



CICD PIPELINE BY USING DECLARATIVE SCRIPT IN JENKINS

- ❖ By using pipeline script, we can integrate the different tools in devops.
- ❖ The time will be saving in developing the project.
- ❖ Continuously integrate and continuously deploy the project by using pipeline script.

JENKINS SHOULD HAVE A NODE:

- For the load and work division purpose we can use the node or agent.
- **Dashboard>manage Jenkins>nodes**, here we can add the node and node credentials also.
- In script we can take the agent following script.

```
pipeline {  
    agent any  
    (or)  
    pipeline {  
        agent {label 'dev'}
```

- If we give agent any the active node will take automatically.

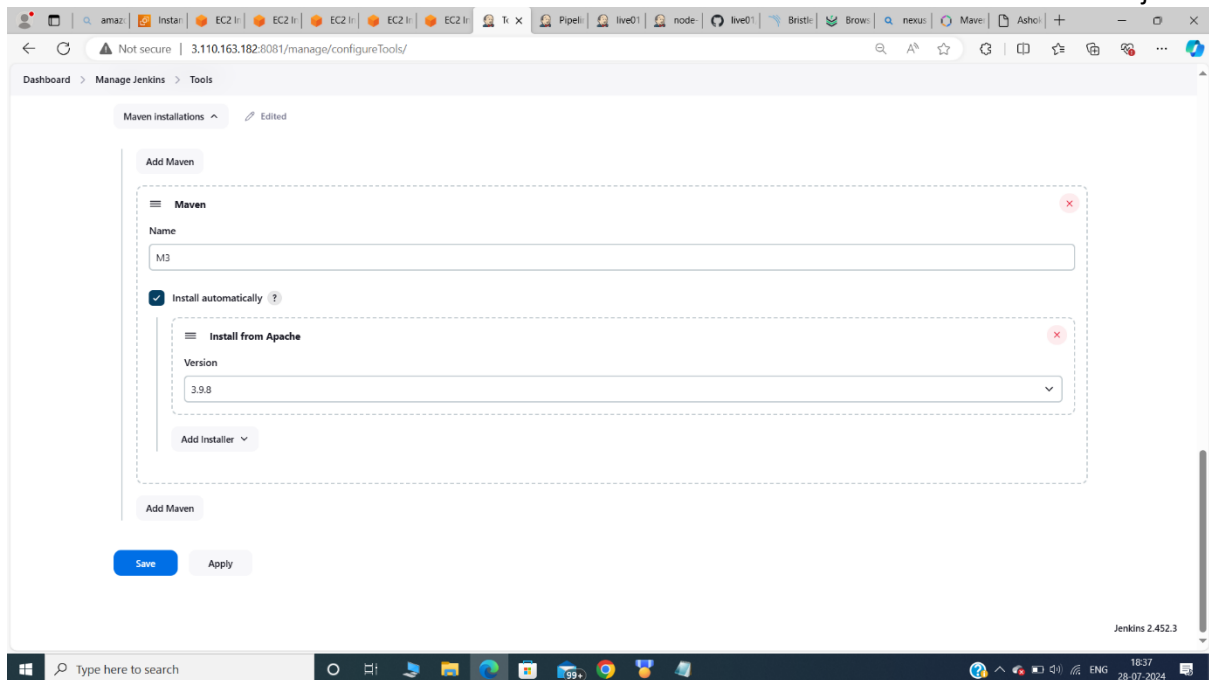
GET THE REPOSITORY CODE FROM GIT HUB:

- First, we can start the Jenkins server and select the new item.
- Give name and select the pipeline option and click on ok.
- The declarative pipeline starts with (**pipeline {**).
- By using git plugin, we can add the GIT repository details as requirements shown.
- Second stage we can do the build steps here we can add the maven like following the steps.
- For deployment we can add the tools below the pipeline as shown in the fig.
- Now go to the dashboard>manage Jenkins>tools here, we can add the maven details here name and version and same name add the tools.

