Step-by-Step Guide to Setting Up Jenkins and SonarQube Integration with Slack Notifications

Overview

This document describes the process to set up a Jenkins CI/CD pipeline integrated with SonarQube for code analysis and Slack for notifications. It also includes hosting a Java application in a Docker container.

Prerequisites

1. **Two EC2 instances:**
   - Instance 1: For Jenkins and hosting the Java application.
   - Instance 2: For SonarQube.

Steps

Step 1: Set Up Jenkins Instance

1. **Launch an EC2 instance (Instance 1) and install the following tools:**
   - Jenkins
   - Git
   - Maven
   - Docker

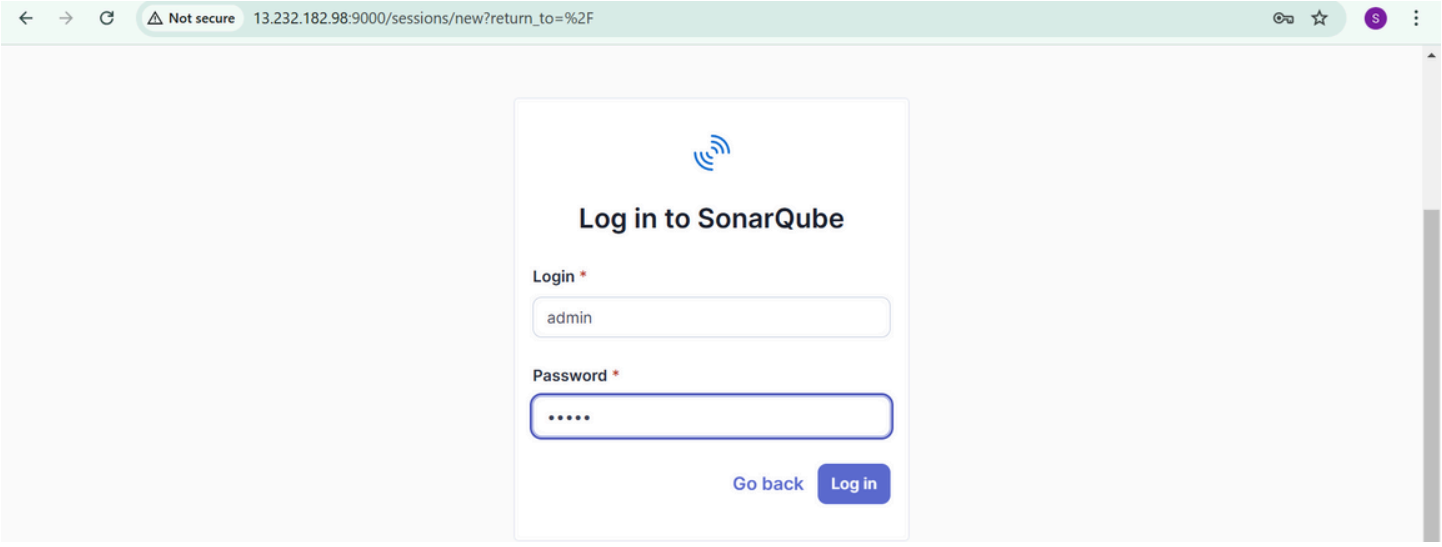
Access SonarQube via the browser at http://<Instance-2-Public-IP>:8080 and log in.

