

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.

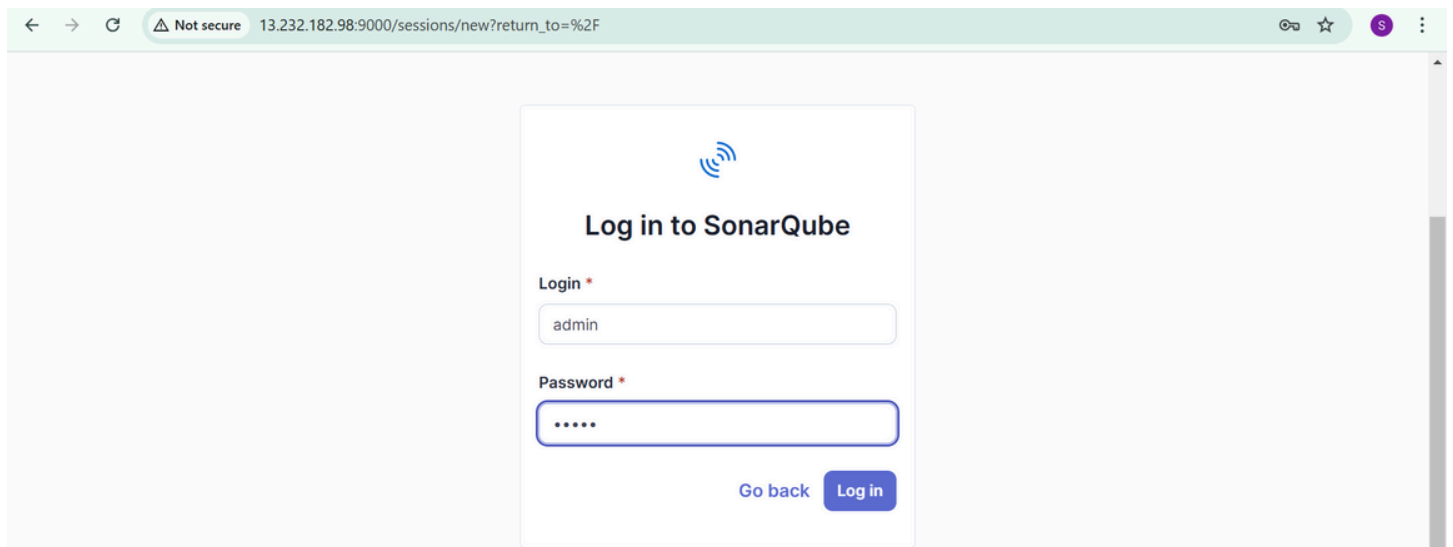
Step 2: Set Up SonarQube Instance

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.