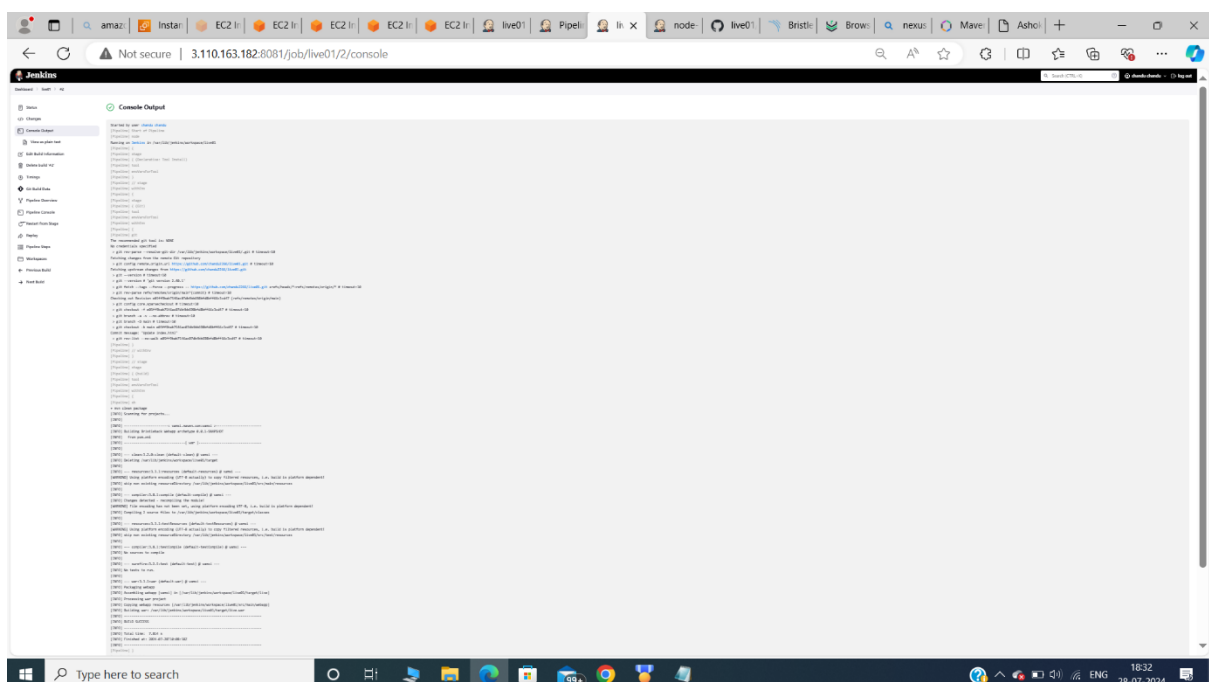


Jenkins

Project-1



- Now we click on save and click on build now the out put will be shown in the console output.



INTEGRATE WITH THE SONARQUBE TOOL:

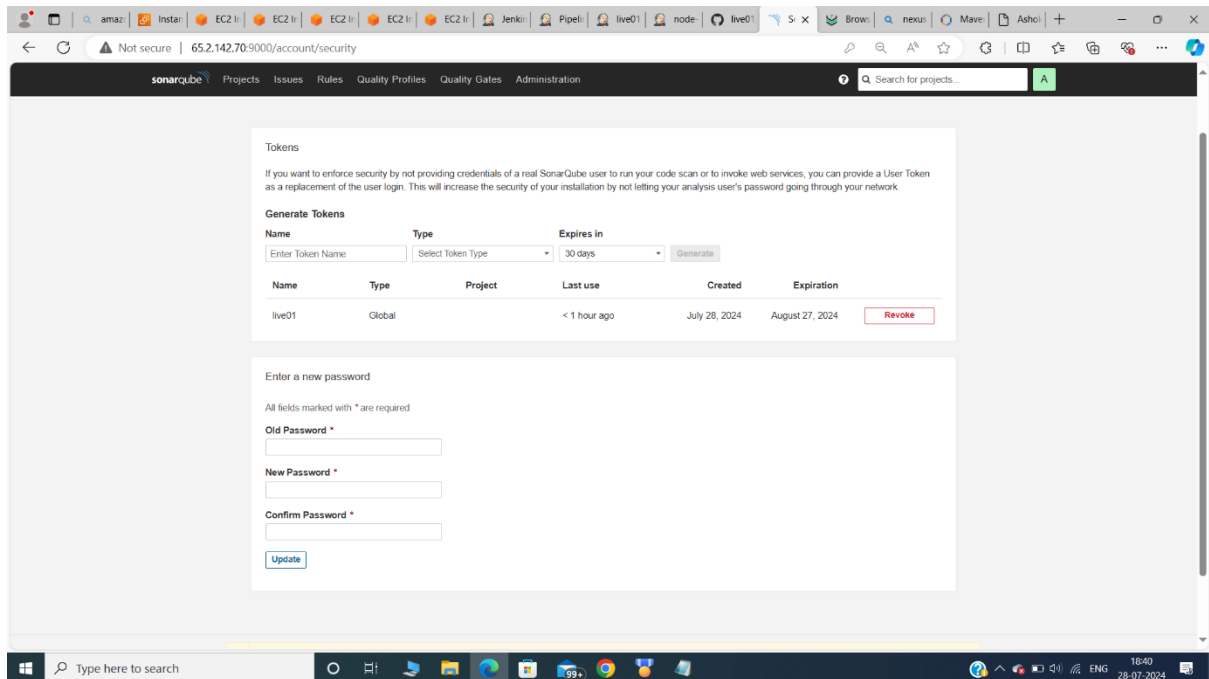
- For this integration we can take the different server and download the SonarQube.
- Start the SonarQube server with allocated port number and login the credentials.



