SonarQube Integration With Jenkins

Introduction:

Integrating SonarQube with Jenkins allows you to incorporate continuous code quality analysis into your build pipeline. SonarQube is a powerful open-source tool for continuous inspection of code quality, providing metrics on various factors such as code coverage, bugs, vulnerabilities, and code smells. Jenkins, a leading open-source automation server, facilitates building, deploying, and automating projects. By integrating SonarQube with Jenkins, you can ensure that code quality is assessed automatically during the build process, enabling early detection and resolution of issues.

Prerequisites:

- ✓ Jenkins Installed
- ✓ AWS Account
- ✓ SonarQube Installed
- ✓ SonarQube Scanner

What is SonarQube?

SonarQube is an open-source platform designed to continuously inspect and measure the quality of source code, providing detailed reports on various aspects of code health. It supports a wide range of programming languages and integrates seamlessly into CI/CD pipelines, making it a vital tool for maintaining high standards of code quality and improving the overall reliability of software projects.

How SonarQube Works ?

Source Code Analysis:
Developers commit code to a version control system. When a build is triggered (manually or via a CI/CD tool), the SonarQube scanner analyzes the source code against a set of predefined or custom rules.

Sending Results to SonarQube Server:
The results of the analysis are sent to the SonarQube server, which processes and stores the data.

Dashboard Visualization:
The processed data is visualized on the SonarQube dashboard, providing insights into various code quality metrics and potential issues.

Feedback and Improvement:
Developers review the reports, identify issues, and make necessary improvements. The cycle continues, fostering continuous improvement in code quality.

Key Features of SonarQube

Code Analysis:
SonarQube performs static code analysis to detect bugs, vulnerabilities, and code smells in your source code. It uses a comprehensive set of rules to identify potential issues.

Multi-language Support:
SonarQube supports a wide array of programming languages, including but not limited to Java, C#, JavaScript, TypeScript, Python, C++, and more. This makes it versatile for projects that involve multiple languages.

Quality Gates:
Quality gates in SonarQube define a set of conditions that code must meet before it can be considered acceptable. This ensures that code changes do not degrade the overall quality.

Dashboards and Reports:
The platform provides detailed dashboards and reports that visualize the quality metrics of the code. This includes metrics like code coverage, duplication, complexity, and maintainability.

Security Vulnerability Detection:
SonarQube identifies security vulnerabilities and suggests ways to fix them, helping to ensure that the software is secure from potential attacks.

Code Coverage:
Integration with various testing tools allows SonarQube to report on code coverage, highlighting untested parts of the codebase.

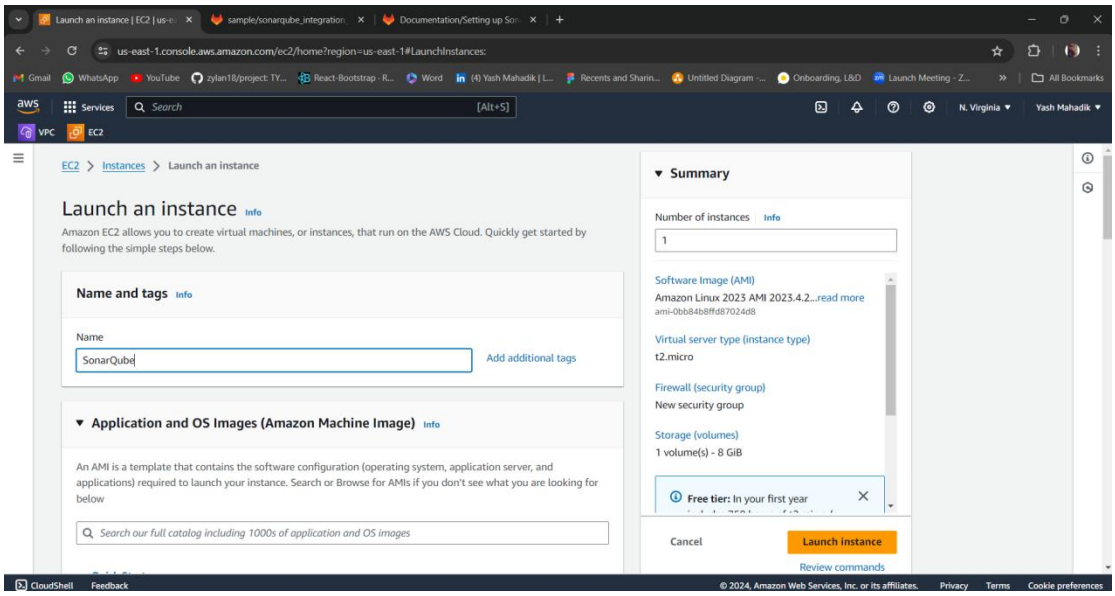
Customizable Rules and Profiles:
Users can create custom rules and quality profiles tailored to their specific needs, providing flexibility in how code quality is assessed.

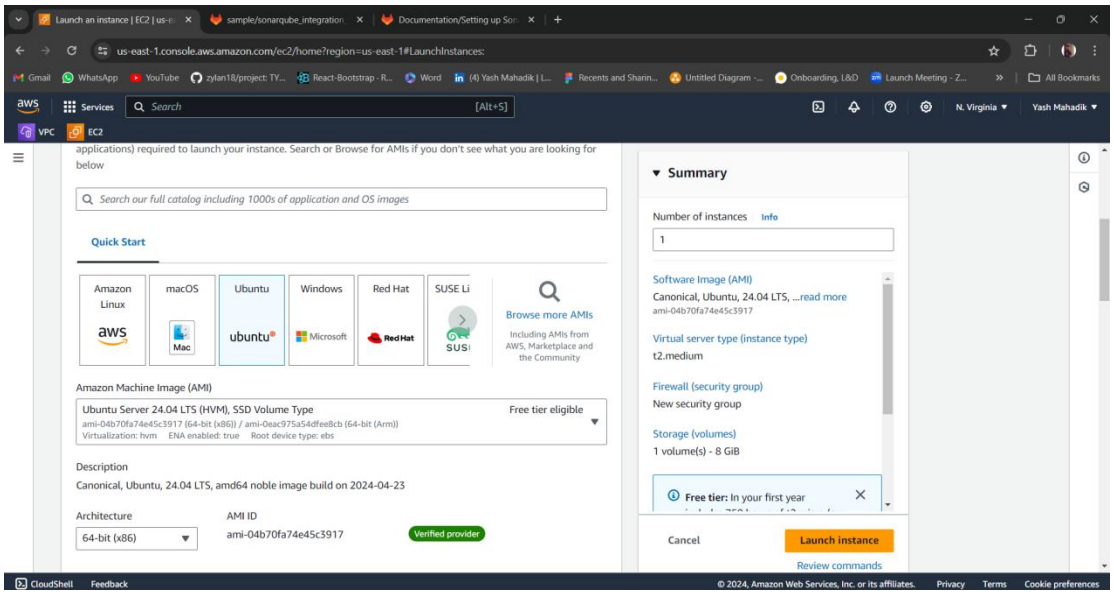
Integration with Development Tools:
SonarQube integrates with popular build tools (e.g., Maven, Gradle), version control systems (e.g., Git), and CI/CD pipelines (e.g., Jenkins, GitLab CI).

Issue Tracking:
Detected issues are categorized and prioritized, allowing developers to address the most critical problems first. SonarQube also tracks historical data, helping teams understand trends in code quality over time.

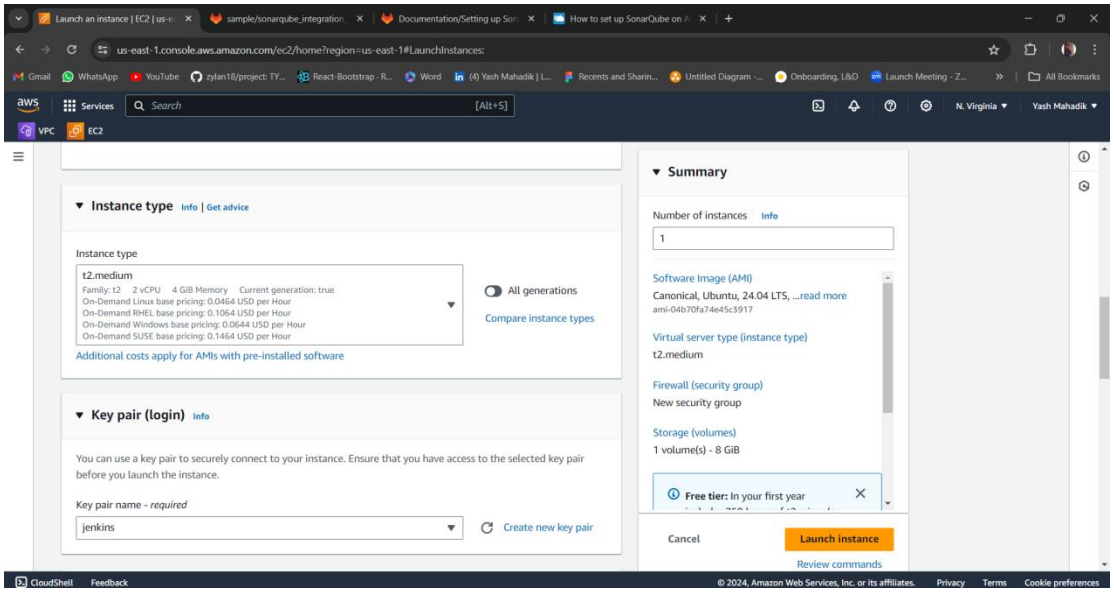
A step-by-step guide to Setting up Sonarqube

1) Login to your AWS account, and **Create Ec2 instance** , name it as **Sonarqube** and use Ubuntu base AMI.

