DEVOPS

JENKINS CI/CD AUTOMATION PROJECT

Project Summary:

Implemented a Complete CI/CD pipeline using Jenkins for a java-based web application. The project involved automating the build ,test,and deployment process using both freestyle jobs and pipeline Scripts to ensure faster and more reliable software delivery.

Tools Involved: Git , Maven, SonarQube, Nexus, Docker, and Tomcat

1. Servers:

- **Jenkins Master:**
- **Purpose:** Acts as the central control unit of Jenkins. It is responsible for scheduling jobs, managing job configurations, and dispatching build tasks to slave nodes.
- Requirements to launch
- Instance Type: t2. Micro
- Storage: 8gb
- AMI Details: Amazon Linux 2 AMI (HVM)-Kernal 5.10,SSD Volume Type.

Jenkins Slave (Agent):

- Purpose: Executes build jobs and relieves the master from resource-intensive tasks.
- Instance Type: t2. Medium,Storage: 8gb
- AMI: Amazon Linux 2023 AMI

3. SonarQube Server:

- **Purpose:** Performs static code analysis and quality checks on the code being built. It provides insights into code smells, bugs, and vulnerabilities.
- Instance Type: t2. Medium,Storage: 8gb
- AMI: Amazon Linux 2023 AMI

4. Nexus Server:

- **Purpose:** Acts as an artifact repository for storing build artifacts like .jar, .war, or .zip files. Nexus supports dependency management and versioning.
- Instance Type: t2. Medium,Storage: 8gb
- AMI Details: Amazon Linux 2023 AMI

5. Tomcat Server:

- **Purpose:** Hosts Java-based applications and deploys .war files generated during builds.
- Instance Type: t2. Micro,Storage: 8gb
- AMI: Amazon Linux 2023

Port numbers:

- Jenkins:8080
- SonarQube:9000
- Nexus:8081
- Tomcat:80

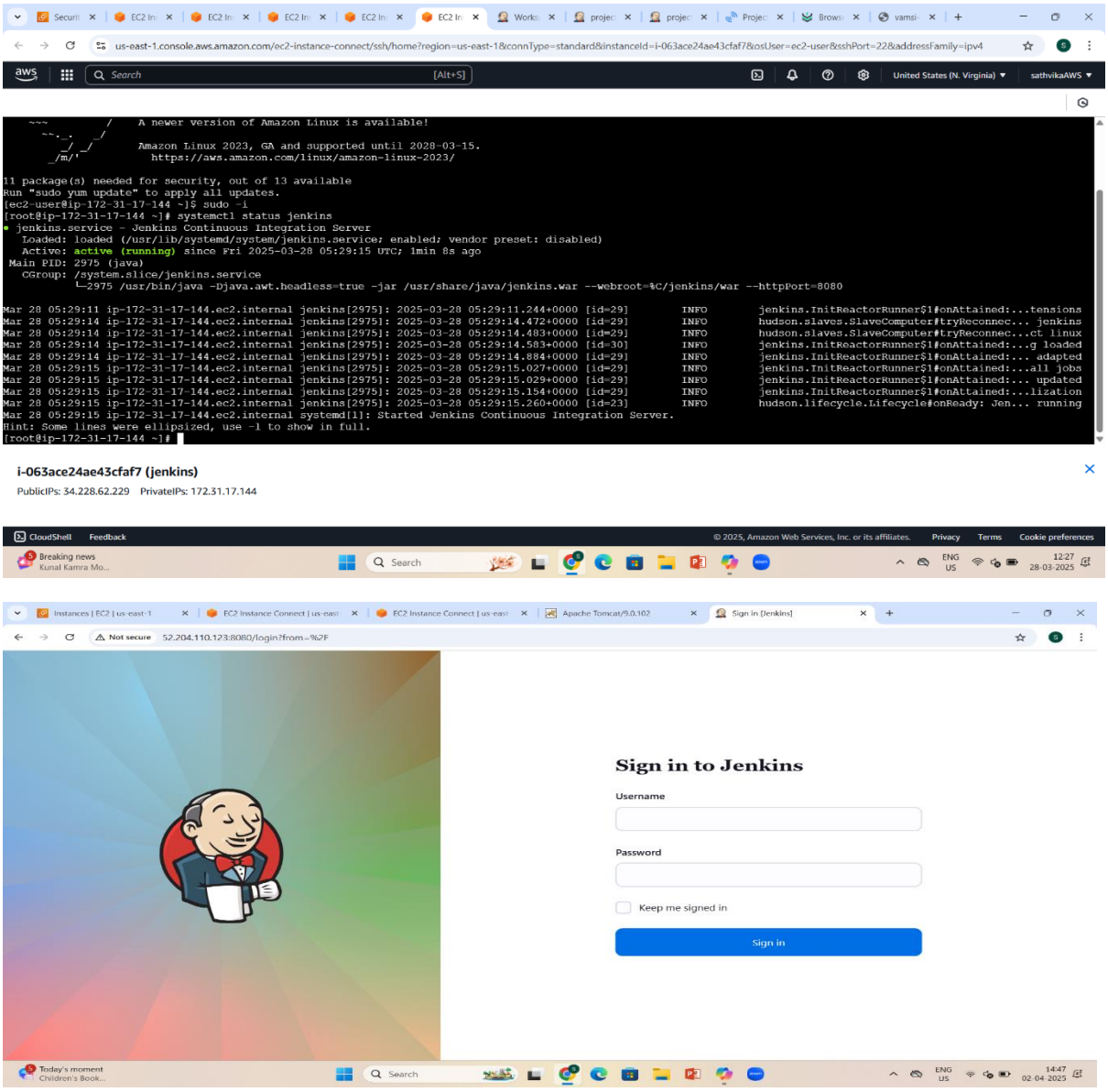
JENKINS SETUP IN AWS:

- **Prepare the AWS Environment**
- Launch an Amazon EC2 Instance
- Log in to your AWS Management Console and create an EC2 instance.
- Use the Amazon Linux AMI for compatibility.

