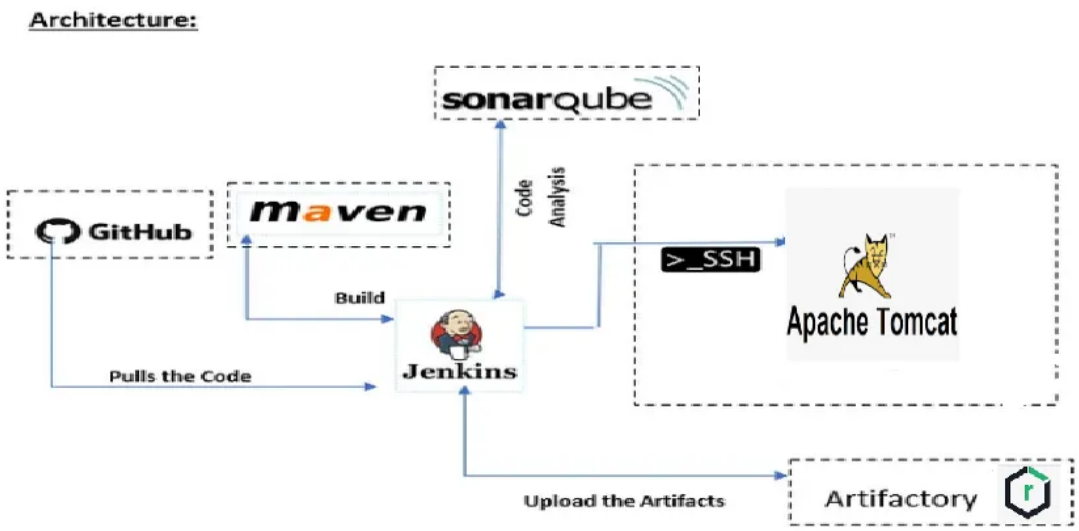
Project

Jenkins CI/CD Automation

Introduction:

This project implements a **Continuous Integration and Continuous Deployment (CI/CD) pipeline** using **Jenkins** with **Git, Maven, SonarQube, Nexus, and Tomcat**. The goal is to automate software development workflows, ensuring faster, more reliable, and efficient releases. By leveraging both **Freestyle and Pipeline jobs**, the project enables smooth integration, quality analysis, artifact management, and deployment.



Architecture

Components

1. Git (Version Control System)

2. Maven (Build Automation Tool)

3. SonarQube (Code Quality & Security Analysis)

4. Nexus (Artifact Repository Manager)

5. Tomcat (Application Server)

6. Jenkins (Automation Server)

Key points:

- **Git**

Git is a tool that helps developers save and track changes in their code.

It allows multiple developers to work on the same project without overwriting each other's work.

Developers can create separate branches to work on new features and merge them later.

Jenkins uses Git to automatically fetch the latest version of the code for building, testing, and deployment.

- **Maven (Build Automation Tool)**

A tool that compiles, tests, and packages applications.

It helps manage dependencies and ensures the software builds correctly.

- **SonarQube (Code Quality & Security Analysis)**

Checks the code for errors, security risks, and best practices.

Helps maintain high-quality and secure code before deployment.

- **Nexus (Artifact Repository Manager)**

Stores built application files (**WAR/JAR** files) for future use.

Acts as a central place for managing project artifacts.

- **Tomcat (Application Server)**

A server where web applications run.

Jenkins automatically deploys the final application to Tomcat after successful builds.

- **Jenkins (Automation Server)**

Jenkins is a tool that automates tasks like fetching code, building applications, testing, and deploying them.

It supports **Freestyle jobs** (manual setup) and **Pipeline jobs** (script-based automation).

Objectives:

- Setup Jenkins
- Setup & configure Maven and Git
- Setup SonarQube
- Integrating GitHub, Maven, Tomcat Server with Jenkins
- Setup tomcat server
- Create a CI and CD job
- Test the deployment

Prerequisite:

- AWS account
- Java-open- jdk
- PowerShell
- GitHub account

Process: Step 1: Install Jenkins on Linux server

Step 2: Install SonarQube on Linux server

Step 3: Install nexus on Linux server

Step 4: Install and configure tomcat

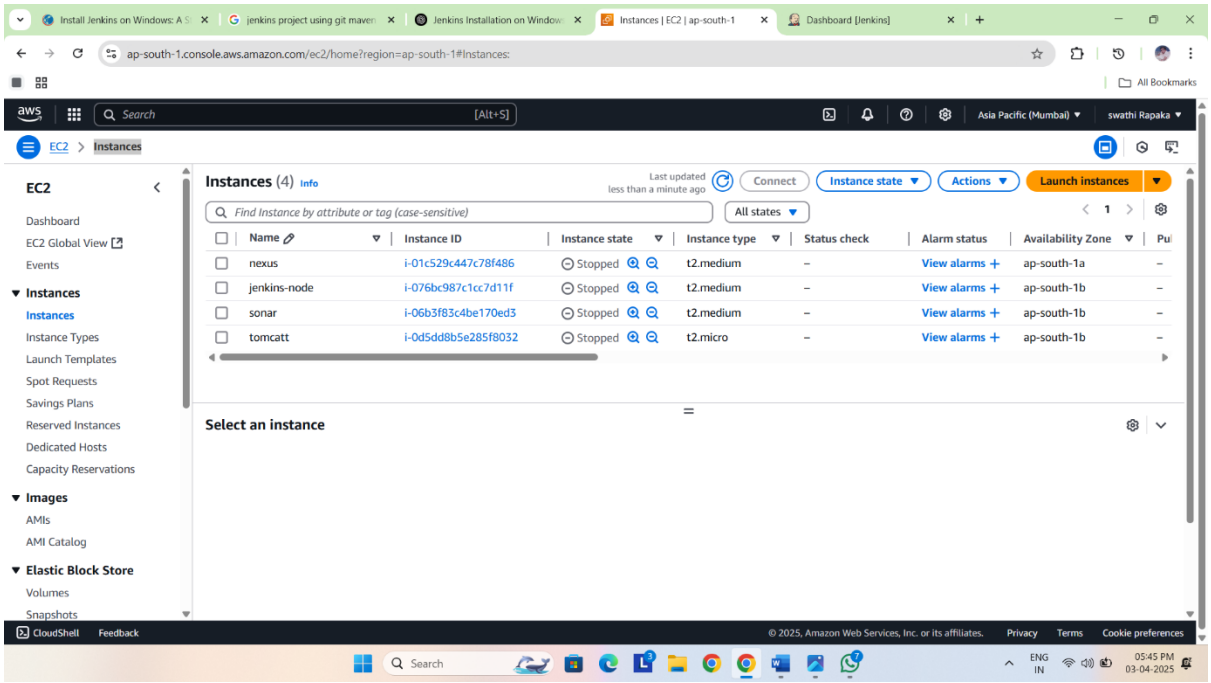
Step 5: Configure SonarQube, maven in Jenkins

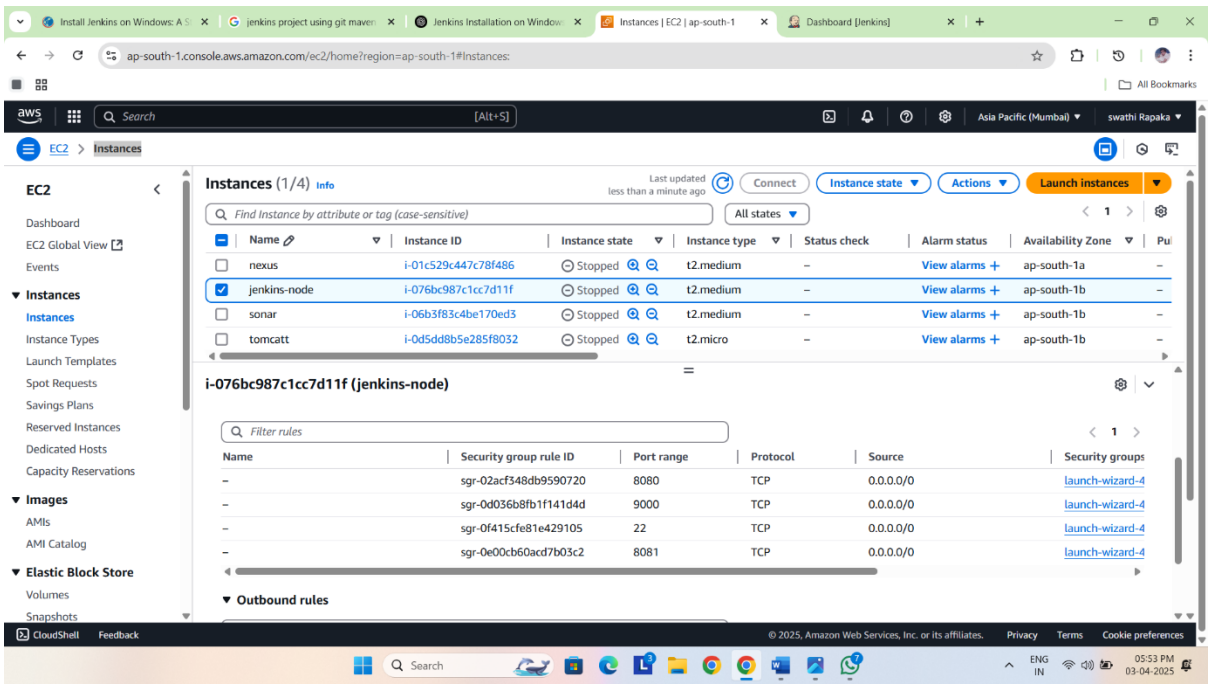
Step 6: Create Pipeline job

Step 1: Set Up AWS EC2 Instances

Log in to **AWS Console** → Navigate to **EC2** → Click **Launch Instance** and configure:

Server	Instance Name	OS	Instance Type	Ports to Open
Jenkins	Jenkins-Server	Amazon Linux 2023	t2.medium	22 (SSH), 8080 (Jenkins)
Jenkins Node	Jenkins-Node	Amazon Linux 2023	t2.medium	22 (SSH)
Maven	Maven-Server	Amazon Linux 2023	t2.medium	22 (SSH)
SonarQube	SonarQube-Server	Amazon Linux 2023	t2.medium	22 (SSH), 9000 (SonarQube)
Nexus	Nexus-Server	Amazon Linux 2023	t2.medium	22 (SSH), 8081 (Nexus)
Tomcat	Tomcat-Server	Amazon Linux 2023	t2.medium	22 (SSH), 8080 (Tomcat)


Instances | EC2 | ap-south-1
Dashboard [Jenkins]
ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:
Search [Alt+S]
Asia Pacific (Mumbai)
swathi Rapaka
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Instances (4) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
nexus i-01c529c447c78f486 Stopped t2.medium – View alarms ap-south-1a
jenkins-node i-076bc987c1cc7d11f Stopped t2.medium – View alarms ap-south-1b
sonar i-06b3f83c4be170ed3 Stopped t2.medium – View alarms ap-south-1b
tomcatt i-0d5dd8b5e285f8032 Stopped t2.micro – View alarms ap-south-1b
Select an instance
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
05:45 PM
03-04-2025


Instances | EC2 | ap-south-1
Dashboard [Jenkins]
ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:
Search [Alt+S]
Asia Pacific (Mumbai)
swathi Rapaka
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Instances (1/4) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
nexus i-01c529c447c78f486 Stopped t2.medium – View alarms ap-south-1a
jenkins-node i-076bc987c1cc7d11f Stopped t2.medium – View alarms ap-south-1b
sonar i-06b3f83c4be170ed3 Stopped t2.medium – View alarms ap-south-1b
tomcatt i-0d5dd8b5e285f8032 Stopped t2.micro – View alarms ap-south-1b
i-076bc987c1cc7d11f (jenkins-node)
Filter rules
Name
Security group rule ID
Port range
Protocol
Source
Security groups
– sgr-02acf348db9590720 8080 TCP 0.0.0.0/0 launch-wizard-4
– sgr-0d036b8fb1f141d4d 9000 TCP 0.0.0.0/0 launch-wizard-4
– sgr-0f415cfe81e429105 22 TCP 0.0.0.0/0 launch-wizard-4
– sgr-0e00cb60acd7b03c2 8081 TCP 0.0.0.0/0 launch-wizard-4
Outbound rules
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
05:53 PM
03-04-2025

I. Jenkins installation process

1.Before installing Jenkins, ensure the following requirements are met:
Java Development Kit (JDK) Installed
Jenkins requires Java 21 or later.

2.Set JAVA_HOME Environment Variable

- Find the Java installation directory.
- Open **Control Panel → System → Advanced System Settings**.
- Click **Environment Variables**, then add a new **System Variable**:
  - **Variable Name**: JAVA_HOME
  - **Variable Value**: Path to Java installation

Step 1: Download Jenkins

1. Open a web browser and go to the **Jenkins official website**:

   https://www.jenkins.io/download/

2. Click on the **Windows** section.
3. Download the **Windows Installer file**.

Step 2: Install Jenkins

1. **Run the Jenkins Installer (.msi)**

   Locate the downloaded file and double-click it to start the installation.

2. **Jenkins Setup Wizard Opens**

   Click **Next** to proceed.

3. **Select Installation Type**

   Choose **"Install as a Windows Service"** (recommended).Click **Next**.

