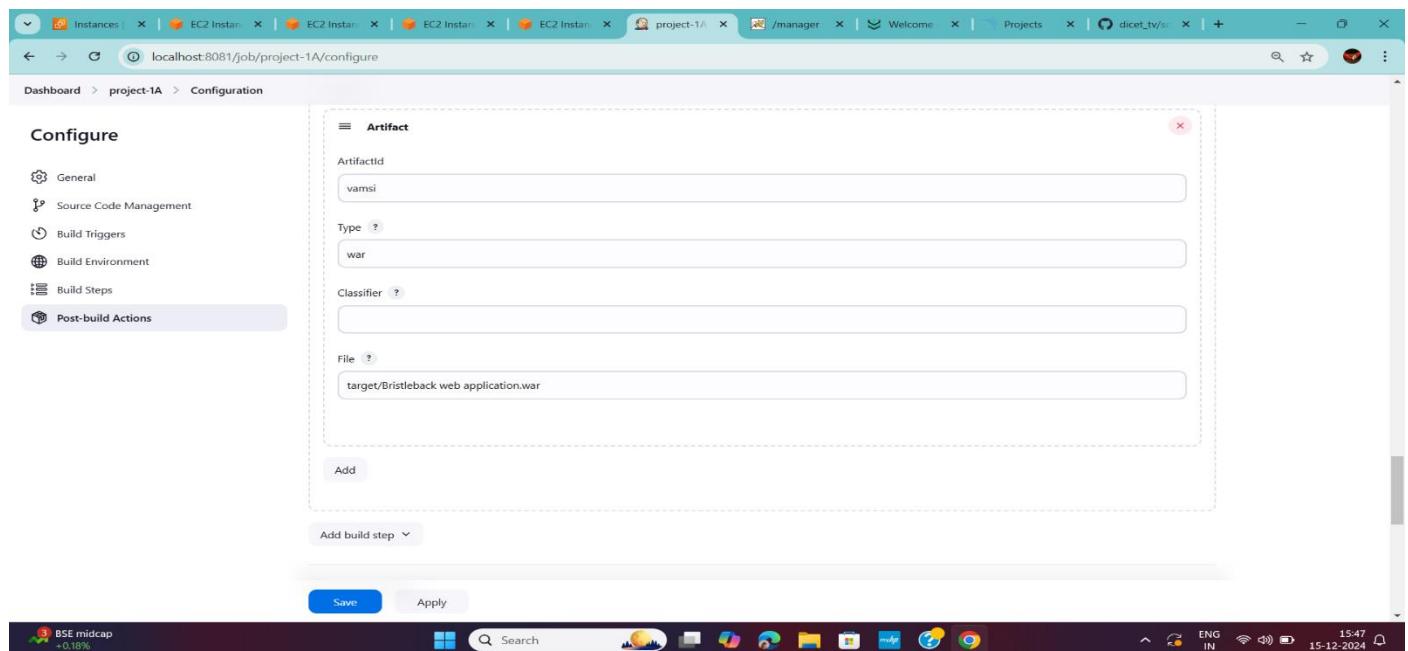
From pom.xml and add respective <distributionManagement> as required to pom.xml along with Nexus repository URL

```
<distributionManagement>
  <snapshotRepository>
    <id>nexus-snapshots</id>
    <url>http://3.110.164.242:8081/repository/maven-snapshots/</url>
  </snapshotRepository>
</distributionManagement>
```

1. Nexus Repository: <Nexus URL>.
2. Repository ID: <repository-id>.
3. Group ID: <group-id>.
4. Artifact ID: <artifact-id>.
5. Version: <version>.
6. File: target/<artifact-name>.war.