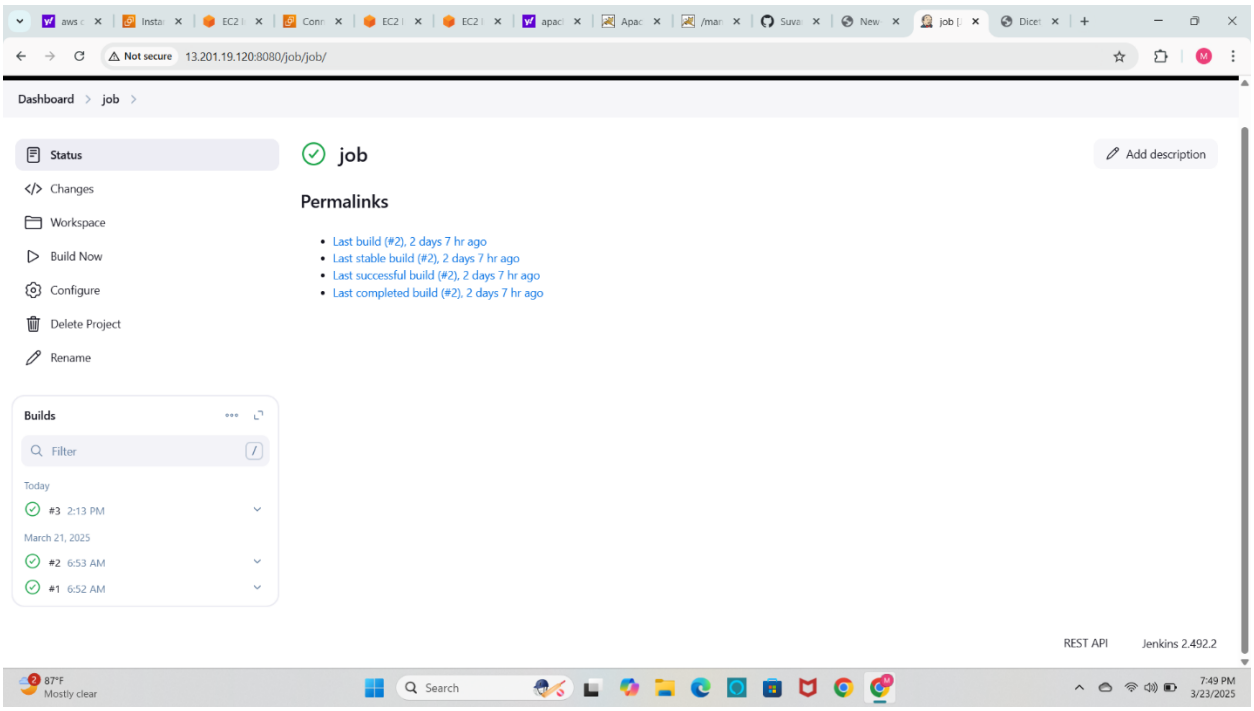
- Configure the instance type (e.g., t2.micro for master).
- Set up a security group that allows inbound traffic on ports 8080 (for Jenkins)
- **Create a Key Pair:**
- Generate a key pair for secure SSH access.
- Save the private key file (.pem) to your local machine.

Install Jenkins on Amazon Linux:

- **Update the System:**
- Sudo yum update –y
- **Install Java (required for Jenkins):**
- Sudo yum install java-23-openjdk –y
- **Add Jenkins Repository**
- Sudo wget -O /etc/yum.repos.d/jenkins.repo
- https://pkg.jenkins.io/redhat-stable/jenkins.repo
- Sudo rpm --import https://pkg.jenkins.io/redhat- stable/jenkins.io- 2023. Key
- Install Jenkins
- sudo yum install jenkins -y
- Start and Enable Jenkins
- sudo systemctl start jenkins
- sudo systemctl enable Jenkins



- GO TO NEW ITEM – SELECT FREESTYLE – CONFIGURE JOB – SAVE
- CLICK ON BUILD NOW ,ONCE BULID IS SUCCESS
- THEN JOB IS CREATED



- TO CONFIGURE/INTEGRATE SONARQUBE WITH JENKINS
- SELECT THE CREATED JOB TO INTEGRATE
- ***Steps to install SonarQube on a T2 Medium AWS EC2 instance:***

1. **Launch an EC2 Instance**:
    - Go to the AWS Management Console and create a new EC2 instance.
    - Choose **Amazon Linux 2023** as the operating system.
    - Select the **T2 Medium** instance type.
    - Configure security groups to allow ports **9000** (SonarQube)
2. **Connect to EC2 Instance:**
    - connect to your EC2 instance.
3. **Install Java**:
    - SonarQube requires Java. Install OpenJDK 11 or a compatible version:
    - sudo yum install java-11-openjdk -y

4. **Download and Install SonarQube**:
    - Download the latest SonarQube version from the official website.
    - Extract the downloaded file:
    - tar -xvzf sonarqube-<version>.zip
5. **Configure SonarQube**:
    - Edit the sonar.properties file in the SonarQube directory to connect it to your database.
6. **Start SonarQube**:
    - Navigate to the SonarQube directory and start the service:
    - ./bin/linux-x86-64/sonar.sh start
7. **Access SonarQube**:
    - Open your browser and go to http://<your-ec2-public-ip>:9000.

Steps to integrate SonarQube with Jenkins:

1. **Install SonarQube Scanner Plugin**:
    - In Jenkins, go to **Manage Jenkins** → **Manage Plugins**.
    - Search for "SonarQube Scanner" in the **Available** tab and install it.
2. **Configure SonarQube in Jenkins**:
    - Go to **Manage Jenkins** → **Configure System**.
    - Scroll to the **SonarQube servers** section.
    - Add a new server, provide a name, and enter the SonarQube server URL.
    - Add the authentication token generated in SonarQube.

3. **Create a Jenkins Project**:
    - Use a **Freestyle Project** in Jenkins.
    - In the project configuration, enable **Execute SonarQube Scanner** under the build section.

4. **Run Analysis**:
    - Save the project and trigger a build.
    - Jenkins will use SonarQube Scanner to analyze the code and send the results to the SonarQube server.

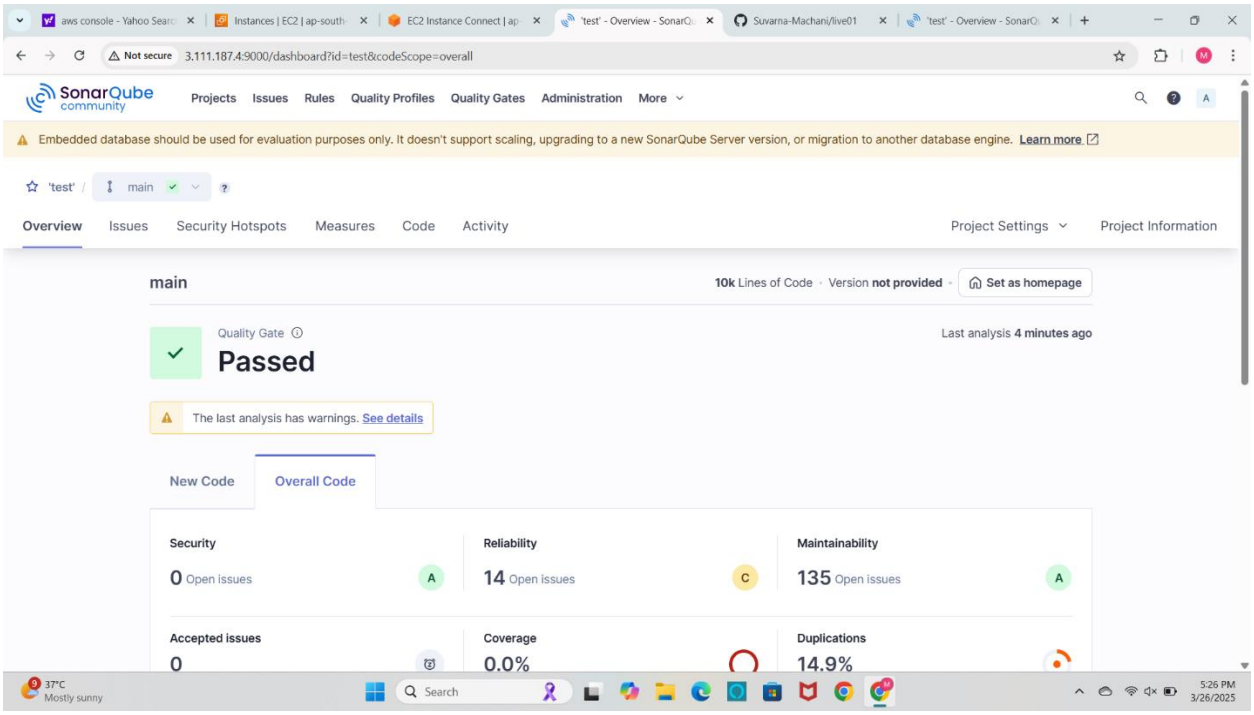
TESTING THE CODE IN SONARQUBE:

1. **Prepare the Environment**:
    - Ensure SonarQube is running and accessible.
    - Confirm the SonarQube Scanner plugin is installed in Jenkins.
2. **Create a Freestyle Project**:
    - Use a Freestyle Project which is created before.
    - Add your source code repository under the **Source Code Management** section.
3. **Configure SonarQube Scanner**:
    - In the project configuration, go to the **Build Environment** section.
    - Check the box for **SonarQube Scanner environment**.
    - Select your SonarQube server from the dropdown.