Step 2: Set Up SonarQube Instance

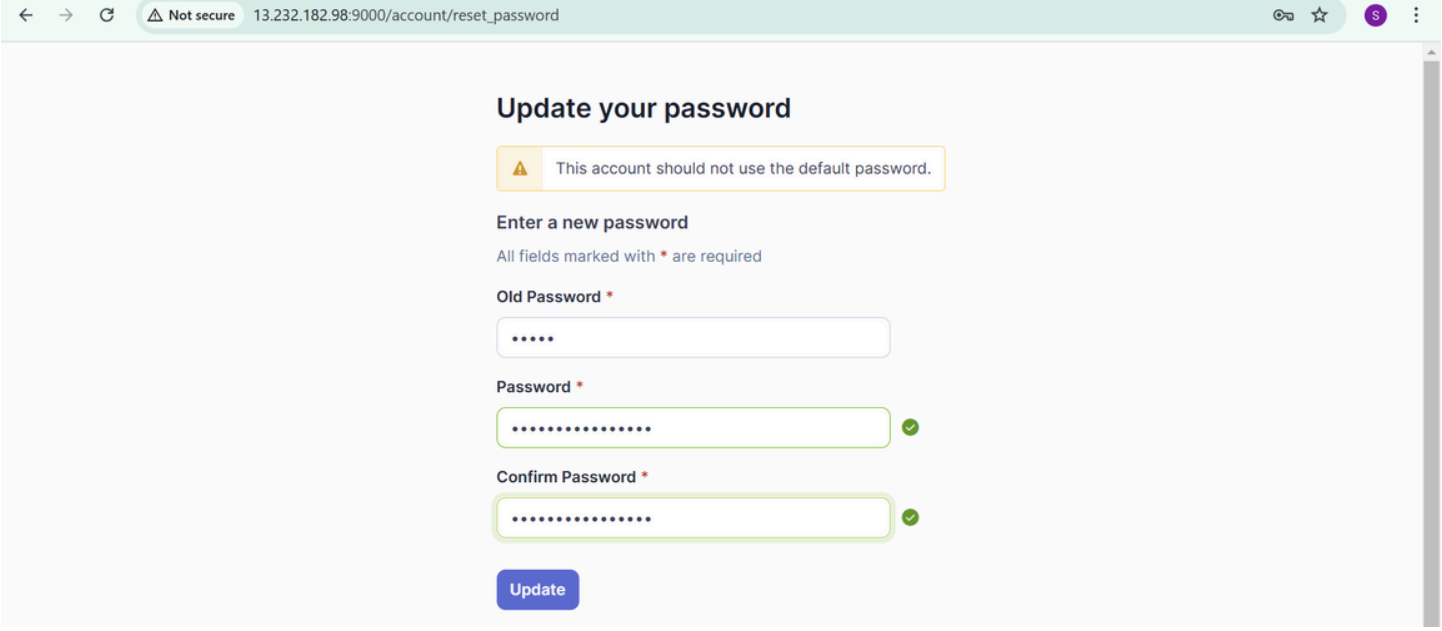
1. Launch a second EC2 instance (Instance 2) and install Docker.
2. Pull the SonarQube Docker image and run a container:

```
docker pull sonarqube
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

Access SonarQube via the browser at http://<Instance-2-Public-IP>:9000 and log in.



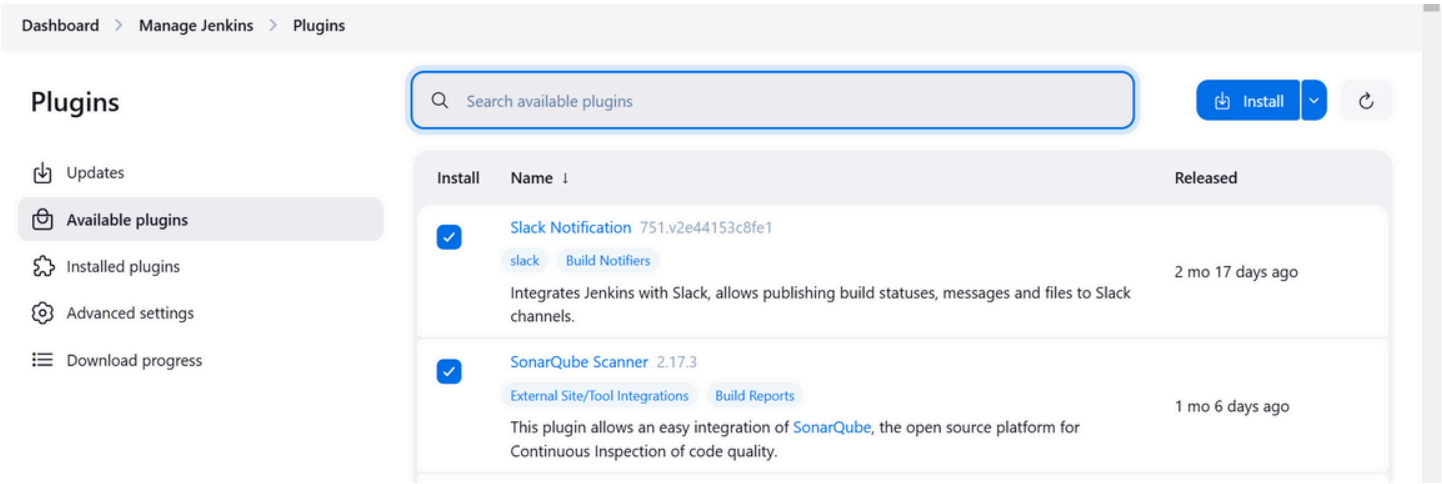
first login is admin and password is admin



Step 3: Configure SonarQube in Jenkins

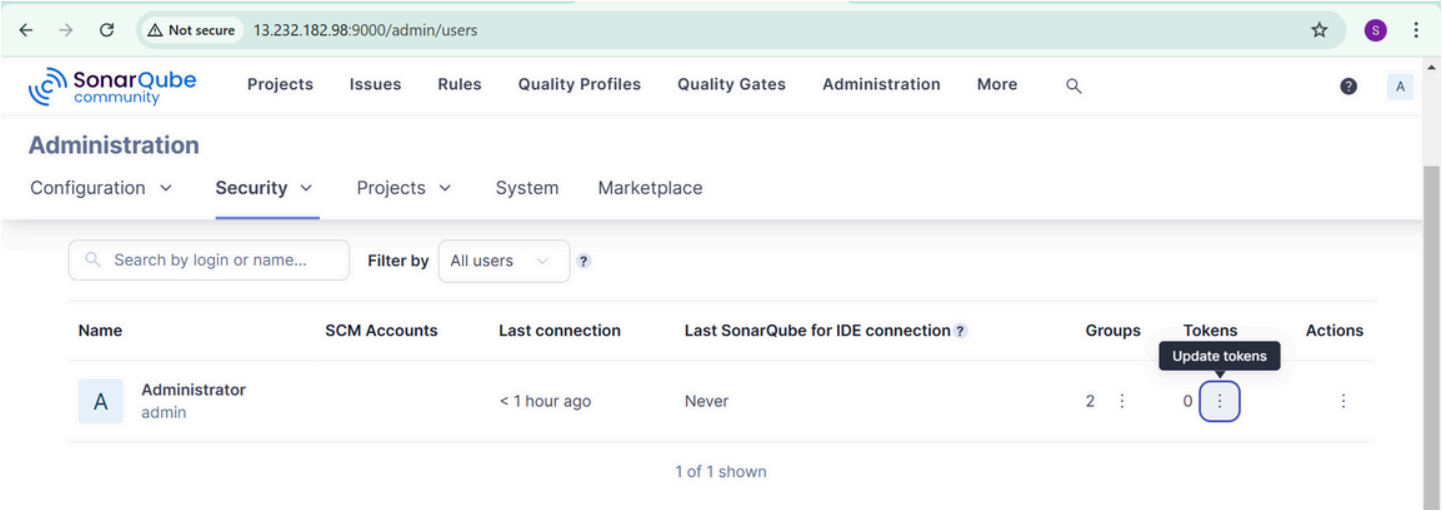
Install the following plugins in Jenkins:

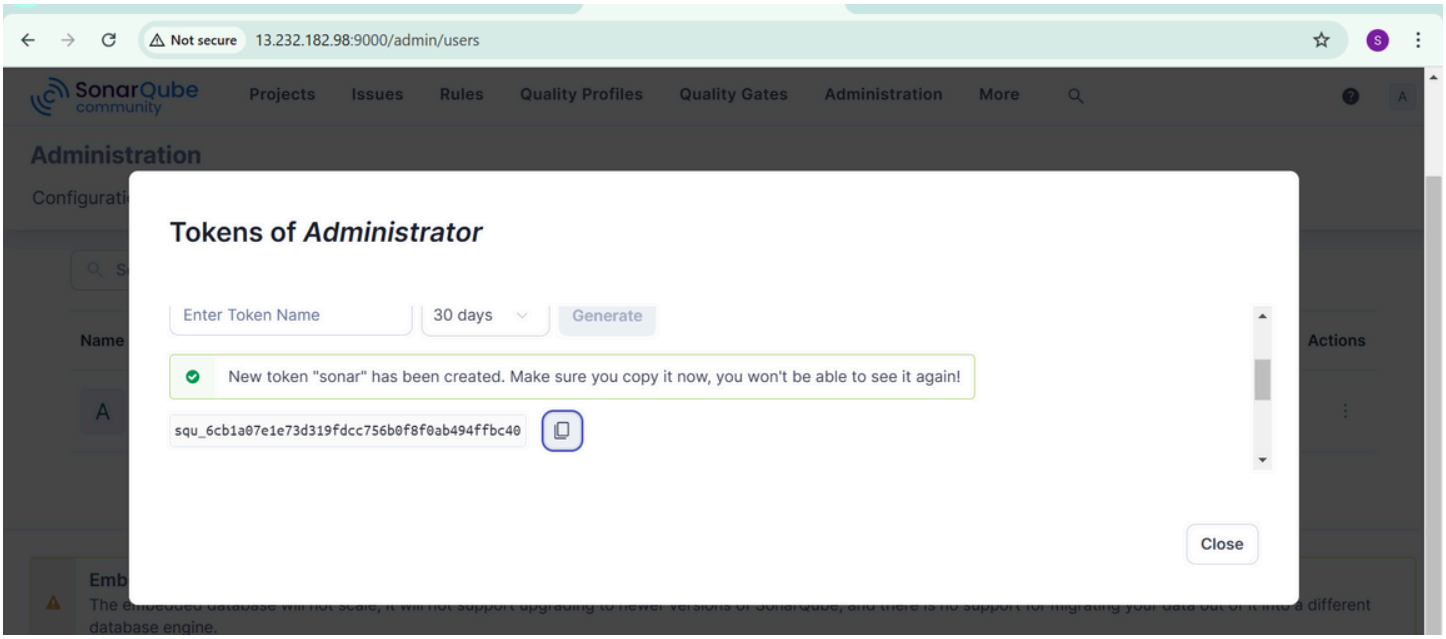
- Slack Notification
- SonarQube Scanner



In SonarQube:

- Go to **Administration > Security > Tokens**.
- Generate a token and copy it.

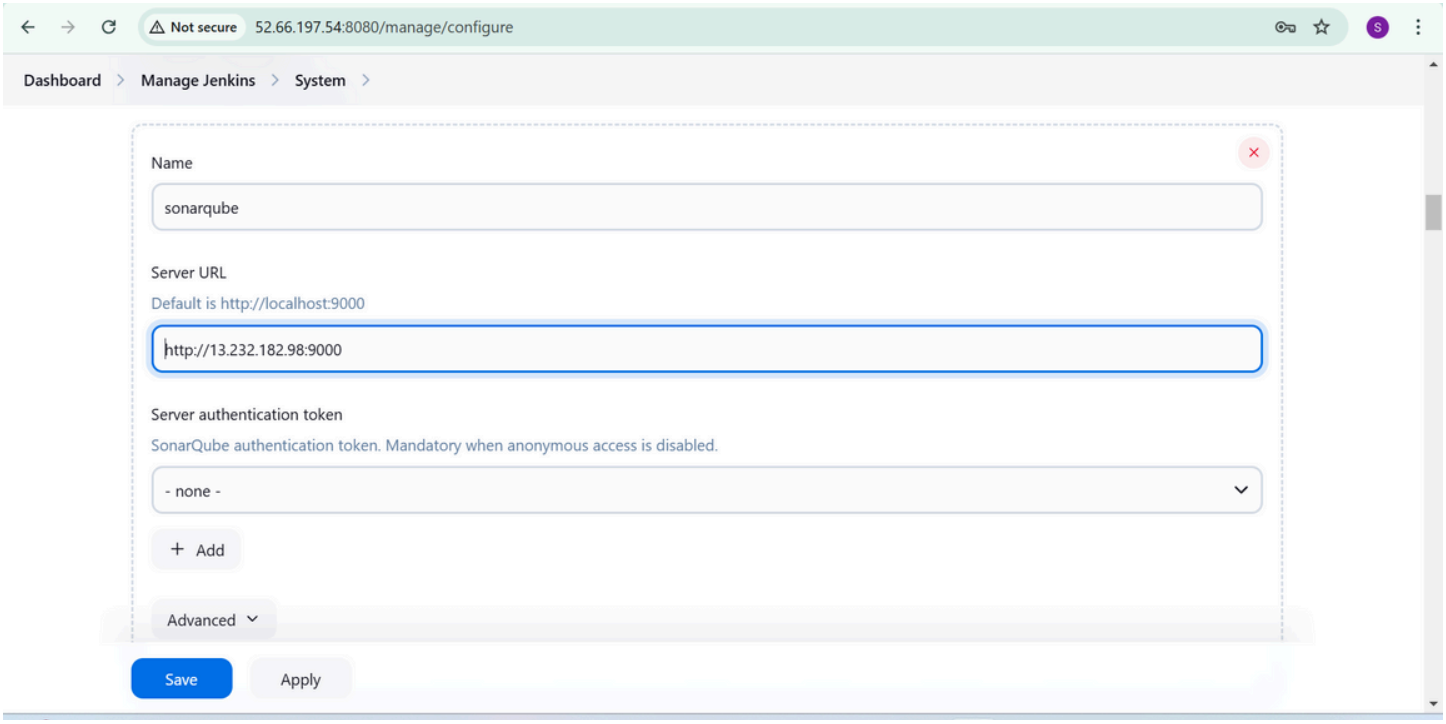