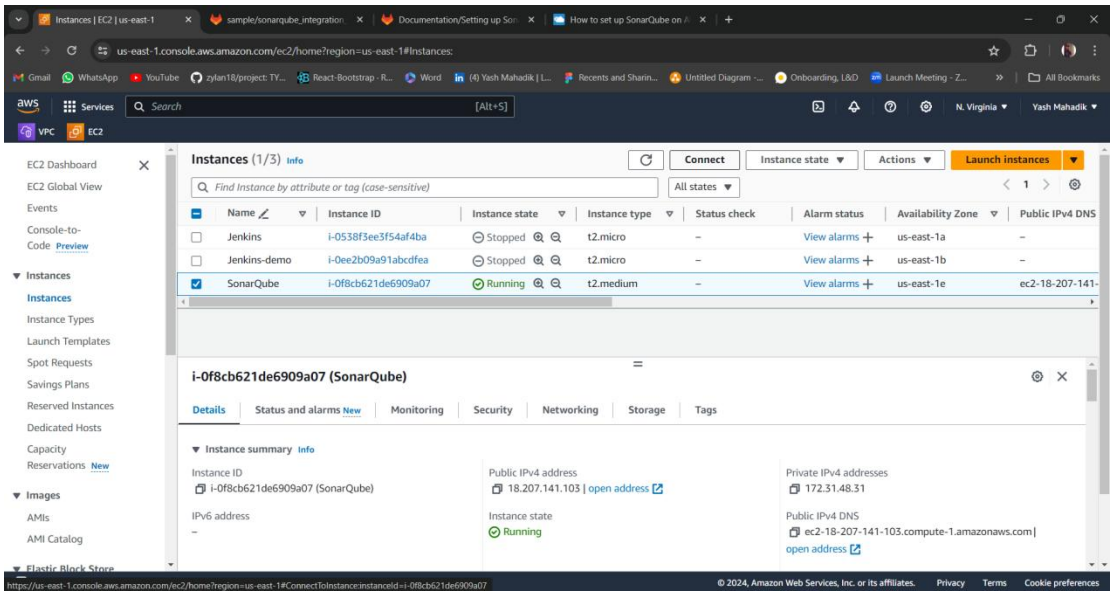




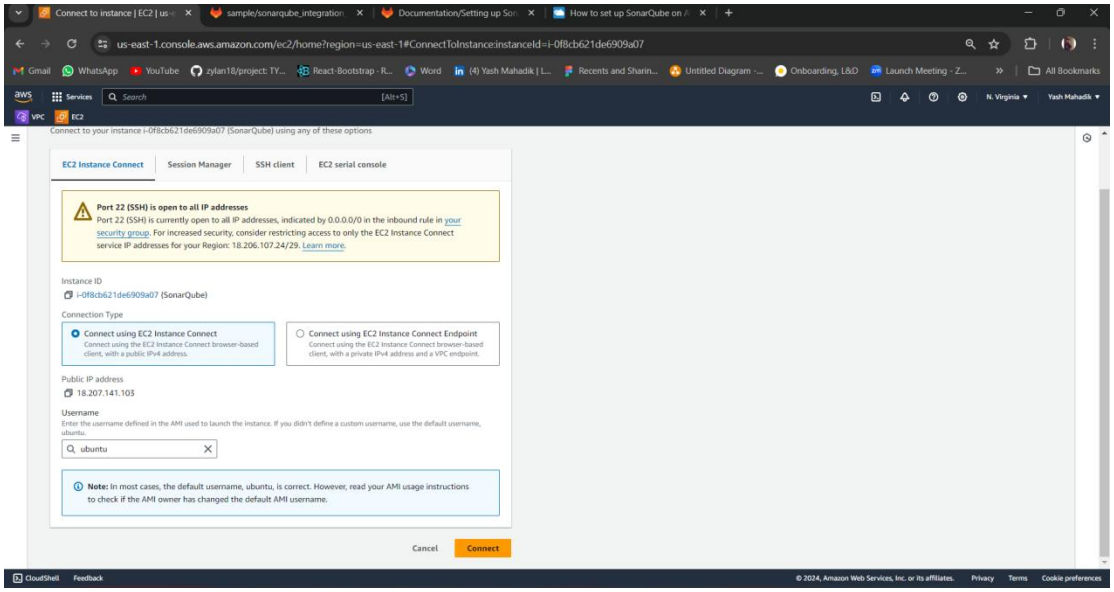
2) To run Sonarqube and Jenkins we need at least 2CPU and 2GB Memory. So we used instance type as t2.medium . Add key pair and Launch the instance.



3) The instance is ready and Running.

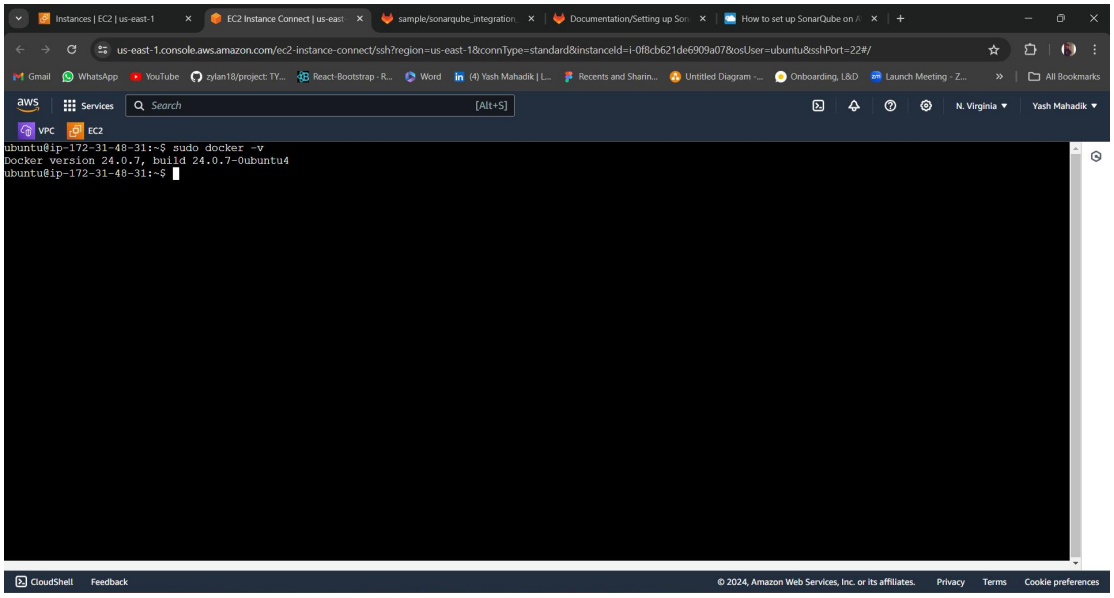


4) Connect instance



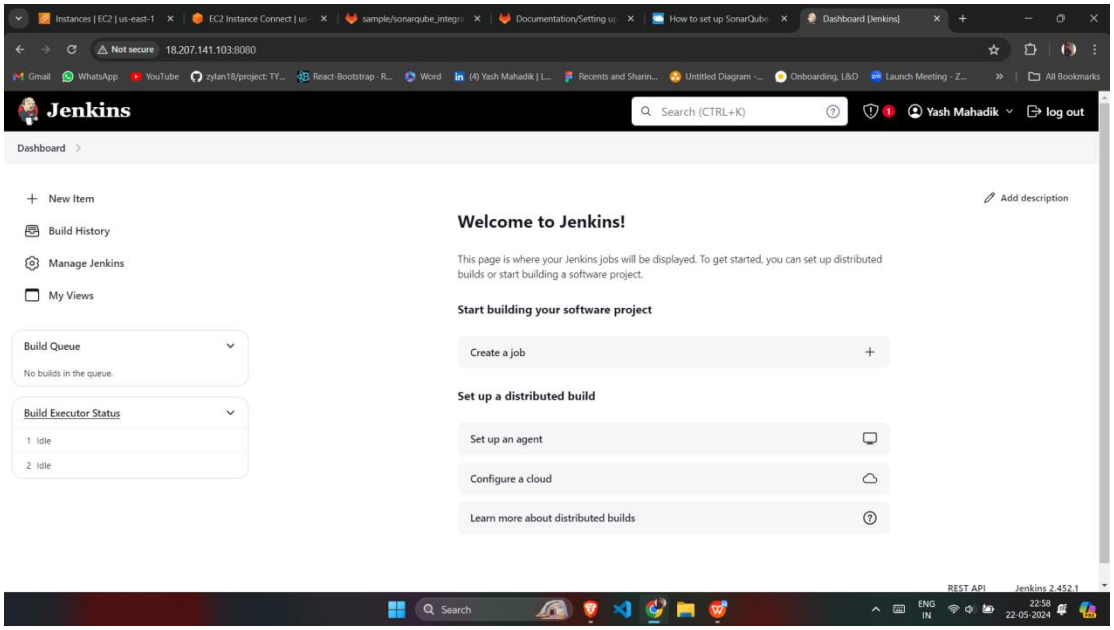
5) You need to Install Docker for pulling Sonarqube image.