4. **Add Build Step**:
    - Add a build step to execute the SonarQube Scanner.
    - Provide the necessary properties, such as:
    - sonar.projectKey=your_project_key
    - sonar.sources=./src
    - sonar.host.url=http://<sonarqube-server-ip>:9000
    - sonar.java/binaries=target/classes
5. **Run the Build**:
    - Save the configuration and trigger a build.
    - Jenkins will analyze the code and send the results to SonarQube.
6. **View Results**:
    - Open the SonarQube dashboard and check the analysis results for your project.

SONARQUBE CODE ANALYSIS/TEST RESULTS :

- CODE IS TESTED AND THE QUALITY GATE HAS BEEN PASSED



Steps to install Nexus Repository Manager on your system:

1. Download Nexus:

- Visit Sonatype's official website and download the appropriate version for your operating system.

2. Install Java:

- Search for Java version using YUM LIST | GREP “JAVA” Command
- Their we find 1.8.0-amazon-corette file copy and paste it after the above command to download java version for nexus repository

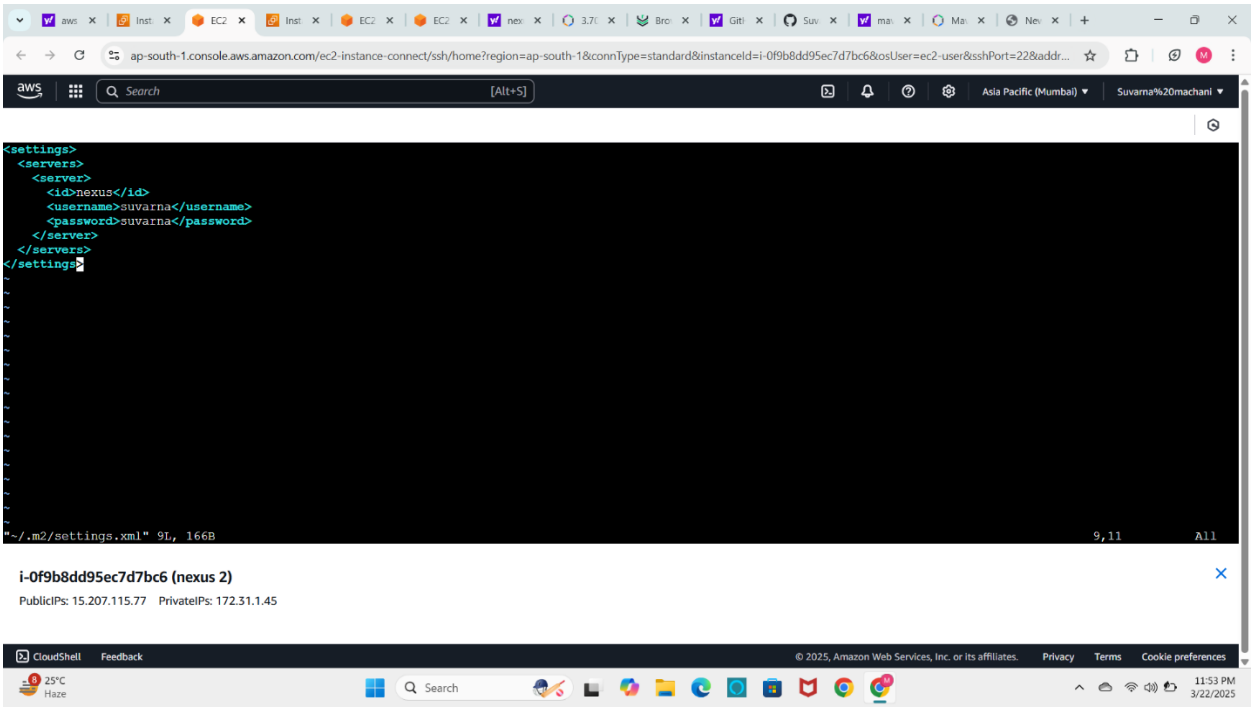
3. Extract Nexus Files

- Extract the downloaded archive
- Using tar -xvzf command and move

4. Run Nexus:

- Using cd /opt/nexus

- Navigate the directory cd bin/
- To Start the Nexus Run the Following Command
    - Sh nexus start – To Start
    - Sh nexus status – To check the status

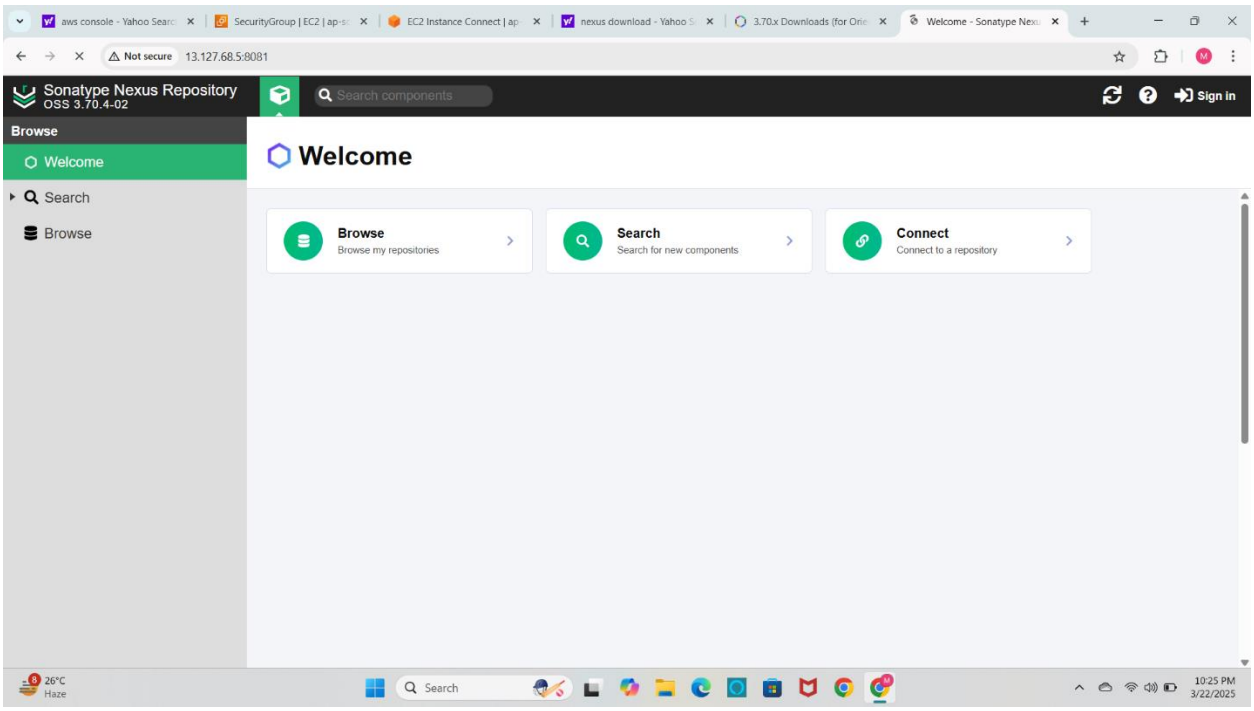


5. Access Nexus

- Open your browser and go to http://localhost:8081.
- Log in using the default credentials:
    - **Username**: admin
    - **Password**: Check the admin.password file in the nexus-data directory.