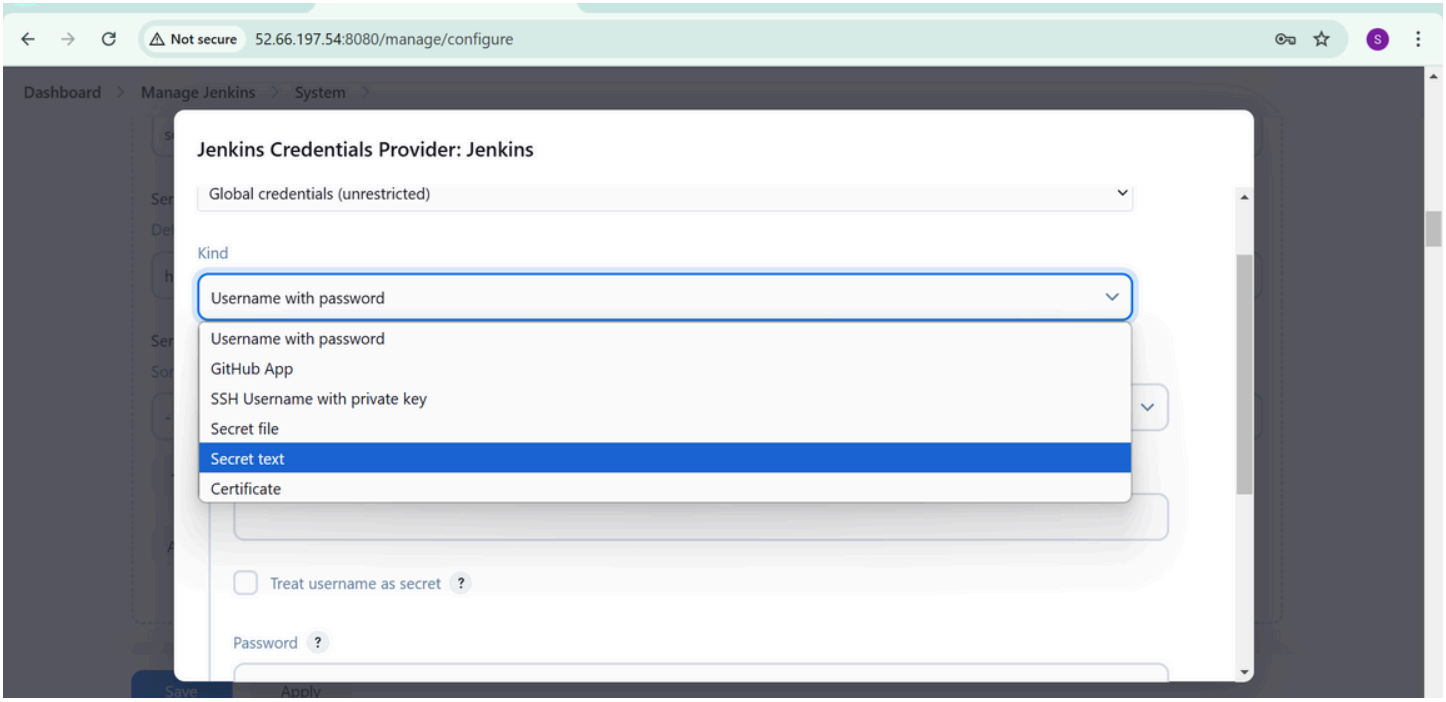


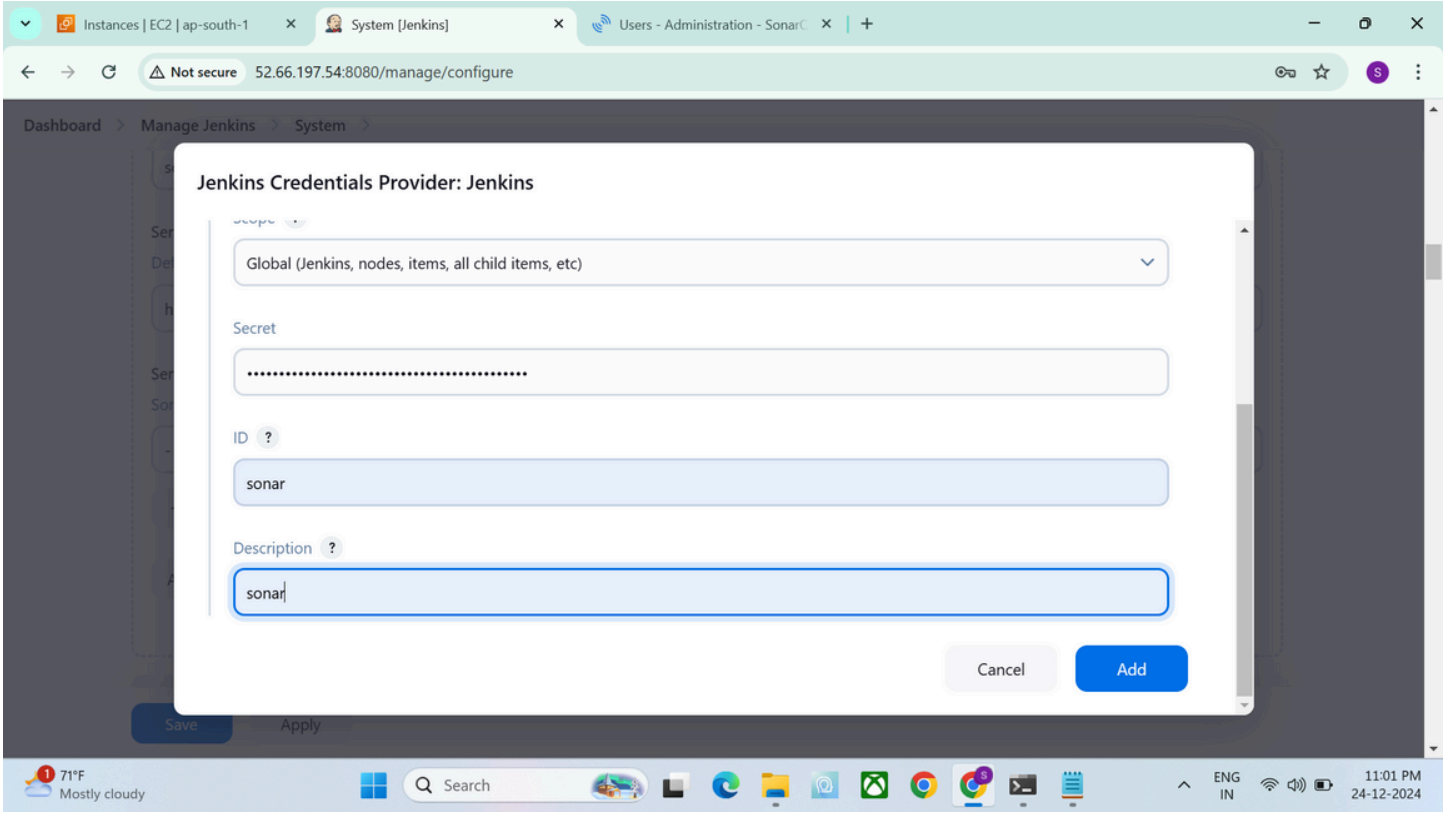
In Jenkins:

- Navigate to Manage **Jenkins > System Configuration > SonarQube Servers**.

- Add a new SonarQube server with the following details:
  - Name: SonarQube
  - url : http://<public-ip>9000
  - Credentials: Add a secret text credential and paste the token.
- Save the configuration.

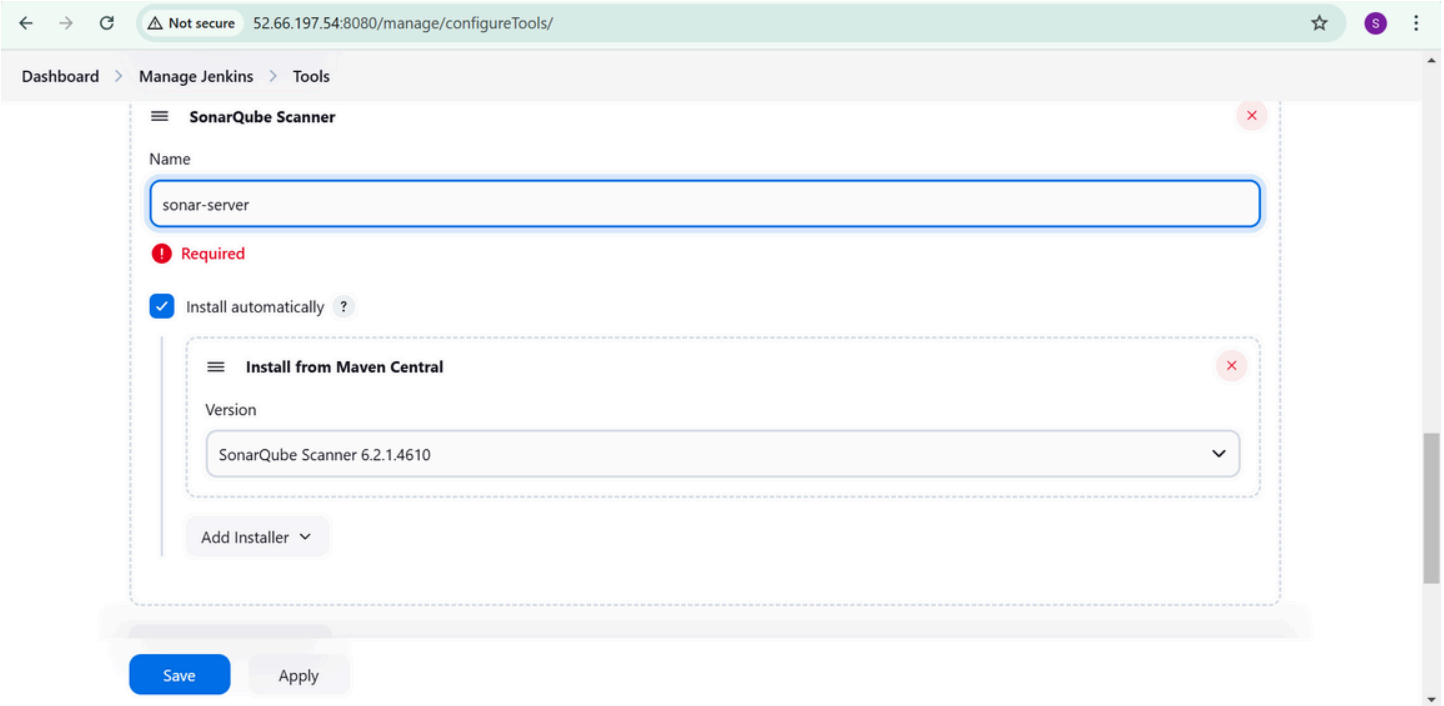






Navigate to Manage Jenkins > Global Tool Configuration.

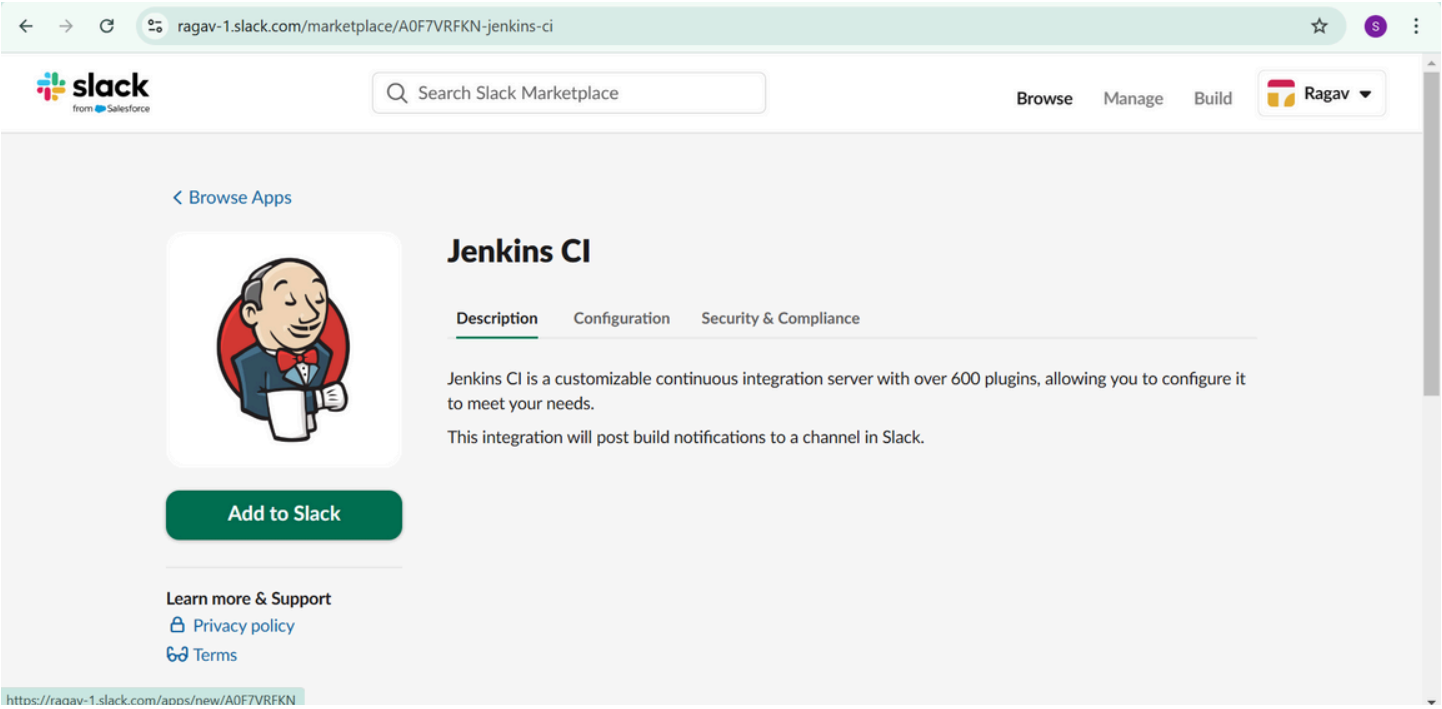
- Under SonarQube Scanner, add a new scanner:
  - Name: sonar-server
  - Install automatically.

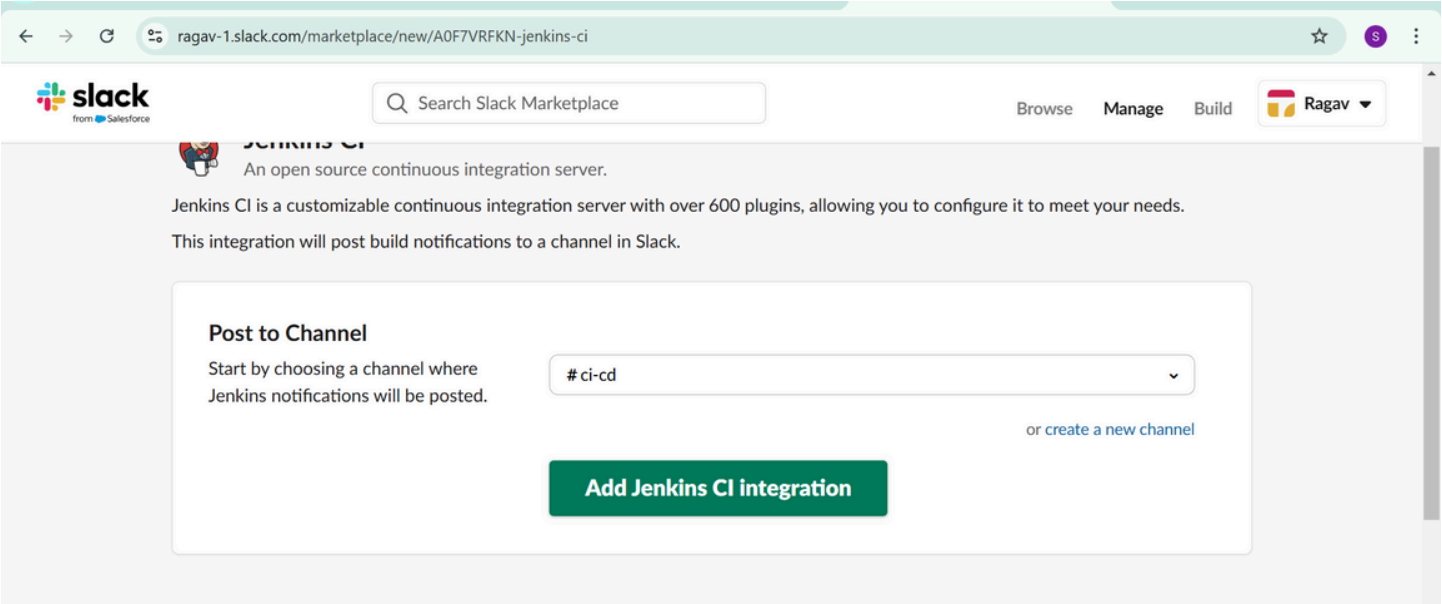


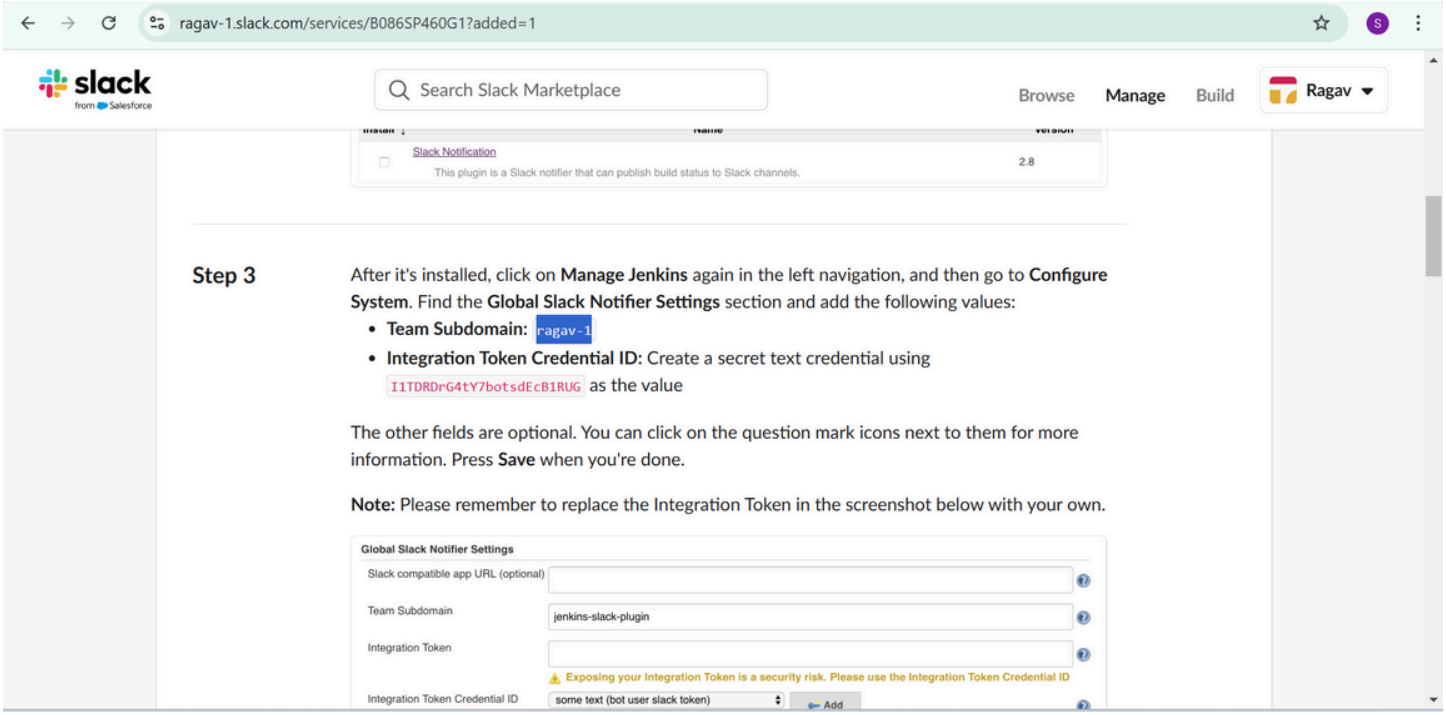
Step 4: Configure Slack Notifications

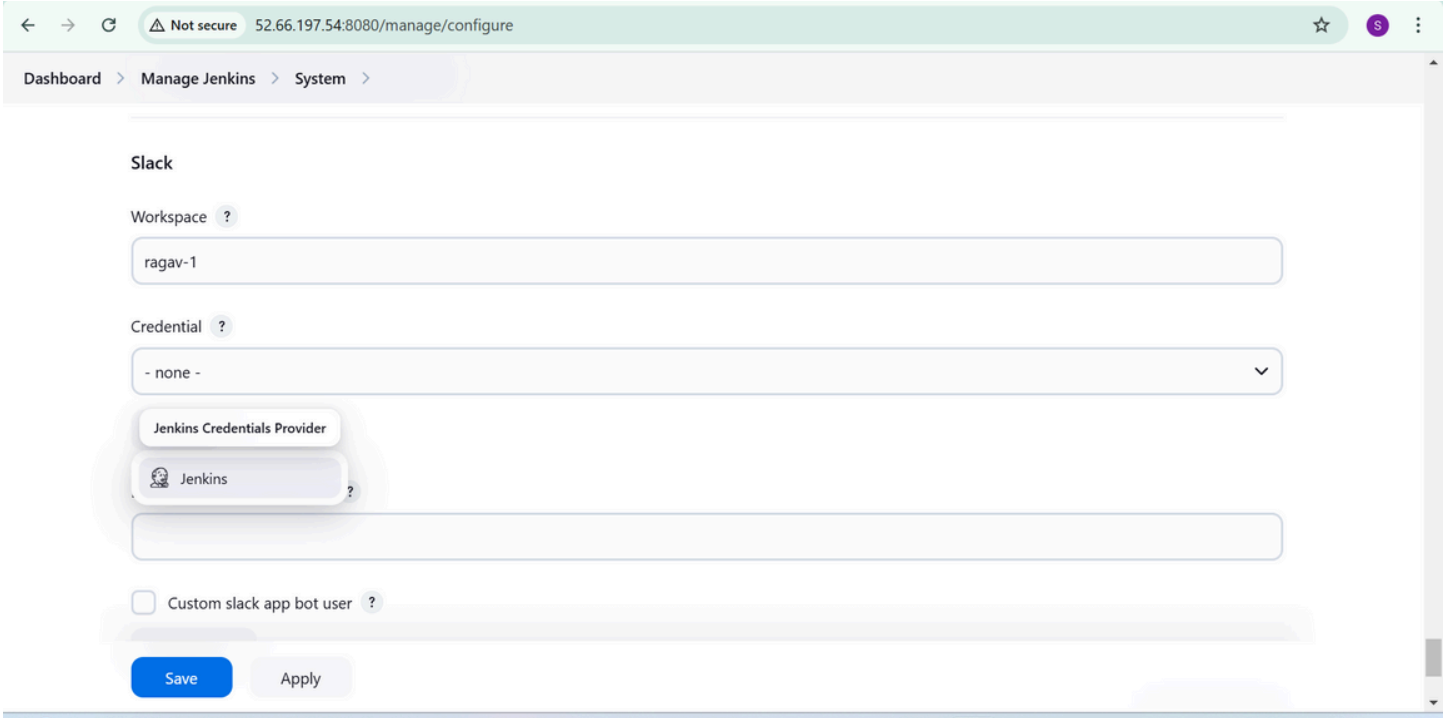
1. **In Slack:**
   - Go to Slack Apps and select Jenkins CI.
