

Sumyak Jain, 500068360, R171218106

Experiment No 10

Jenkins Integration with Nexus

STEP-1) open Jenkins

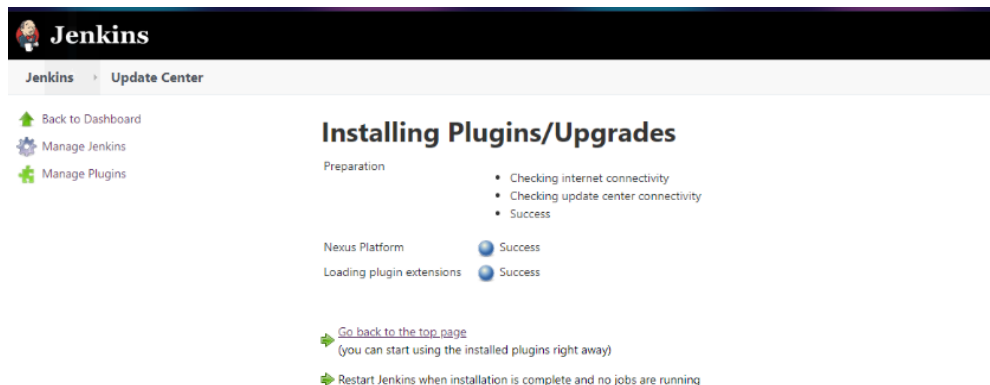
```
Administrator: Command Prompt - java -jar jenkins.war --httpPort=8383
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd Jenkins

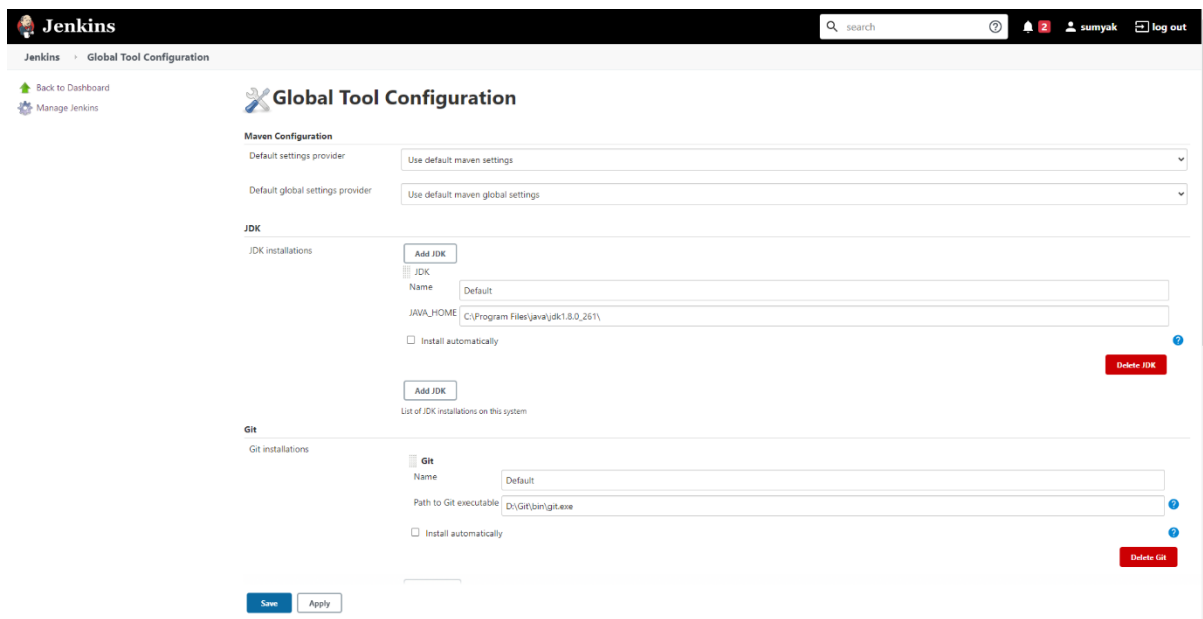
C:\Jenkins>java -jar jenkins.war --httpPort=8383
Error: Could not find or load main class jar
Caused by: java.lang.ClassNotFoundException: jar

C:\Jenkins>java -jar jenkins.war --httpPort=8383
Running from: D:\Jenkins\jenkins.war
Webroot: $user.home/.jenkins
020-10-06 05:05:05.625+0000 [id-1] INFO org.eclipse.jetty.util.log.Log#initialized: Logging initialized @1112ms
020-10-06 05:05:05.710+0000 [id-1] INFO org.eclipse.jetty.util.log.JavaUtilLog
020-10-06 05:05:05.750+0000 [id-1] INFO winstone.Logger#logInternal: Beginning extraction from war file
020-10-06 05:05:05.840+0000 [id-1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
020-10-06 05:05:05.840+0000 [id-1] INFO org.eclipse.jetty.server.Server#doStart: jetty-9.4.30.v20200611; built:
020-10-06 11T12:34:51.929Z; g1t: 271836e4c1f4612f12b7bb13ef5a92a927634b0d; jvm 11.0.8+10-LTS
020-10-06 05:05:06.930+0000 [id-1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /,
did not find org.eclipse.jetty.jsp.JettyJspServlet
020-10-06 05:05:07.000+0000 [id-1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager worke
Name=node0
020-10-06 05:05:07.000+0000 [id-1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, usin
defaults
020-10-06 05:05:07.000+0000 [id-1] INFO o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every
660000ms
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.thoughtworks.xstream.core.util.Fields (file:/C:/Users/GAURAV/.jenkins/war/WEB-INF/
```