6. Configure Repositories

- Once logged in, configure the necessary repositories (e.g., Maven) based on your project's requirements.



Integrating Nexus with Jenkins :

1.Install Jenkins: Ensure Jenkins and nexus in launched with same security groups .

2.**Add Nexus Credentials to Jenkins**:

- Go to Jenkins Dashboard > Manage Credentials.
- Add Nexus credentials (username and token/password) under the appropriate scope.

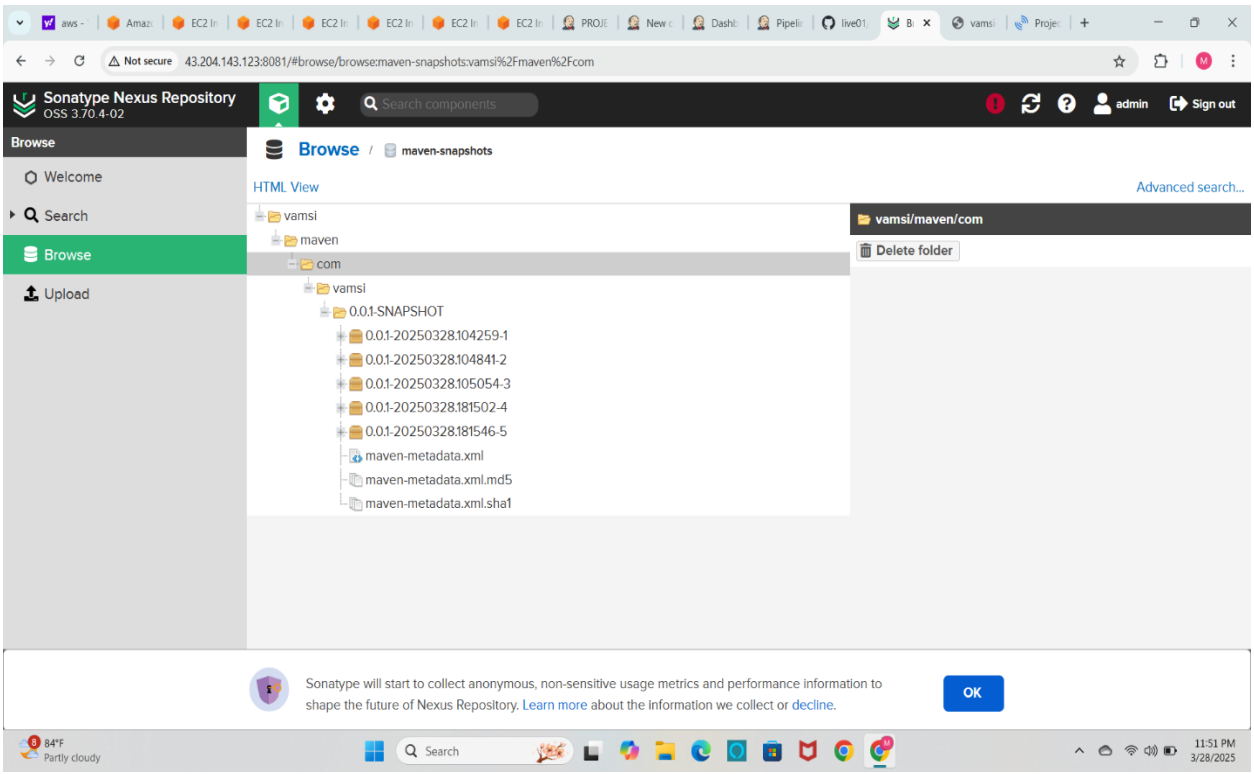
3.Install Required Plugins:

In Jenkins, go to Manage Jenkins > Manage Plugins.

- Install relevant plugins like Nexus *Artifact Uploaders* for easier integration.

4**. Verify Artifacts in Nexus**

- Log in to the Nexus Repository Manager.
- Navigate to the repository where the artifacts (e.g., WAR files) were uploaded (e.g., maven-snapshots or maven-releases).
- Confirm the presence of the newly built artifacts with correct versioning.



Steps to install Apache Tomcat on an EC2 instance:

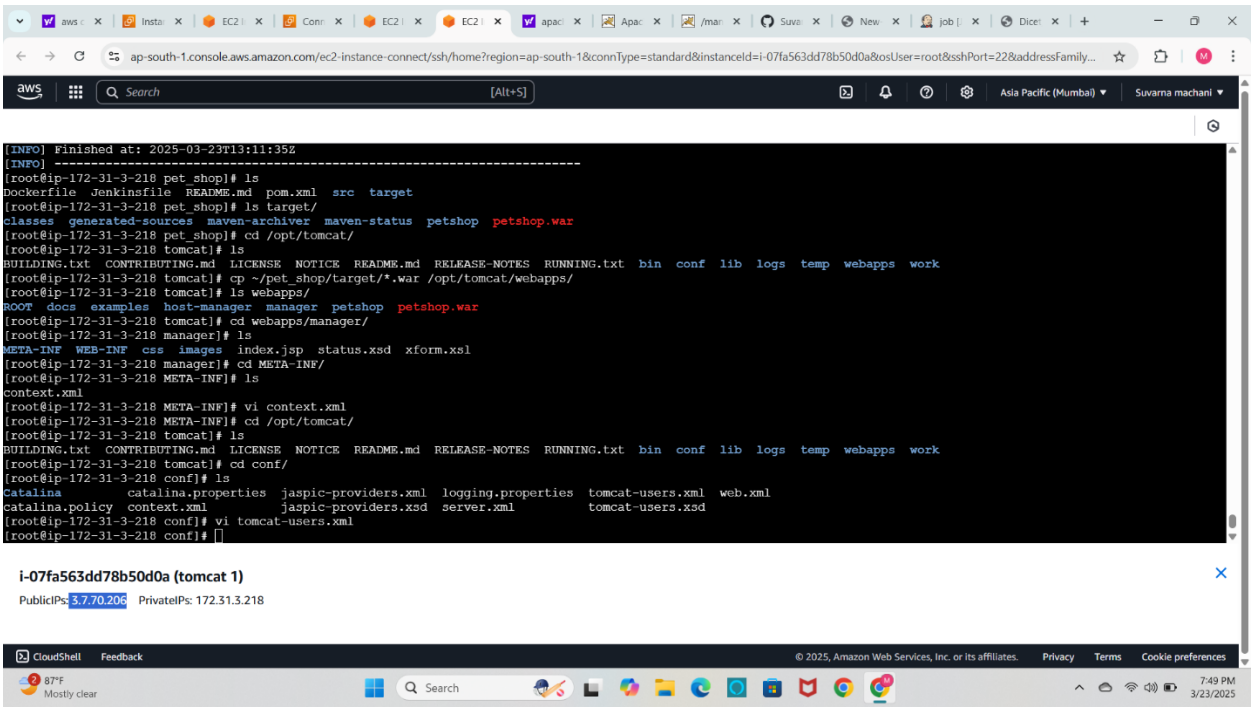
1. **Launch an EC2 Instance**:
   - Log in to your AWS Management Console.