Jenkins

Project-1

- Here we can create the token in the SonarQube.
- Click on **administration>security>users**.



- Here we can create the token copy the token, paste in the node pad for the further use.
- Now go to Jenkins we can download the **SonarQube Scanner plugin**.
- **Dashboard>manage Jenkins>plugins.**
- Now we can go to the Jenkins add the SonarQube details in the **dashboard>manage Jenkins>system**.
- Under the SonarQube servers we can add the details about SonarQube as show in the fig.
- We can add the token of the SonarQube in the **dashboard>manage Jenkins>credentials** and click on global credentials and next click on the add credentials as show in the image.

Note: here we can add the SonarQube token by using secret text in kind.



Jenkins

Project-1

Not secure | 3.110.163.182.8081/manage/pluginManager/installed

Jenkins

Search (CTRL+Q) | chandu chandu | log out

Dashboard > Manage Jenkins > Plugins

Plugins

sonar

Name	Enabled
SonarQube Scanner for Jenkins 3.17.2 This plugin allows an easy integration of SonarQube, the open source platform for Continuous inspection of code quality. Report an issue with this plugin	<input checked="" type="checkbox"/>

Updates 2
Available plugins
Installed plugins
Advanced settings

REST API Jenkins 2.452.3

Type here to search

Not secure | 3.110.163.182.8081/manage/configure

Dashboard > Manage Jenkins > System

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

SonarQube installations

List of SonarQube installations

Name

sonarqube-live01

Server URL

Default is http://localhost:9000

http://65.2.142.70:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

live01-sonartoken

+ Add

Advanced

Save Apply

Type here to search



Jenkins

Project-1

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > live01-sonartoken

Update credentials

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

Secret
.....

ID ?
live01

Description ?
live01-sonartoken

Save

REST API Jenkins 2.452.3

- Now we can go to the git hub -we selected git repository(eg:live01) add the plugin in the plugin management.as shown in the fig.

```
48 <version>3.8.1</version>
49 <scope>test</scope>
50 </dependency>
51
52 </dependencies>
53
54 <build>
55   <finalName>live</finalName>
56   <pluginManagement>
57     <plugins>
58       <plugin>
59         <groupId>org.apache.maven.plugins</groupId>
60         <artifactId>maven-compiler-plugin</artifactId>
61         <version>3.8.1</version>
62         <configuration>
63           <source>1.8</source>
64           <target>1.8</target>
65         </configuration>
66       </plugin>
67       <plugin>
68         <groupId>vams1.maven.com</groupId>
69         <artifactId>sonar-maven-plugin</artifactId>
70         <version>4.0.0.4121</version>
71       </plugin>
72     </plugins>
73   </pluginManagement>
74   <plugins>
75     <plugin>
76       <groupId>org.apache.maven.plugins</groupId>
77       <artifactId>maven-war-plugin</artifactId>
78       <version>3.3.1</version>
79     </plugin>
80     <plugin>
81       <groupId>org.mortbay.jetty</groupId>
82       <artifactId>jetty-maven-plugin</artifactId>
83       <version>8.1.10.v20130312</version>
84     </plugin>
85   </plugins>
86 </build>
87
88 </project>
```