4. **Choose Installation Directory**

   By default, Jenkins installs in

   C:\Program Files\Jenkins

5.Select Java Runtime Environment (JRE)

- If Java is installed, the installer will detect it automatically.
- If not detected, manually browse and select the Java installation folder.
- Click Next.

6. Configure Port for Jenkins

- Default Jenkins port: 8080.
- Click Next.

7. **Select Windows User for Jenkins Service**

- Choose a Windows user account under which Jenkins will run.
- If unsure, select **Run as Local System** (recommended for basic setup).
- Click **Next**.

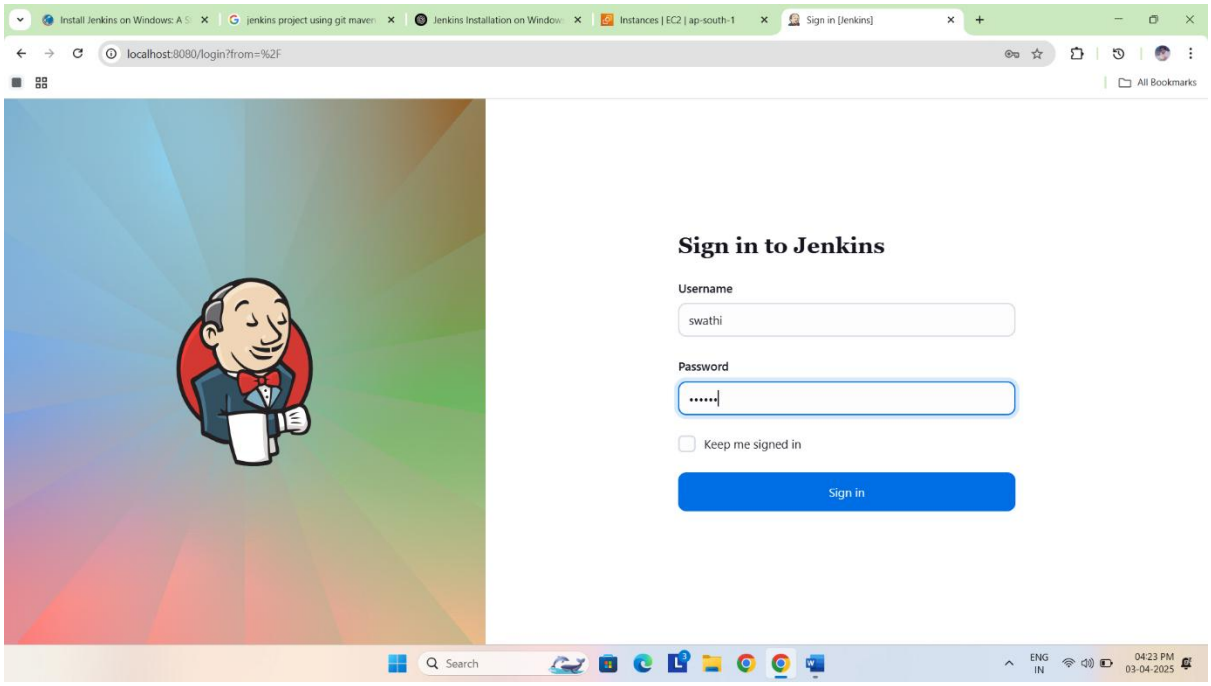
8. **Complete Installation**

- Click **Install** and wait for the installation process to complete.
- Once finished, click **Finish** to close the setup wizard

Step 3: Start Jenkins and Initial Setup

1. **Start Jenkins Service**
    - If installed as a service, Jenkins starts automatically.
2. **Open Jenkins in a Web Browser**
    - Open a browser and go to

http://localhost:8080



3. **Unlock Jenkins**

   The first time Jenkins runs, it requires an **administrator password**.

   Find the password in this file

C:\Program Files\Jenkins\secrets\initial Admin Password

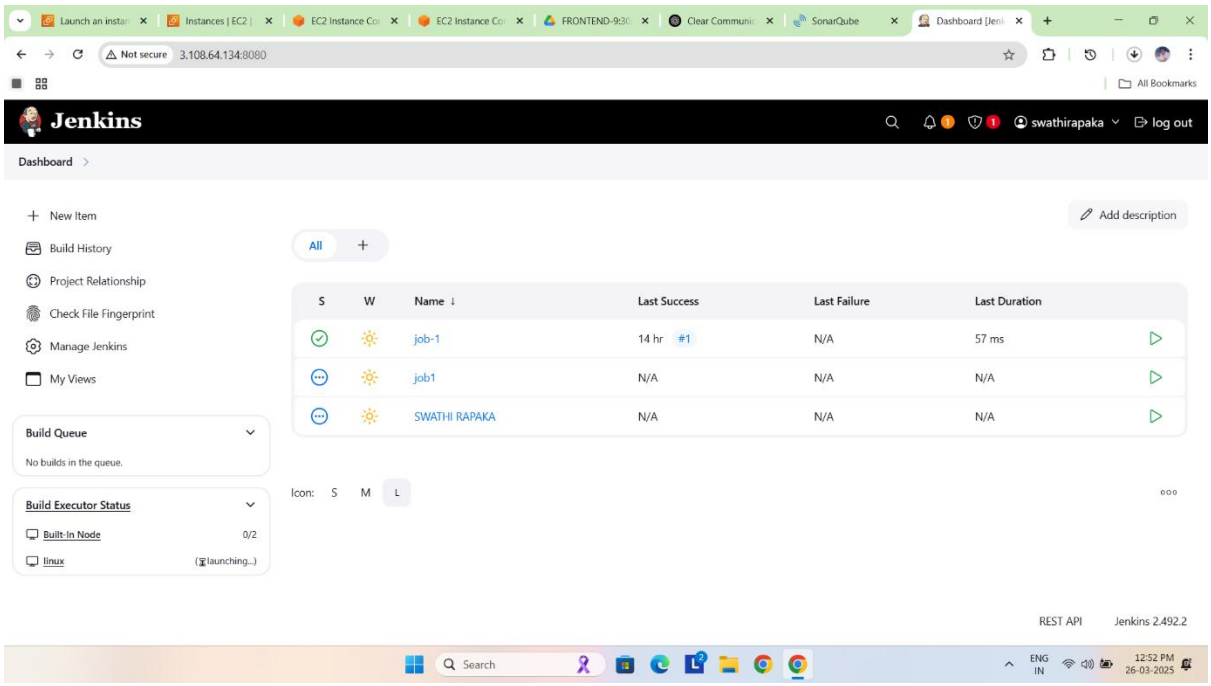
Open this file in **Notepad** and copy the password.

Paste it into the Jenkins setup page and click **Continue**.

Step 4: Install Plugins and Create Admin User

1. **Install Suggested Plugins**
   - Jenkins will prompt you to install recommended plugins.
   - Click **"Install Suggested Plugins"**.
2. **Create Admin User**
   - Enter the following details:
     - **Username:** (Swathi)
     - **Password:** (Swathi)
     - **Full Name:** (Swathi Rapaka)
     - **Email Address:** (swathi@gmail.com)
   - Click **Save and Continue**.

3. **Instance Configuration**
    - Confirm the Jenkins URL (default: http://localhost:8080).
    - Click **Save and Finish**.
4. **Start Using Jenkins**
    - Click **"Start using Jenkins"** to go to the Jenkins dashboard



Set Up a Jenkins Node (Agent) in Linux

Introduction

A **Jenkins Node (Agent)** is a separate machine that runs Jenkins jobs, allowing workload distribution. This guide explains how to configure a Jenkins agent on an **AWS EC2 Linux instance**.

Step 1: Launch an AWS EC2 Instance

To create a Linux server for the Jenkins agent, follow these steps:

1.1 Log in to AWS Console

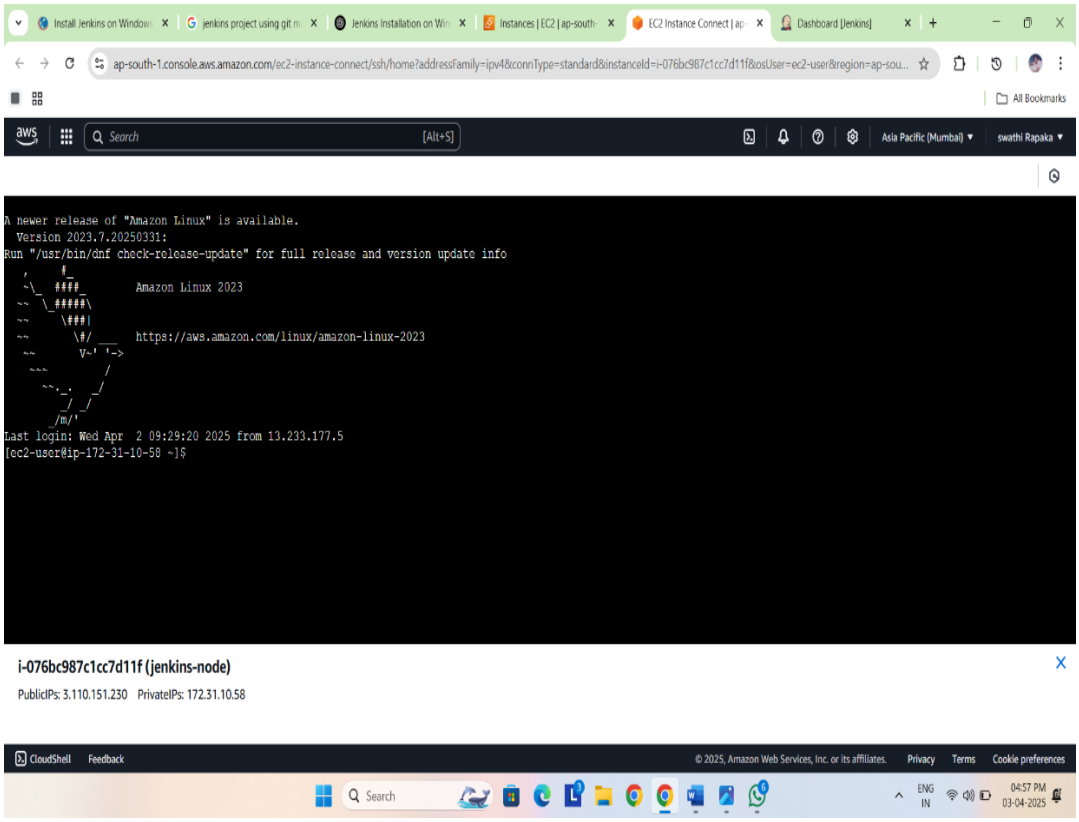
1. Open a browser and go to AWS Console.
2. Sign in to your **AWS account**.
3. Navigate to **EC2 Dashboard**.

1.2 Create a New EC2 Instance

1. Click on **Launch Instance**.
2. Configure the instance with the following details:

Setting	Value
Instance Name	Jenkins-node
Operating System	Amazon Linux 2023
Instance Type	t2. medium
Key Pair	Select an existing key
Security Group Rule	Jenkins (8080)

3. Install the java git and maven is mandatory



Configure the Node in Jenkins

1.Navigate to Node Management

Step Action

✓ Log in to **Jenkins Dashboard**.

Click on **Manage Jenkins** in the left menu.

Select **Manage Nodes and Clouds**.

Create a New Node

✓ Click **New Node**.

Enter **Node Name** (e.g., Linux).

Select **Permanent Agent**.

Click **OK**.

Configure Node Settings

Field	**Value**
Remote Root Directory	Jenkins
Label	Dev
Launch Method	Launch agent via SSH
Host	https://15.207.199.130 (Public IP)
Credentials	Add **SSH username and private key** for authentication
HostKeyVerification Strategy	Non-Verifying Verification Strategy

Add SSH Credentials

Click **Add → Jenkins → SSH Username with Private Key**.

Enter **Username**: ec2-user.

Select **Enter directly** and paste the private key from you. Pem

Click **Add**.