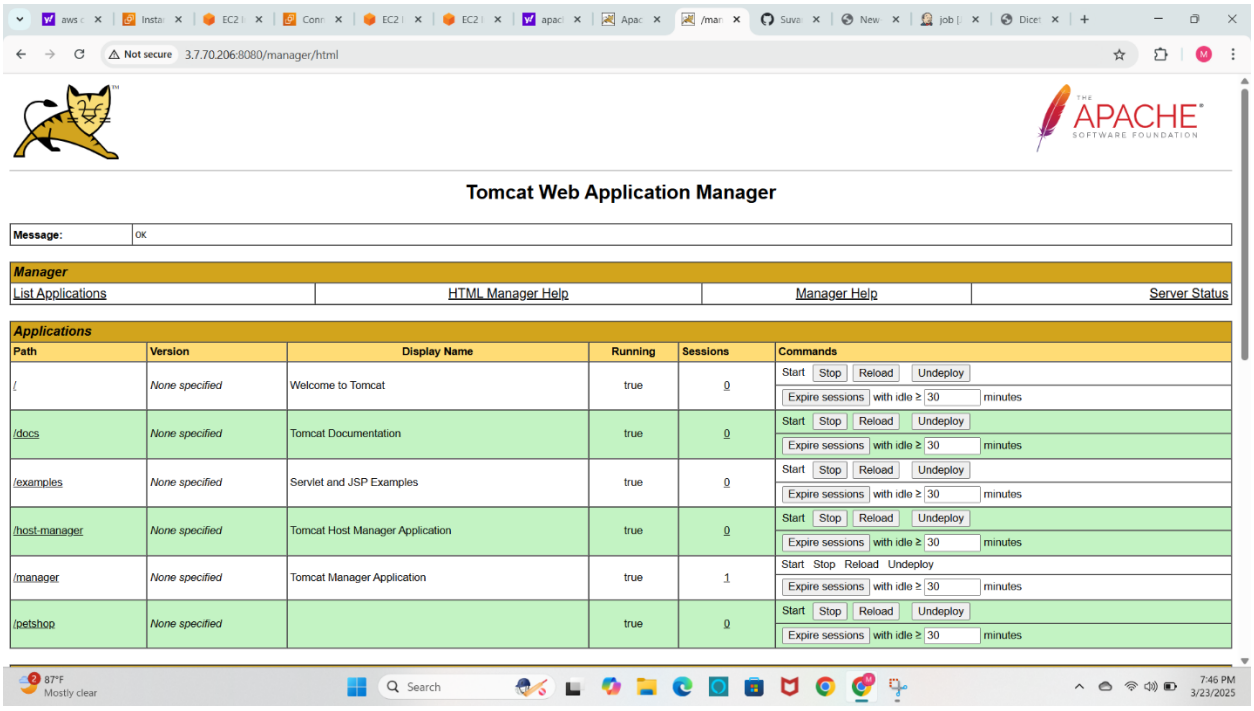
- Navigate to the EC2 Dashboard and click "Launch Instance."
- Choose an Amazon Linux 2023
- Configure the instance type (e.g., t2.medium) and security group to allow inbound traffic on ports and add 8080 (Tomcat).

2. **Connect to the EC2 Instance**:
   - connect to your instance:
   - path/to/your-key.pem ec2-user@your-ec2-public-ip
3. **Install Java**:
   - Update the package list:
   - sudo yum update -y # For Amazon Linux
   - Install Java:
   - sudo amazon-linux-extras install java-openjdk11 -y # Amazon Linux
   - Verify the installation:
   - java -version
4. **Download and Install Tomcat**:
   - Navigate to the /opt directory:
   - cd /opt
   - Download the latest Tomcat binary:
   - wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.11/bin/apache-tomcat-10.1.11.tar.gz
   - Extract the downloaded file:
   - tar -xvzf apache-tomcat-10.1.11.tar.gz
5. **Set Permissions**:
   - Navigate to the bin directory of Tomcat:

- cd apache-tomcat-10.1.11/bin
- Add execute permissions to the startup and shutdown scripts:
- chmod +x startup.sh shutdown.sh

6. **Start Tomcat**:
   - Start the Tomcat server:
   - ./startup.sh
7. **Access Tomcat**:
   - Open your web browser and navigate to:
   - http://<ipaddress>:8080
   - You should see the Tomcat welcome page.
8. **Configuration**:
   - Edit cd webapps/manager/META-INF/context.xml
   - Comment the <valve></valve> part
   - Edit the tomcat-users.xml file to set up user roles and credentials for the Tomcat (Manager –GUI,Manager—Script).