- Now come to the job click on configure, here we can add the next stage for SonarQube.
- Click on pipeline syntax select the with SonarQube scanner plugin add the requirement details. click on generate pipeline script. copy the script and paste in script.
- Here we can add some line as shown in the image.



Jenkins

Project-1

```
live01-declarative pipeline script.txt - Notepad
File Edit Format View Help
}
stages {
  stage('Git') {
    steps {
      git branch: 'main', url: 'https://github.com/chandu2266/Live01.git'
    }
  }
  stage('build') {
    steps {
      sh 'mvn clean package'
    }
  }
  stage('code qualirt analysis') {
    steps {
      withSonarQubeEnv(installationName: 'sonarqube-Live01', credentialsId: 'Live01') {
        sh 'mvn sonar:sonar'
      }
    }
  }
  stage('upload to nexus ') {
    steps {
      nexusArtifactUploader artifacts: [[artifactId: 'vamsi', classifier: '\\', file: 'target/Live.war', ty
    }
  }
  stage('deploy') {
    steps {

```

- Installation name is taken which given for the system settings for SonarQube server.
- After completed the script click apply and save. Click on build now. The build we success as shown in the fig.

```
Dashboard > live01 > #3
[INFO] 18:30:35.763 0 source files to be analyzed
[INFO] 18:30:35.861 0/0 source files have been analyzed
[INFO] 18:30:35.861 Sensor jac Docker Sensor [jac] (done) | time=143ms
[INFO] 18:30:35.867 Run sensors on project
[INFO] 18:30:36.806 Sensor Analysis Warnings Import [csharp] (done) | time=0ms
[INFO] 18:30:36.807 Sensor Analysis Warnings Import [csharp] (done) | time=0ms
[INFO] 18:30:36.807 Sensor Zero Coverage Sensor
[INFO] 18:30:36.825 Sensor Zero Coverage Sensor (done) | time=13ms
[INFO] 18:30:36.825 Sensor Java CPD Block Indexer
[INFO] 18:30:36.851 Sensor Java CPD Block Indexer (done) | time=26ms
[INFO] 18:30:36.861 SCM Publisher SCM provider for this project is: git
[INFO] 18:30:36.864 SCM Publisher SCM provider for this project is: git
[INFO] 18:30:41.518 SCM Publisher 98/98 source files have been analyzed (done) | time=5442ms
[INFO] 18:30:41.559 CPD Executor 2 files had no CPD blocks
[INFO] 18:30:41.559 CPD Executor Calculating CPD for 8 files
[INFO] 18:30:41.708 CPD Executor CPD calculation finished (done) | time=143ms
[INFO] 18:30:41.984 Analysis report generated in 252ms, dir size=934.0 KB
[INFO] 18:30:42.436 Analysis report compressed in 440ms, zip size=318.9 KB
[INFO] 18:30:42.782 Analysis report uploaded in 34ms
[INFO] 18:30:42.798 ANALYSIS SUCCESSFUL, you can find the results at: http://65.2.142.79:9000/dashboard?id=vamsi.maven.com%3Avamsi
[INFO] 18:30:42.798 Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] 18:30:42.798 More about the report processing at http://65.2.142.79:9000/api/ce/task?id=X2M4SH8ZmWU7_y1741_
[INFO] 18:30:46.069 Analysis total time: 30.088 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 36.388 s
[INFO] Finished at: 2024-07-28T18:30:46Z
[INFO] -----
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Now we can go to SonarQube server refresh it the output we will see here. As shown in the fig.



Jenkins

Project-1

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

My Favorites All

Search by project name or key

Create Project

1 project(s) Perspective: Overall Status Sort by: Name

☆ Bristleback Webapp archetype **Passed** Last analysis: 23 minutes ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
11	0	0.0%	3	0.0%	15.0%	8.5k

1 of 1 shown

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.

SonarQube™ technology is powered by SonarSource SA