STEP-2) download nexus plugin

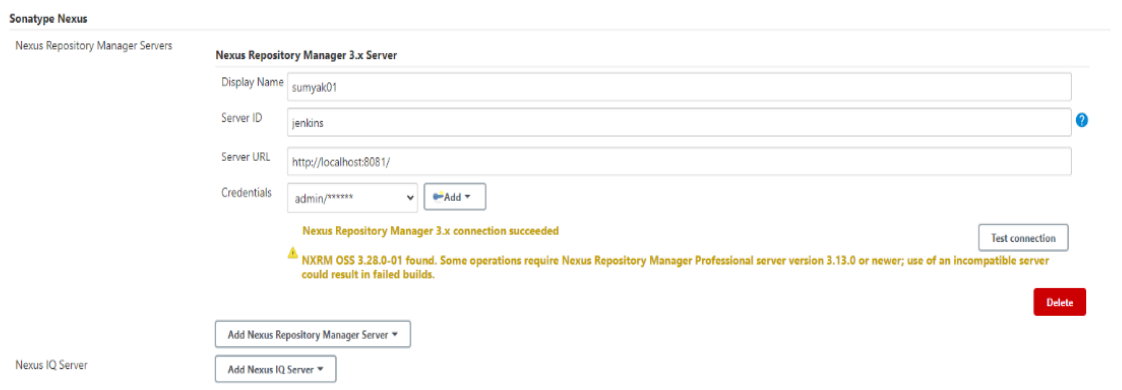


STEP-3) then go to global configuration setting and provide JDK 8 path



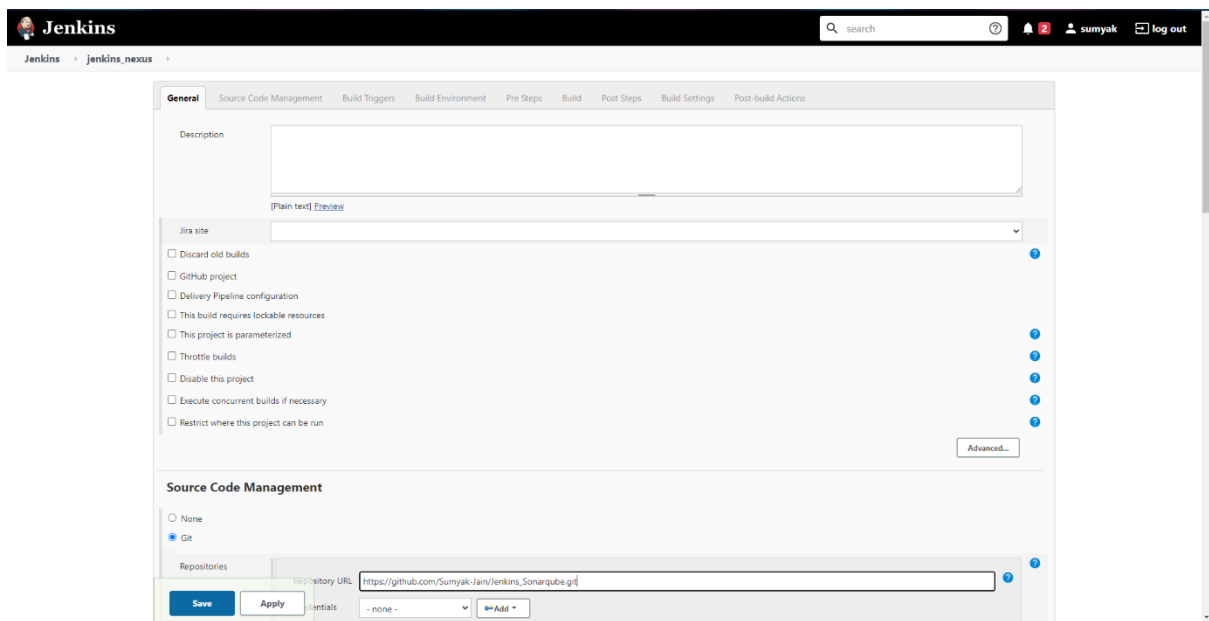
The image shows the Jenkins Global Tool Configuration page. The header includes the Jenkins logo, a search bar, and user information (sumyak). The main section is titled "Global Tool Configuration". Under "Maven Configuration", there are two dropdown menus for "Default settings provider" and "Default global settings provider", both set to "Use default maven settings". Under "JDK", there is a section for "JDK installations" with an "Add JDK" button. Below this, a table shows a single installation with Name "Default", JAVA_HOME "C:\Program Files\java\jdk1.8.0_261", and "Install automatically" unchecked. A "Delete JDK" button is next to it. Below the table is another "Add JDK" button and a note "List of JDK installations on this system". Under "Git", there is a section for "Git installations" with a "Git" icon. Below this, a table shows a single installation with Name "Default", Path to Git executable "D:\Git\bin\git.exe", and "Install automatically" unchecked. A "Delete Git" button is next to it. At the bottom, there are "Save" and "Apply" buttons.