sudo apt update -y
sudo apt install docker.io -y
sudo docker -v



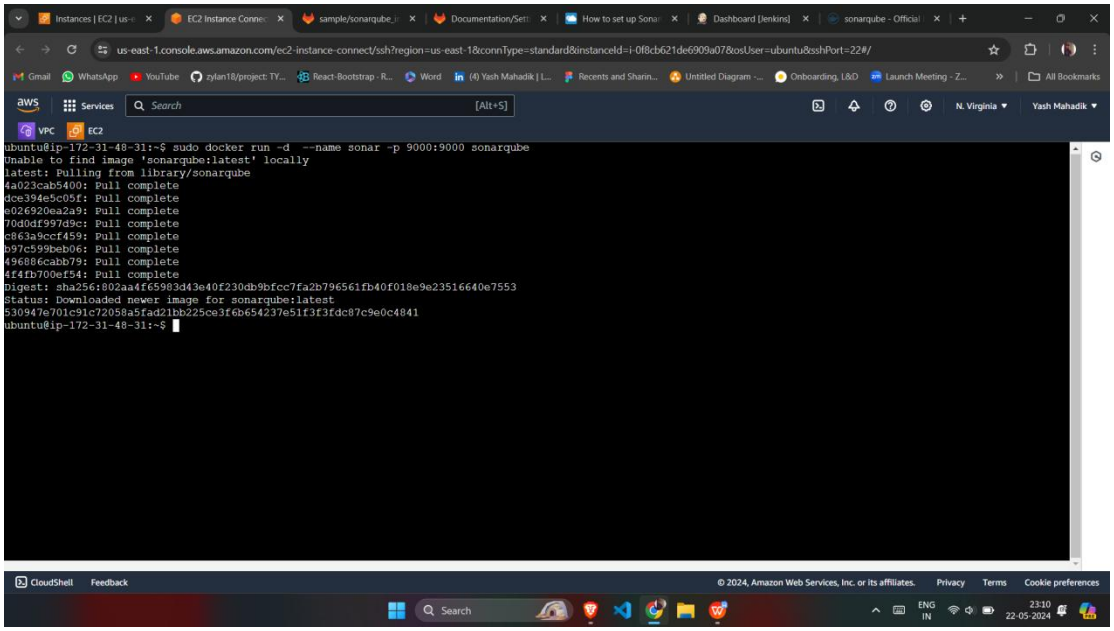
6)Also Install Jenkins on the same Instance.