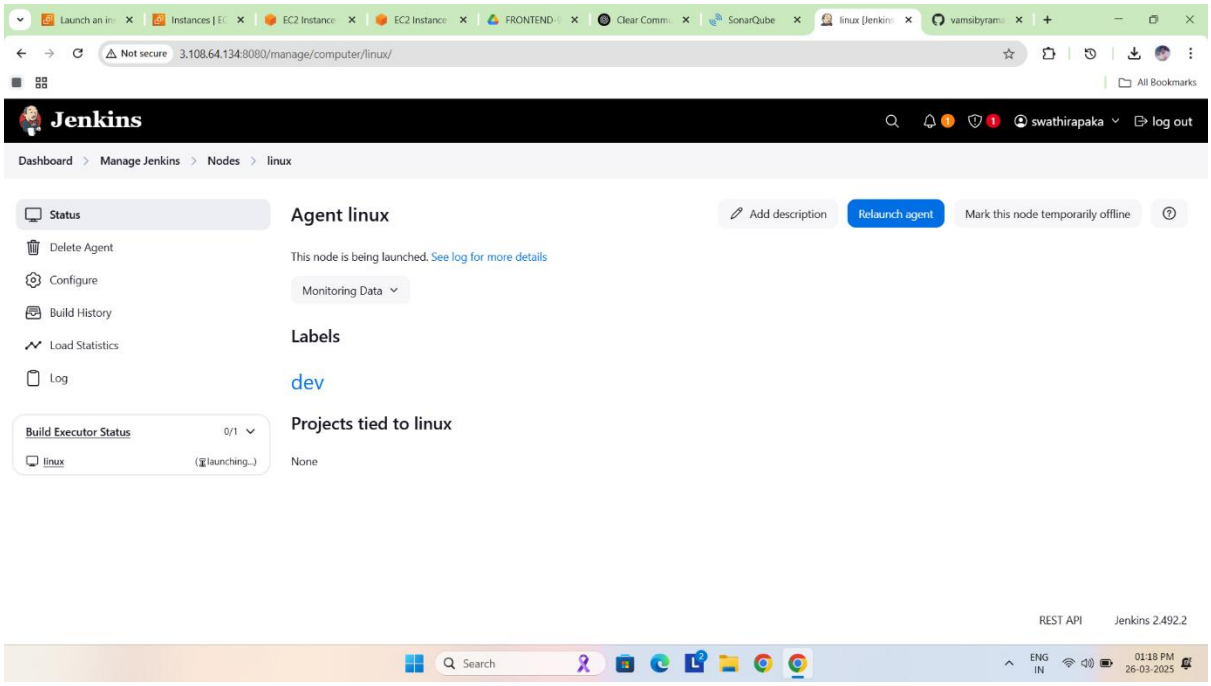
Save and Test the Connection

✓ Click **Save** to store the configuration.

Click **Launch Agent** to establish the connection.

If the connection is successful, the **agent status turns green**.

Click **Build Now** to test the connection.



SonarQube Installation on Linux

Step 1: Install Java

1. yum list | grep "java"
2. yum install java-17-amazon-corretto.x86_64 -y

3.yum install git -y

4. yum install maven -y

5.wgethttps://dlcdn.apache.org/tomcat/tomcat-9/v9.0.102/bin/apache-tomcat-9.0.102.tar.gz

6. tar -zxvf apache-tomcat-9.0.102.tar.gz
7. mv apache-tomcat-9.0.102 /opt/tomcat
8. cd /opt/tomcat/bin/

Step 2: Create a SonarQube User

- useradd sonarqube

Step 3: Change Ownership to SonarQube User

- chown -R sonarqube: sonarqube sonar/.

Step 4: Start SonarQube

Now that SonarQube is installed and the ownership is set, start SonarQube.

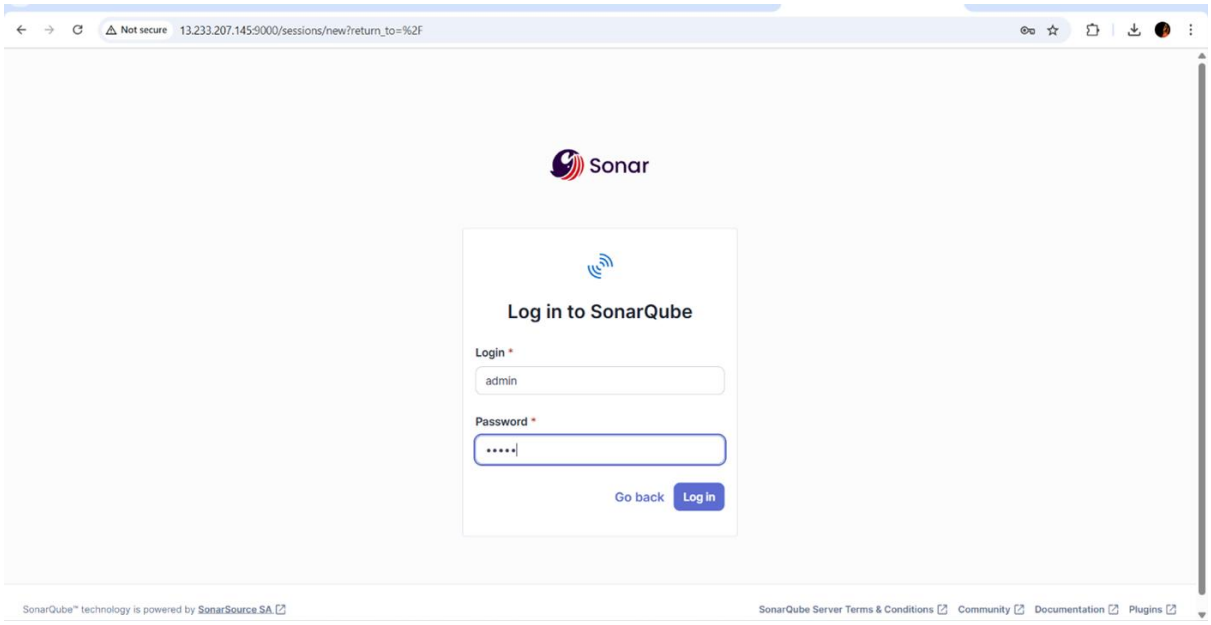
- Switch to the SonarQube user:
  su sonarqube
- Navigate to the SonarQube directory:
  cd /opt/sonar/
- Start SonarQube:
  sh bin/linux-x86-64/sonar.sh start

step 5: Access SonarQube

1.from https://3.111.58.238

2.define Username: admin

Password: admin



Nexus Repository Manager Installation & Setup on Linux

Step 1: install java

Nexus requires Java 1.8

1.install OpenJDK 1.8: yum install java 1.8.0 OpenJDK

2.verify the installation: java version

Step 2: Download Nexus Repository Manager

1. **Download Nexus Repository Manager with Orient DB:**

Versions of Nexus Repository Manager prior to 3.71.0 use Orient DB as their embedded database. Starting from version 3.71.0, Nexus transitioned to using H2 as the default embedded database. If you specifically need a version with Orient DB, you'll want to download a 3.70.x release.help.sonatype.com

You can find the 3.70.x versions here: help.sonatype.com

Choose the appropriate archive for your operating system and Java version. For instance, if you're using a Unix-based system with Java 8, you might select:

- nexus-3.70.4-02-java8-unix.tar.gz
- wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz
    - Extract Nexus Archive: tar -xzvf nexus-3.70.4-02-java8-unix.tar.gz
    - Move Nexus to the Desired Directory: sudo mv nexus-3.70.4-02 /opt/nexus
    - Start Nexus: /opt/nexus/bin/nexus start

Step3: Access Nexus Repository Manager

1. **Access Nexus** in your web browser: http://localhost:8081
2. **Login credentials**:
    - **Username**: admin
    - **Password**: Found
    - in:/opt/nexus/sonatype-work/nexus3/admin.password
3. Retrieve the password using:
4. cat /opt/nexus/sonatype-work/nexus3/admin. password

Step 4: Configure Nexus with Maven

To deploy to a Nexus repository using Maven

1. Configure pom.xml

In your Maven project, add the following distribution Management section to your pom.xml file:

```
<distribution Management>
  <repository>
    <id>nexus</id>
    <name>Releases</name>
    <url>http://localhost:8081/repository/maven-releases</url>
  </repository>
  <snapshot Repository>
```

```
    <id>nexus</id>
    <name>Snapshot</name>
    <url>http://52.66.44.57:8081/repository/maven-snapshots</url>
  </snapshot Repository>
</distribution Management>
```

2. Configure settings.xml

Next, configure Maven's settings.xml to provide Nexus credentials.

1. **Edit your Maven settings.xml file** (located in ~/.m2/settings.xml).
2. **Add the server configuration**

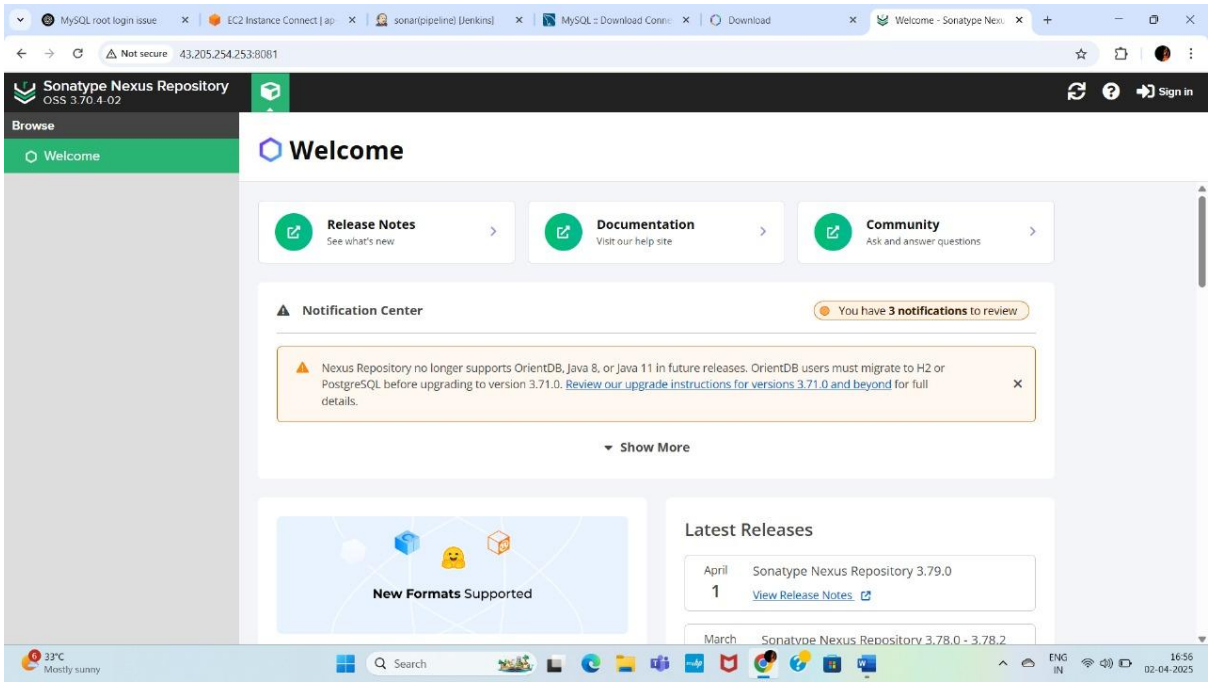
```
<settings>
  <servers>
    <server>
      <id>nexus</id>
      <username>admin</username>
      <password>admin</password>
    </server>
  </servers>
</settings>
```

Step 5: Deploying to Nexus

Deploy the project to Nexus using Maven:

To deploy project to the Nexus repository: **mvn clean deploy**

This will upload artifacts to the appropriate repository (Releases or Snapshots) in Nexus, based on the version in pom.xml



Apache Tomcat Installation and Configuration

Step 1: Install Java 17 (Amazon Corretto)

1. **Tomcat requires Java to run. Install Amazon Corretto 17:** yum install java-17-amazon-corretto-devel
2. **Verify the installation by checking the Java version**: java -version

Step 2: Download Apache Tomcat

1. Visit the official Apache Tomcat website and download the latest **Tomcat 9** version.

Step 3: Extract the Archive

1. After downloading, extract the Tomcat archive using: tar -xvzf apache-tomcat-9.tar.gz
2. Move the extracted folder to /opt/tomcat and rename it: mv apache-tomcat-9 /opt/tomcat

Step 4: Start Tomcat

1. Navigate to the bin directory and start Tomcat: cd /opt/tomcat/bin
2. To start the tomcat: sh startup.sh
3. Check the Tomcat status: sh startup.sh status

Step 5: Access Tomcat Dashboard

1. Open your browser and enter the following URL to access the Tomcat dashboard: http://13.203.84.5

Step 6: Enable GUI Deployment in "Manage Apps"

To enable GUI-based application deployment:

1. Navigate to the META-INF directory of the manager app: cd /opt/tomcat/webapps/manager/META-INF/
2. Open the context.xml file using vi editor: vi context.xml
3. Comment out the following <value> save, and exit.