The screenshot shows the Jenkins configuration interface for a project named 'project-1A'. The 'Configuration' section is open, specifically the 'Build Steps' tab. A modal window titled 'Nexus artifact uploader' is displayed. It contains fields for 'Nexus Version' (set to 'NEXUS3'), 'Protocol' (set to 'HTTP'), 'Nexus URL' (set to '3.110.164.242:8081'), 'Credentials' (set to 'admin/***** (nexus credentials)'), 'Groupid' (set to 'vamsi.maven.com'), 'Version' (set to '0.0.1-SNAPSHOT'), 'Repository' (set to 'maven-snapshots/'), and 'Artifacts' (set to 'target/Bristleback web application.war'). At the bottom of the modal are 'Save' and 'Apply' buttons.



The screenshot shows the Jenkins configuration interface for a project named 'project-1A'. The 'Configuration' section is open, specifically the 'Post-build Actions' tab. A modal window titled 'Artifact' is displayed. It contains fields for 'ArtifactId' (set to 'vamsi'), 'Type' (set to 'war'), 'Classifier' (empty), and 'File' (set to 'target/Bristleback web application.war'). At the bottom of the modal are 'Add' and 'Save' buttons. Below the modal, there is a 'Save' and 'Apply' button for the main configuration page.

Step 5: Deploy WAR File to Tomcat

1. Add a post-build action: Deploy WAR/EAR to a container.

1. WAR/EAR File: target/<artifact-name>.war.
2. Context Path: /project.
3. Container: Apache Tomcat.
 1. Provide the Tomcat server details and credentials configured in Jenkins.

The screenshot shows the Jenkins configuration interface for a job named 'project-1A'. The 'Post-build Actions' section is active, displaying a 'Deploy war/ear to a container' step. Within this step, the 'Containers' section is expanded, showing a 'Tomcat 9.x Remote' configuration. This configuration includes a 'Credentials' dropdown set to 'admin/*** (tomcat credentials)', a 'Tomcat URL' input field containing 'http://43.204.130.230:8080', and an 'Advanced' dropdown. At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

➤ Execution and Verification

1. Trigger the Jenkins job manually.

The screenshot shows the Jenkins dashboard for the 'project-1A' job. The job status is 'Passed'. The 'Builds' section shows the most recent build (#10) was successful. A SonarQube Quality Gate card indicates 'Passed' with a green status bar. The SonarQube icon has a blue ribbon banner above it. The dashboard also includes sections for Status, Changes, Workspace, Build Now, Configure, Delete Project, Job Config History, SonarQube, and Rename.

2. Monitor the console output for each step. #log information is attached in below link

<https://drive.google.com/file/d/1GMSgEZE3XDHUkrFDGjtH-Rk3tshEGzOb/view?usp=sharing>

3. Validate the following:

1. Code is cloned from the GitHub repository.
2. Build is successful, and the artifact is generated.
3. Code analysis results are updated in the SonarQube dashboard.

The screenshot shows the SonarQube interface. On the left, there's a sidebar with filters for Quality Gate (Passed, Failed), Reliability (A-E rating), Security (A-E rating), and Security Review (≥ 80%, 70%-80%, 0%). The main area displays two projects: 'cake' (Passed) and 'project1' (not yet analyzed). The 'cake' project details are as follows:

Category	Value
Bugs	56 (B)
Vulnerabilities	0 (A)
Hotspots Reviewed	0.0% (E)
Code Smells	0 (A)
Coverage	0.0% (Red)
Duplications	4.0% (Green)
Lines	16k (M) CSS, HTML, XML

A yellow warning box states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine."

At the bottom, it says "SonarQube™ technology is powered by SonarSource SA Community Edition - v9.9.8 (build 100196) - LGPLv3 - Community - Documentation - Plugins - Web API".

The screenshot shows the SonarQube dashboard for the 'cake' project. At the top, it says "Last analysis of this Branch had 3 warnings December 15, 2024 at 3:53 PM Version 1.0". Below that, there are tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity. The Overview tab is selected.

The Quality Gate Status is **Passed**. The Overall Code section shows "New Code Since 0.0.1-SNAPSHOT... Started 2 days ago".