STEP-4) provide sonatype nexus details



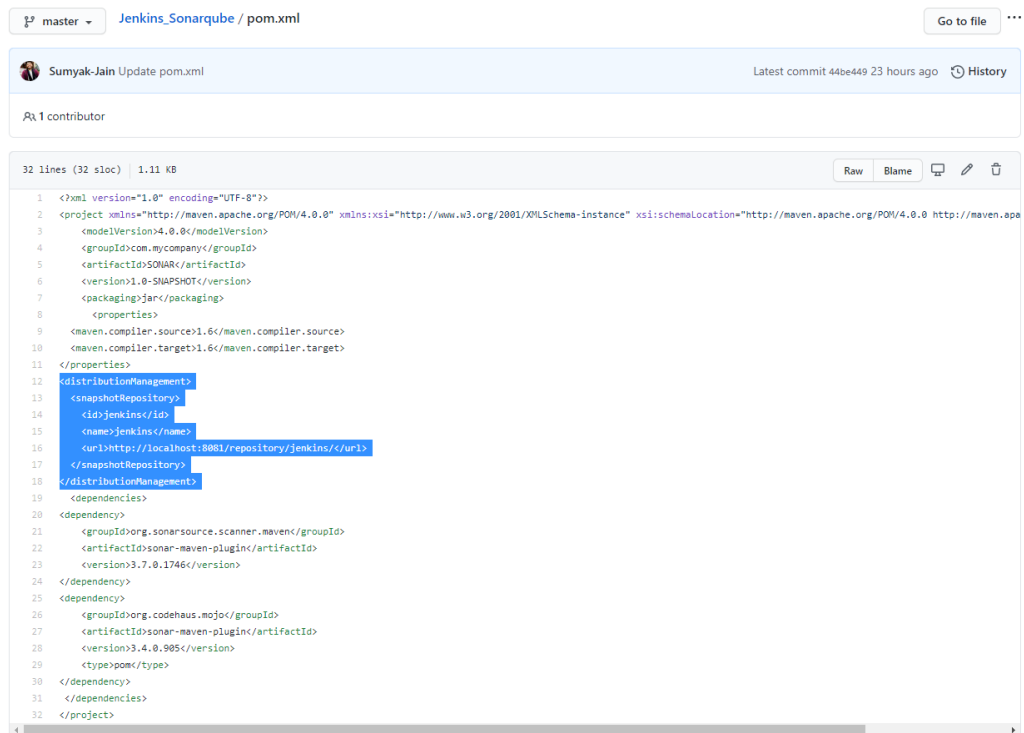
The image shows the Sonatype Nexus configuration page. The header includes the Sonatype Nexus logo and the text "Nexus Repository Manager Servers". The main section is titled "Nexus Repository Manager 3.x Server". Below this, there are fields for "Display Name" (sumyak01), "Server ID" (jenkins), "Server URL" (http://localhost:8081/), and "Credentials" (admin/*****). A "Test connection" button is next to the "Credentials" field. Below the "Test connection" button, there is a message: "Nexus Repository Manager 3.x connection succeeded". Below this message, there is a warning icon and text: "NXRM OSS 3.28.0-01 found. Some operations require Nexus Repository Manager Professional server version 3.13.0 or newer; use of an incompatible server could result in failed builds." At the bottom, there are two buttons: "Add Nexus Repository Manager Server" and "Add Nexus IQ Server".