Community Edition - v9.9.6 (build 92038) - LGPL V3 - Community - Documentation - Plugins - Web API

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

Bristleback Webapp archetype main

Last analysis of this Branch had 1 warning July 28, 2024 at 6:23 PM Version 0.0.1

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code
Since 0.0.1-SNAPSHOT
Started 3 hours ago

Overall Code

New Bugs	New Vulnerabilities	New Security Hotspots	Added Debt	New Code Smells	Reliability	Security	Security Review	Maintainability
0	0	0	0	0	A	A	A	A

UPLOAD THE ARTIFACT TO NEXUS REPOSITORY:

- For nexus tool we can install the nexus in the new server and start it, open the nexus and login.
- Go to the settings click on create the repository here, and add the details as show in the fig.



Jenkins

Project-1

The screenshot shows the Sonatype Nexus Repository OSS 3.70.1-02 interface. The left sidebar contains navigation options: Administration, Repository, Repositories, Blob Stores, Proprietary Repositories, Content Selectors, Cleanup Policies, Routing Rules, Security, Privileges, Roles, Users, Anonymous Access, LDAP, Realms, and SSL Certificates. The main content area is titled 'Create Repository: maven2 (hosted)'. It includes a 'Name' field with the value 'live01', an 'Online' checkbox that is checked, and a 'Maven 2' section with 'Version policy' set to 'Release', 'Layout policy' set to 'Strict', and 'Content Disposition' set to 'Inline'. The 'Storage' section shows 'Blob store' as 'default' and 'Strict Content Type Validation' as checked. A 'Hosted' section is also visible at the bottom.

- After creating repository click on the copy and copy the URL and paste in notepad for the further use.

The screenshot shows the Sonatype Nexus Repository OSS 3.70.1-02 interface. The left sidebar contains navigation options: Welcome, Search, Browse, and Upload. The main content area is titled 'Browse Browse assets and components'. It displays a table with columns: Name, Type, Format, Status, URL, Health check, and Firewall Re... The table lists several repositories, including 'project-1'. A modal dialog box is open, displaying the URL 'http://3108.61142:8081/repository/project-1' and a 'Copy to clipboard: Ctrl+C, Enter' message. The dialog also includes a 'Close' button.

- Now come to the Jenkins we can install the plugin **nexus artifact uploader**. as shown in the fig.



Jenkins

Project-1

Now we can add the credentials in Jenkins for the nexus. here we can add the nexus login details (username and password).

Now we can edit the pom.xml file in the selected repository as shown in the fig.

[illegible]

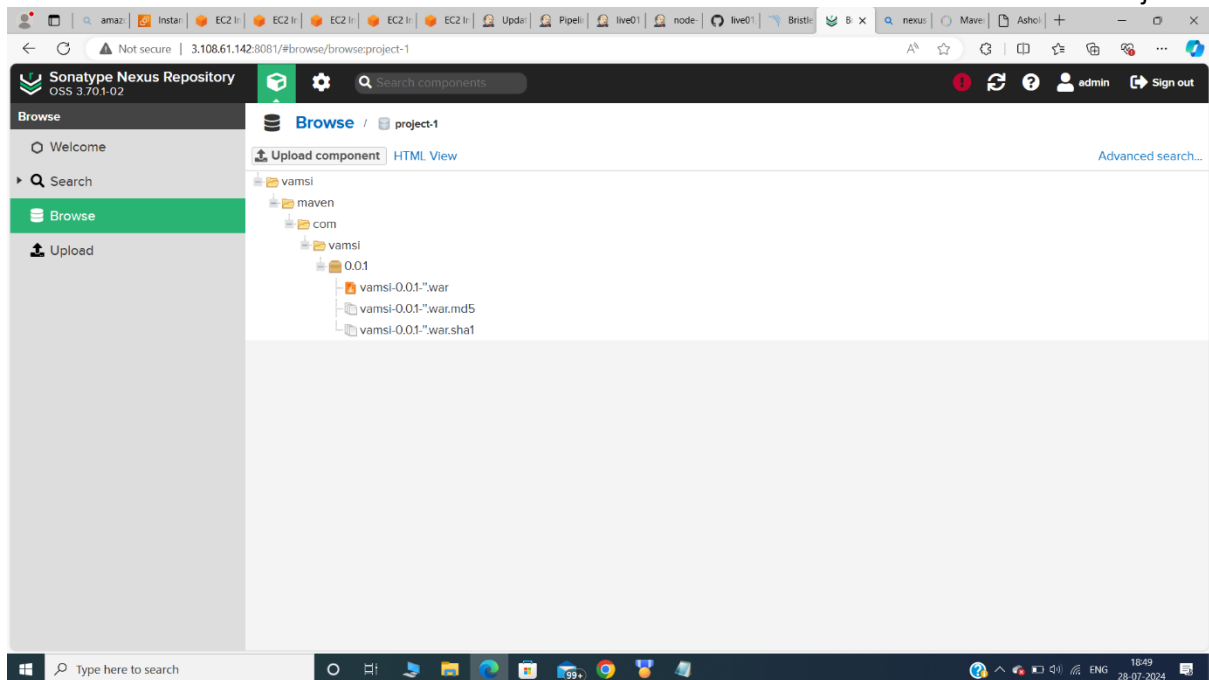
- ```
Dashboard > live01 > #6
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 24.744 s
[INFO] Finished at: 2024-07-28T12:22:14Z
[INFO]
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // withBov
[Pipeline] }
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
[Pipeline] tool
[Pipeline] mvnarsPortTool
[Pipeline] mvnw
[Pipeline] {
[Pipeline] newArtifactUploader
Uploading artifact live-war started....
GroupID: vamsi.sare.com
ArtifactName: vamsi
Classifier: ''
Type: war
Version: 0.0.1
File: live-war
RepositoryProjectId
uploading: https://180.41.142.1882/repository/project-1/vamsi/newcom/vamsi/0.0.1/vamsi-0.0.1.war
10 % completed (1.9 MB / 17 MB).
20 % completed (3.6 MB / 17 MB).
30 % completed (5.3 MB / 17 MB).
40 % completed (7.1 MB / 17 MB).
50 % completed (8.8 MB / 17 MB).
[Pipeline] }
[Pipeline] // withBov
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withBov
[Pipeline] }
[Pipeline] // mvnw
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Page | 10