← → ↻ ⚠ Not secure 13.232.182.98:9000/sessions/new?return_to=%2F 🔑 ☆ S ⋮



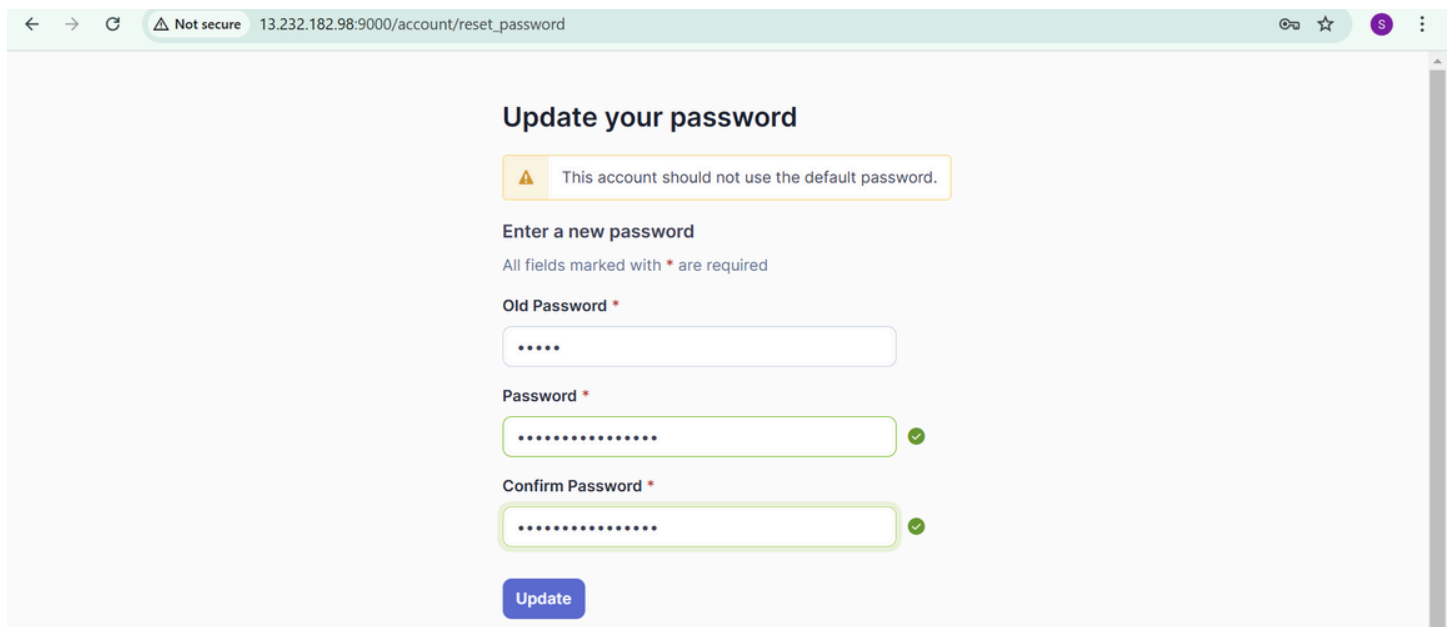
Log in to SonarQube

Login *

Password *


[Go back](#) [Log in](#)

first login is admin and password is admin



← → ↻ ⚠ Not secure 13.232.182.98:9000/account/reset_password 🔑 ☆ S ⋮

Update your password

 This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

Password *

 ✓

Confirm Password *

 ✓

[Update](#)

Step 3: Configure SonarQube in Jenkins

Install the following plugins in Jenkins:

- Slack Notification
- SonarQube Scanner

Dashboard > Manage Jenkins > Plugins

Plugins

Search available plugins

Install

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Install	Name	Released
<input checked="" type="checkbox"/>	Slack Notification 751.v2e44153c8fe1 slack Build Notifiers Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels.	2 mo 17 days ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.3 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	1 mo 6 days ago

In SonarQube:

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

← → ↻ Not secure 13.232.182.98:9000/admin/users