The Measures section lists the following:

Measure	Value	Status
New Bugs	0	Reliability (A)
New Vulnerabilities	0	Security (A)
New Security Hotspots	0	Security Review (A)
Added Debt	0	Maintainability (A)
New Code Smells	0	(Green)

At the bottom, it says "SonarQube™ technology is powered by SonarSource SA Community Edition - v9.9.8 (build 100196) - LGPLv3 - Community - Documentation - Plugins - Web API".

4. Artifact is uploaded to the Nexus repository.

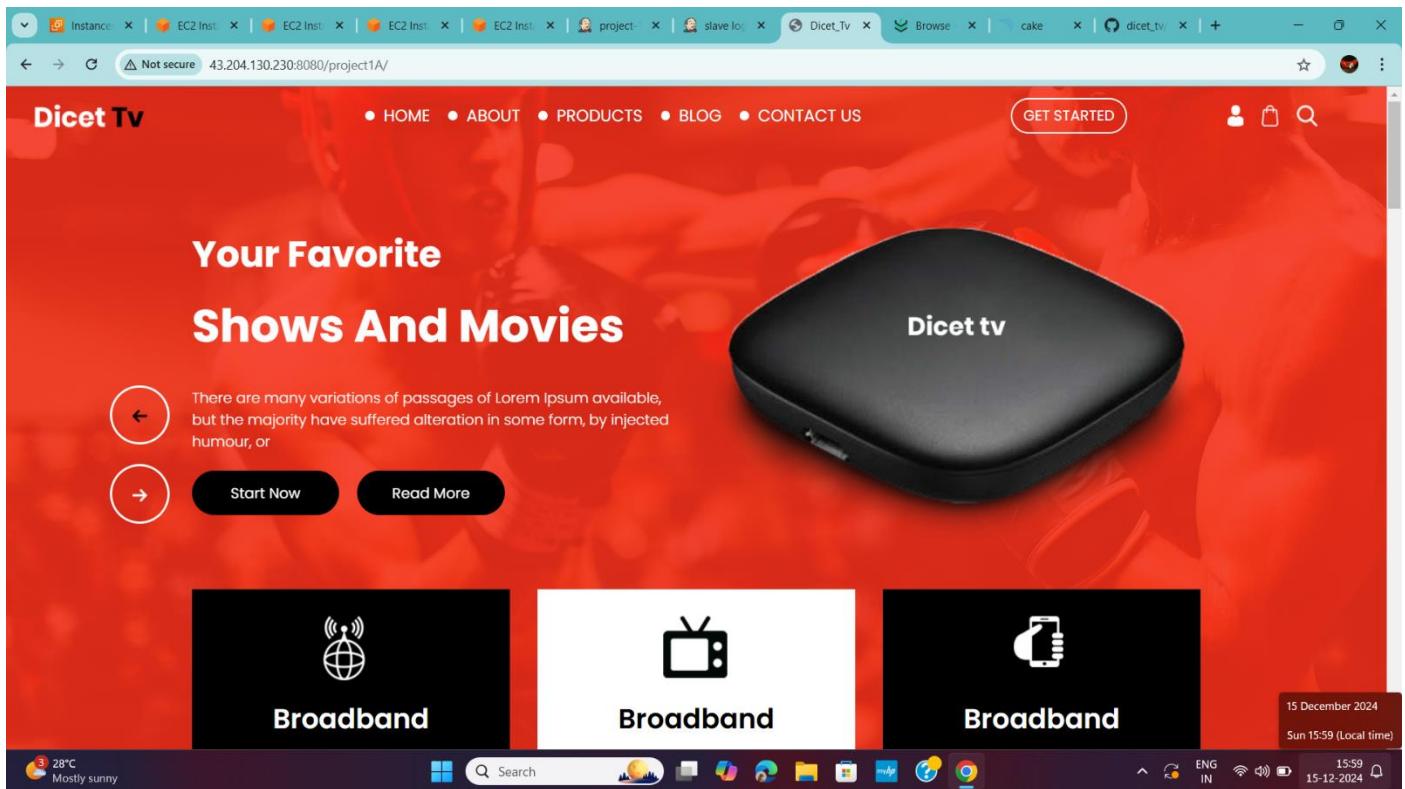
The screenshot shows the Sonatype Nexus Repository interface. The left sidebar has a green 'Browse' button selected. The main area shows a tree view under 'vamsi/maven/com/vamsi/0.01-SNAPSHOT'. Inside this directory are several artifacts: 0.01-20241211.172813-1, 0.01-20241212.105729-2, 0.01-20241212.120750-3, 0.01-20241212.122055-4, 0.01-20241215.102351-5, maven-metadata.xml, maven-metadata.xml.md5, and maven-metadata.xml.sha1. A message at the bottom says: 'Sonatype will start to collect anonymous, non-sensitive usage metrics and performance information to shape the future of Nexus Repository. Learn more about the information we collect or decline.' An 'OK' button is present. The taskbar at the bottom shows weather (28°C, mostly sunny), system icons, and the date (15-12-2024).