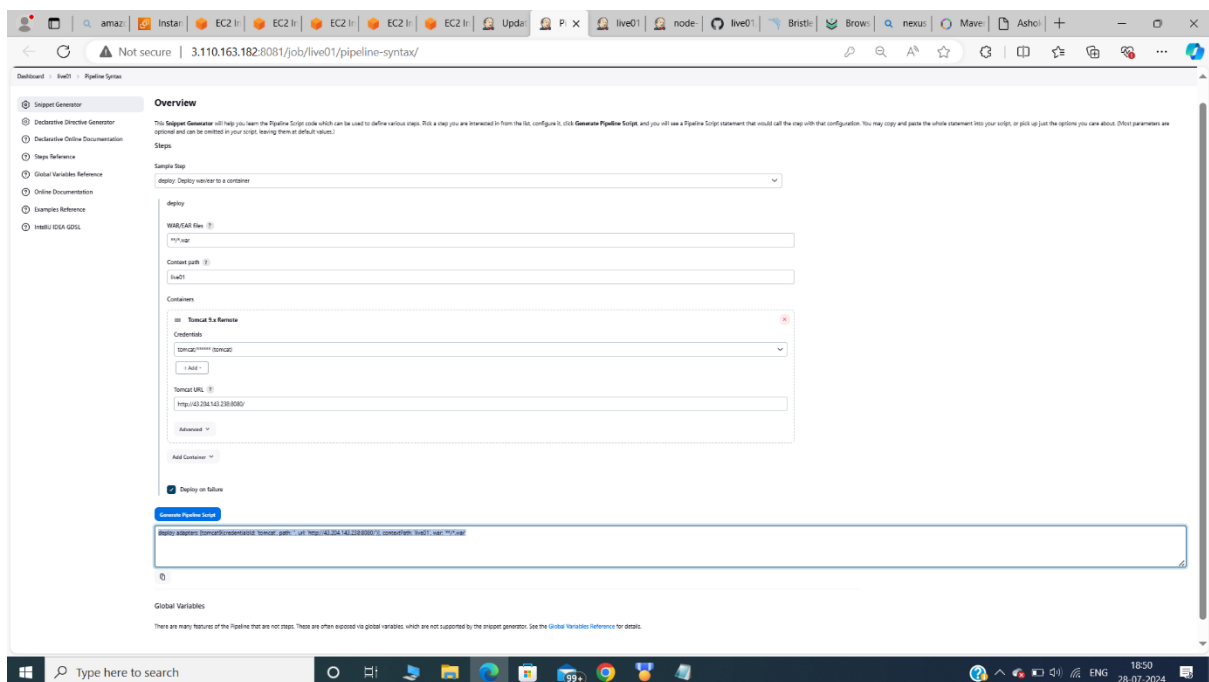
# Jenkins

Project-1



## DEPLOY THE ARTIFACT TO THE TOMCAT SERVER:

- Now we can take a new instance for the tomcat and install tomcat server.
- Add the permissions open the manager and GUI script options also.
- Now come to the Jenkins install the deploy plugin and add the tomcat credentials in credentials in the Jenkins.