SonarQube community

Projects Issues Rules Quality Profiles Quality Gates Administration More

Administration

Configuration Security Projects System Marketplace

Search by login or name... Filter by All users

Name	SCM Accounts	Last connection	Last SonarQube for IDE connection ?	Groups	Tokens	Actions
 Administrator admin		< 1 hour ago	Never	2	0 	

1 of 1 shown

← → ↻ Not secure 13.232.182.98:9000/admin/users

SonarQube community


Projects Issues Rules Quality Profiles Quality Gates Administration More


Administration

Configuration

Tokens of Administrator

Enter Token Name 30 days Generate

 New token "sonar" has been created. Make sure you copy it now, you won't be able to see it again!

squ_6cb1a07e1e73d319fdcc756b0f8f0ab494ffbc40 

Close

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.

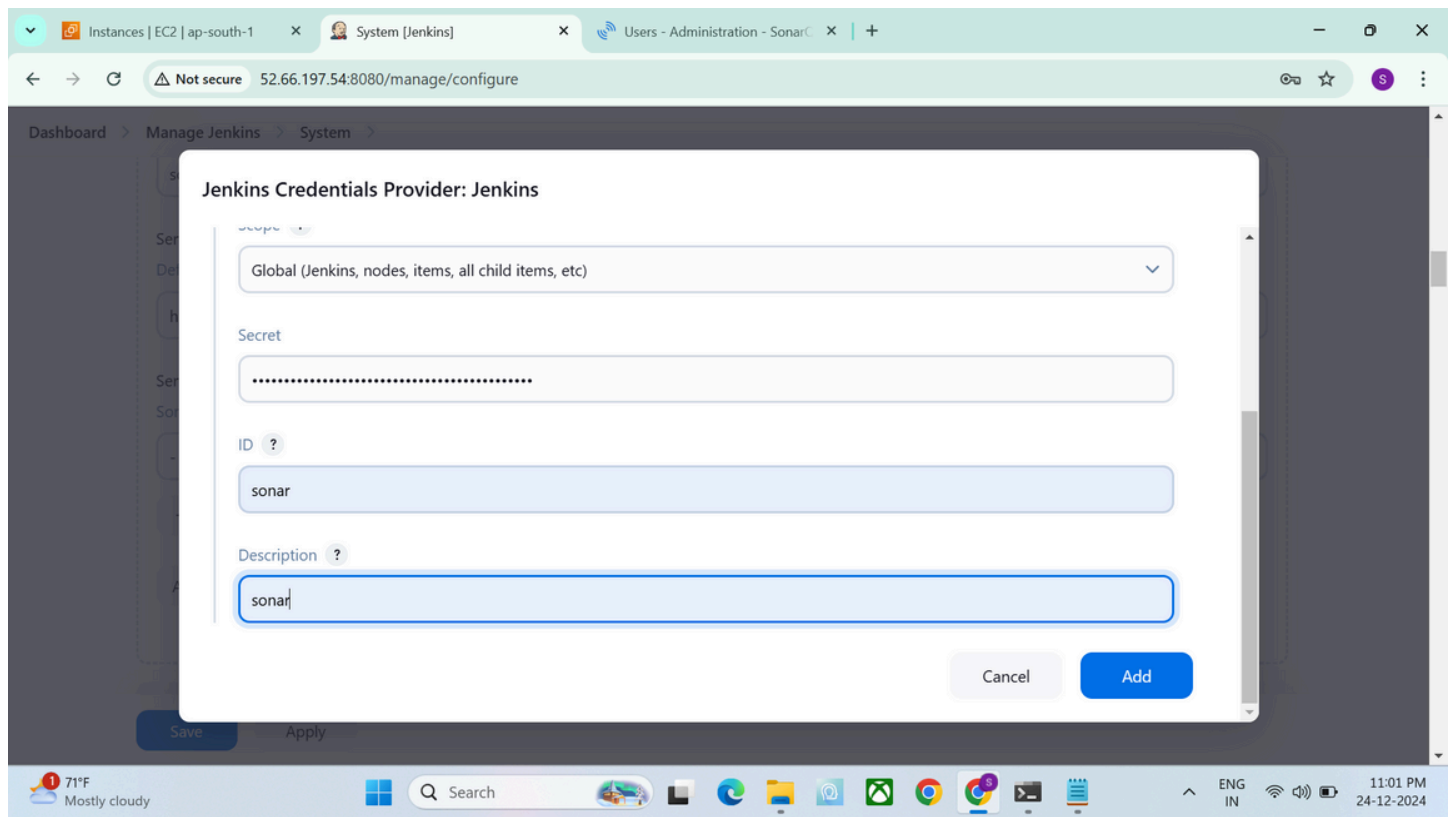
The screenshot shows the Jenkins configuration page for SonarQube. The browser address bar indicates the URL is 52.66.197.54:8080/manage/configure. The breadcrumb navigation shows Dashboard > Manage Jenkins > System > SonarQube. The configuration form includes the following fields:

- Name:** sonarqube
- Server URL:** http://13.232.182.98:9000 (Default is http://localhost:9000)
- Server authentication token:** - none - (Mandatory when anonymous access is disabled)
- + Add** button
- Advanced** dropdown menu
- Save** and **Apply** buttons at the bottom.

The screenshot shows the 'Jenkins Credentials Provider: Jenkins' dialog box. The 'Global credentials (unrestricted)' dropdown is selected. The 'Kind' dropdown is open, showing the following options:

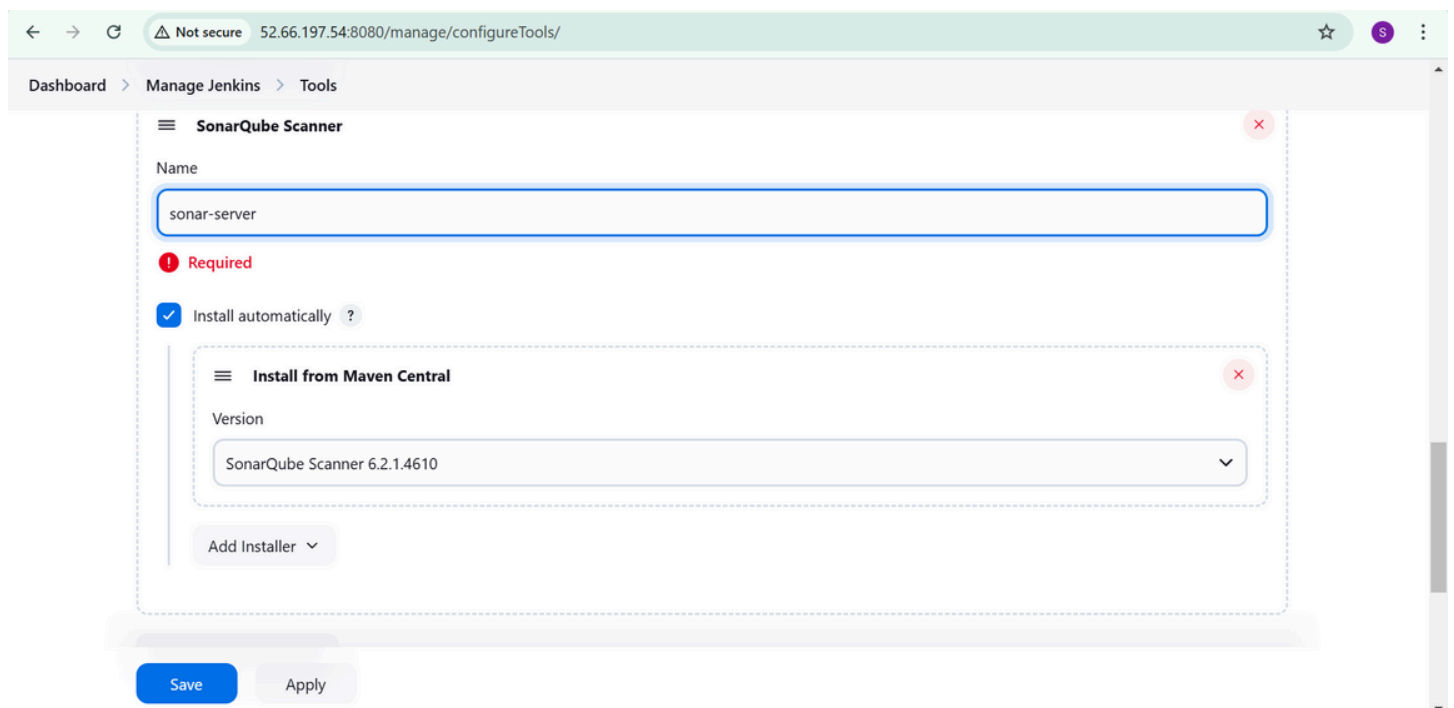
- Username with password
- Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text** (highlighted)
- Certificate