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update
 sudo apt-get install fontconfig openjdk-17-jre
 sudo apt-get install jenkins
```

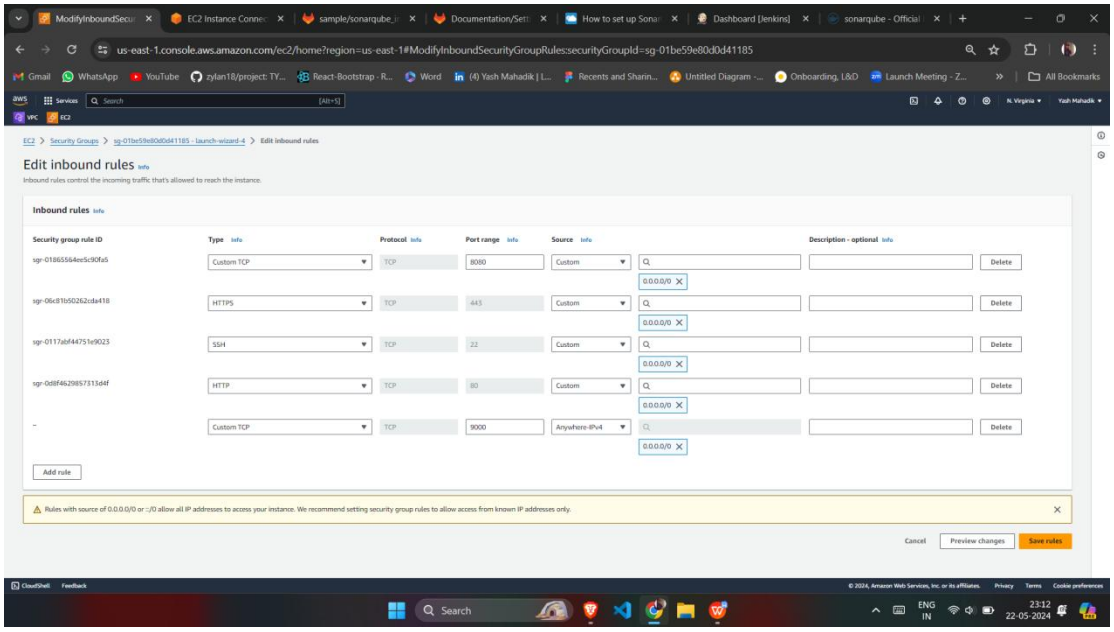


7)Install Sonarqube

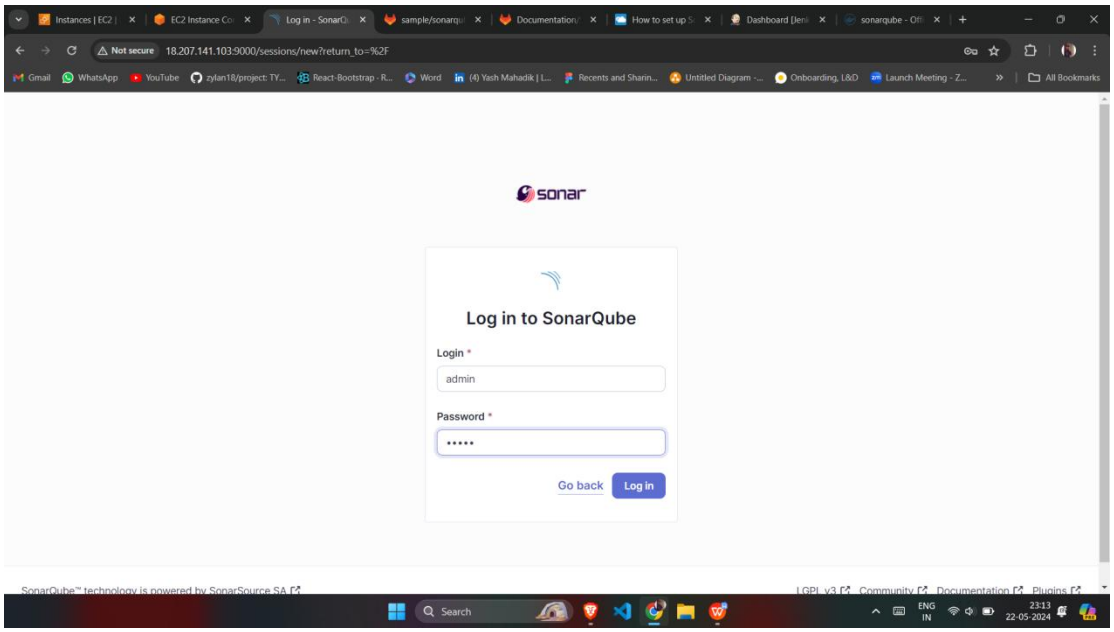
sudo docker run -d --name sonar -p 9000:9000 sonarqube



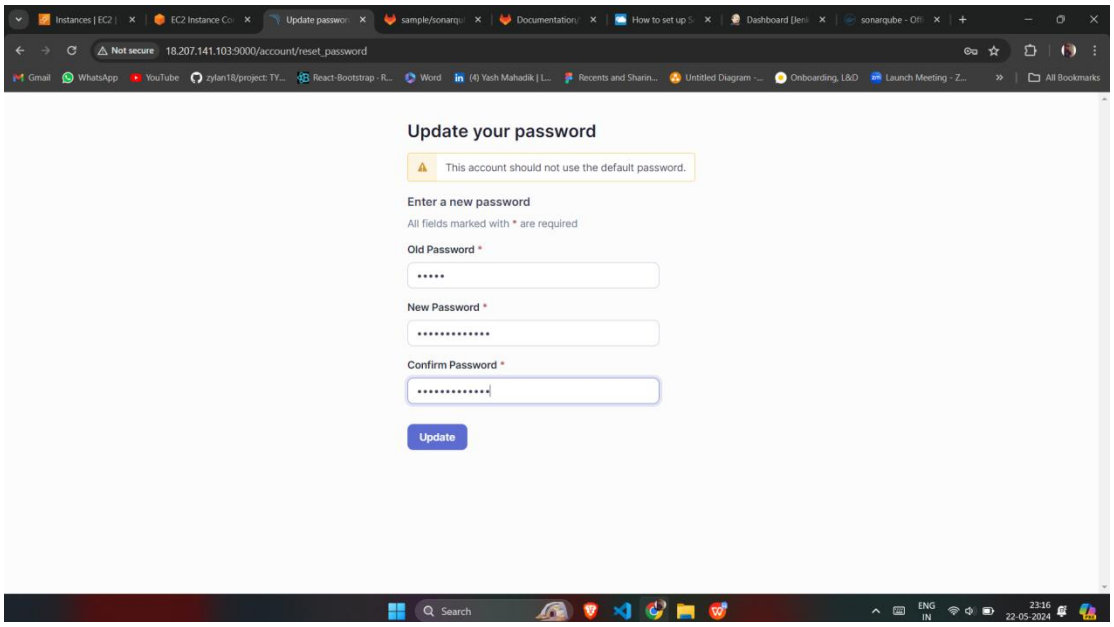
8)As we know Jenkins runs on port 8080 and Sonarqube run on port 9000 , we do need to add inbound traffics.



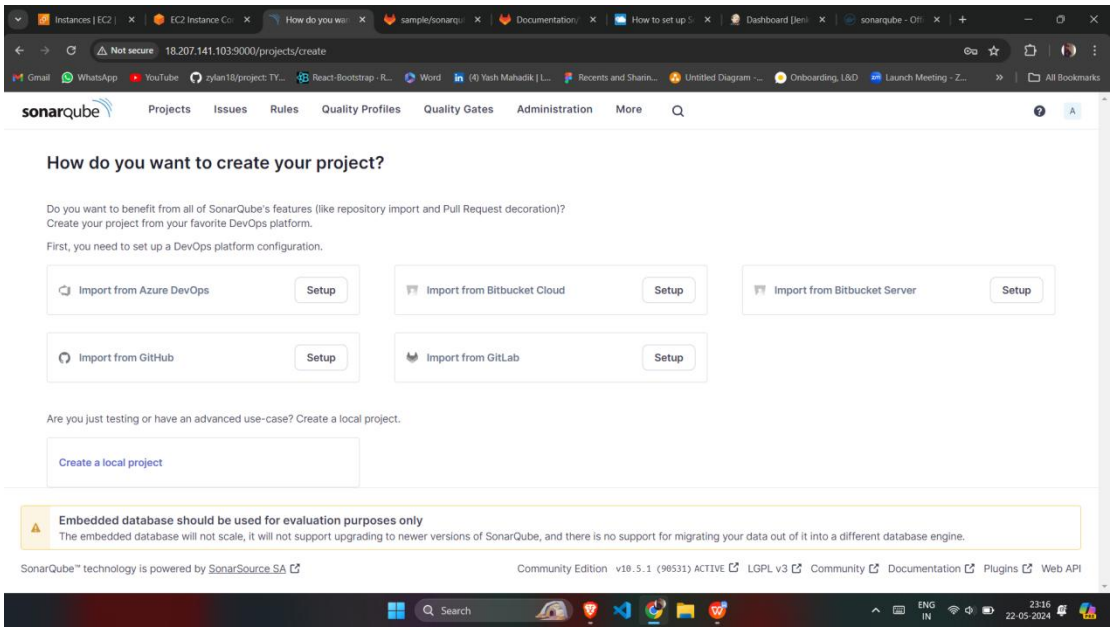
9)Sonarqube will run on port 9000 , by default **login = admin** and **password = admin**



10)You can Setup your own Password .

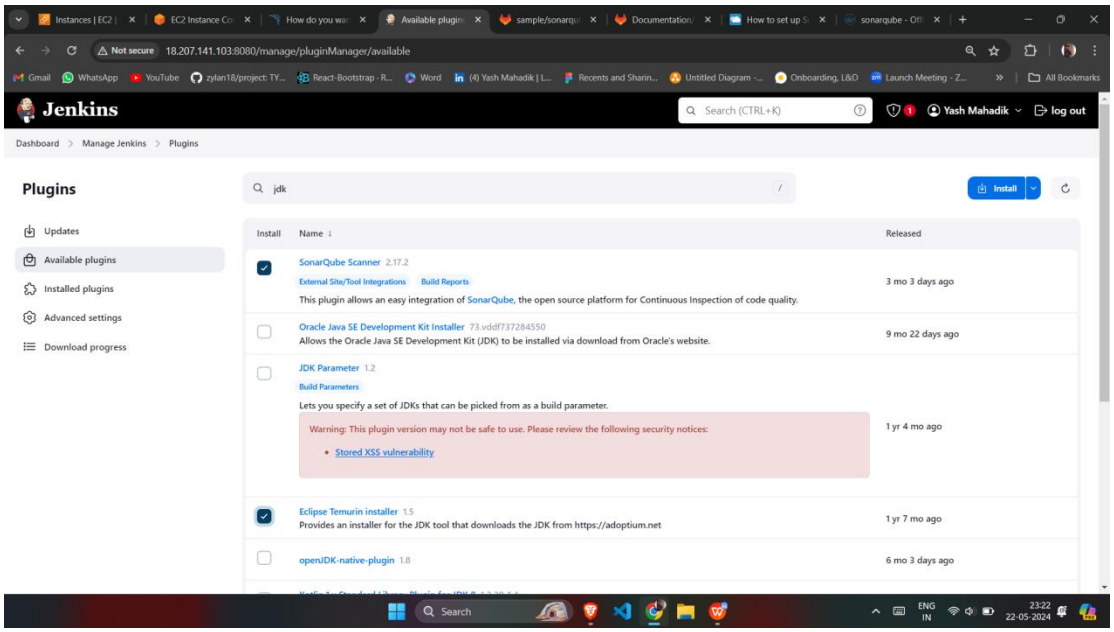


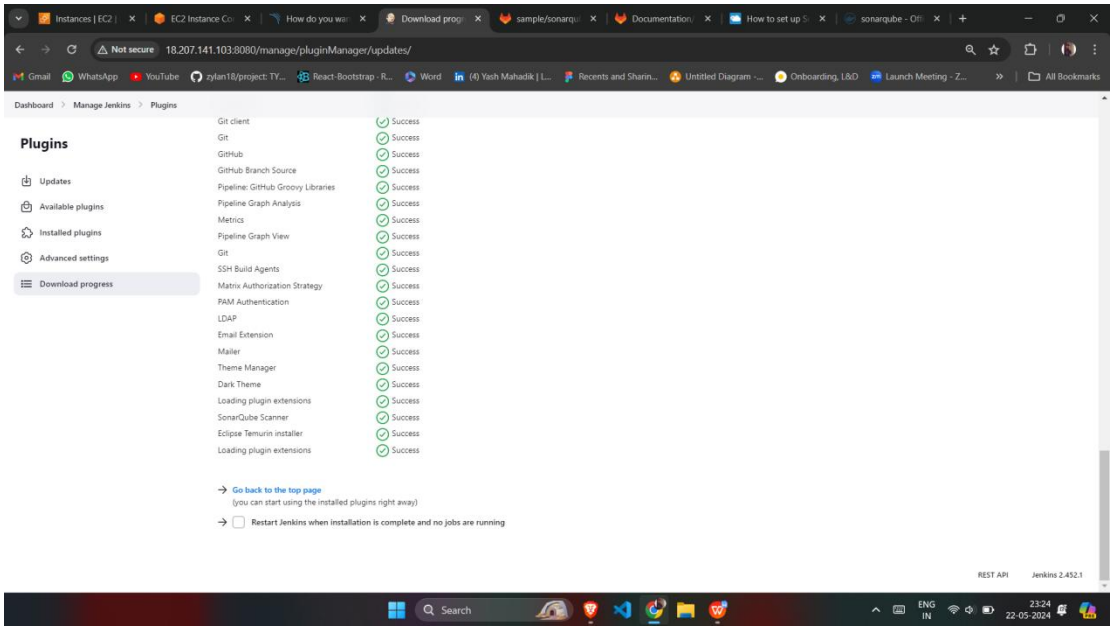
11)This is how Sonarqube Dashboard looks like.



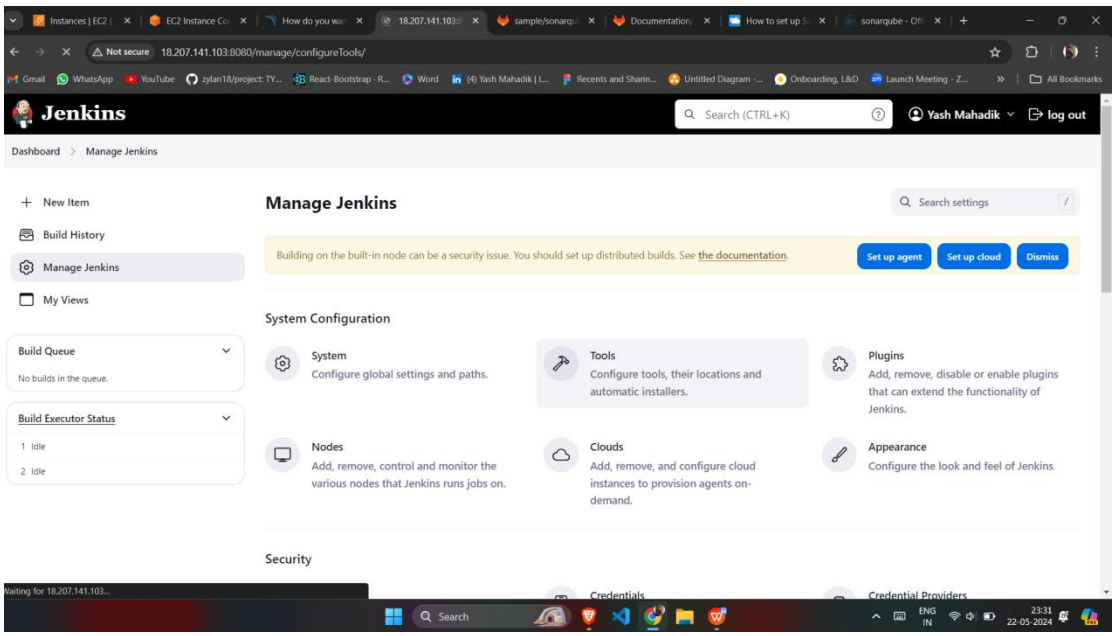
#Configure Sonarqube with Jenkins

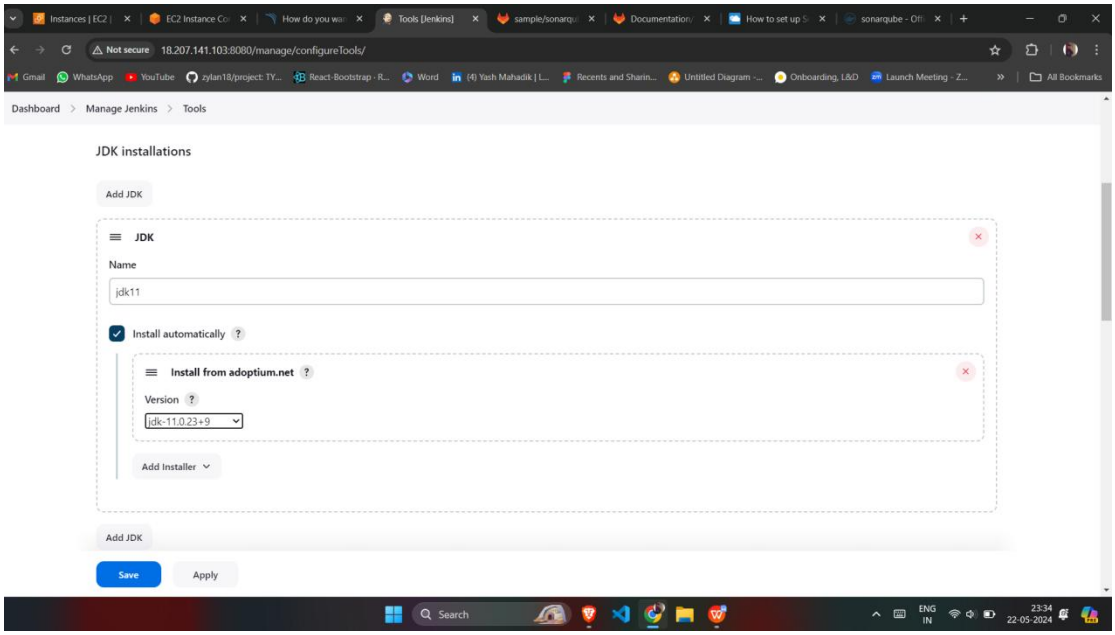
1) Install SonarQube Scanner Plugins : In Jenkins Navigate to Manages Jenkins > Plugins and Install **Sonarqube Scanner plugin** along with **Eclipse Temurin** for **jdk.**



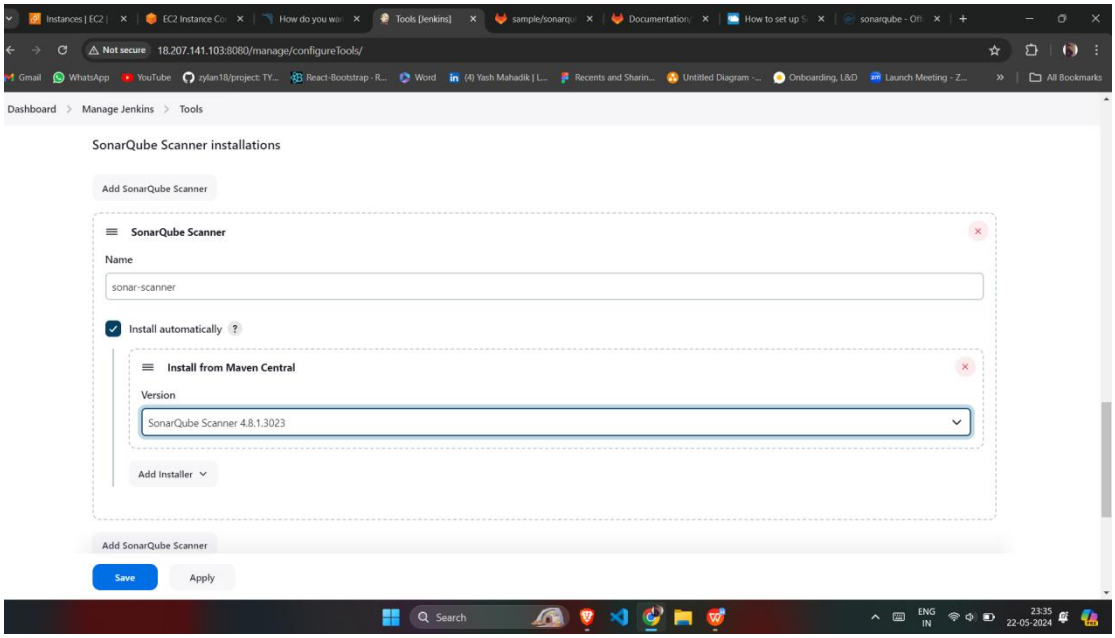