- Now come to the job and configure it add the next stage for the deploy and click on the pipeline syntax select the deploy plugin and give the details for tomcat.
- Click on generate pipeline script. Copy it and paste in script.

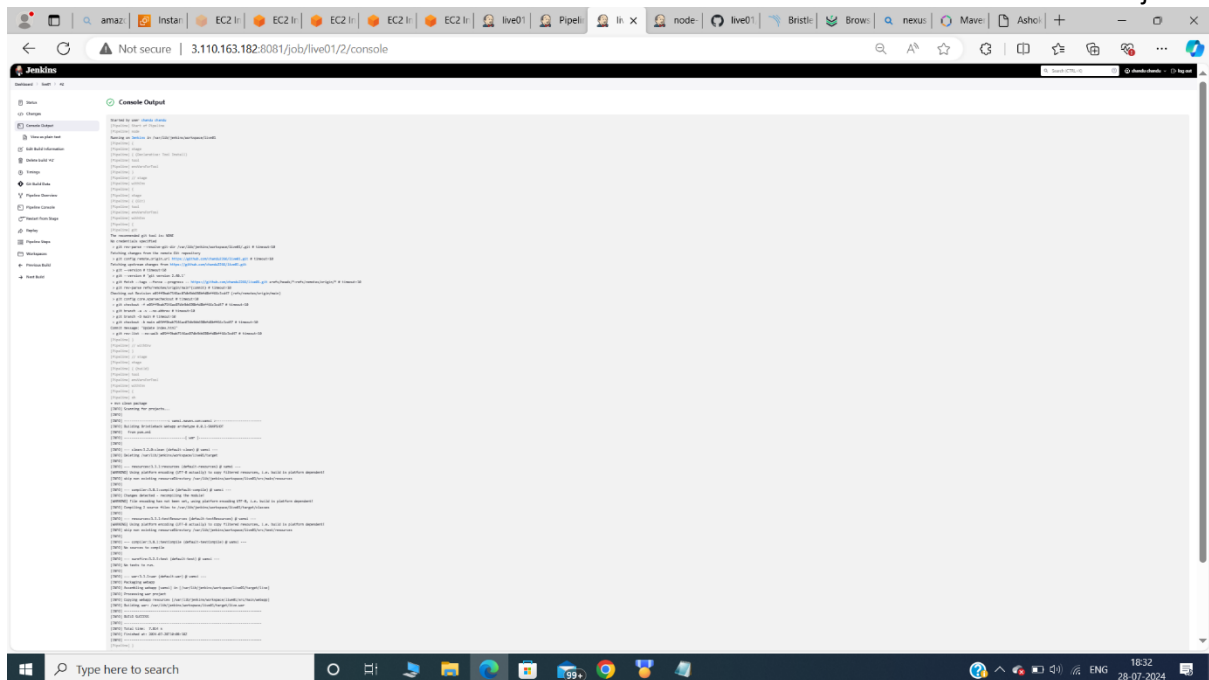


- Now click on build now after click on save and apply.