STEP-5) make a new project



The image shows the Jenkins 'New Project' configuration page. The 'General' tab is selected. The 'Description' field is empty. The 'Jira site' dropdown is set to 'None'. A list of checkboxes for project settings is visible, including 'Discard old builds', 'GitHub project', 'Delivery Pipeline configuration', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', 'Execute concurrent builds if necessary', and 'Restrict where this project can be run'. The 'Source Code Management' section has 'Git' selected. The 'Repository' field is set to 'https://github.com/Sumyak-Jain/jenkins_Sonarqube.git'. The 'Credentials' dropdown is set to 'none'. The 'Save' button is highlighted.

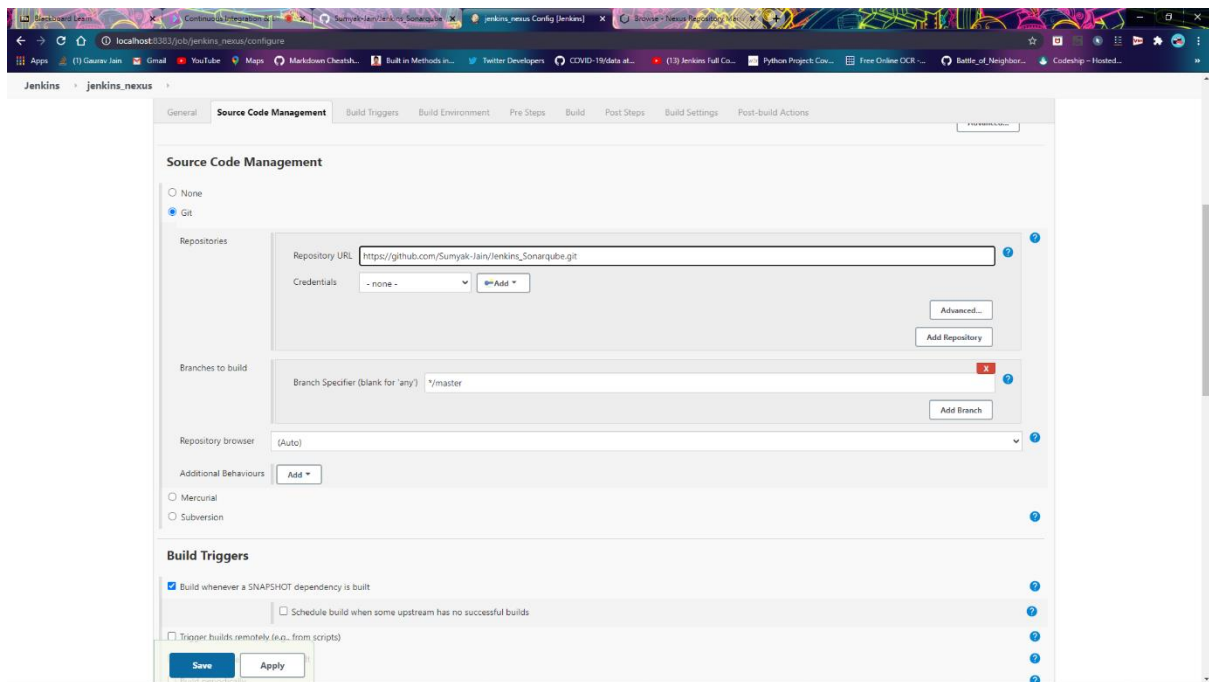
STEP-6) make changes in pom.xml



The image shows the GitHub web interface for the file 'Jenkins_Sonarqube/pom.xml'. The commit history shows a recent update by Sumyak-Jain. The file content is displayed in a code editor with line numbers 1 through 32. The XML content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <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">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.mycompany</groupId>
5   <artifactId>SONAR</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <packaging>jar</packaging>
8   <properties>
9     <maven.compiler.source>1.6</maven.compiler.source>
10    <maven.compiler.target>1.6</maven.compiler.target>