5. The WAR file is deployed to the Tomcat server.

The screenshot shows the Tomcat Web Application Manager interface. At the top, it says 'Tomcat Web Application Manager'. Below that is a message box: 'Message: OK'. The main area is titled 'Manager' and contains tabs for 'List Applications', 'HTML Manager Help', 'Manager Help', and 'Server Status'. The 'Applications' section lists the following deployed applications:

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/project-1B	None specified	Bristleback Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/project1A	None specified	Bristleback Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

At the bottom, there's a 'Deploy' section with a 'Deploy directory or WAR file located on server' input field and a 'Context Path' dropdown. The taskbar at the bottom shows weather (28°C, mostly sunny), system icons, and the date (15-12-2024).



Create a Pipeline Project in Jenkins

➤ Overview

This Jenkins Pipeline job is designed to automate the tasks needed to perform for *project-1*. It utilizes the master-slave architecture and integrates tools like Git, Maven, SonarQube, Nexus, and Apache Tomcat to achieve the following tasks:

➤ Tasks Overview

- Assign Job to a Node using Label
- Clone the GitHub repository.
- Build the project and create Maven artifacts.
- Perform code analysis using SonarQube.
- Upload the artifact to a Nexus repository.
- Deploy the WAR file to a Tomcat server

➤ Pre-requisites

Create a Node using an AWS Instance, using SSH connection with Private key and make agent online.

Jenkins Setup: A running Jenkins instance with the following plugins:

- Git
- Maven Integration
- SonarQube Scanner
- Nexus Artifact Uploader
- Deploy to Container

MAKE PLUGINS INSTALLATION SAME AS ABOVE FREE_STYLE PROJECT & CONFIGURE TOOLS (SonarQube, Nexus, Tomcat) With respected Credentials as Above

1. Create a New Pipeline Project

- On the Jenkins dashboard, click **New Item**.
- Enter a name for your project (e.g. project-1B).
- Select **Pipeline** as the project type and click **OK**.

The screenshot shows the Jenkins 'New Item' creation interface. The 'item name' field is filled with 'project_1B'. The 'item type' dropdown is open, showing four options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', and 'Folder'. The 'Pipeline' option is highlighted. At the bottom, there is a large blue 'OK' button.

2. Configure General Settings and Triggers

- Add a description if needed.
- I not made any changes in general settings.
- I did not use any of Build Trigger

The screenshot shows the Jenkins 'Configuration' page for the 'project-1B' job. The 'General' tab is selected. The 'Description' field contains the following text:
project-1B
Create a pipeline - (Make slave to do task for master)
Task 1.1 - To clone a GitHub Repository
Task 1.2 - Create Artifact(achieve maven targets)
Task 1.3 - Need to analyze code using SonarQube
Task 1.4 - Need to save a snapshot of artifact in nexus
Task 1.5 - Need to deploy war file into a tomcat server