Below the dropdown, there is a checkbox for 'Treat username as secret' and a 'Password' field with a question mark icon. The 'Apply' button is visible at the bottom.



Navigate to Manage Jenkins > Global Tool Configuration.

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



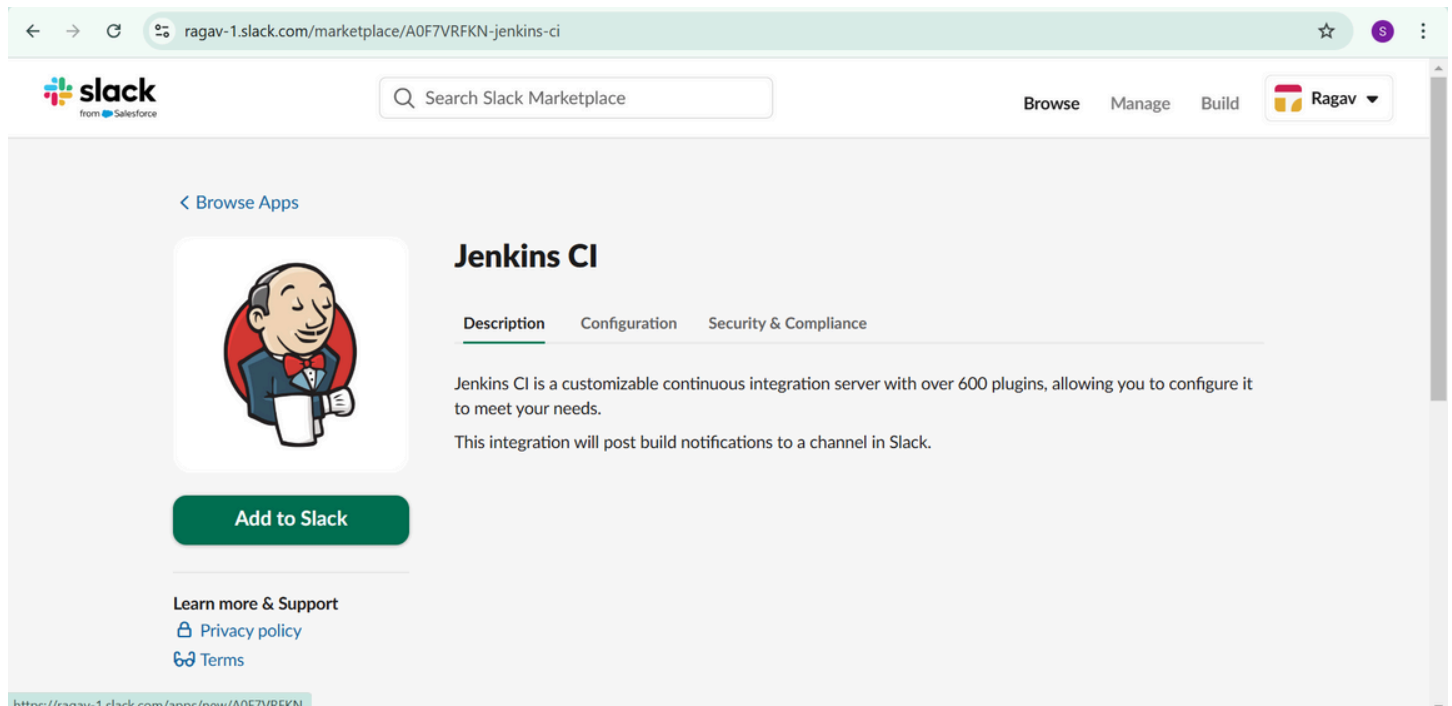
Step 4: Configure Slack Notifications

1. In Slack:

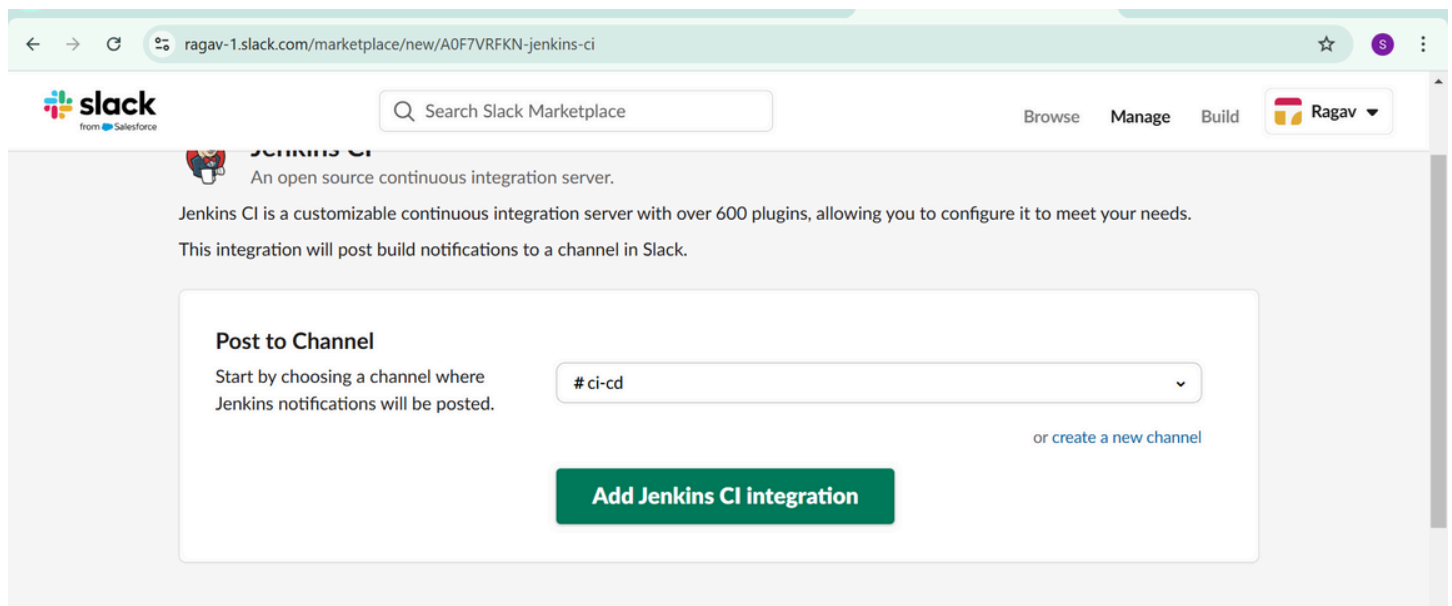
- Go to Slack Apps and select Jenkins CI.
- Configure it for your desired channel and generate a token.

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.




The screenshot shows the Slack Marketplace page for the Jenkins CI app. The browser address bar displays 'ragav-1.slack.com/marketplace/A0F7VRFKN-jenkins-ci'. The Slack logo and 'from Salesforce' tag are in the top left. A search bar labeled 'Search Slack Marketplace' is in the top center. Navigation links 'Browse', 'Manage', and 'Build' are in the top right, along with a user profile 'Ragav'. The main content area features the Jenkins CI app card with its logo (a man in a tuxedo) and a green 'Add to Slack' button. To the right of the card, the 'Description' tab is active, showing text about Jenkins CI's capabilities and how the integration posts build notifications to a Slack channel. Below the card, there are links for 'Learn more & Support', 'Privacy policy', and 'Terms'.