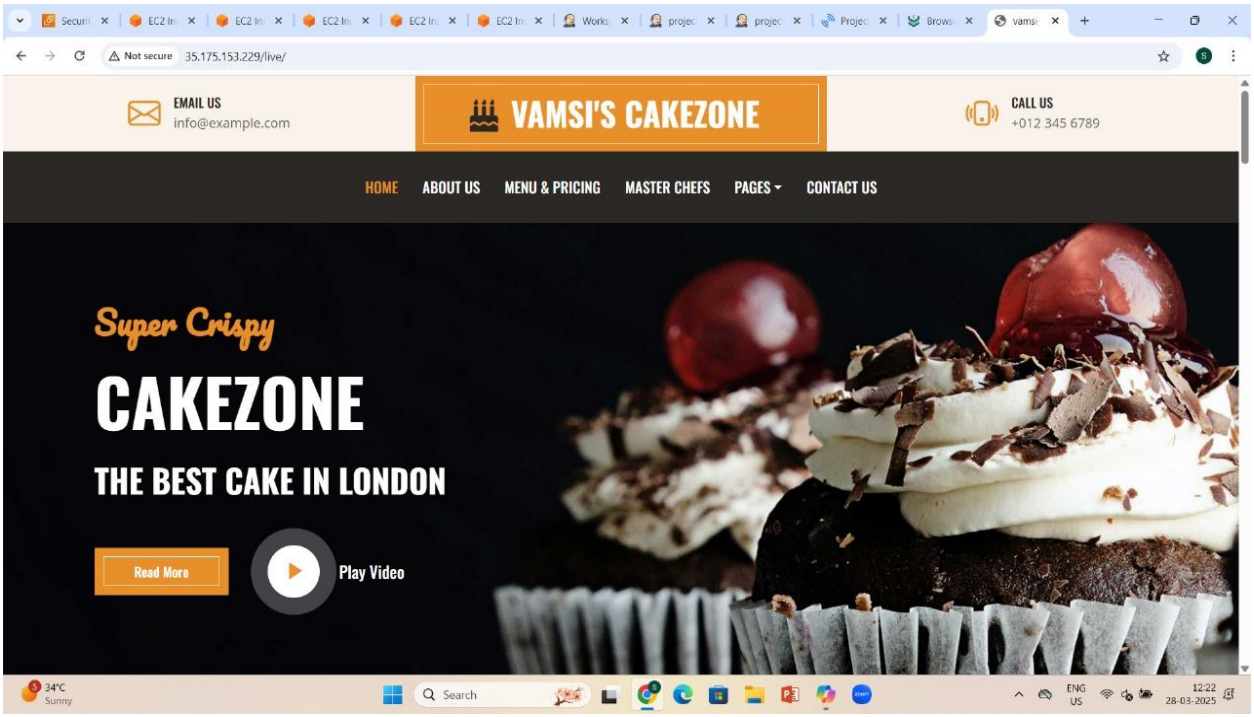

[INFO] Finished at: 2025-03-23T13:11:35Z
[INFO] --
[root@ip-172-31-3-218 pet_shop]# ls
Dockerfile Jenkinsfile README.md pom.xml src target
[root@ip-172-31-3-218 pet_shop]# ls target/
classes generated-sources maven-archiver maven-status petshop petshop.war
[root@ip-172-31-3-218 pet_shop]# cd /opt/tomcat/
[root@ip-172-31-3-218 tomcat]# ls
BUILDING.txt CONTRIBUTING.md LICENSE NOTICE README.md RELEASE-NOTES RUNNING.txt bin conf lib logs temp webapps work
[root@ip-172-31-3-218 tomcat]# cp ~/pet_shop/target/*.war /opt/tomcat/webapps/
[root@ip-172-31-3-218 tomcat]# ls webapps/
ROOT docs examples host-manager manager petshop petshop.war
[root@ip-172-31-3-218 tomcat]# cd webapps/manager/
[root@ip-172-31-3-218 manager]# ls
META-INF WEB-INF css images index.jsp status.xsd xform.xsl
[root@ip-172-31-3-218 manager]# cd META-INF/
[root@ip-172-31-3-218 META-INF]# ls
context.xml
[root@ip-172-31-3-218 META-INF]# vi context.xml
[root@ip-172-31-3-218 META-INF]# cd /opt/tomcat/
[root@ip-172-31-3-218 tomcat]# ls
BUILDING.txt CONTRIBUTING.md LICENSE NOTICE README.md RELEASE-NOTES RUNNING.txt bin conf lib logs temp webapps work
[root@ip-172-31-3-218 tomcat]# cd conf/
[root@ip-172-31-3-218 conf]# ls
catalina catalina.properties jaspic-providers.xml logging.properties tomcat-users.xml web.xml
catalina.policy context.xml jaspic-providers.xsd server.xml tomcat-users.xsd
[root@ip-172-31-3-218 conf]# vi tomcat-users.xml
[root@ip-172-31-3-218 conf]#
i-07fa563dd78b50d0a (tomcat 1)
PublicIPs: 3.7.70.206 PrivateIPs: 172.31.3.218


Not secure 3.7.70.206:8080/manager/html
Tomcat Web Application Manager
Message: OK
Manager
List Applications HTML Manager Help Manager Help Server Status
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
/petshop None specified true 0 Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Steps to integrate Apache Tomcat with Jenkins for deploying applications:

1. **Install Jenkins and Tomcat**:
    - Ensure both Jenkins and Tomcat are installed and running on your server.
    - Verify that Java is installed, as both Jenkins and Tomcat require it.
2. **Install the "Deploy to Container" Plugin in Jenkins**:
    - Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins.
    - Search for the "Deploy to Container" plugin under the "Available Plugins" and install it.
    - Restart Jenkins to apply the changes.
3. **Configure Tomcat for Deployment**:
    - Edit the tomcat-users.xml file located in the conf directory of your Tomcat installation.
    - Add a user with the manager-script role for Jenkins to deploy applications:
    - <user username="deployer" password="your_password" roles="manager-script"/>
    - Restart Tomcat to apply the changes.
4. **Set Up a Jenkins Job**:
    - Create a new Jenkins job (e.g., a Freestyle project).
    - In the "Source Code Management" section, configure your repository (e.g., GitHub).
    - Add a build step to compile your application (e.g., using Maven or Gradle).

5. **Add a Post-Build Action**:
    - In the Jenkins job configuration, scroll to the "Post-build Actions" section.
    - Select "Deploy war/ear to a container."
    - Provide the following details:
        - **WAR/EAR file**: Path to the WAR file generated by your build (e.g., target/*.war).
        - **Containers**: Choose "Tomcat 9”
        - **Manager URL**: URL of your Tomcat Manager (e.g., http://<your-server-ip>:8080/manager/text).
        - **Credentials**: Add the username and password you configured in tomcat-users.xml.
6. **Test the Integration**:
    - Save the Jenkins job configuration and trigger a build.
    - If everything is set up correctly, Jenkins will deploy the WAR file to Tomcat automatically.



Step-by-step guide to implement a Jenkins pipeline integrating Maven, SonarQube, Nexus, and Tomcat for deploying applications:

1. Set Up Jenkins

- Install Jenkins on your server and ensure it's running.
- Install the following plugins:
  - **SonarQube Scanner**: For code analysis.
  - **Nexus Artifact Uploader**: For uploading artifacts to Nexus.
  - **Deploy to Container**: For deploying to Tomcat.
- Restart Jenkins after installing the plugins.

2. Configure SonarQube in Jenkins

- Go to **Manage Jenkins → Configure System → SonarQube Servers**.
- Click "Add SonarQube" and provide:
    - A name for the server (e.g., SonarQube).
    - The URL of your SonarQube server (e.g., http://<sonarqube-server-ip>:9000).
    - A token generated from SonarQube (navigate to SonarQube Dashboard → User Settings → Generate Token).

3. Configure Nexus Credentials

- Navigate to **Manage Jenkins → Credentials → System → Global Credentials**.
- Add Nexus credentials (username and password) with:
    - Scope: Global.
    - ID: Something meaningful like nexus-credentials.

4. Configure Tomcat Manager User

- Edit the conf/tomcat-users.xml file in your Tomcat installation directory.
- Add a user with the manager-script role:
- <user username="deployer" password="your_password" roles="manager-script"/>
- Save the file and restart Tomcat.

5. Prepare Maven Project:

- Ensure your Maven project has a pom.xml with:
    - Proper dependencies.
    - SonarQube plugin configuration.

- Distribution management section for Nexus.

6. Create a Jenkins Pipeline

- Go to **Jenkins Dashboard → New Item → Pipeline**.
- Provide a name and select "Pipeline" as the type.

8. Test Your Pipeline

- Trigger a build in Jenkins.
- Validate each stage:
    1. **Checkout Code**: Ensure the code repository is fetched.
    2. **Build Application**: Verify that the .war file is generated in the target/ directory.
    3. **SonarQube Analysis**: Check the SonarQube dashboard for analysis results.
    4. **Quality Gate**: Ensure the pipeline stops if quality gate fails.
    5. **Nexus Upload**: Confirm the artifact is uploaded to Nexus.
    6. **Tomcat Deployment**: Verify the application is deployed to Tomcat.