Below the description, there is a list of build triggers, all of which are currently unchecked:

- Discard old builds
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override
- Preserve stashes from completed builds
- This project is parameterised
- Throttle builds

At the bottom, there are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration interface for a project named 'project-1B'. In the 'Build Triggers' section, several options are listed with checkboxes: 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', 'Poll SCM', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'. Below this is the 'Advanced Project Options' section with an 'Advanced' dropdown menu. The main focus is the 'Pipeline' section, which is currently active. It contains a 'Definition' tab with 'Save' and 'Apply' buttons. The pipeline definition area is empty, indicated by a large grey box.

3. Set Up Pipeline Definition

- Scroll down to the **Pipeline** section.
- Choose how you want to define the pipeline:
 - **Pipeline Script:** Enter the pipeline script directly in the Jenkins UI.
 - **Pipeline Script from SCM:** Use a version control system (e.g., Git) to manage the Jenkinsfile

I used Pipeline Script (Declarative Format) with Five Stages

I used pipeline syntax - **Directive Generator** and **Snippet Generator** for generating Pipeline Script

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator. On the left, there's a sidebar with links: Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main area is titled 'Overview' and contains a note about the Snippet Generator. It shows a dropdown for 'Sample Step' with 'git:Git' selected. Below it are fields for 'Repository URL' (set to 'https://github.com/DHANASRI-SUNKARI/dice_tv.git'), 'Branch' (set to 'master'), and 'Credentials' (set to '+ none'). At the bottom, there are two checked checkboxes: 'Include in polling?' and 'Include in changelog?'. A 'Generate Pipeline Script' button is located at the very bottom.

The screenshot shows the Jenkins Directive Generator interface. On the left, there's a sidebar with links like Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSDL. The main area has a title 'Overview' and a sub-section 'Directives'. Under 'Directives', there's a 'Sample Directive' section with a dropdown menu set to 'agent: Agent'. Below it, there's a note about the 'agent' directive and a detailed configuration form for 'label: Run on an agent matching a label'. The 'Label' field contains 'dev', and the 'Retry Count' field contains '1'. At the bottom, a blue button says 'Generate Declarative Directive', and below it is the generated Groovy code:

```
agent {  
    label 'dev'  
}
```

➤ Pipeline Script: (Descriptive)

The screenshot shows the Jenkins Configuration page for a project named 'project-1B'. The top navigation bar includes links for Instant, EC2, project, slave, Dicet, Browse, cake, DHAN, and log out. The main content area has a title 'Configuration' and a 'Configure' section with checkboxes for 'Quiet period' and 'Trigger builds remotely'. Below this is a 'General' section with links for Advanced Project Options and Pipeline. The 'Pipeline' section is expanded, showing a 'Definition' tab with a dropdown menu set to 'Pipeline script'. A large text area displays the pipeline script code, which includes environment variables and stages. A checkbox 'Use Groovy Sandbox' is checked. At the bottom are 'Save' and 'Apply' buttons.

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. The left sidebar contains links for Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA DSL. The main content area has a heading 'Overview' with a sub-section 'Steps'. A 'Sample Step' dropdown is set to 'git'. Below it, a 'git' step configuration is shown with fields for 'Repository URL' (set to 'https://github.com/DHANASRI-SUNKARI/dicet_tv.git'), 'Branch' (set to 'master'), and 'Credentials' (set to 'none'). There are also checkboxes for 'Include in polling?' and 'Include in changelog?'. A blue 'Generate Pipeline Script' button is at the bottom. The status bar at the bottom shows system information like weather (28°C Sunny), date (15-12-2024), and time (16:09).

This screenshot shows the same Jenkins Pipeline Syntax Snippet Generator interface after generating the pipeline script. The generated code is displayed in a large text area at the bottom: 'git \'https://github.com/DHANASRI-SUNKARI/dicet_tv.git\''. The rest of the interface is identical to the first screenshot, including the sidebar links and the step configuration.