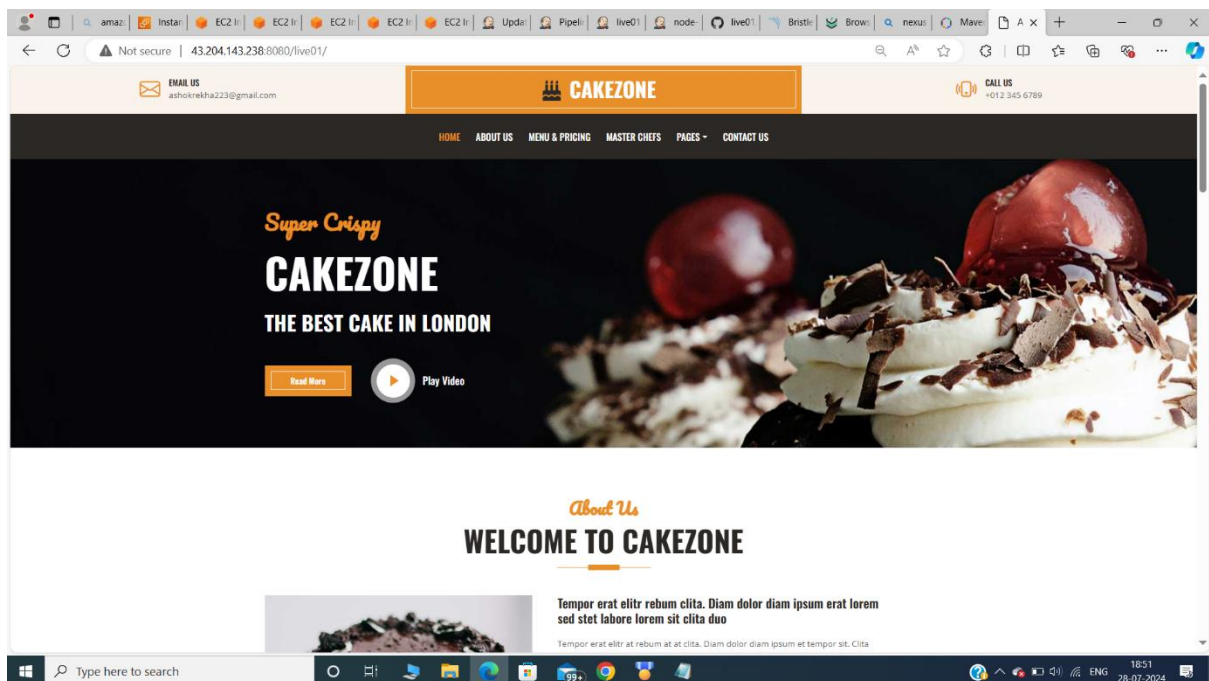


# Jenkins

Project-1



- Now go to the tomcat server add the path which giving in the tomcat script(eg:live01).click on enter.





# Jenkins

Project-1

- We can see the total script like this.

```
pipeline {
 agent any
 tools {
 maven 'M3'
 }
 stages {
 stage('Git') {
 steps {
 git branch: 'main', url: 'https://github.com/chandu2266/live01.git'
 }
 }
 stage('build') {
 steps {
 sh 'mvn clean package'
 }
 }
 stage('code qualirt analysis') {
 steps {
 withSonarQubeEnv(installationName: 'sonarqube-live01', credentialsId: 'live01') {
 sh 'mvn sonar:sonar'
 }
 }
 }
 stage('upload to nexus ') {
 steps {
 nexusArtifactUploader artifacts: [[artifactId: 'vamsi', classifier: '\'', file: 'target/live.war',
type: 'war']], credentialsId: 'nexus', groupId: 'vamsi.maven.com', nexusUrl: '3.108.61.142:8081',
nexusVersion: 'nexus3', protocol: 'http', repository: 'project-1', version: '0.0.1'
]
 }
 }
 stage('deploy') {
 steps {
 deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url:
'http://43.204.143.238:8080/)], contextPath: 'live01', war: '**/*.war'
]
 }
 }
}
```