The screenshot shows the configuration page for the Jenkins CI integration in Slack. The browser address bar displays 'ragav-1.slack.com/marketplace/new/A0F7VRFKN-jenkins-ci'. The Slack logo and 'from Salesforce' tag are in the top left. A search bar labeled 'Search Slack Marketplace' is in the top center. Navigation links 'Browse', 'Manage', and 'Build' are in the top right, along with a user profile 'Ragav'. The main content area has a heading 'Jenkins CI' and a subheading 'An open source continuous integration server.' Below this, it states 'Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.' and 'This integration will post build notifications to a channel in Slack.' A section titled 'Post to Channel' contains the instruction 'Start by choosing a channel where Jenkins notifications will be posted.' and a dropdown menu showing '# ci-cd'. A link 'or create a new channel' is next to the dropdown. At the bottom, there is a large green button labeled 'Add Jenkins CI integration'.

← → ↻ ragav-1.slack.com/services/B086SP460G1?added=1 ☆ S

 Search Slack Marketplace Browse Manage Build Ragav

Slack Notification 2.8
This plugin is a Slack notifier that can publish build status to Slack channels.

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 I1TD0RDrG4tY7botsdEcB1RUG 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 ☆ S

Dashboard > Manage Jenkins > System >

Slack


Workspace ?

ragav-1

Credential ?