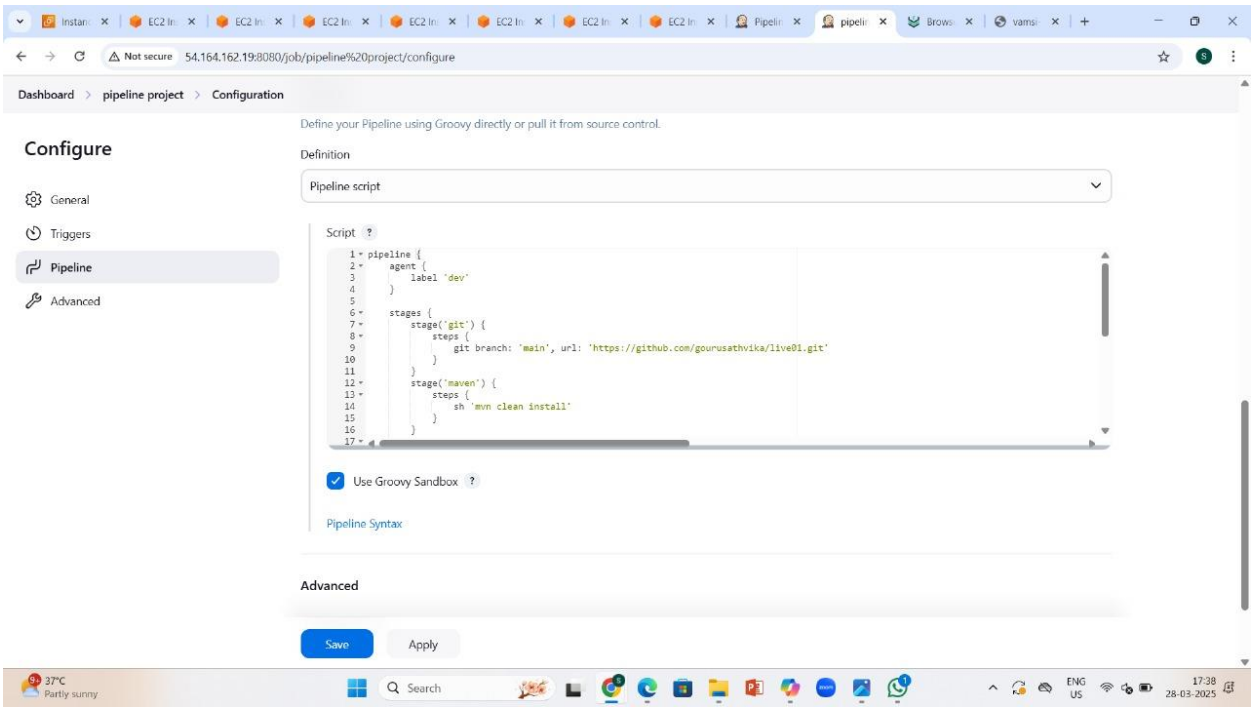
9. Troubleshooting:

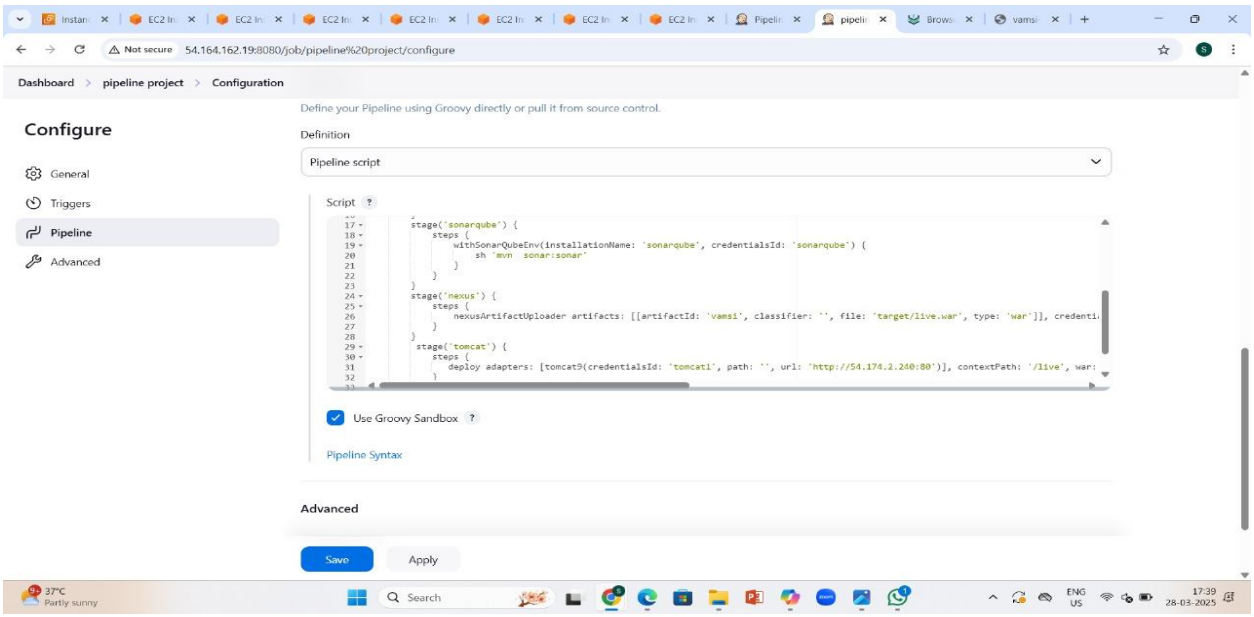
- **SonarQube Issues**: Ensure the server URL and token are correct. Check SonarQube logs if the pipeline fails.
- **Nexus Upload Issues**: Validate the repository URL and credentials.
- **Tomcat Deployment Issues**: Confirm the manager-script user is configured correctly in tomcat-users.xml