11  </properties>
12  <distributionManagement>
13    <snapshotRepository>
14      <id>jenkins</id>
15      <name>jenkins</name>
16      <url>http://localhost:8081/repository/jenkins</url>
17    </snapshotRepository>
18  </distributionManagement>
19  <dependencies>
20    <dependency>
21      <groupId>org.sonarsource.scanner.maven</groupId>
22      <artifactId>sonar-maven-plugin</artifactId>
23      <version>3.7.0.1746</version>
24    </dependency>
25    <dependency>
26      <groupId>org.codehaus.mojo</groupId>
27      <artifactId>sonar-maven-plugin</artifactId>
28      <version>3.4.0.905</version>
29      <type>pom</type>
30    </dependency>
31  </dependencies>
32 </project>
```

STEP-7) give the URL of your repo in SCM



The screenshot shows the Jenkins configuration page for 'jenkins_nexus' under the 'Source Code Management' tab. The 'Git' option is selected under 'None' or 'Git'. The 'Repository URL' is set to 'https://github.com/Sunyak-Jain/jenkins_Sonarqube.git'. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. The 'Build Triggers' section has 'Build whenever a SNAPSHOT dependency is built' checked. The 'Save' and 'Apply' buttons are at the bottom.

General Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Build Settings Post-build Actions

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for \'any\')

Repository browser

Additional Behaviours

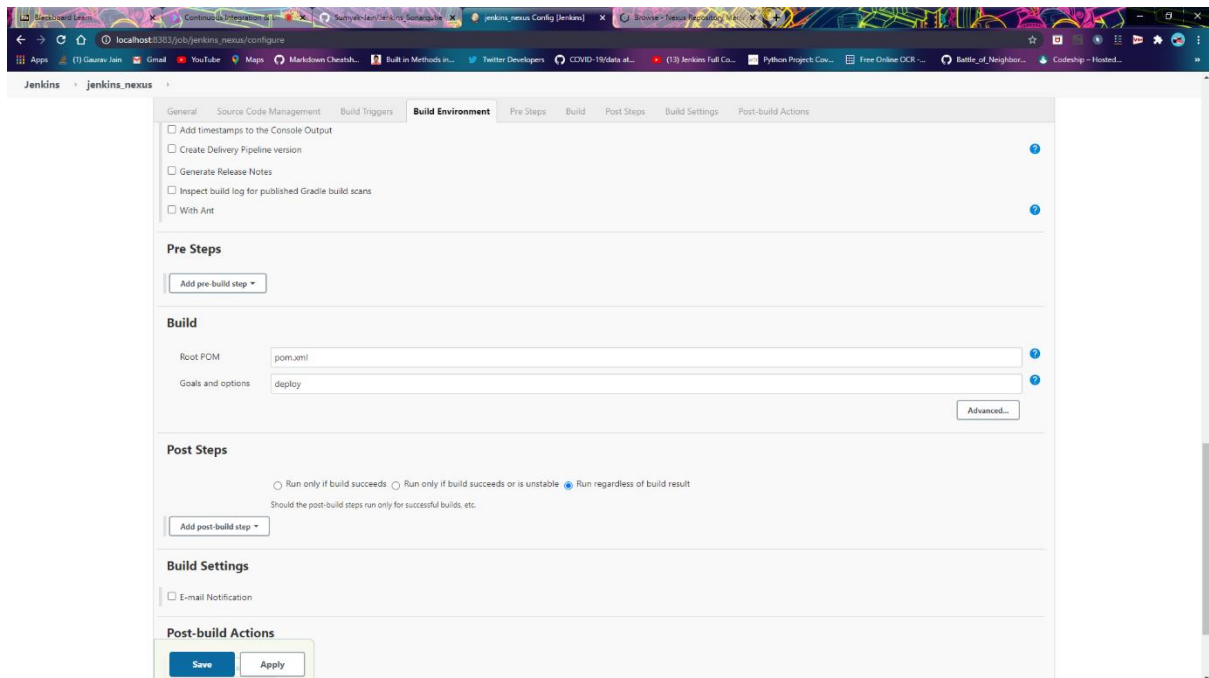