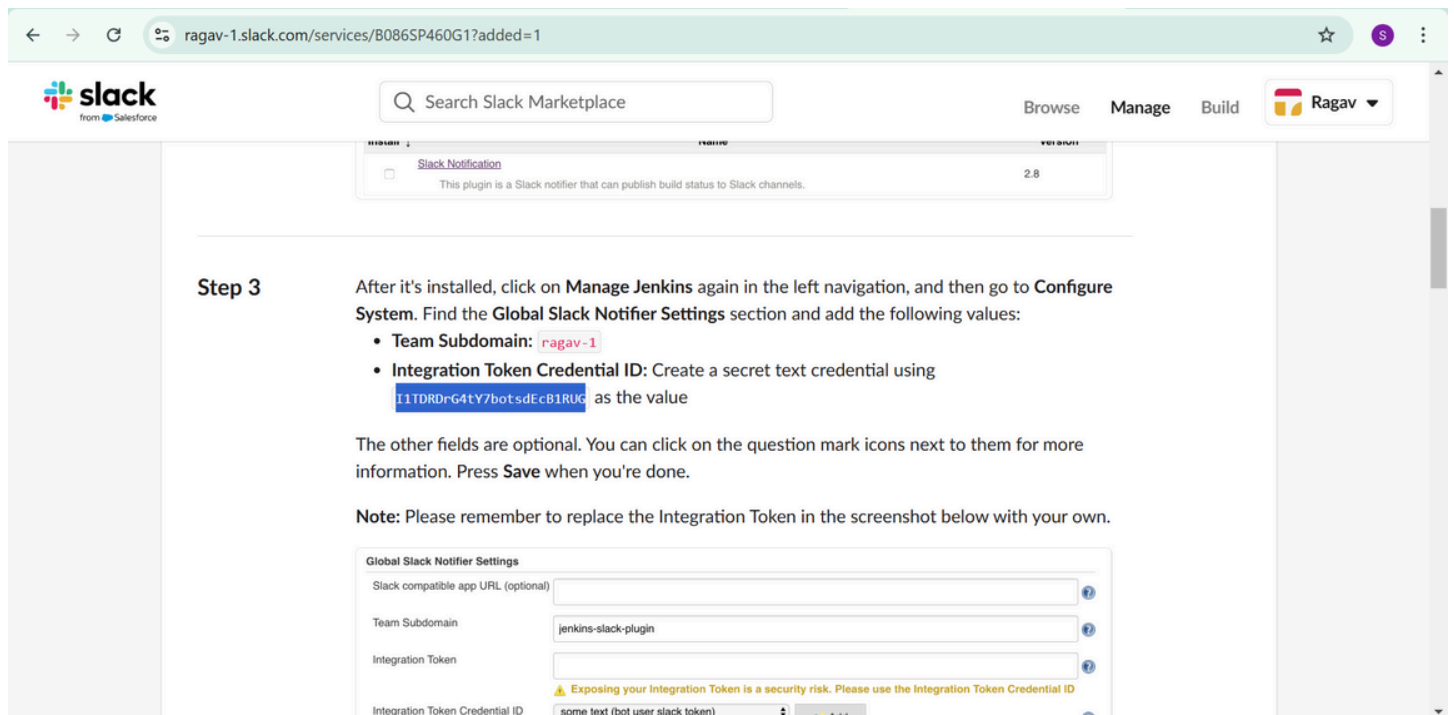
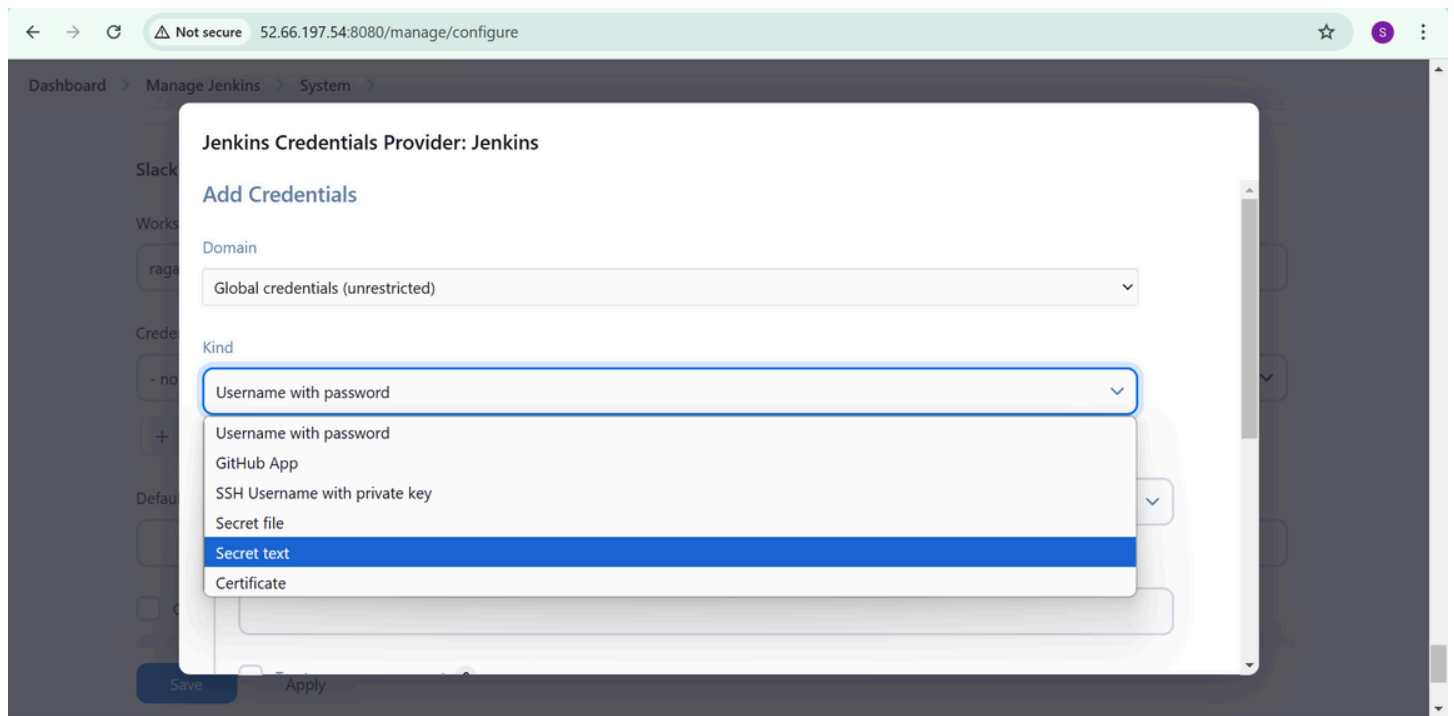
- none -