The screenshot shows the Jenkins Directive Generator interface. On the left, a sidebar lists various Jenkins tools: Snippet Generator, Declarative Directive Generator (selected), Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main content area has a heading 'Overview' and a detailed description of the 'Directive Generator'. It explains how to generate Pipeline code for Declarative Pipeline directives like 'agent', 'options', 'when', etc. A 'Generate Declarative Directive' button is present. Below it, a code editor shows the generated Pipeline code for the 'tools' directive, specifically 'maven'. The code is:`tools {
 maven 'maven'
}`



The screenshot shows the Jenkins Pipeline Syntax page. The sidebar on the left includes Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main content area has a heading 'Overview' and a detailed description of the 'Snippet Generator'. It explains how to learn Pipeline Script code for defining steps. A 'Generate Pipeline Script' button is present. Below it, a code editor shows the generated Pipeline script for the 'sh' step, which is 'mvn clean package'. The code is:`sh 'mvn clean package'`



➤ SonarQube

```
environment {
    SONAR_SERVER = 'sonarqube' // Name of the SonarQube instance in Jenkins configuration
    SONAR_PROJECT_KEY = 'temp' // Replace with your unique SonarQube project key
    SONAR_TOKEN = credentials('sonarqube_analysis') // Jenkins credential ID for SonarQube token
}

script {
    withSonarQubeEnv(SONAR_SERVER) {
        sh """
            mvn sonar:sonar \
                -Dsonar.projectKey=${SONAR_PROJECT_KEY} \
                -Dsonar.host.url=${env.SONAR_HOST_URL} \
                -Dsonar.login=${SONAR_TOKEN}
        """
    }
}
```

The screenshot shows a Jenkins Pipeline Syntax configuration page. The URL in the browser is `localhost:8081/job/project-1B/pipeline-syntax/`. The configuration includes a 'nexusArtifactUploader' step with the following details:

- Nexus Details**:
 - Nexus Version: NEXUS3
 - Protocol: HTTP
 - Nexus URL: 3.110.164.242:8081
 - Credentials: admin/******** (nexus credentials)
- Groupid**: vamsi.maven.com
- Version**: 0.0.1-SNAPSHOT
- Repository**: maven-snapshots
- Artifacts**: (empty)

The screenshot shows a Jenkins Pipeline Syntax configuration page. The URL in the browser is `localhost:8081/job/project-1B/pipeline-syntax/`. The configuration includes a 'deploy' step with the following details:

- Sample Step**: deploy: Deploy war/ear to a container
- deploy**:
 - WAR/FAR files: **/*.war
 - War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like **/*/*.war
- Context path**: project-1B
- Containers**:
 - Tomcat 9.x Remote**:
 - Credentials: admin/******** (tomcat credentials)
- Tomcat URL**: http://43.204.120.230:8080

➤ Jenkinsfile :

```
pipeline {
    agent {label 'dev'}
    tools { maven 'maven'}
    environment {

        // SonarQube environment variables
        SONAR_SERVER = 'sonarqube' // Name of the SonarQube instance in Jenkins configuration
        SONAR_PROJECT_KEY = 'temp' // Replace with your unique SonarQube project key
        SONAR_TOKEN = credentials('sonarqube_analysis') // Jenkins credential ID for SonarQube token

    }

    stages {
        stage('stage-1 GIT') {
            steps {
                git 'https://github.com/DHANASRI-SUNKARI/dicet_tv.git'
            }
        }
        stage('stage-2 MAVEN'){
            steps{
                sh 'mvn clean package'
            }
        }
        stage('stage-3 SONARQUBE'){
            steps{
                script {
                    echo 'Running SonarQube analysis...'
                    withSonarQubeEnv(SONAR_SERVER) {
                        sh """
                            mvn sonar:sonar \
                            -Dsonar.projectKey=${SONAR_PROJECT_KEY} \
                            -Dsonar.host.url=${env.SONAR_HOST_URL} \
                            -Dsonar.login=${SONAR_TOKEN}
                        """
                    }
                }
            }
        }
        stage('stage-4 NEXUS'){
            steps{
                nexusArtifactUploader artifacts: [[artifactId: 'vamsi', classifier: '', file: 'target/Bristleback web application.war', type: 'war']], credentialsId: 'nexus_credentials', groupId: 'vamsi.maven.com', nexusUrl: '13.201.227.14:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots/', version: '0.0.1-SNAPSHOT'
            }
        }
        stage('stage-5 TOMCAT'){
            steps{
                deploy adapters: [tomcat9(credentialsId: 'tomcat_server', path: "", url: 'http://65.0.131.250:8080')], contextPath: 'project-1B', war: '**/*.war'
            }
        }
    }
}
```