Step 7: Configure Tomcat Users

1. Navigate to the conf directory: cd /opt/tomcat/conf
2. Open the tomcat-users.xml file: vi tomcat-users.xml
3. Add the following lines: <role rolename="manager-gui"/>

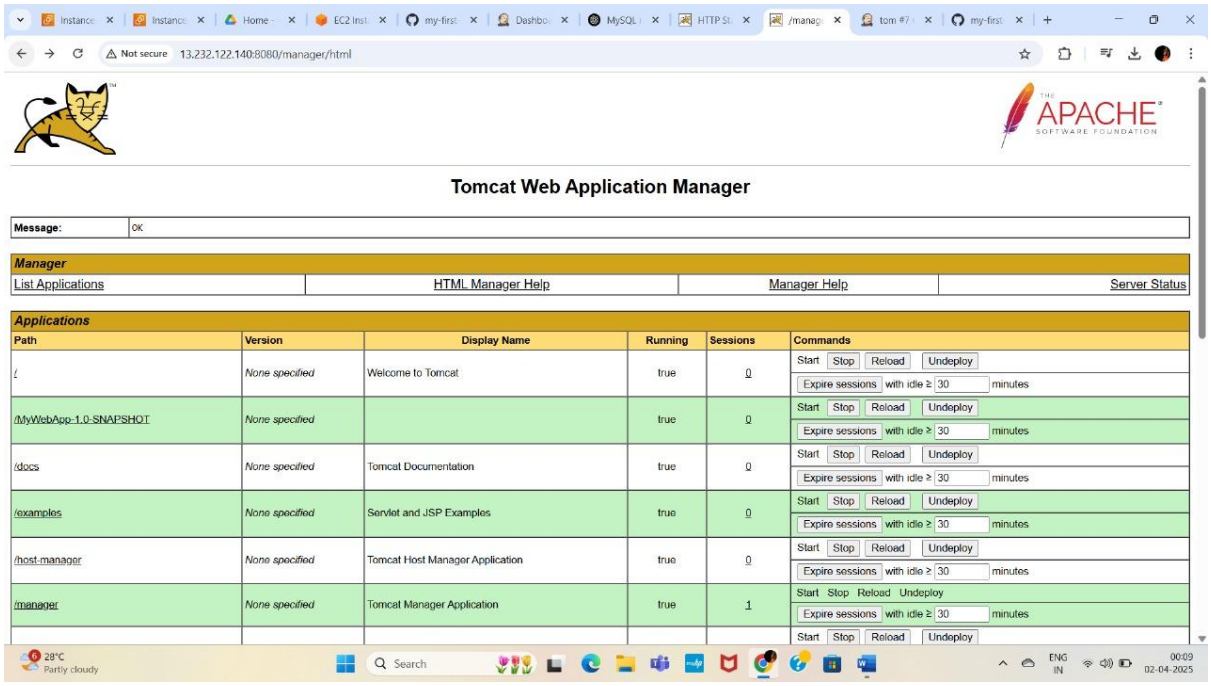
<user username="tomcat" password="tomcat" roles="manager-gui, manager-script"/>

4. Save and exit the file.

Step 8: Access Tomcat Dashboard

Now, open the following URL in your browser and log in using the credentials: http://13.203.84.5

Username:tomcat
Password: tomcat



Git Integration in Jenkins:

Git Plugin Installation

- Open Jenkins Dashboard → Navigate to Manage Jenkins → Click Manage Plugins
- Restart Jenkins after installation

Git Configuration in Jenkins

- Go to Manage Jenkins → Global Tool Configuration
- Under Git, click Add Git and provide the path if not detected automatically
- Save the configuration

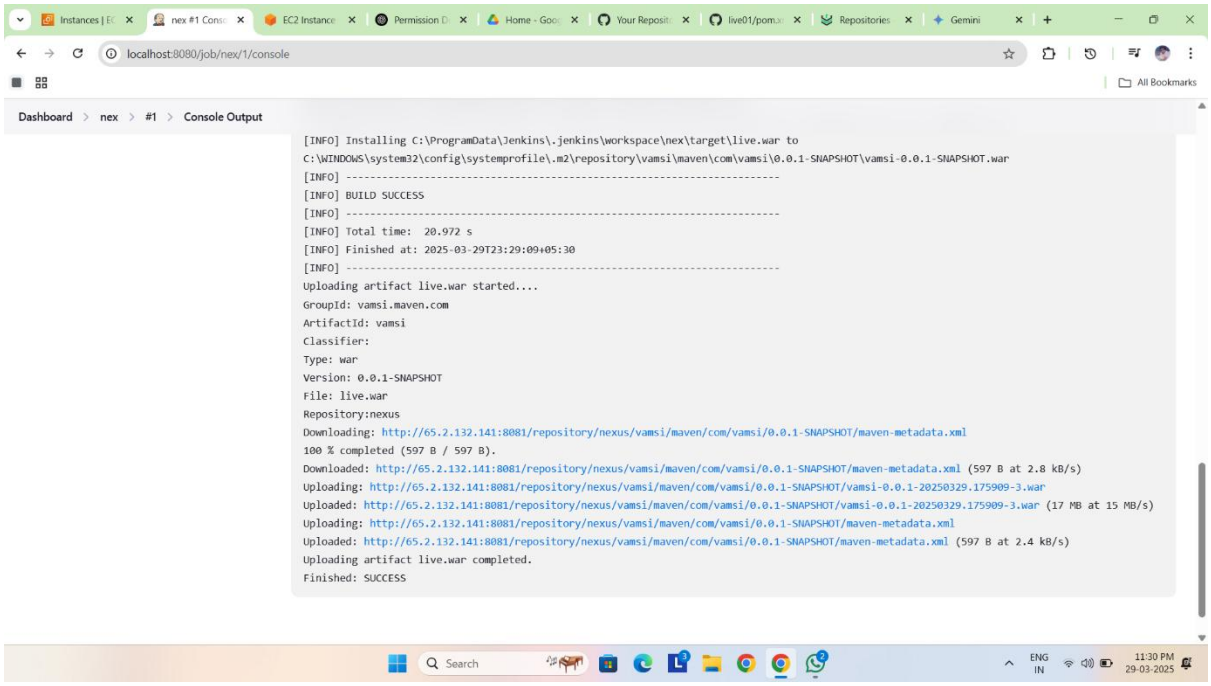
Jenkins Pipeline for Git Checkout

1. Create a New Pipeline Job in Jenkins
2. Add the following stage in the Jenkinsfile:

```
stage('git') {
   steps {
      gitbranch:'main', url:'https://github.com/metta-anusha/pet_shop.git'
   }
}
```

Git Stage - Console Output

In this stage, the Jenkins pipeline fetches the code from the Git repository. Below is the console output when the Git stage is executed:



Maven Installation in Jenkins

- Navigate to **Manage Jenkins → Global Tool Configuration**
- Under **Maven**, click **Add Maven**
- Set **Name** as Maven and provide installation path
- Save the configuration

Jenkins Pipeline for Maven Build

1. Add the following stage in the Jenkins file:

```
stage('maven') {
   steps {
      sh 'mvn clean package'
   }
}
```

Maven Stage - Console Output

In this stage, the Jenkins pipeline runs Maven to clean and package the project. Below is the console output when the Maven stage is executed

Creating a SonarQube Pipeline in Jenkins

Prerequisites:

Before setting up the pipeline, ensure the following:

- **SonarQube Scanner Plugin** is installed in Jenkins

- **SonarQube Server** is configured in Jenkins

Step 1: Install SonarQube Scanner Plugin

1. Open **Jenkins Dashboard** → Navigate to **Manage Jenkins** → Select **Manage Plugins**.
2. Go to the **Available Plugins** tab and search for **SonarQube Scanner**.
3. Click **Install** and wait for the installation to complete.

Step 2: Configure SonarQube Server in Jenkins

1. Navigate to **Manage Jenkins → Global Tool Configuration**.
2. Locate **SonarQube Scanner** and click **Add SonarQube**.
3. Enter the **SonarQube Server Details**.
4. Generate an authentication token in **SonarQube**:
    - Open **http://13.232.3.106:9000/account/security/**.
    - Create a new token and copy it.
5. Configure credentials in Jenkins:
    - Go to **Manage Jenkins → Manage Credentials**.
    - Add the generated token under **Secret Text Credentials**.

Step 3: Create a Jenkins Pipeline Job

1. Open **Jenkins Dashboard** → Click **New Item**.
2. Select **Pipeline**, provide a name (e.g., **Sonar**), and click **OK**.
3. Scroll down to the **Pipeline Section** and choose **Pipeline Script**.

Step 4: Define the Jenkins Pipeline Script

To integrate SonarQube into the Jenkins pipeline, add the following stage:

Pipeline Script:

```
stage ('SonarQube Analysis') {
   steps {
      with SonarQube Env('sonarqube') {
         sh 'mvn sonar: sonar'
      }
```

```
    }
}
```

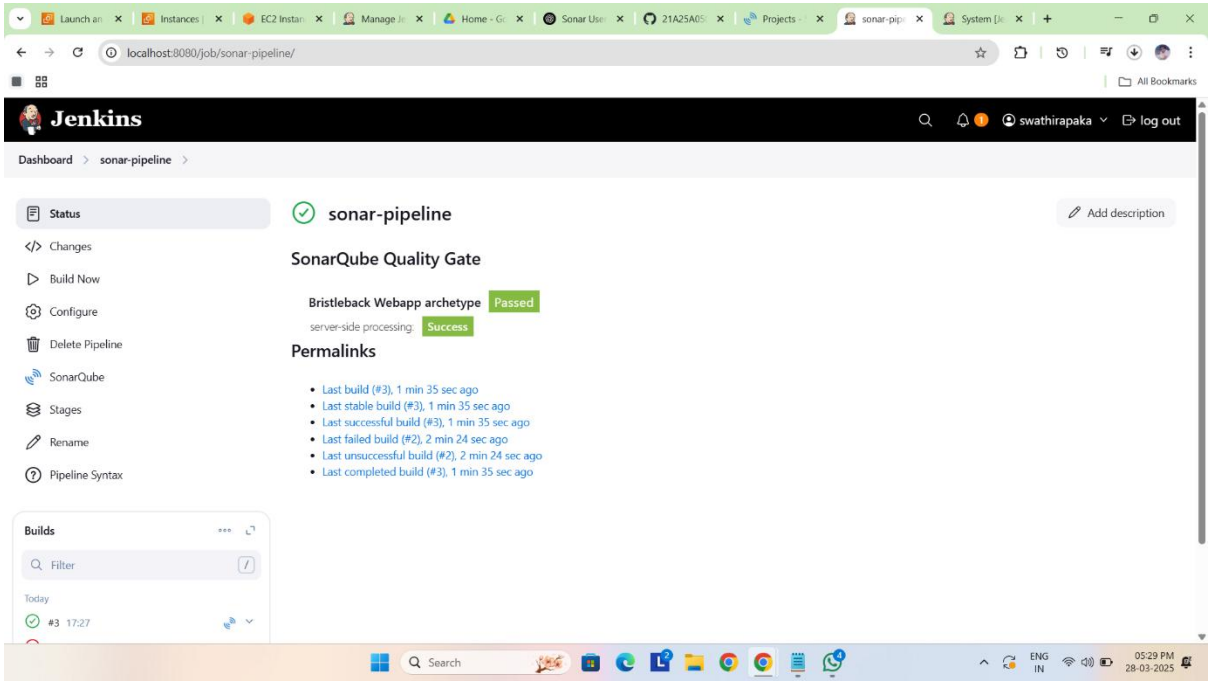
Click **Save** and then **Build Now** to execute the pipeline.

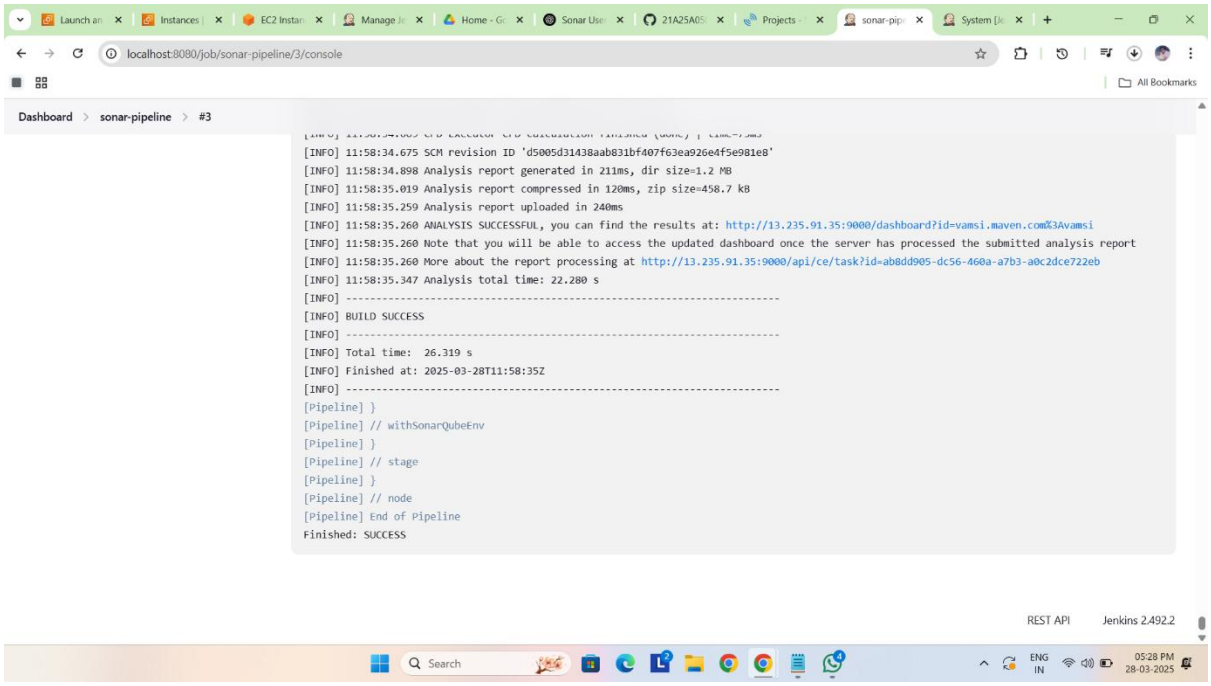
Additional Notes:

- To generate the correct **pipeline syntax**, go to **Pipeline Syntax** → Use **Snippet Generator**.
- If necessary, download and install the **SonarQube Scanner Plugin** from the plugin manager.