   - Configure it for your desired channel and generate a token.

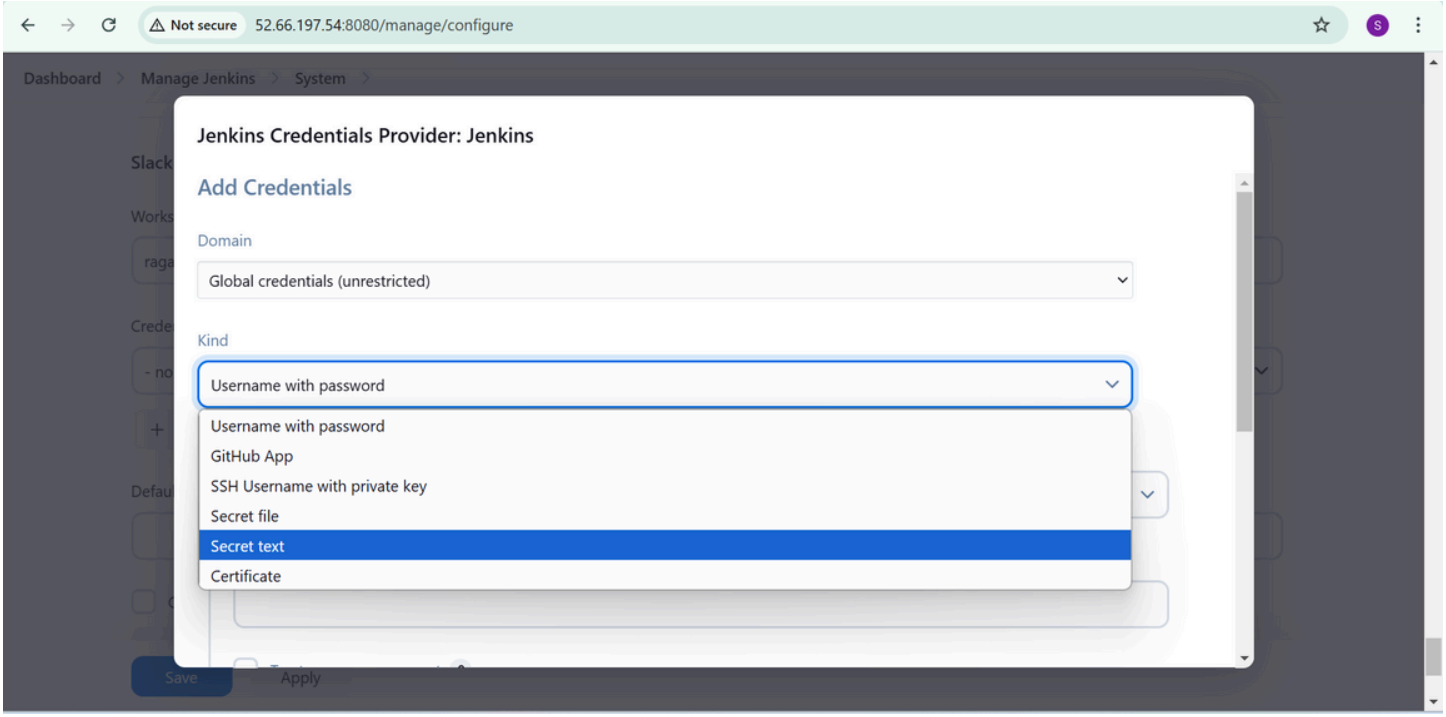
2. **In Jenkins:**
   - Navigate to **Manage Jenkins > System Configuration > Slack**.
   - Add Slack workspace and channel credentials:
     - Credential Type: Secret text.
     - Secret: Paste the Slack token.
   - Save the configuration.

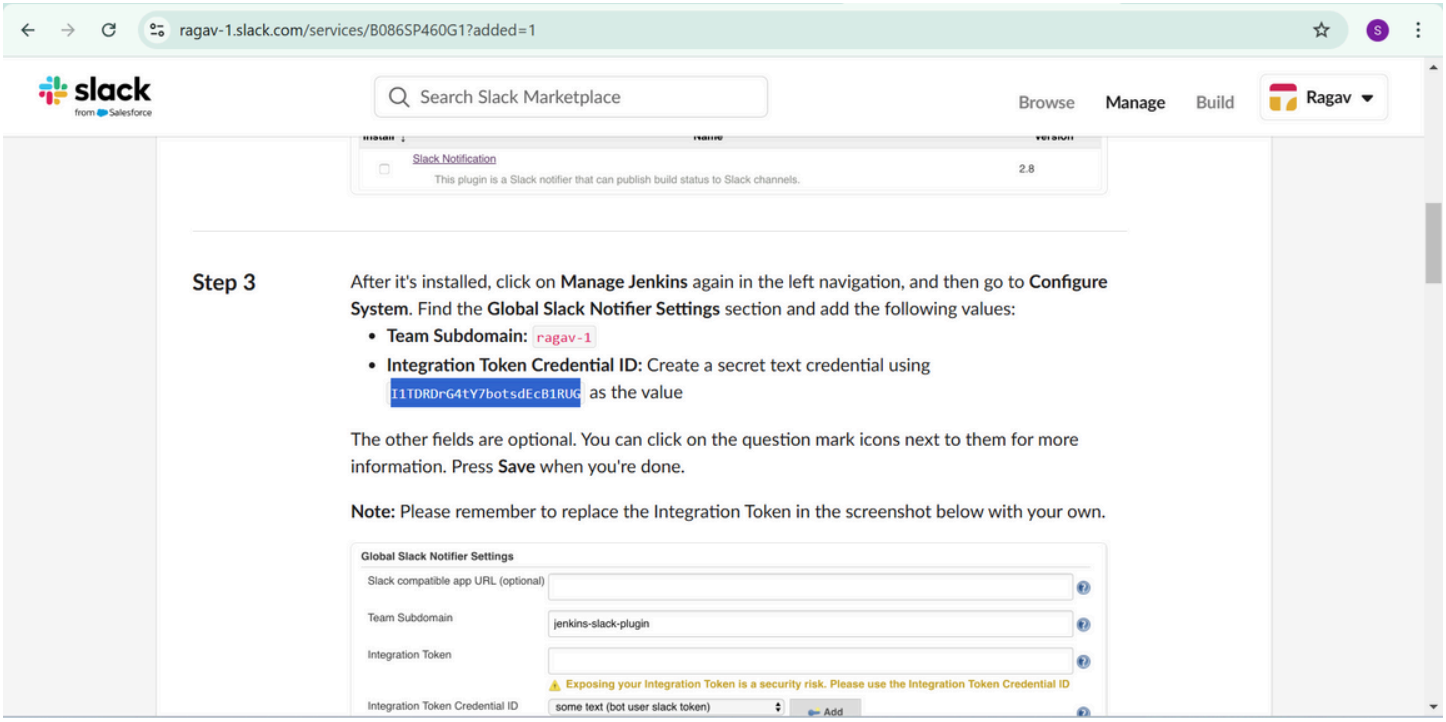


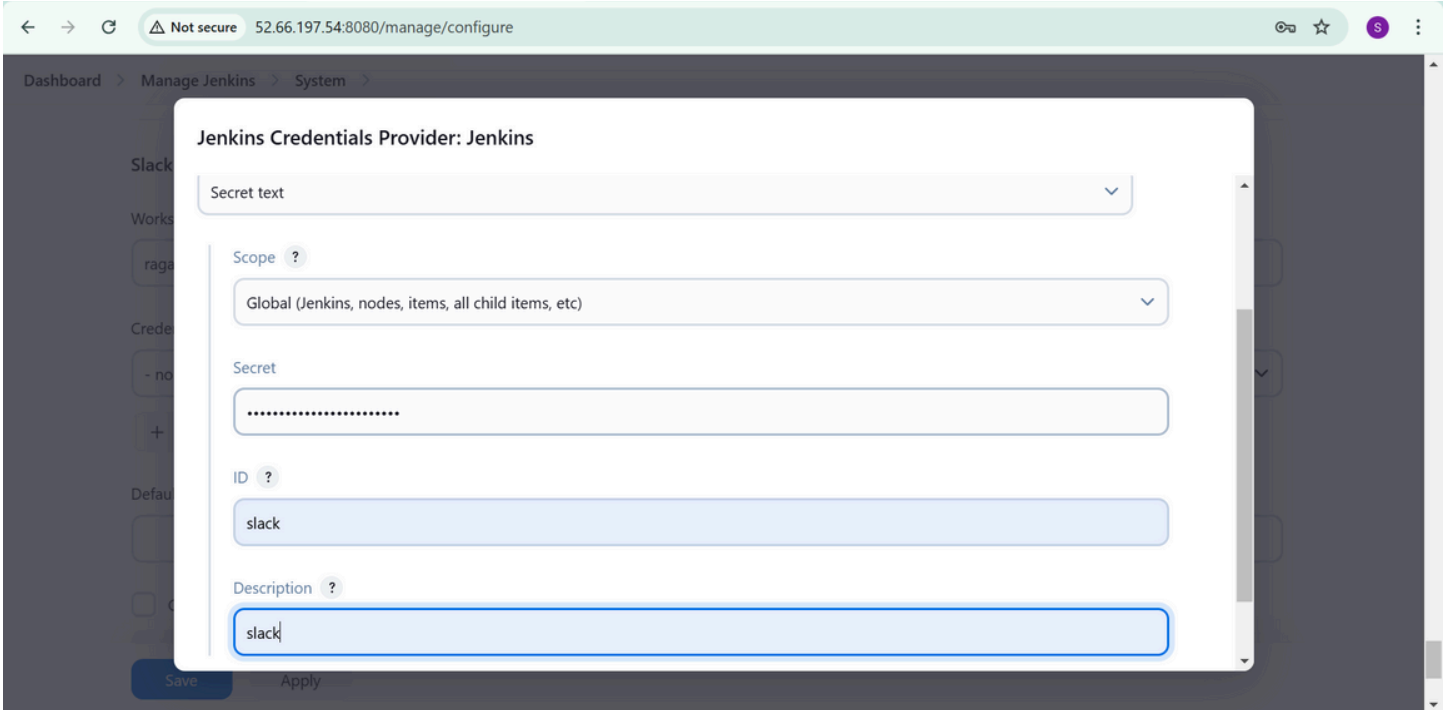


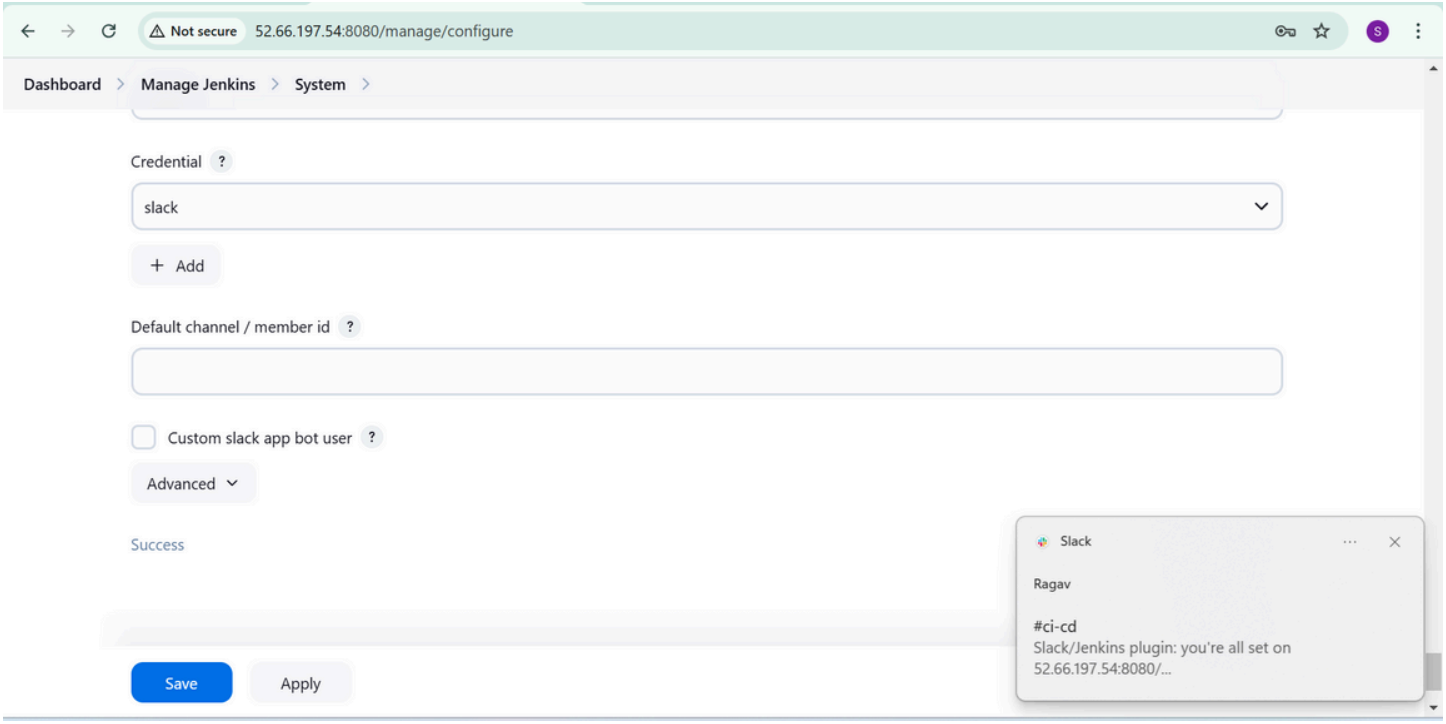

ragav-1.slack.com/services/B086SP460G1?added=1
slack
from Salesforce
Search Slack Marketplace
Browse
Manage
Build
Ragav
Slack Notification
This plugin is a Slack notifier that can publish build status to Slack channels.
2.8
Step 3
After it's installed, click on Manage Jenkins again in the left navigation, and then go to Configure System. Find the Global Slack Notifier Settings section and add the following values:
Team Subdomain: ragav-1
Integration Token Credential ID: Create a secret text credential using I1TDRDrG4tY7botsdEcB1RUG as the value
The other fields are optional. You can click on the question mark icons next to them for more information. Press Save when you're done.
Note: Please remember to replace the Integration Token in the screenshot below with your own.
Global Slack Notifier Settings
Slack compatible app URL (optional)
Team Subdomain
jenkins-slack-plugin
Integration Token
Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID
Integration Token Credential ID
some text (bot user slack token)
Add


Not secure
52.66.197.54:8080/manage/configure
Dashboard
Manage Jenkins
System
Slack
Workspace
ragav-1
Credential
- none -
Jenkins Credentials Provider
Jenkins
Custom slack app bot user
Save
Apply


Not secure
52.66.197.54:8080/manage/configure
Dashboard
Manage Jenkins
System
Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Username with password
Username with password
GitHub App
SSH Username with private key
Secret file
Secret text
Certificate
Save
Apply


ragav-1.slack.com/services/B086SP460G1?added=1
slack
from Salesforce
Search Slack Marketplace
Browse
Manage
Build
Ragav
Slack Notification
This plugin is a Slack notifier that can publish build status to Slack channels.
2.8
Step 3
After it's installed, click on Manage Jenkins again in the left navigation, and then go to Configure System. Find the Global Slack Notifier Settings section and add the following values:
Team Subdomain: ragav-1
Integration Token Credential ID: Create a secret text credential using I1TDRDrG4tY7botsdEcB1RUG as the value
The other fields are optional. You can click on the question mark icons next to them for more information. Press Save when you're done.
Note: Please remember to replace the Integration Token in the screenshot below with your own.
Global Slack Notifier Settings
Slack compatible app URL (optional)
Team Subdomain
jenkins-slack-plugin
Integration Token
Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID
Integration Token Credential ID
some text (bot user slack token)
Add





Step 5: Create and Run Jenkins Pipeline Job

1. In Jenkins, create a new Pipeline job.
2. Select Pipeline script from SCM and provide the repository URL:
   - https://github.com/RagavMuthukumar/java-spring-boot.git
3. Use the following Jenkinsfile for the pipeline:

```
def COLOR_MAP = [

  'SUCCESS': 'good',

  'FAILURE': 'danger'

]
```

```
pipeline {
  agent any
  environment {
    SCANNER_HOME = tool 'sonar-server'
  }
  stages {
    stage('Git Checkout') {
      steps {
        git 'https://github.com/RagavMuthukumar/java-spring-boot.git'
      }
    }
    stage('Compile') {
      steps {
        sh 'mvn compile'
      }
    }
    stage('Code Analysis') {
      steps {
        withSonarQubeEnv('SonarQube') {
          sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Java-Springboot \
          -Dsonar.java.binaries=. \
          -Dsonar.projectKey=Java-Springboot'''
        }
      }
    }
    stage('Package') {
```

```
        steps {
            sh 'mvn package'
        }
    }
    stage('Docker Build') {
        steps {
            sh 'docker build -t java-spring .'
        }
    }
    stage('Run Docker Container') {
        steps {
            sh 'docker run -itd --name ci-cd-container -p 8085:8080 java-spring'
        }
    }
  }
  post {
    always {
        echo 'Slack Notification.'
        slackSend channel: '#cicd',
        color: COLOR_MAP[currentBuild.currentResult],
        message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} build ${env.BUILD_NUMBER} \nMore info at: ${env.BUILD_URL}"
    }
  }
}
```

Save and run the job.

Step 6: Resolve Docker Permission Issue

If the error Permission denied: /var/run/docker.sock occurs:

```
[Pipeline] sh
+ docker build -t java-spring .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:
Get "http://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
[Pipeline] }
```

1. Grant Docker permissions to Jenkins:

sudo usermod -aG docker jenkins

Step 7: Dockerfile for Java Application

Use the following Dockerfile for the Java application:

```
FROM openjdk:17-alpine

WORKDIR /build

COPY target/demo-0.0.1-SNAPSHOT.jar /build

ENTRYPOINT ["java", "-jar", "demo-0.0.1-SNAPSHOT.jar"]

EXPOSE 80
```