4. Save the Pipeline

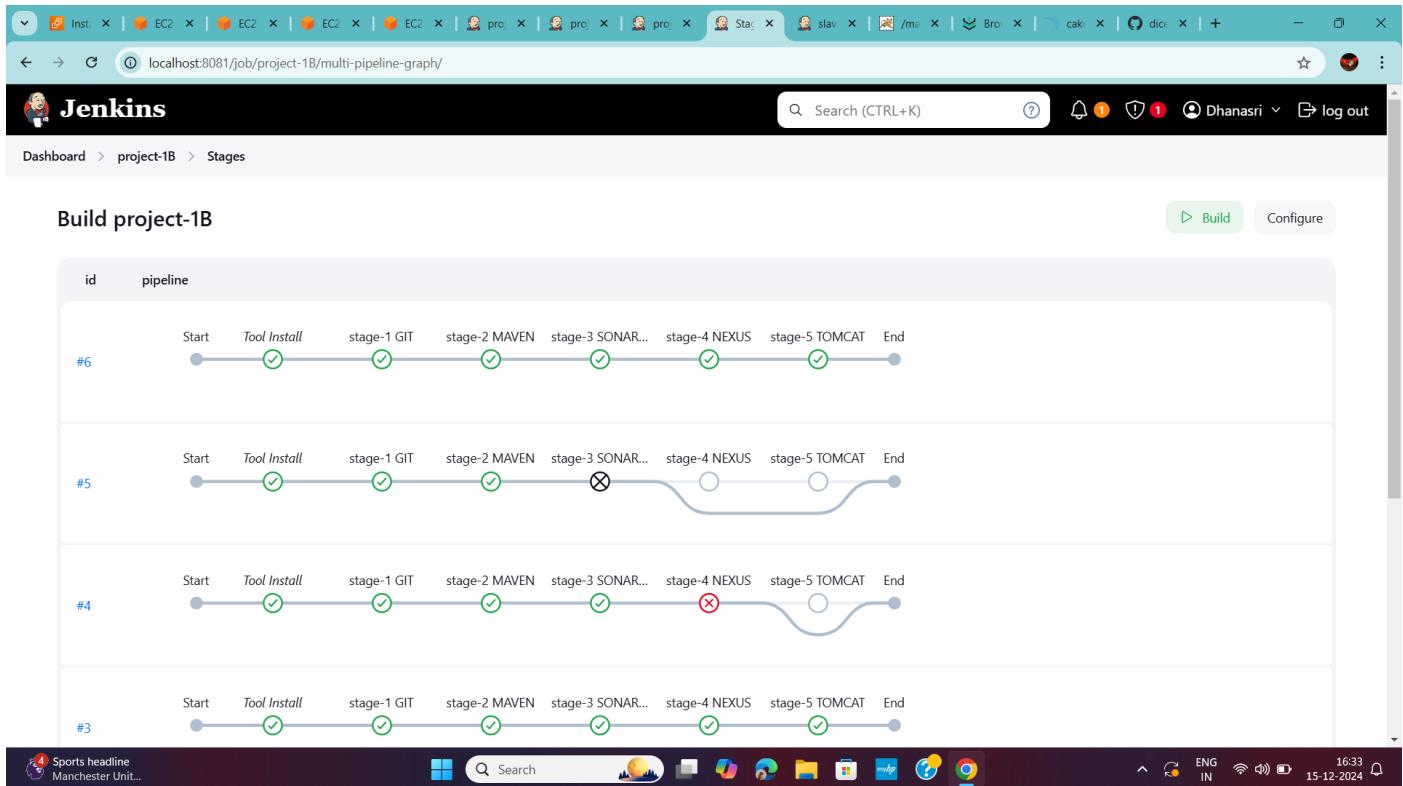
- Click **Save** to create the pipeline.