SonarQube Stage - Console Output

In this stage, the Jenkins pipeline runs the SonarQube analysis using Maven. Below is the console output when the SonarQube stage is executed:





SonarQube Freestyle:

Create a New Freestyle Job

Open **Jenkins** → Click **New Item**

- Enter a **Job Name** (e.g., Sonar)
- Select **Freestyle project** → Click **OK**

Configure the Job

General Section:

- Check **Restrict where this project can be run**
- In the **Label Expression**, enter: dev

Source Code Management:

- Select **Git**
- In the **Repository URL**, enter:
  - https://github.com/21A25A0509/live01.git
- In the **Branches to build**, enter

Build Steps:

Step 1: Maven Build

- Click **Add Build Step** → Select **Invoke top-level Maven targets**
- In **Goals**, enter:

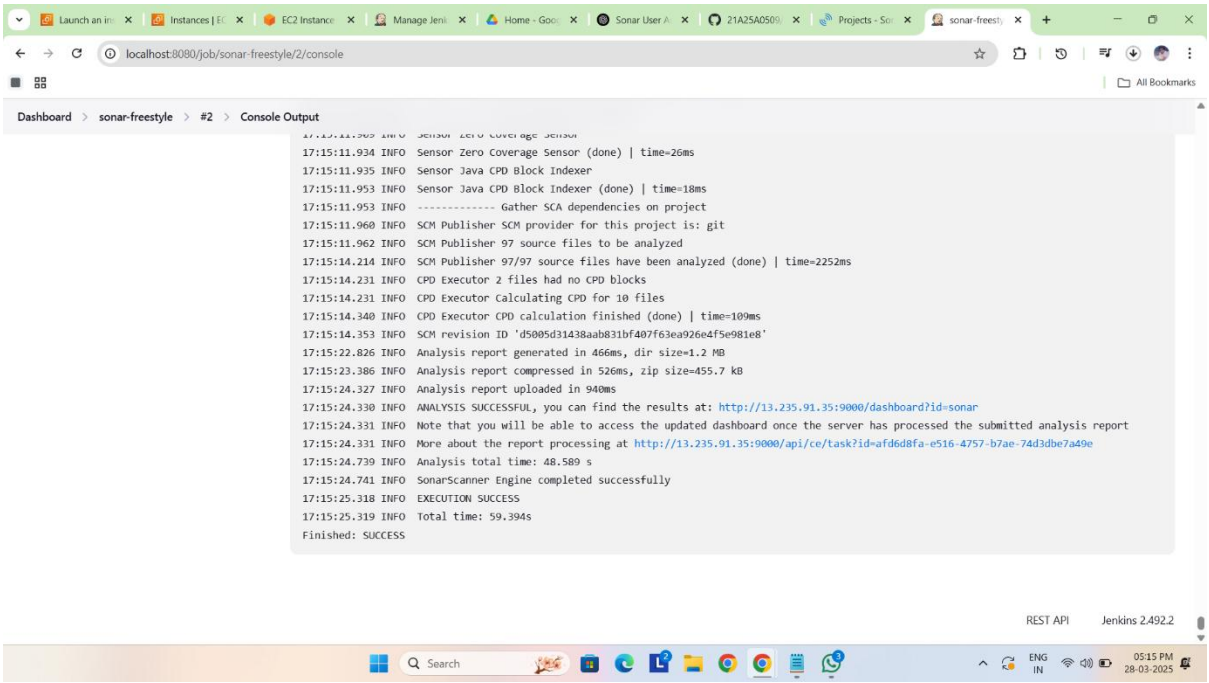
Step 2: Execute SonarQube Scanner

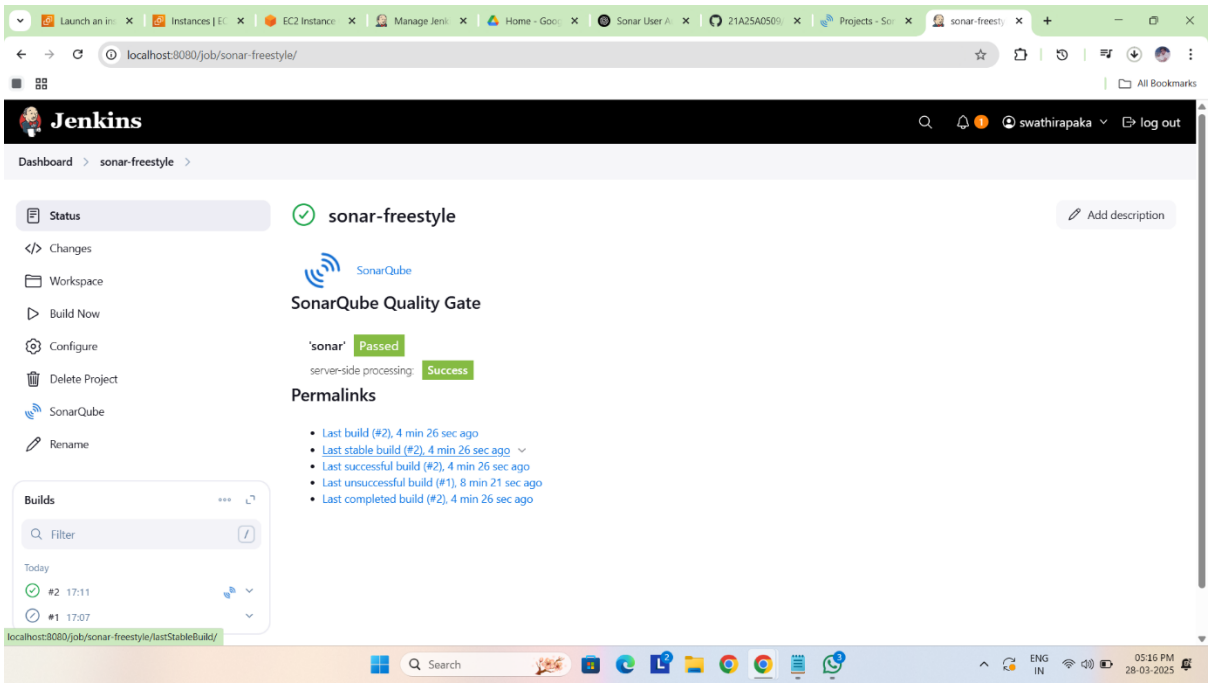
- Click **Add Build Step** → Select **Execute SonarQube Scanner**
- In **Analysis Properties**, enter the following
- mvn clean package
- sonar. project Key=sonar
- sonar. Project Name='sonar'
- sonar.host.url=http://65.0.29.251:9000/
- sonar. Sources=src
- sonar.java. binaries=target/classes

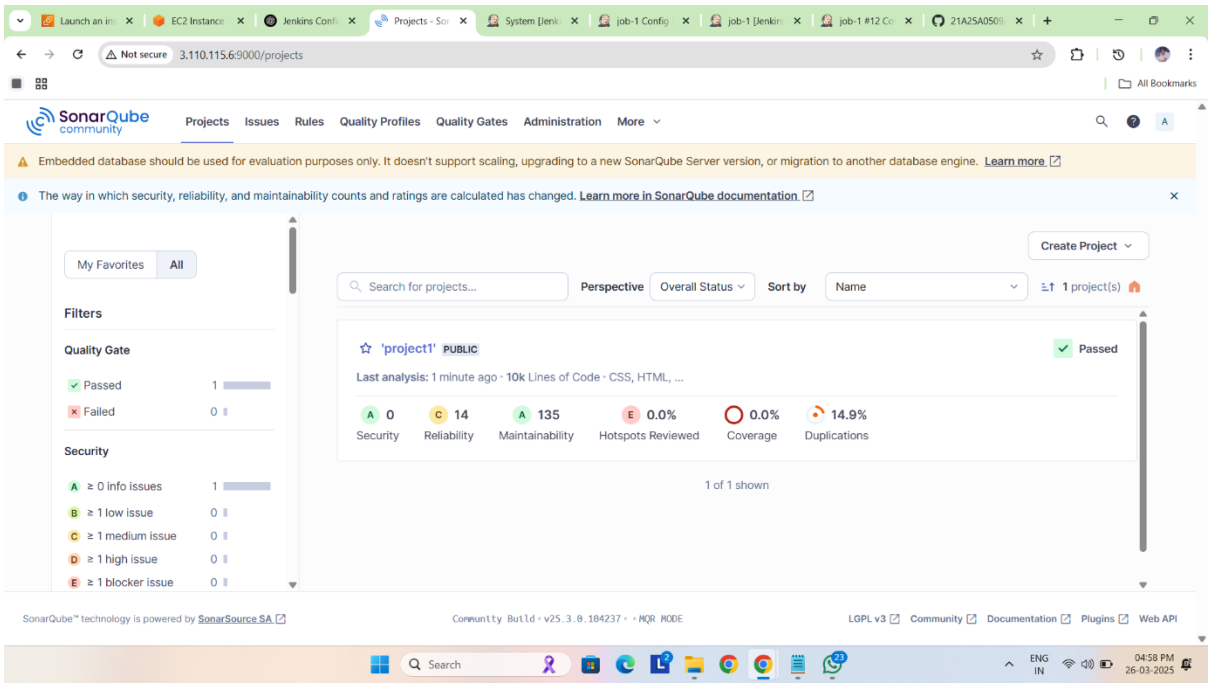
4. **Save and Build the Job**

   Click **Save**

- Click **Build Now** to start the process
- Once the build is complete.


localhost:8080/job/sonar-freestyle/2/console
Dashboard > sonar-freestyle > #2 > Console Output
17:15:11.934 INFO Sensor Zero Coverage Sensor (done) | time=26ms
17:15:11.935 INFO Sensor Java CPD Block Indexer
17:15:11.953 INFO Sensor Java CPD Block Indexer (done) | time=18ms
17:15:11.953 INFO ------------ Gather SCA dependencies on project
17:15:11.960 INFO SCM Publisher SCM provider for this project is: git
17:15:11.962 INFO SCM Publisher 97 source files to be analyzed
17:15:14.214 INFO SCM Publisher 97/97 source files have been analyzed (done) | time=2252ms
17:15:14.231 INFO CPD Executor 2 files had no CPD blocks
17:15:14.231 INFO CPD Executor Calculating CPD for 10 files
17:15:14.340 INFO CPD Executor CPD calculation finished (done) | time=109ms
17:15:14.353 INFO SCM revision ID 'd5005d31438aab831bf407f63ea926e4f5e981e8'
17:15:22.826 INFO Analysis report generated in 466ms, dir size=1.2 MB
17:15:23.386 INFO Analysis report compressed in 526ms, zip size=455.7 kB
17:15:24.327 INFO Analysis report uploaded in 940ms
17:15:24.330 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://13.235.91.35:9000/dashboard?id=sonar
17:15:24.331 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
17:15:24.331 INFO More about the report processing at http://13.235.91.35:9000/api/ce/task?id=afd6d8fa-e516-4757-b7ae-74d3dbe7a49e
17:15:24.739 INFO Analysis total time: 48.589 s
17:15:24.741 INFO SonarScanner Engine completed successfully
17:15:25.318 INFO EXECUTION SUCCESS
17:15:25.319 INFO Total time: 59.394s
Finished: SUCCESS
REST API
Jenkins 2.492.2


localhost:8080/job/sonar-freestyle/
Jenkins
swathirapaka
log out
Dashboard > sonar-freestyle >
Status
Changes
Workspace
Build Now
Configure
Delete Project
SonarQube
Rename
Builds
Filter
Today
#2 17:11
#1 17:07
sonar-freestyle
Add description
SonarQube
SonarQube Quality Gate
'sonar' Passed
server-side processing: Success
Permalinks
Last build (#2), 4 min 26 sec ago
Last stable build (#2), 4 min 26 sec ago
Last successful build (#2), 4 min 26 sec ago
Last unsuccessful build (#1), 8 min 21 sec ago
Last completed build (#2), 4 min 26 sec ago
localhost:8080/job/sonar-freestyle/lastStableBuild/



Creating a Nexus Pipeline in Jenkins

To integrate **Nexus with Jenkins**, follow these steps to set up a Jenkins pipeline for code analysis.