☐ Mercurial
☐ Subversion

Build Triggers

☒ Build whenever a SNAPSHOT dependency is built
☐ Schedule build when some upstream has no successful builds
☐ Trigger builds remotely (e.g., from scripts)

STEP-7) give the goal for your pom.xml in build location

deploy



The screenshot shows the Jenkins configuration page for 'jenkins_nexus' under the 'Build Environment' tab. The 'Pre Steps' section has an 'Add pre-build step' button. The 'Build' section has 'Root POM' set to 'pom.xml' and 'Goals and options' set to 'deploy'. The 'Post Steps' section has 'Run regardless of build result' selected. The 'Build Settings' section has 'E-mail Notification' unchecked. The 'Post-build Actions' section has a 'Save' button and an 'Apply' button.

General Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Build Settings Post-build Actions

Build Environment

☐ Add timestamps to the Console Output
☐ Create Delivery Pipeline version
☐ Generate Release Notes
☐ Inspect build log for published Gradle build scans
☐ With Ant

Pre Steps

Build

Root POM

Goals and options

Post Steps

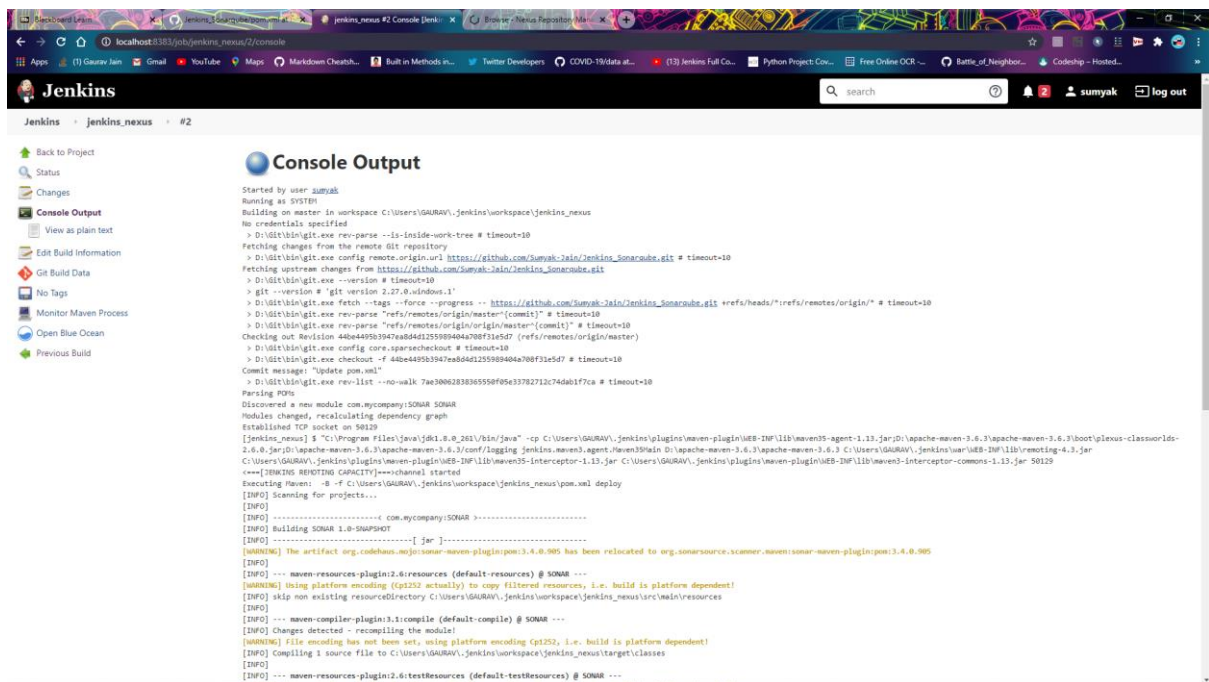
☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result
Should the post-build steps run only for successful builds, etc.

Build Settings

☐ E-mail Notification

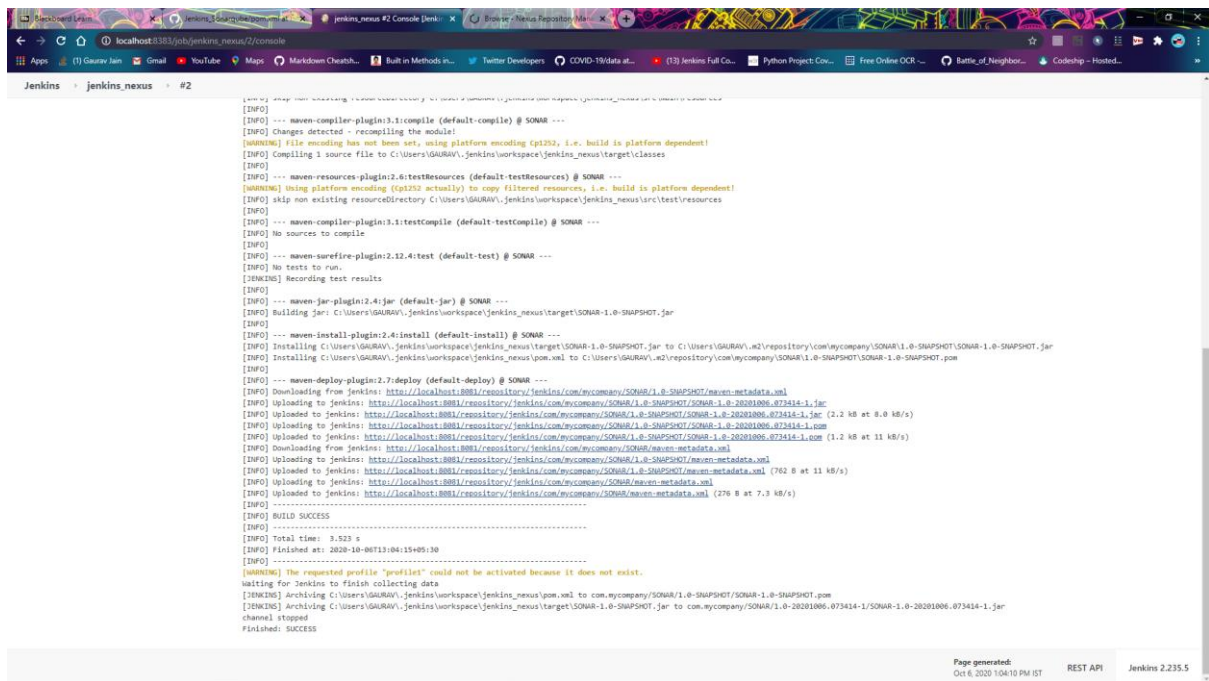
Post-build Actions

STEP-8) build your project