5.Run the Pipeline

- Navigate to the project dashboard.
- Click **Build Now** to run the pipeline.

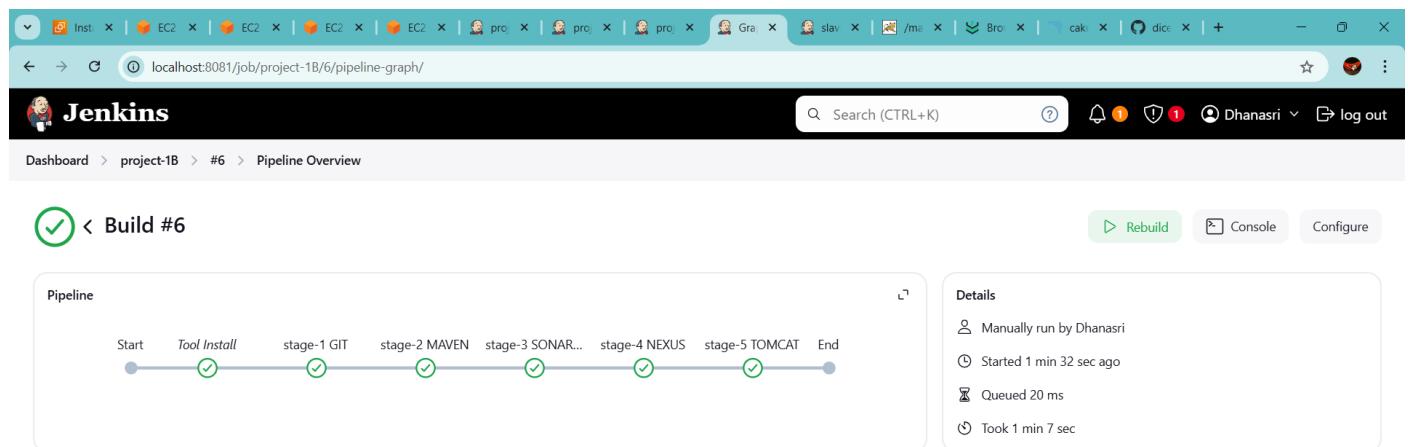
6.View Pipeline Results

- Click on the build number to view logs and results.
- Click on stages to find information about stages



7.Execution and Verification

After Successful Build we can find (Same output as above Free Style Job)



- Code is cloned from the GitHub repository.
- Build is successful, and the artifact is generated.
- Code analysis results are updated in the SonarQube dashboard.
- Artifact is uploaded to the Nexus repository.
- The WAR file is deployed to the Tomcat server

Monitor the console output for each step. #log information is attached in below link

<https://drive.google.com/file/d/1czTZKfh5T3cAFcNHXCq1eE0JuJmEvT1O/view?usp=sharing>

Troubleshooting

Common Issues

1. **Git Clone Error:** Check credentials and repository URL.
2. **Maven Build Failure:** Review the build logs and resolve compilation issues.
3. **SonarQube Issues:** Verify the SonarQube server configuration and project properties.
4. **Nexus Upload Failure:** Ensure correct credentials and repository configuration.
5. **Tomcat Deployment Error:** Validate the Tomcat server's user roles and permissions.

SUBMITTED BY:

- **DHANASRI SUNKARI**