2) In Jenkins, navigate to **Manage Jenkins > Tools**, add jdk11 and version.

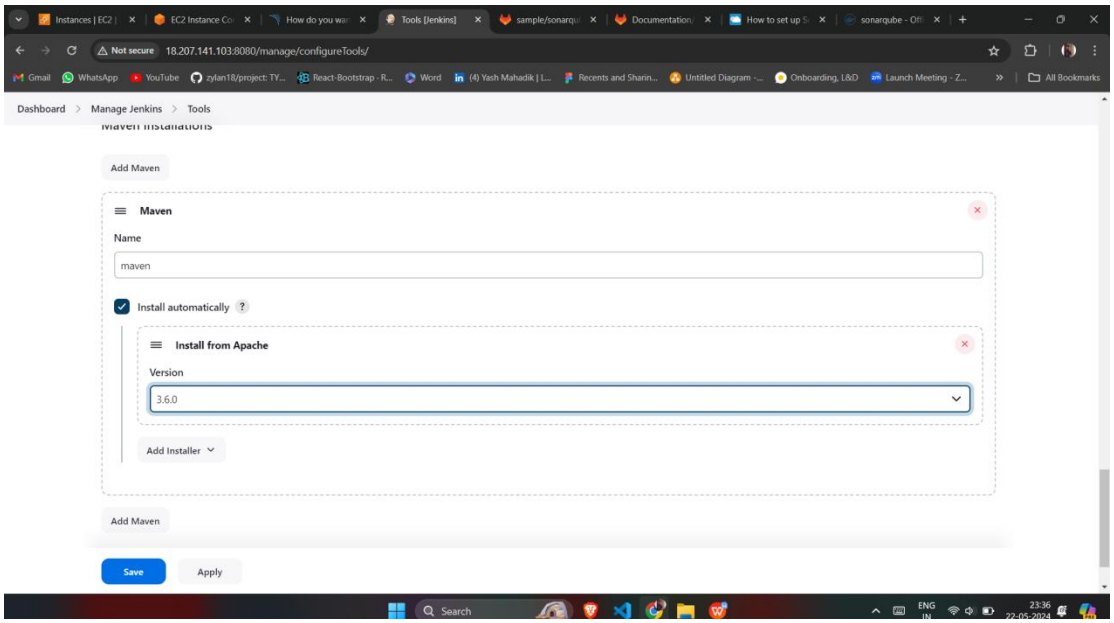




3) In Jenkins, navigate to **Manage Jenkins > Tools**, add Sonarqube Scanner with version.

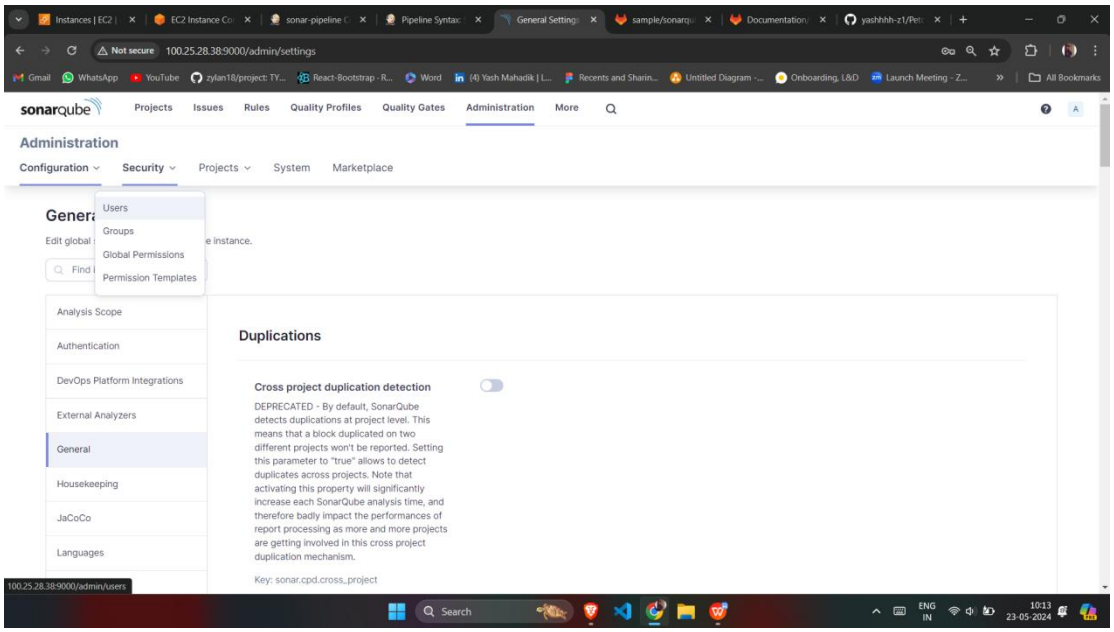


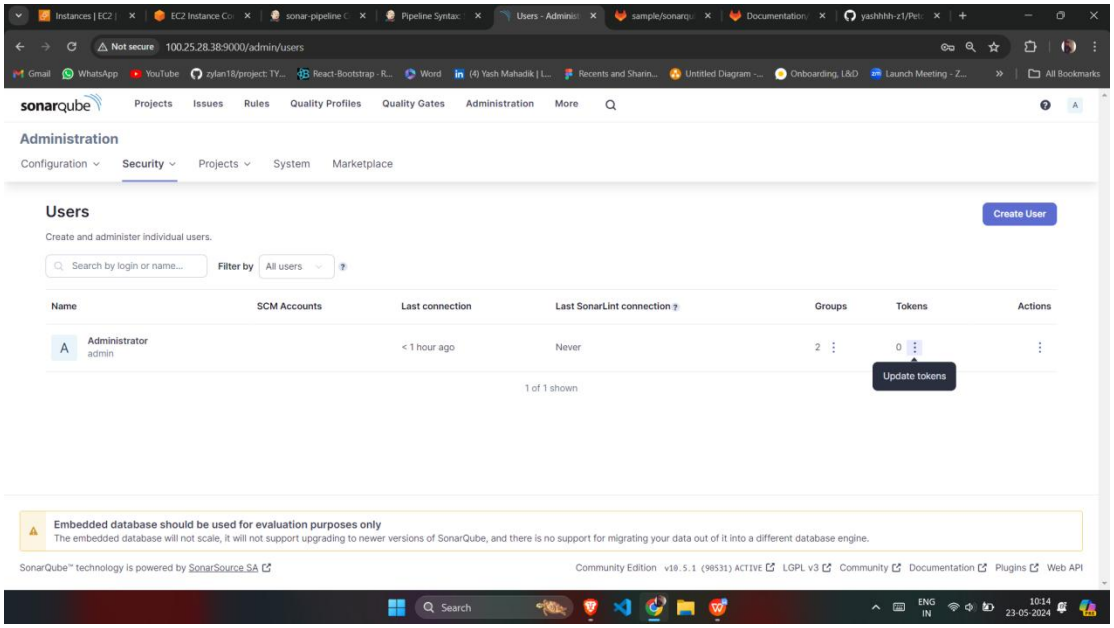
4) In Jenkins, navigate to **Manage Jenkins > Tools**, add maven and version.

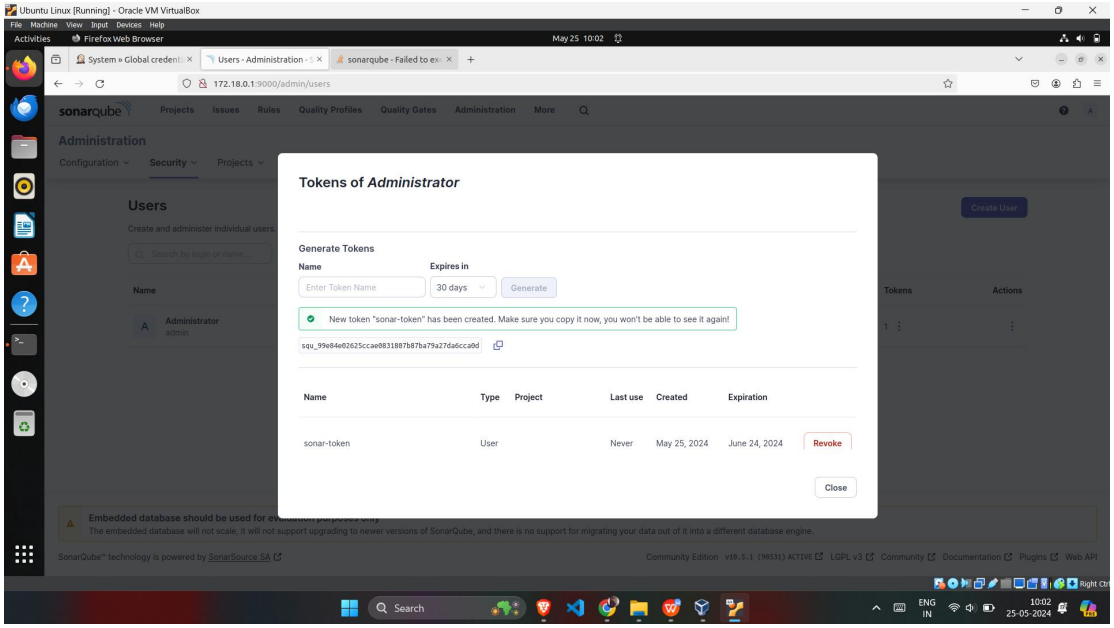


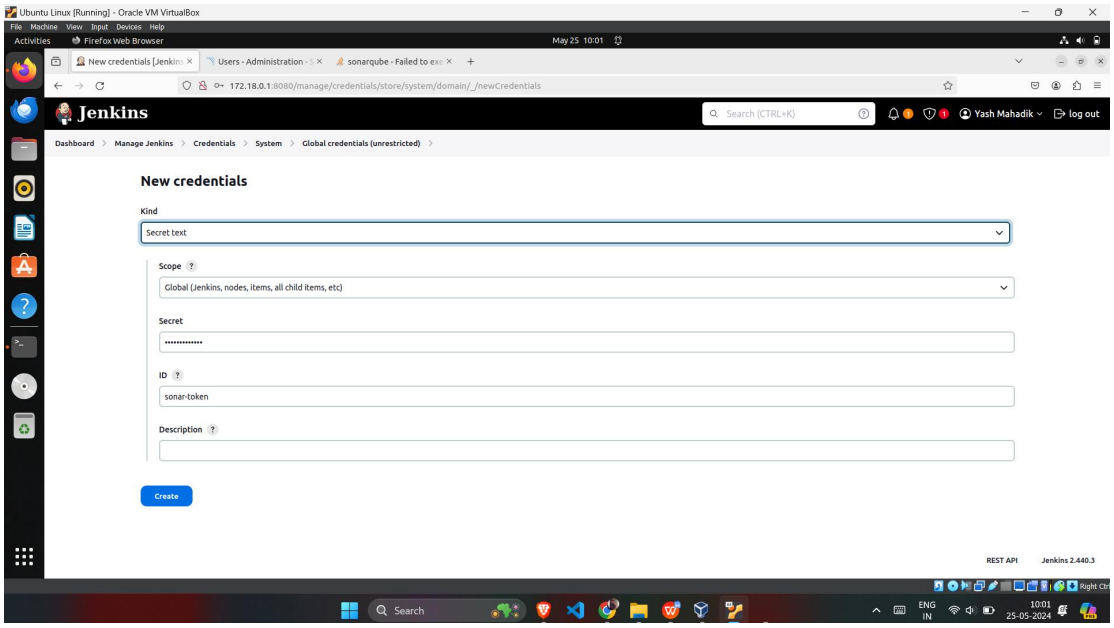
Generate SonarQube Token

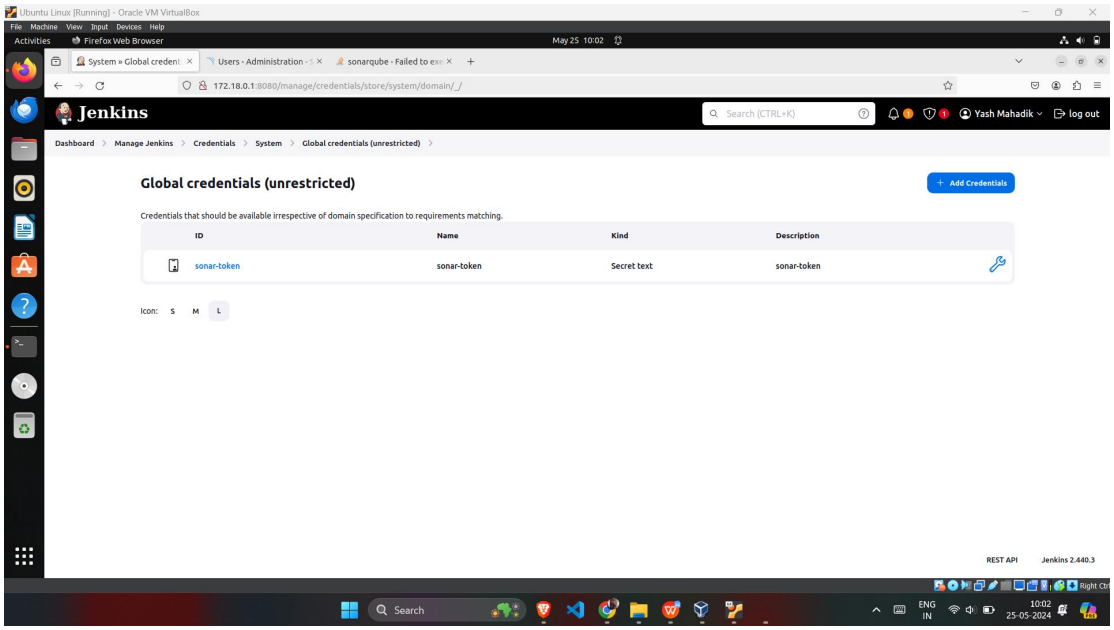
In SonarQube Navigate to **Administration > Security > Users**, create a new Token and use that token in Jenkins configuration.



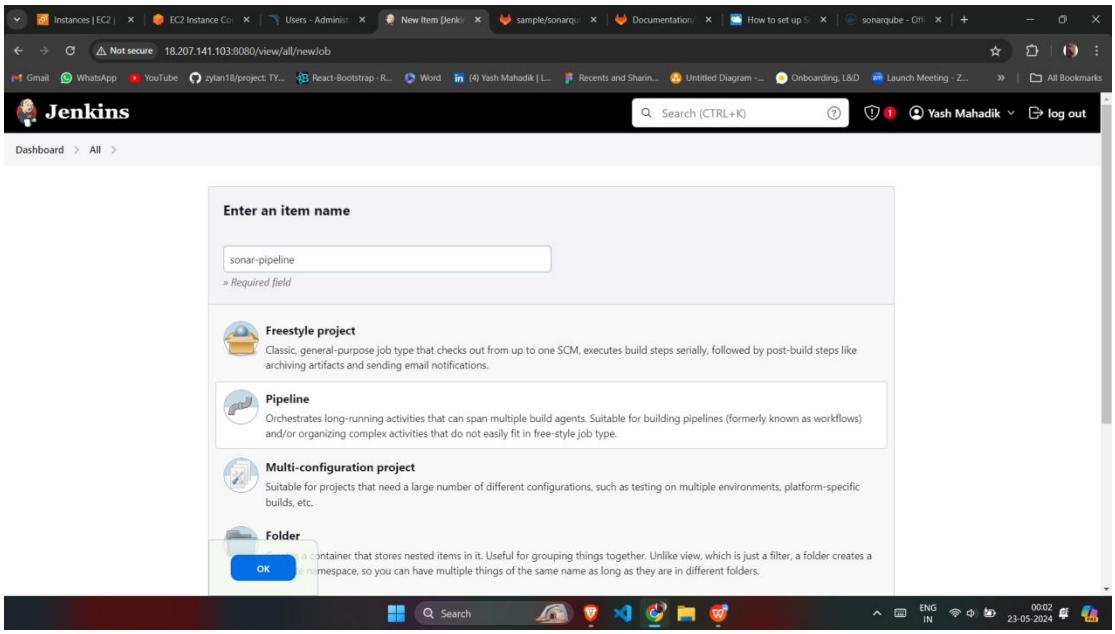

Not secure 100.25.28.38:9000/admin/users
sonarqube
Projects Issues Rules Quality Profiles Quality Gates Administration More
Administration
Configuration Security Projects System Marketplace
Users
Create and administer individual users.
Create User
Search by login or name...
Filter by All users
Name SCM Accounts Last connection Last SonarLint connection Groups Tokens Actions
Administrator
admin
< 1 hour ago
Never
2
0