The screenshot shows the Jenkins web interface for a job named 'jenkins_nexus' with build number '#2'. The 'Console Output' tab is selected, displaying the build log. The log starts with 'Started by user sumyak' and 'Running as SYSTEM'. It shows the build process on a master node, including fetching changes from a remote Git repository, checking out the code, and parsing POM files. The build process involves Maven, with various plugins like 'maven-resources-plugin', 'maven-compiler-plugin', 'maven-surefire-plugin', and 'maven-jar-plugin' being executed. The log also shows the installation of the 'maven-deploy-plugin' and the deployment of the build artifacts to the Jenkins repository. The build concludes with 'BUILD SUCCESS'.

BUILD SUCCESS



This screenshot continues the Jenkins build log from the previous one. It shows the execution of the 'maven-compiler-plugin:3.1:compile' goal, which compiles the source code. This is followed by the 'maven-resources-plugin:2.7:resources' goal, which copies the resources. The log then shows the execution of the 'maven-surefire-plugin:2.12.4:test' goal, which runs the unit tests. After the tests pass, the 'maven-jar-plugin:2.4:jar' goal is executed to create the JAR file. The log also shows the installation of the 'maven-deploy-plugin' and the deployment of the build artifacts to the Jenkins repository. The build concludes with 'BUILD SUCCESS'.

STEP-9) check your nexus for build artifact