➢ Nexus Plugin Installed in Jenkins

- In Jenkins Dashboard → Go to Manage Jenkins → Manage Plugins
- Search for Nexus Artifact Uploader→ Install it

Credentials:

- Configure credentials in Jenkins (Manage Jenkins → Manage Credentials)

Step 1: Create a Jenkins Pipeline Job

- Open **Jenkins Dashboard** → Click **New Item**
- Select **Pipeline** → Name it (e.g., Nexus) → Click **OK**
- Scroll down to the **Pipeline** section → Choose **Pipeline Script**

Step 2: Define the Jenkins Pipeline Script

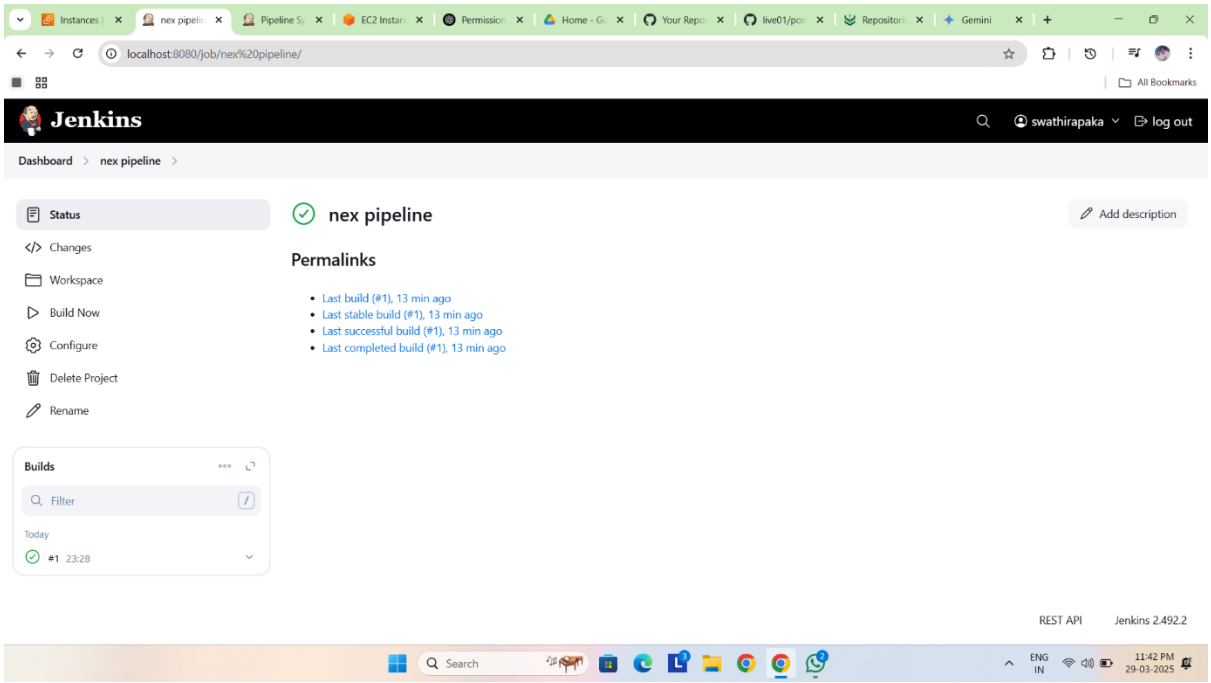
Pipeline Script:

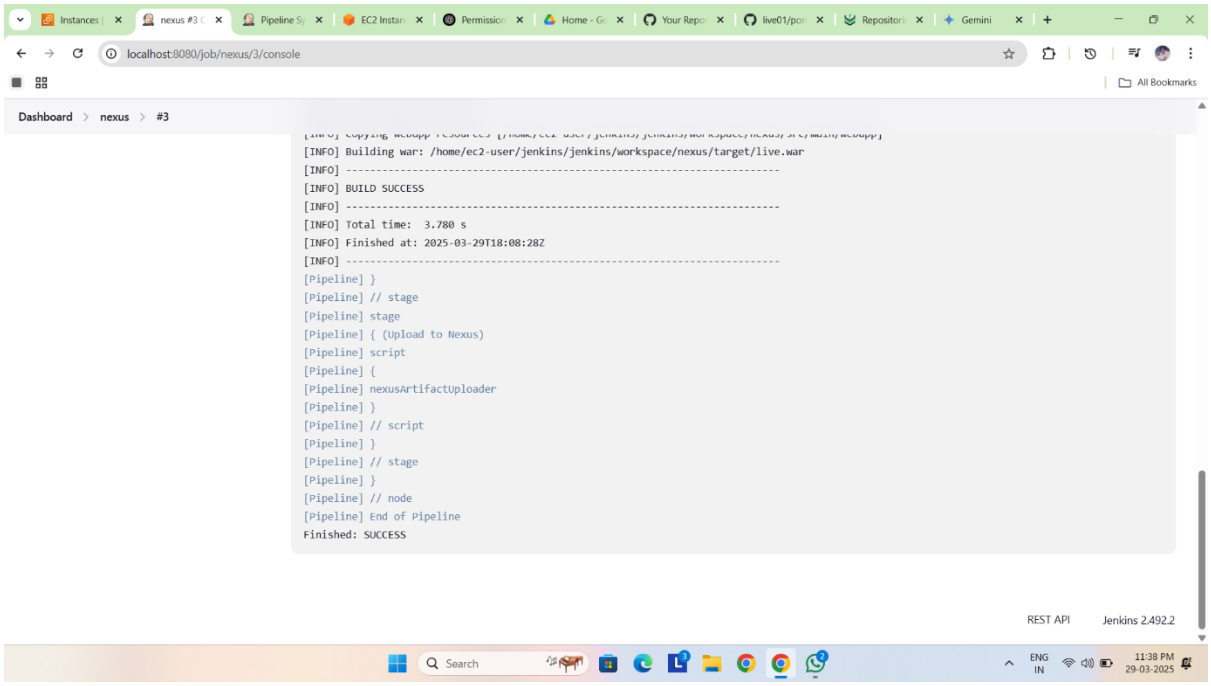
```
pipeline {

    agent {
```

```
        label 'dev}
stages {
        stage('git') {
            steps {
                git branch: 'main', url: 'https://github.com/21A25A0509/live01.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn package'
            }
        stage('Upload to Nexus') {
            steps {
                script {
          nexus Artifact Uploader artifacts: [[artifact Id: ' Vamsi ', classifier: '', file: 'target/live. war', type: 'war']], credentials Id: 'nexus1', group Id: 'vamsi.maven.com', nexus Url: '65.2.149.25:8081', nexus Version: 'nexus3', protocol: 'http', repository: 'nexus', version: '0.0.1-SNAPSHOT'
                }
            }
        }
    }
}
```

Nexus Artifact Uploader artifacts:

- Nexus **Version:** NEXUS3
- Protocol**:** HTTP
- Nexus **URL:** http://43.205.254.253:8081/
- Credentials**:** admin/****** (Stored as nexus-new)
- Group **ID:** vamsi.maven.com
- Version**:** 0.0.1-SNAPSHOT
- Repository**:** nexus2
- Artifact **Details:**
- **Artifact ID:** Vamsi
- **Type:** .war
- **Classifier:** *(Not specified)*
- **File Path:** target/live

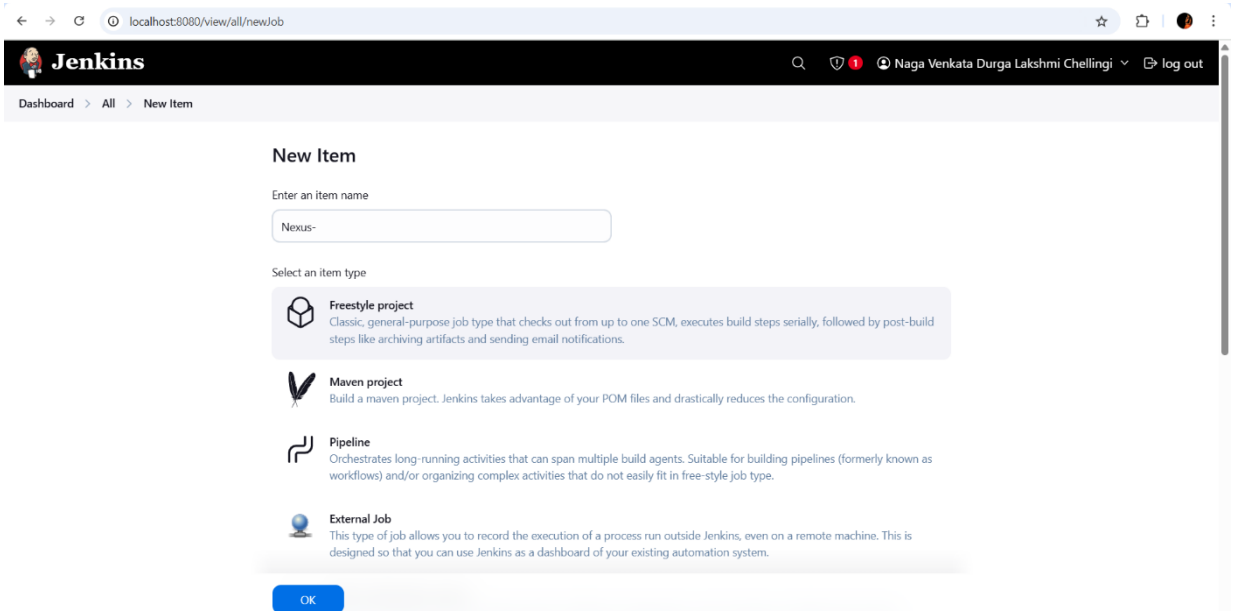




Creating a Nexus Freestyle Job in Jenkins

1. Create a New Freestyle Job

- Open **Jenkins** → Click **New Item**
- Enter a **Job Name** (e.g., Nexus)
- Select **Freestyle project** → Click **OK**



2. Configure the Job

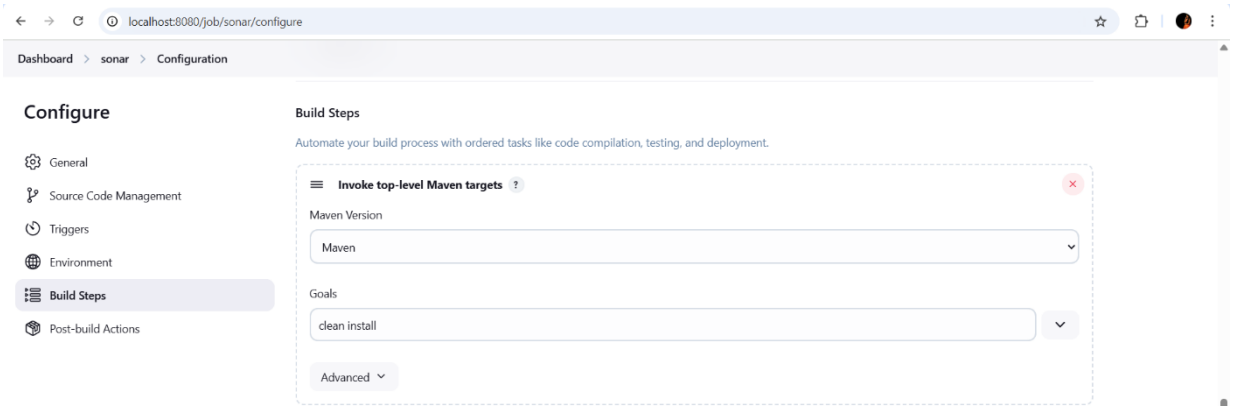
General Section:

- Check **Restrict where this project can be run**
- In the **Label Expression**, enter: dev

Build Steps:

Step 1: Maven Build

- Click **Add Build Step** → Select **Invoke top-level Maven targets**
- In **Goals**, enter:
    - mvn clean



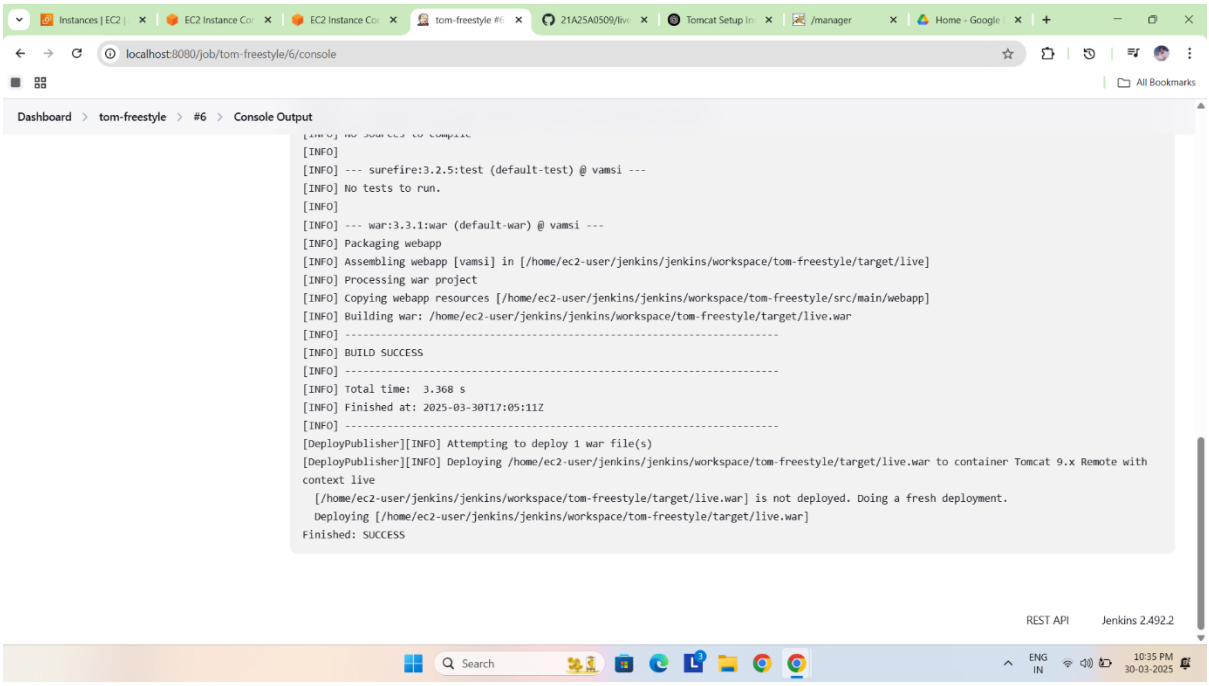
Step2: Nexus artifact uploader

- Click **Add Build Step** → Select Nexus artifact uploader

  Fill the Nexus details

Step3: Save and Build the Job

- Click **Save**
- Click **Build Now** to start the process
- Once the build is complete.