Update tokens
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 v10.5.1 (90531) ACTIVE LGPL v3 Community Documentation Plugins Web API

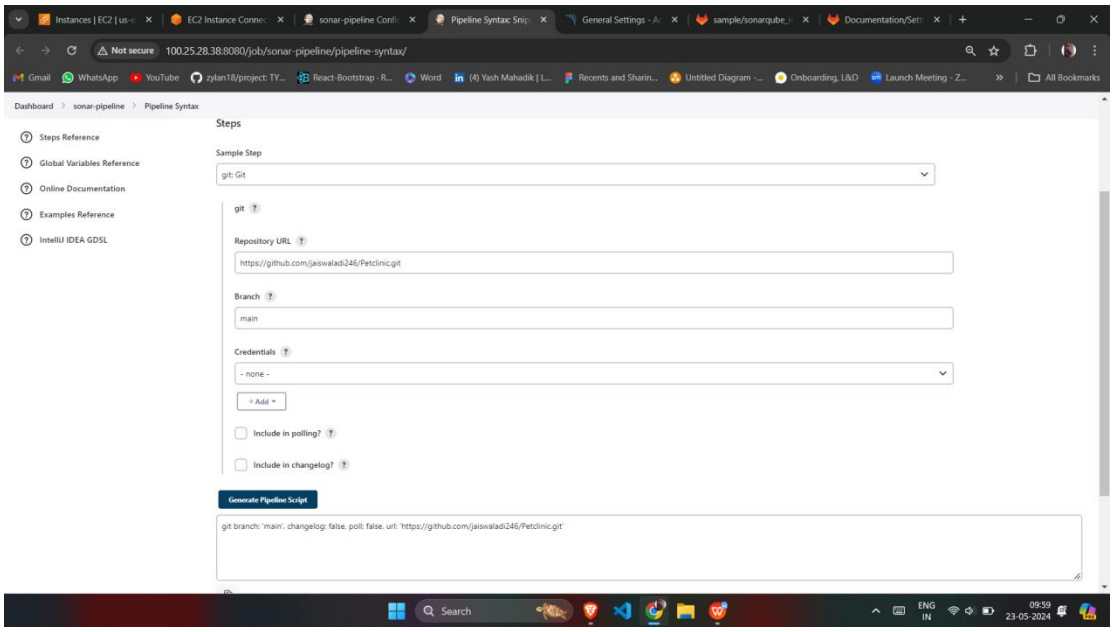

Ubuntu Linux [Running] - Oracle VM VirtualBox
Firefox Web Browser
172.18.0.1:9000/admin/users
Tokens of Administrator
Generate Tokens
Name
Expires in
Enter Token Name
30 days
Generate
New token "sonar-token" has been created. Make sure you copy it now, you won't be able to see it again!
Name Type Project Last use Created Expiration
sonar-token User Never May 25, 2024 June 24, 2024 Revoke
Close


Jenkins
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
New credentials
Kind
Secret text
Scope
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID
sonar-token
Description
Create


Jenkins
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
sonar-token
sonar-token
Secret text
sonar-token

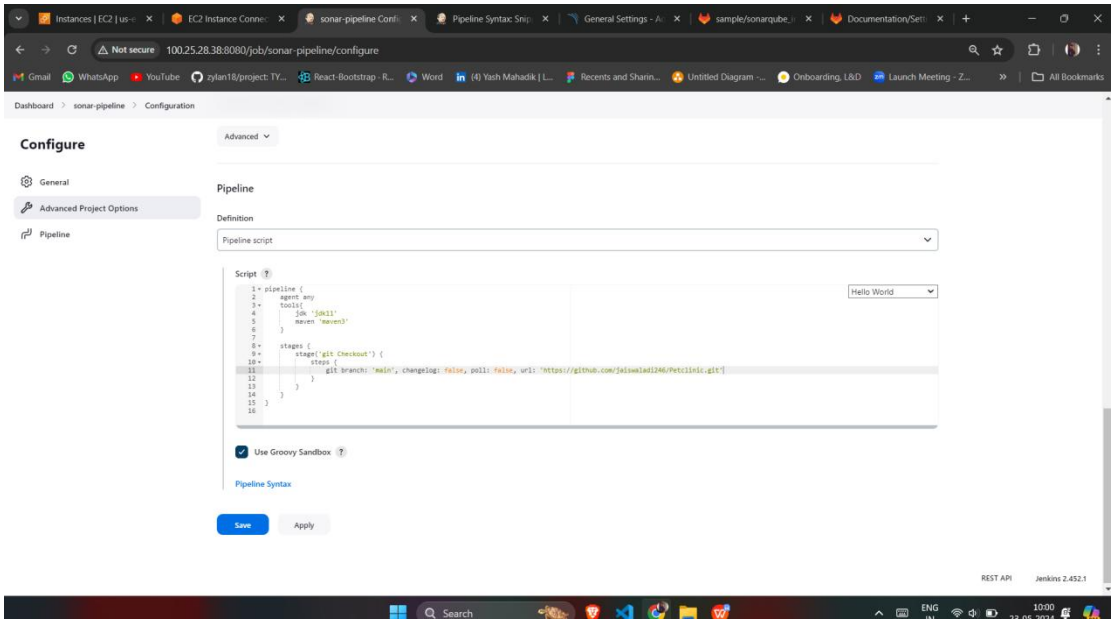
Create Pipeline and name it as sonar-analyis





Git checkout

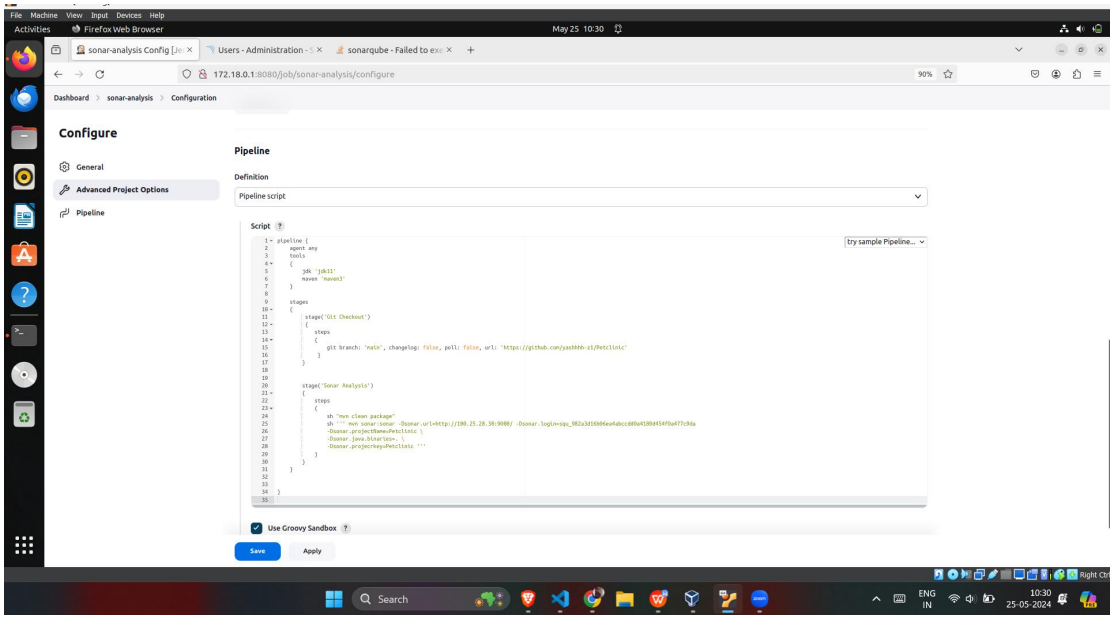
```
pipeline {
   agent any
   tools{
      jdk 'jdk11'
      maven 'maven3'
   }