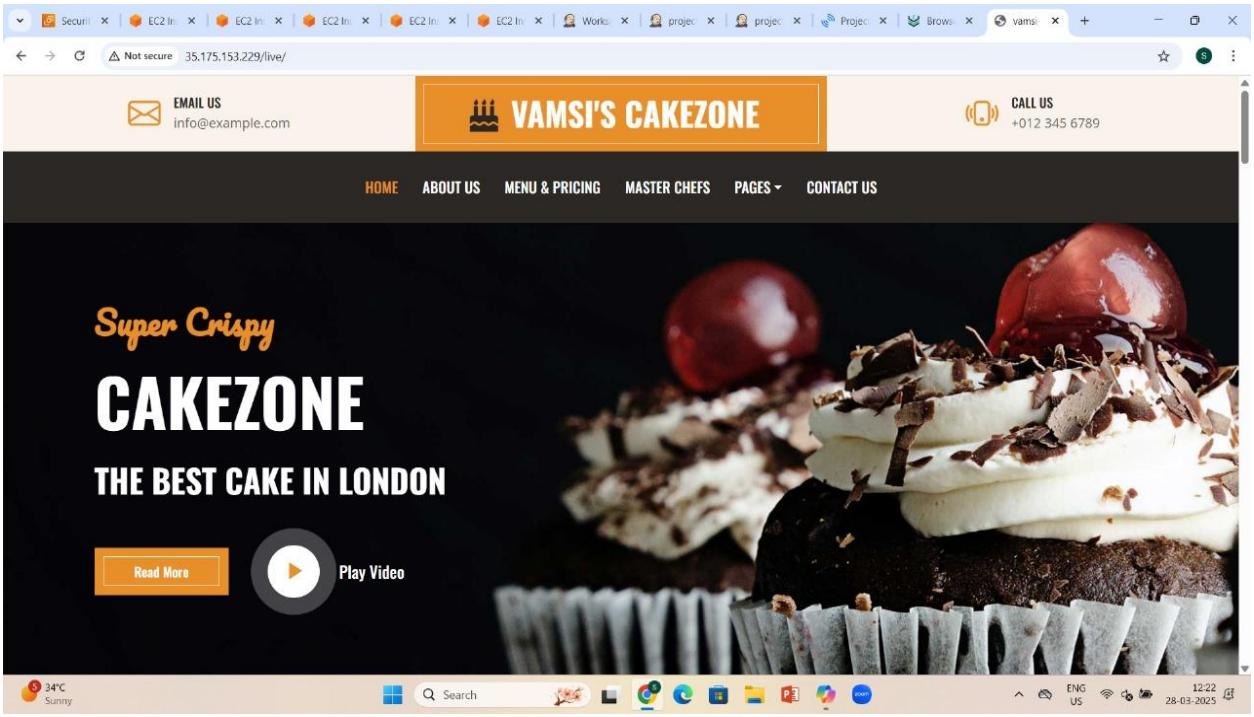
PIPELINE SCRIPT:

```
pipeline {
    agent {
        label 'dev'
    }
    stages {
        stage('git') {
            steps {
                git branch: 'main', url: 'https://github.com/Suvarna-Machani/live01.git'
            }
        }
        stage('maven') {
            steps {
                sh 'mvn clean install'
            }
        }
        stage('sonarqube') {
            steps {
                withSonarQubeEnv(installationName: 'sonarqube', credentialsId: 'sonarqube') {
                    sh 'mvn  sonar:sonar'
                }
            }
```

```
    stage('nexus') {
        steps {
            nexusArtifactUploader artifacts: [[artifactId: 'vamsi', classifier: '', file: 'target/live.war', type: 'war']], credentialsId: 'admin', groupId: 'vamsi.maven.com', nexusUrl: '54.147.17.206:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots', version: '0.0.1-SNAPSHOT'
        }
    }
    stage('tomcat') {
        steps {
            deploy adapters: [tomcat9(credentialsId: 'tomcat1', path: '', url: 'http://54.174.2.240:80')], contextPath: '/live', war: 'target/live.war'
        }
    }
}
```


Not secure 54.164.162.19:8080/job/pipeline%20project/configure
Dashboard > pipeline project > Configuration
Configure
General
Triggers
Pipeline
Advanced
Define your Pipeline using Groovy directly or pull it from source control.
Definition
Pipeline script
Script
pipeline {
agent {
label 'dev'
}
stages {
stage('git') {
steps {
git branch: 'main', url: 'https://github.com/gourusathvika/live01.git'
}
}
stage('maven') {
steps {
sh 'mvn clean install'
}
}
Use Groovy Sandbox
Pipeline Syntax
Advanced
Save
Apply
Search


Not secure 54.164.162.19:8080/job/pipeline%20project/configure
Dashboard > pipeline project > Configuration
Configure
General
Triggers
Pipeline
Advanced
Define your Pipeline using Groovy directly or pull it from source control.
Definition
Pipeline script
Script
stage('sonarqube') {
steps {
withSonarQubeEnv(installationName: 'sonarqube', credentialsId: 'sonarqube') {
sh 'mvn sonar:sonar'
}
}
}
stage('nexus') {
steps {
nexusArtifactUploader artifacts: [[artifactId: 'vamsi', classifier: '', file: 'target/live.war', type: 'war']], credenti
}
}
stage('tomcat') {
steps {
deploy adapters: [tomcat9(credentialsId: 'tomcat1', path: '', url: 'http://54.174.2.240:80')], contextPath: '/live', war:
}
Use Groovy Sandbox
Pipeline Syntax
Advanced
Save
Apply
Search



MYSQL Integration with Tomcat:

Step 1: Install Java:

- **Command:** sudo yum install java -y
- **Download and Set Up Apache Tomcat**

1. Download Tomcat:

- wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.102/bin/apache-tomcat-9.0.102.tar.gz

2. Extract Tomcat Files:

- tar -zxvf apache-tomcat-9.0.102.tar.gz

3. Start Tomcat:

- sh apache-tomcat-9.0.102/bin/startup.sh

Step 4: Configure Tomcat Users and Roles

1. Edit context.xml

- vi apache-tomcat-9.0.102/webapps/manager/META-INF/context.xml

2. Edit tomcat-users.xml

- vi apache-tomcat-9.0.102/conf/tomcat-users.xml

Step 5: Install MySQL Server

1. Download MySQL Repository using :

- Sudo wget https://dev.mysql.com/get/mysql80- community-release-el9-1.noarch.rpm
- This downloads the MySQL repository package needed to install MySQL.

2. Install Repository:

- Repository
  - rpm -ivh mysql80-community-release-el9-1.noarch.rpm
  - Installs the MySQL repository package using the RPM package manager, enabling access to

3. MySQL software:

- Import MySQL GPG Key using
  - sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023
  - This adds the GPG key to validate MySQL packages before installation.
- Install MySQL Client and Server

- dnf install mysql-community-client -y
- dnf install mysql-community-server -y
- Installs the client and server components of MySQL

4. Start MySQL :

- systemctl start mysqld.service
  - Starts the MySQL database service.
- Get Temporary Password by using the command
  - grep 'temporary password' /var/log/mysqld.log
  - This helps in searching for and retrieves the temporary password for the MySQL root user from the MySQL logs.
- Login to MySQL
  - mysql -u root -p
  - Connects to the MySQL server using the root user and prompts for the password

5. Download and Configure MySQL Connector/J and create database:

- Download Connector/J
  - `wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j-9.2.0.tar.gz
  - Downloads the MySQL Connector/J package, required for Java applications to connect to MySQL.
- Extract Connector/J Files
  - tar -zxvf mysql-connector-j-9.2.0.tar.gz
  - Extracts the .tar.gz package and provides access to the connector .jar file.

- Copy the Connector File to Tomcat

6. Create a Database:

  - CREATE DATABASE my_database;
  - This initializes a database to store your table.
  - Switch to the newly created database.
  - USE my_database;

- Create a Table
  - CREATE TABLE users (

    id INT AUTO_INCREMENT PRIMARY KEY,

    name VARCHAR(255) NOT NULL,

    email VARCHAR(255) NOT NULL

    );

- By this when application is deployed it will take the users.

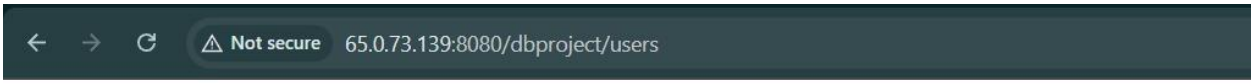
7. Restart Tomcat:

- Stop Tomcat
  - sh apache-tomcat-9.0.102/bin/shutdown.sh
  - Stops the Tomcat server gracefully.

- Start Tomcat
  - sh apache-tomcat-9.0.102/bin/startup.sh
  - Starts the Tomcat server again to apply the changes.

8. Deploy Application:

- Deploy the Java application that connects to the database



Users List

ID	Name	Email
1	suvarna	suvarna@gmail.com

Add New User

Name:
Email:
Add User