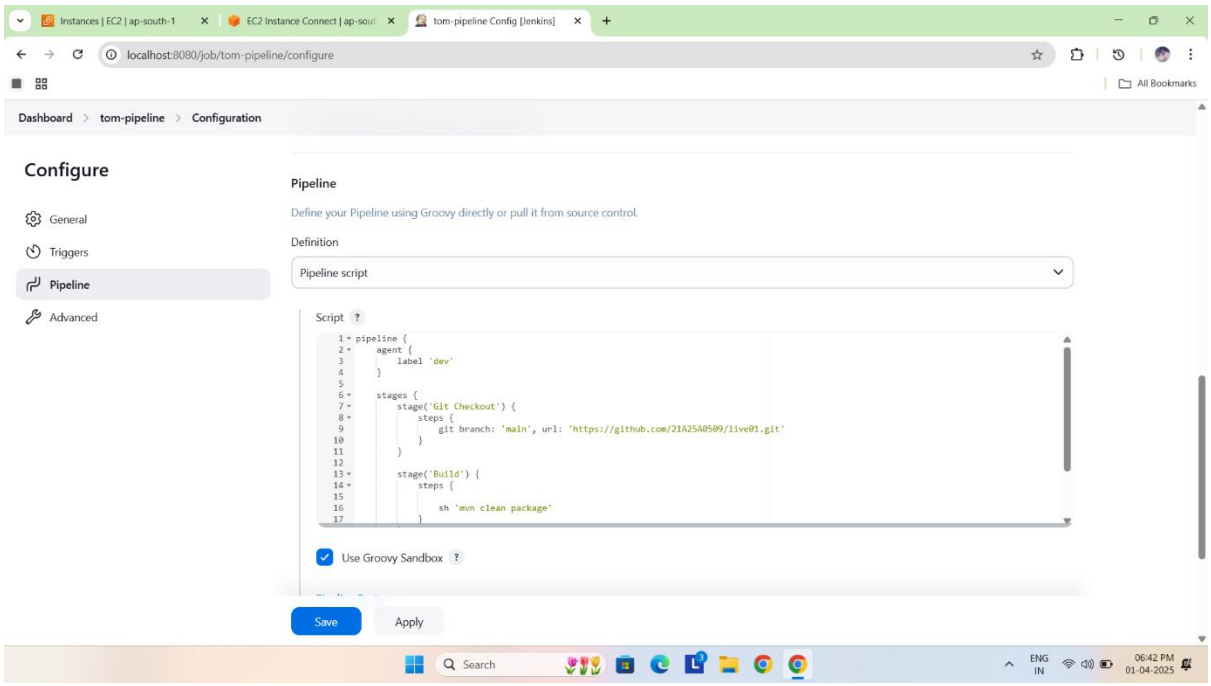
Creating a Tomcat Pipeline in Jenkins To integrate **Tomcat with Jenkins**, follow these steps to set up a Jenkins pipeline for code analysis.

- ➢ Tomcat Plugin Installed in Jenkins
- In Jenkins Dashboard → Go to Manage Jenkins → Manage Plugins
- Deploy war/ear to a container→ Install it

Credentials:

Configure credentials in Jenkins (Manage Jenkins → Manage Credentials)

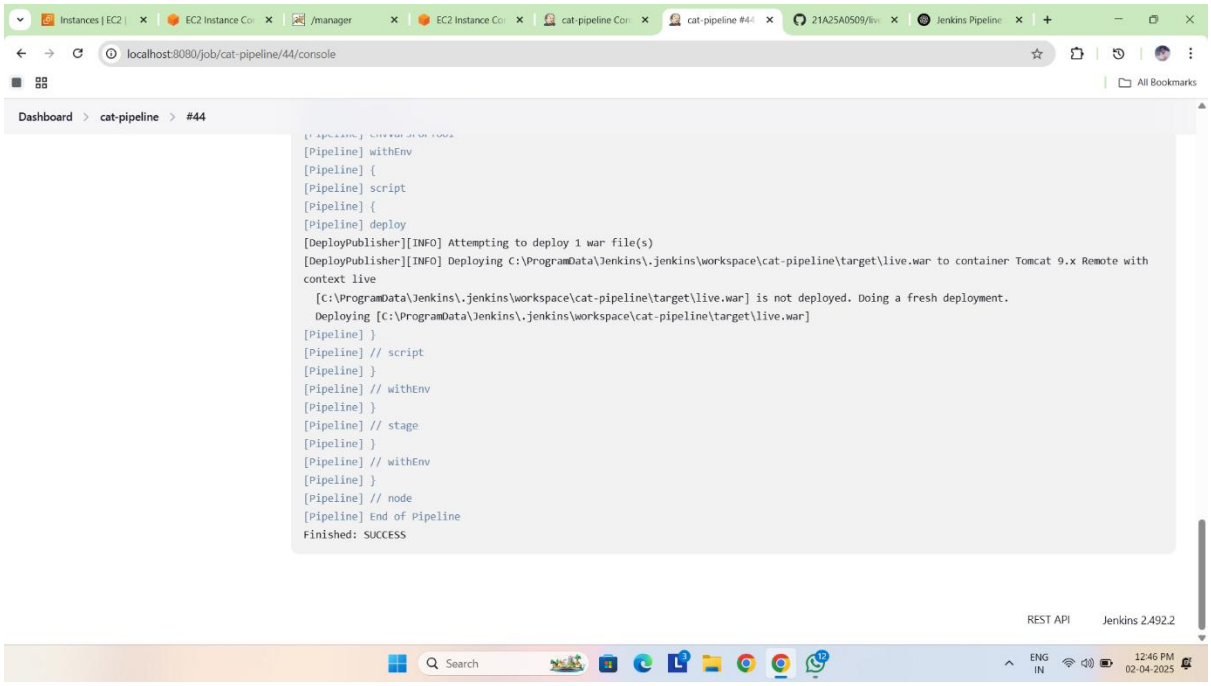
Step 2: Define the Jenkins Pipeline Script

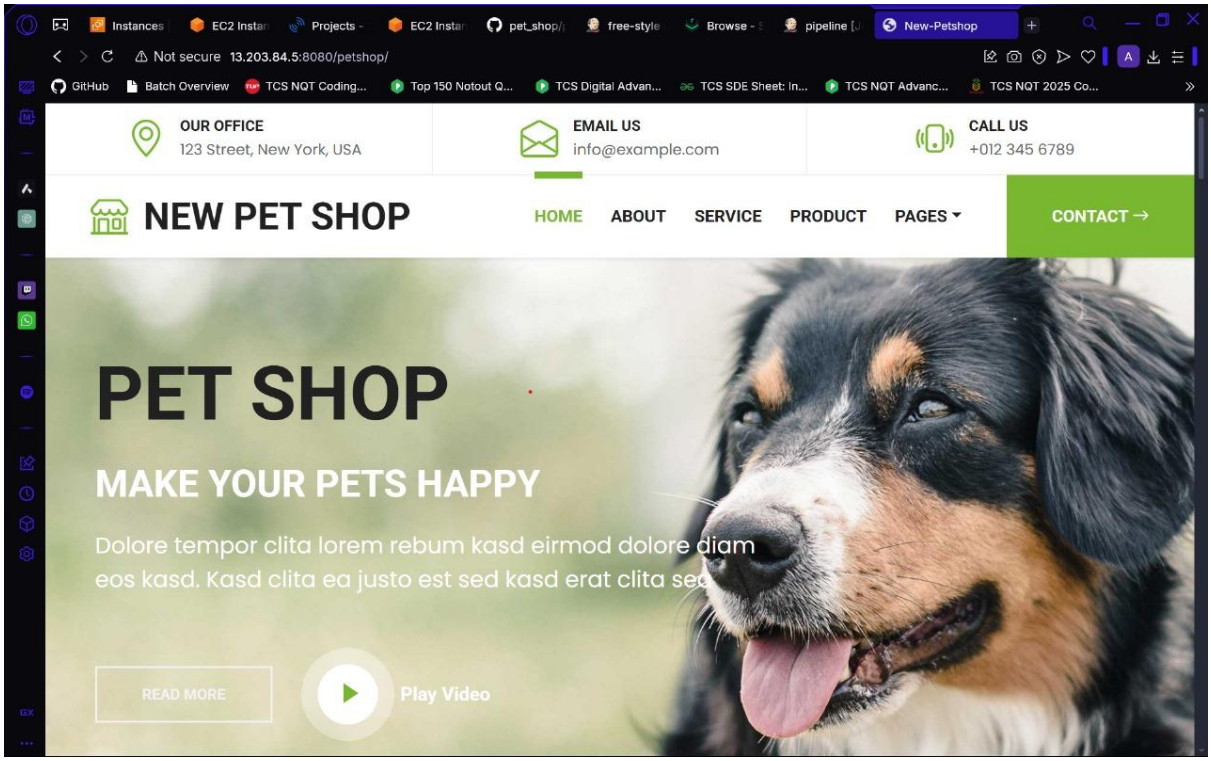


Pipeline Script:

```
pipeline {
    agent {
        label 'dev'
    }
    stages {
        stage('git') {
            steps {
                git branch: 'main', url: 'https://github.com/21A25A0509/live01.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn package'
            }
        }
```

```
        stage('Tomcat') {
            steps {
                deploy adapters: [tomcat9(credentials Id: 'tomcat', url:
'http://172.31.11.188:8080')],
                context Path: 'live', war: '**/*.war'
            }
        }
    }
}
```





Creating a Tomcat Freestyle Job in Jenkins

1. Create a New Freestyle Job

- Open **Jenkins** → Click **New Item**
- Enter a **Job Name** (e.g., Tomcat)
- Select **Freestyle project** → Click **OK**

2.Configure the Job

General Section:

- Check **Restrict where this project can be run**
- In the **Label Expression**, enter: dev

Source Code Management:

- Select **Git**
- In the **Repository URL**, enter:
  - https://github.com/21A25A0509/live01.git
- In the **Branches to build**, enter:

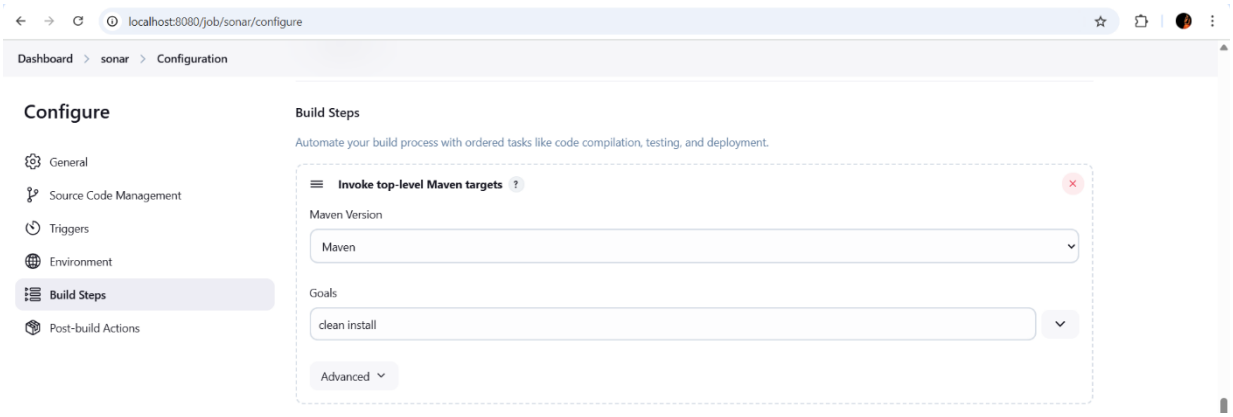
- */main

Build Steps:

Step 1: Maven Build

- Click **Add Build Step** → Select **Invoke top-level Maven targets**
- In **Goals**, enter:

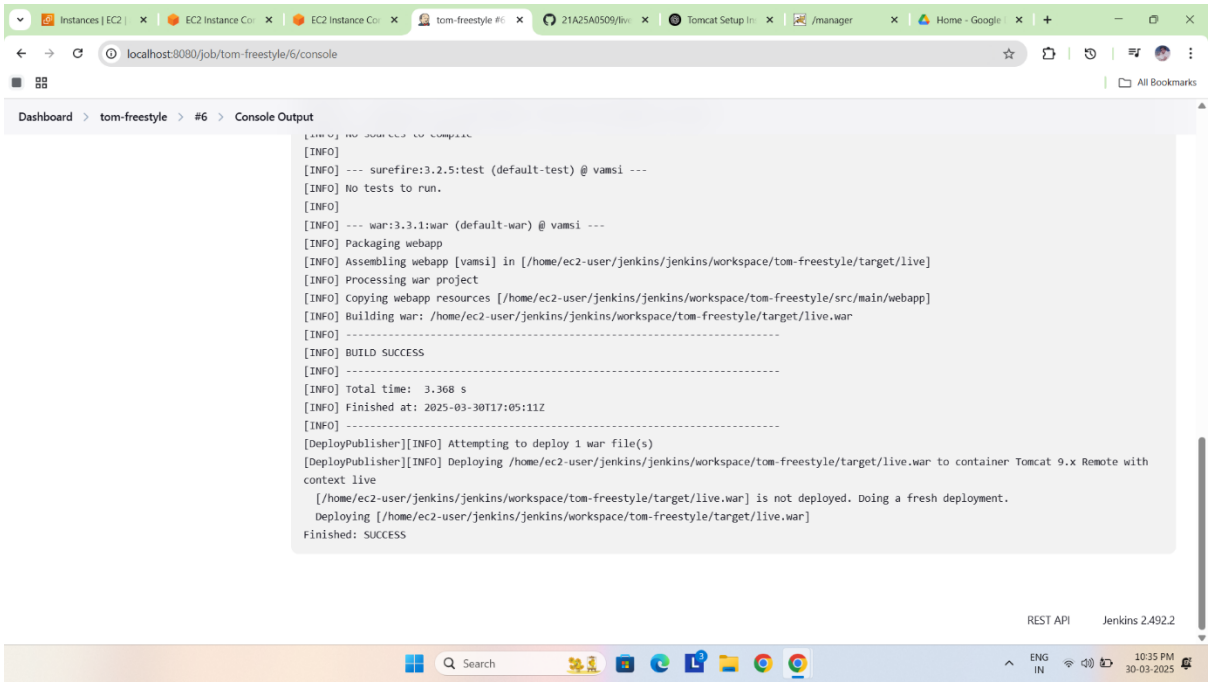
    - mvn clean

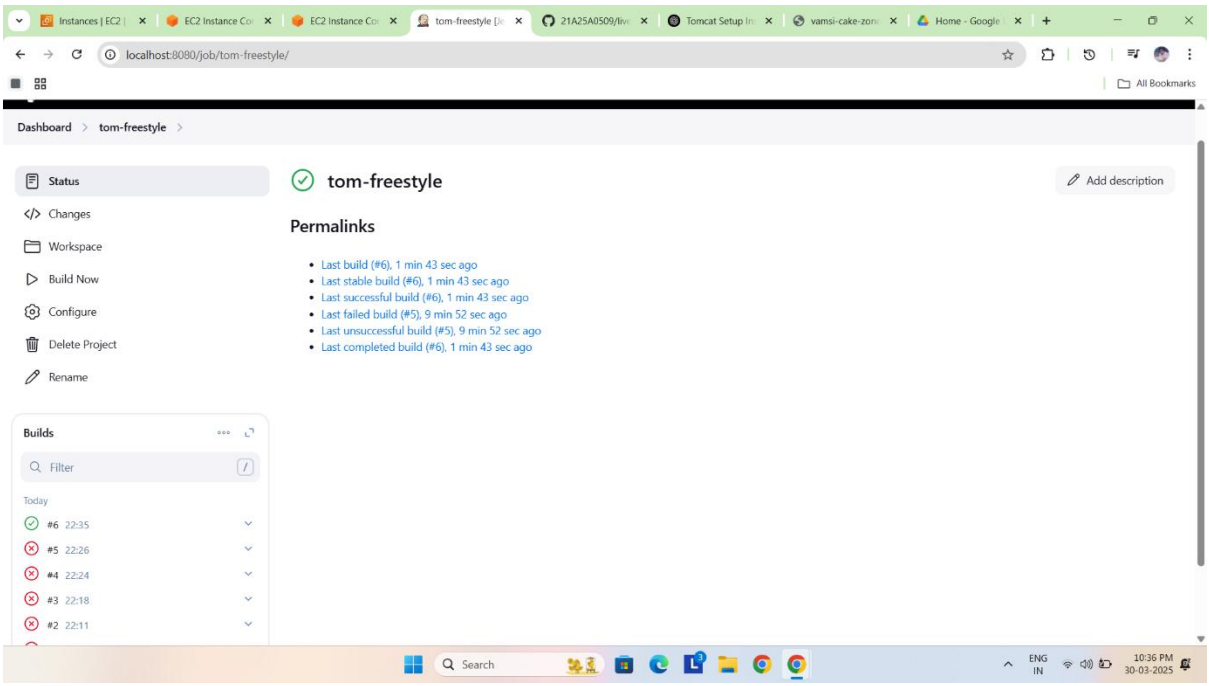


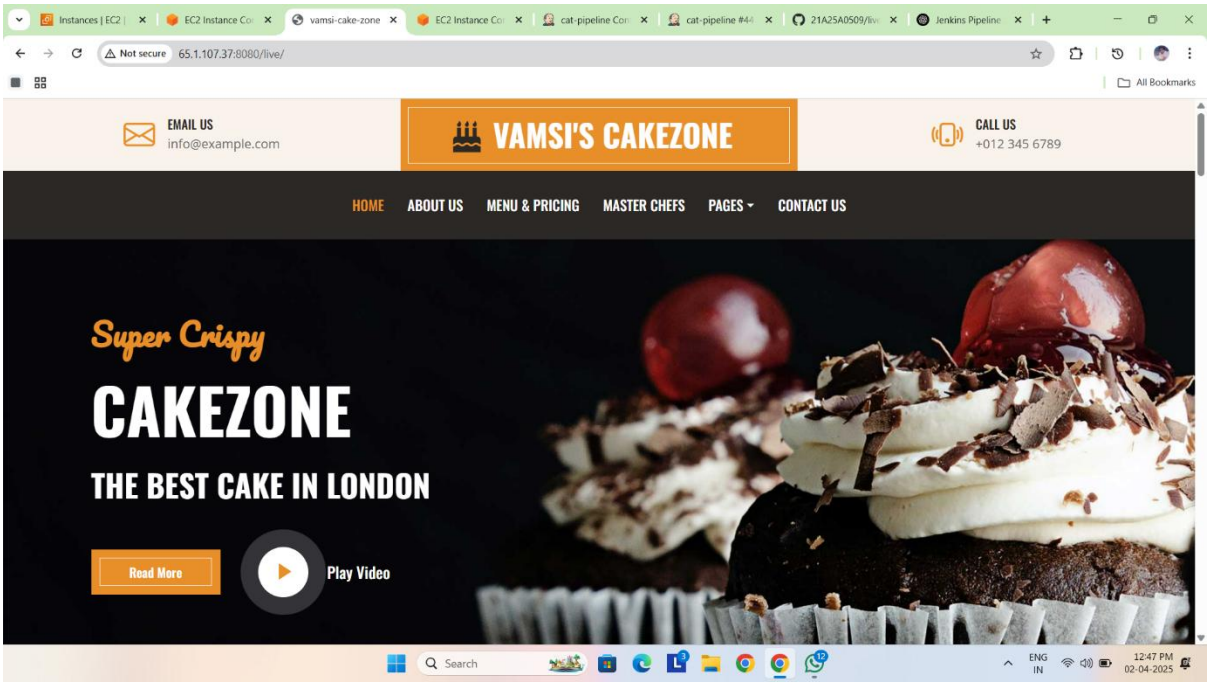
Step3: Save and Build the Job

- Click **Save**
- Click **Build Now** to start the process

Once the build is complete


localhost:8080/job/tom-freestyle/6/console
Dashboard > tom-freestyle > #6 > Console Output
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ vamsi ---
[INFO] No tests to run.
[INFO]
[INFO] --- war:3.3.1:war (default-war) @ vamsi ---
[INFO] Packaging webapp
[INFO] Assembling webapp [vamsi] in [/home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/target/live]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/src/main/webapp]
[INFO] Building war: /home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/target/live.war
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 3.368 s
[INFO] Finished at: 2025-03-30T17:05:11Z
[INFO] --
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/target/live.war to container Tomcat 9.x Remote with context live
[/home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/target/live.war] is not deployed. Doing a fresh deployment.
Deploying [/home/ec2-user/jenkins/jenkins/workspace/tom-freestyle/target/live.war]
Finished: SUCCESS
REST API
Jenkins 2.492.2


localhost:8080/job/tom-freestyle/
Dashboard > tom-freestyle >
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
tom-freestyle
Add description
Permalinks
Last build (#6), 1 min 43 sec ago
Last stable build (#6), 1 min 43 sec ago
Last successful build (#6), 1 min 43 sec ago
Last failed build (#5), 9 min 52 sec ago
Last unsuccessful build (#5), 9 min 52 sec ago
Last completed build (#6), 1 min 43 sec ago
Builds
Filter
Today
#6 22:35
#5 22:26
#4 22:24
#3 22:18
#2 22:11



Installing MySQL & Integrating with Tomcat 9

1. Install MySQL on Amazon Linux 2023

   Step1: Download the MySQL Repository

   Sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm

   Step 2: Install the Repository

   sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y

   Step 3: Import the MySQL GPG Key

   sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023

   Step 4: Install MySQL Server

   sudo dnf install mysql-community-server -y

   Step 5: Start MySQL Service

   sudo systemctl start MySQL. service

   Step 6: Enable MySQL to Start on Boot

   Sudo systemctl enable mysqld

2.**Access MySQL & setup Database**

After installation, MSQL generates a temporary password for the root user

Step 1: Find Temporary password

Sudo grep ‘temporary password’/var/log/mysql/log copy the password show in the output

Step 2: log into MySQL

mysql -u root -p Paste the temporary password when prompted.

Step 3: Change the Default Password

ALTER USER 'root'@'localhost' IDENTIFIED BY 'NewPassword@123';

Note: The password must include uppercase, lowercase, a number, and a special character.

3. Create a Database & Table

Once logged into MySQL, create a database and table for storing user data.

Step 1: Create a New Database

CREATE DATABASE Vamsi; USE Vamsi;

Step 2: Create a Table

CREATE TABLE users (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), email VARCHAR(255));

Step 3: View Databases & Tables

SHOW DATABASES;

SHOW TABLES;

5. Configure MySQL with Tomcat 9

Now, we need to connect MySQL with Apache Tomcat 9.

Step1: Download the MySQL Connector JAR

Wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.33.tar.gz

Step 2: Extract the JAR File

tar -xvzf mysql-connector-java-8.0.33.tar.gz

Step 3: Move the JAR to Tomcat's Library Folder

Sudo mv mysql-connector-java-8.0.33/mysql-connector-java-8.0.33.jar /opt/tomcat/lib/

Step 4: Restart Tomcat

sudo systemctl restart tomcat

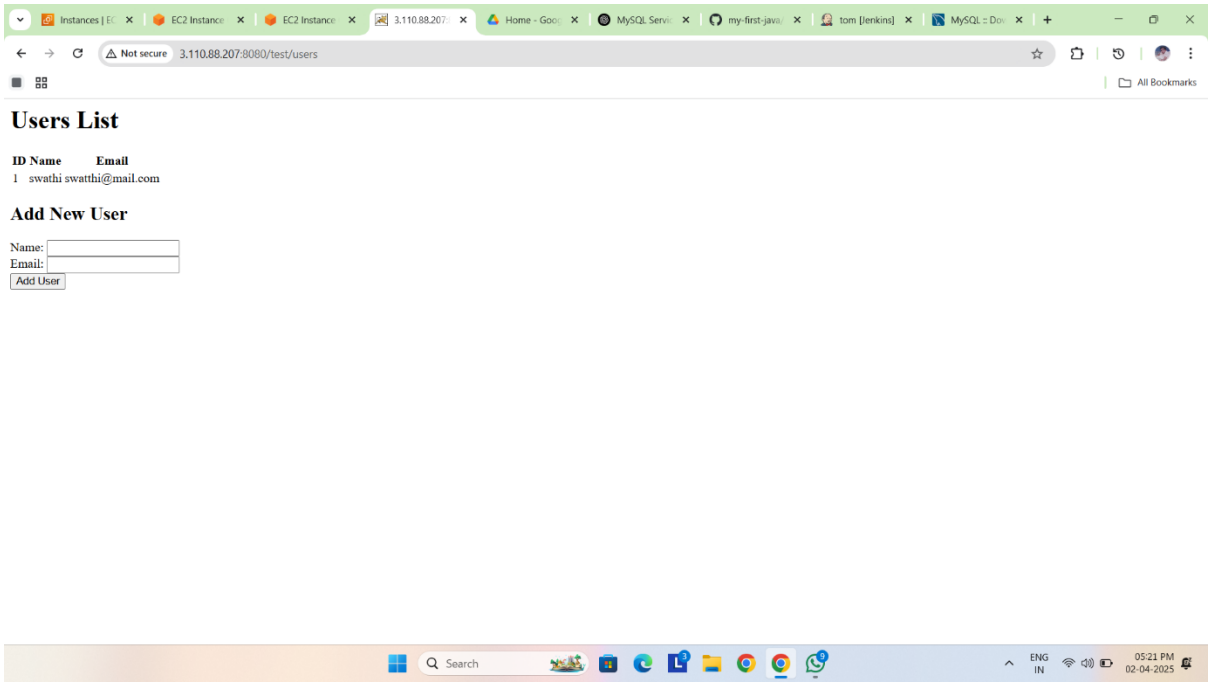
5. Verify Everything

1. Open **Tomcat Manager** (http://your-server-ip:8080).

2. Click on View Users.

3. Add new users.

4. Data should be stored in MySQL


Not secure
3.110.88.207:8080/test/users
All Bookmarks
Users List
ID Name Email
1 swathi swatthi@mail.com
Add New User
Name:
Email:
Add User
Search
ENG
IN
05:21 PM
02-04-2025