   stages {
      stage('git Checkout') {
         steps {
            git branch: 'main', changelog: false, poll: false, url:
'https://github.com/jaiswaladi246/Petclinic.git'
         }
      }
   }
}
```



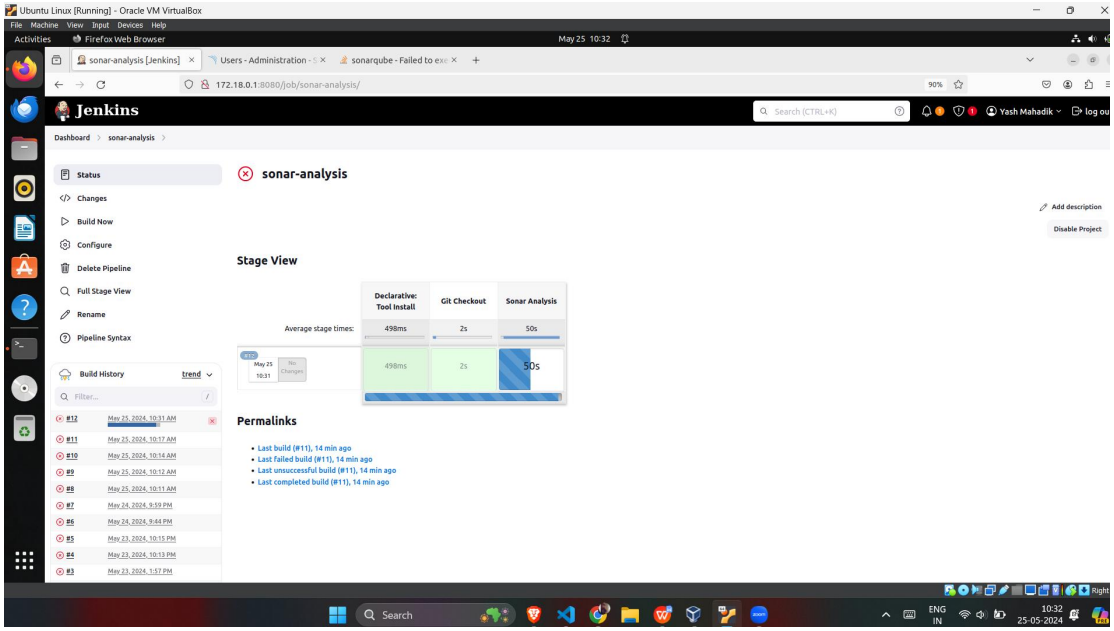
Add script sonarqube analysis

```
pipeline {
   agent any
   tools
   {
      jdk 'jdk11'
      maven 'maven3'
   }
```

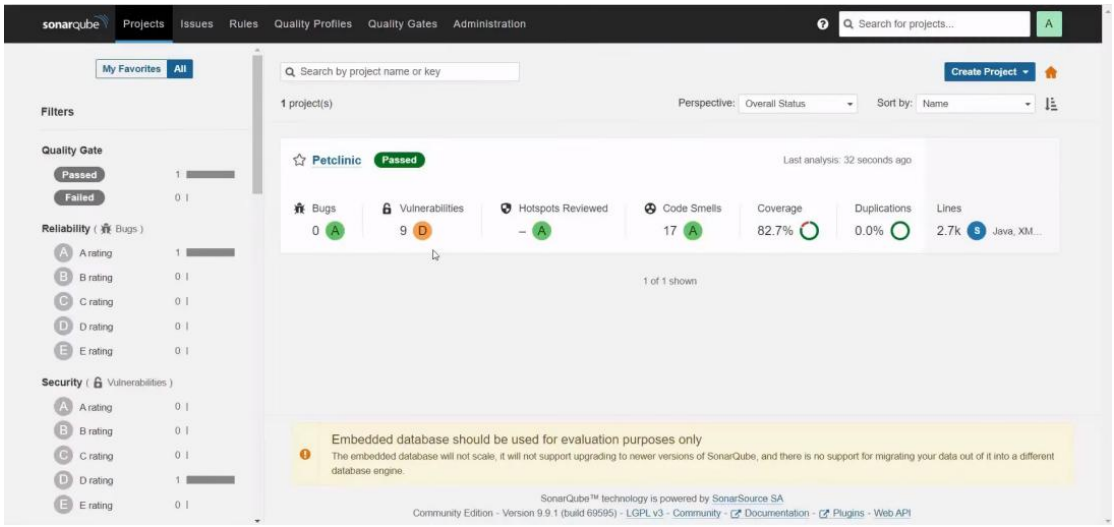
```
  stages
  {
      stage('Git Checkout')
      {
        steps
        {
           git branch: 'main', changelog: false, poll: false, url:
'https://github.com/yashhhh-z1/Petclinic'
         }
      }


      stage('Sonar Analysis')
      {
        steps
        {
           sh "mvn clean package"
           sh ''' mvn sonar:sonar -Dsonar.url=http://100.25.28.38:9000/ -
Dsonar.login=squ_982a3d16b06ea4abccdd0a4180d454f0a477c9da
           -Dsonar.projectName=Petclinic \
           -Dsonar.java.binaries=. \
           -Dsonar.projecrkey=Petclinic '''
        }
      }
  }


}
```



Run Analysis and View Results: Run your Jenkins job. After the build completes, SonarQube will analyze the code.



You can view the results and quality gate status on the SonarQube dashboard



Conclusion:

By integrating Jenkins with SonarQube, you can continuously monitor code quality and enforce best practices, leading to more maintainable and reliable software. This setup allows developers to catch and address issues early in the development cycle, ultimately improving the overall quality of the codebase.