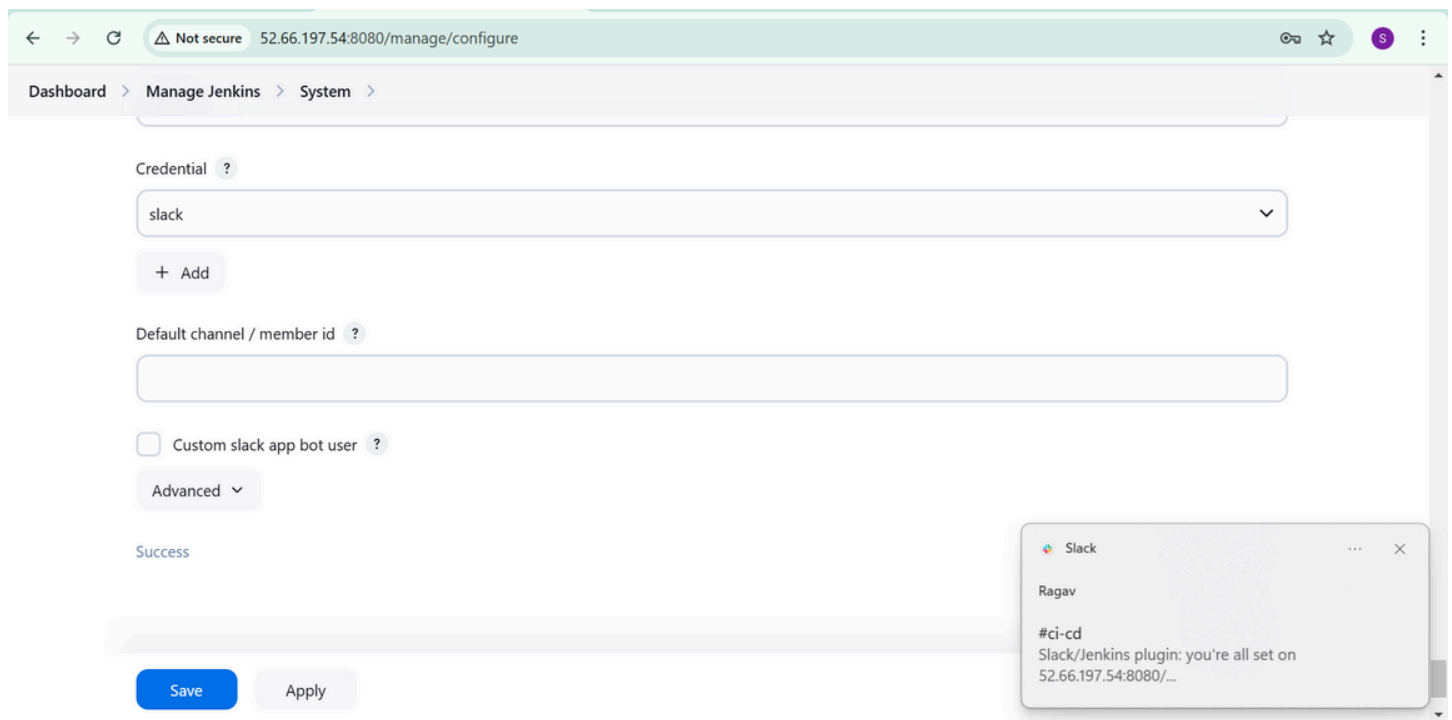
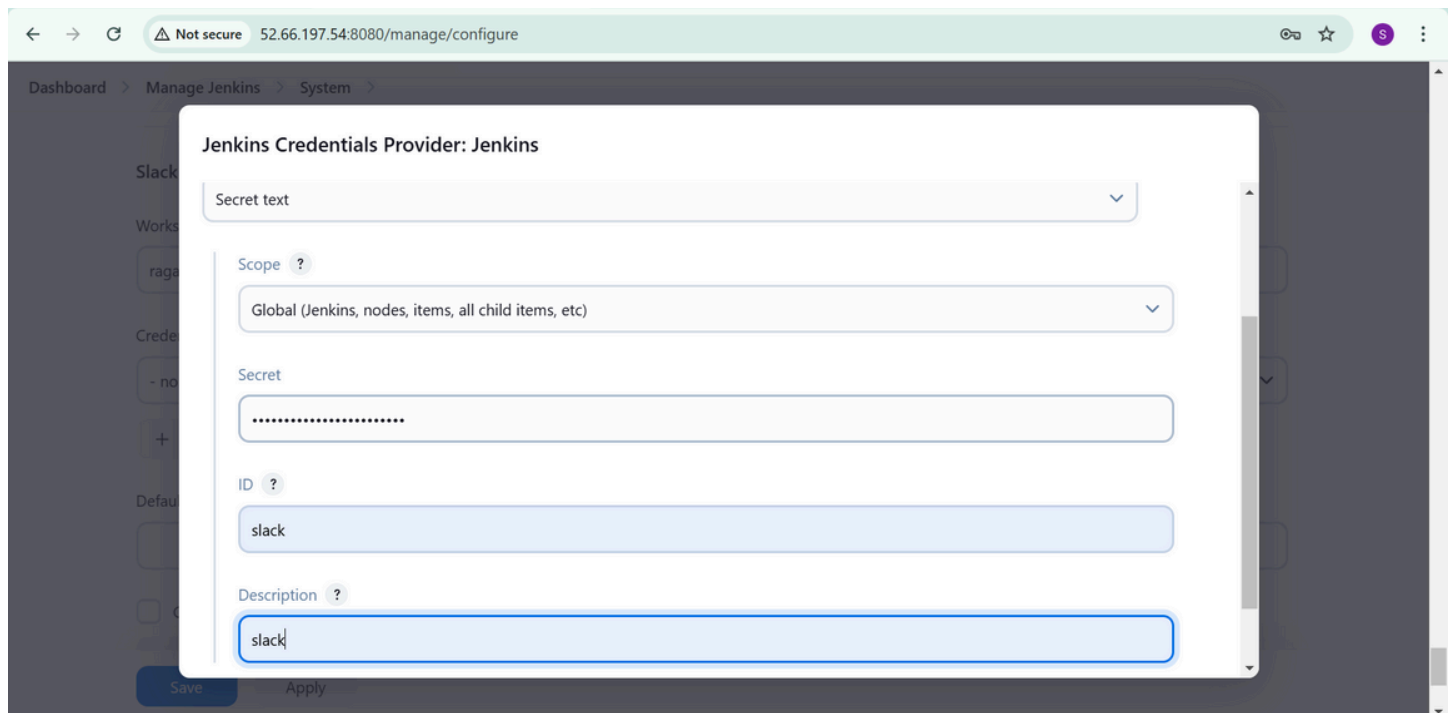
Jenkins Credentials Provider

 Jenkins ?

☐ Custom slack app bot user ?

Save Apply





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 { label 'slave-1' }  
  
    environment {  
  
        SCANNER_HOME = tool 'sonarqube'  
    }  
  
    stages {  
  
        stage('git checkout') {  
  
            steps {  
  
                git 'https://github.com/RagavMuthukumar/java-spring-boot.git'  
            }  
        }  
  
        stage('compile') {  
  
            steps {  
  
                sh 'mvn clean compile'  
            }  
        }  
  
        stage('code analysis') {  
  
            steps {  
  
                withSonarQubeEnv('sonar-server') {  
  
                    sh "'$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=java-spring-boot \  
                    -Dsonar.java.binaries=. \  
                    -Dsonar.projectKey=java-spring-boot'"  
                }  
            }  
        }  
  
        stage('docker clean') {
```

```
steps {

  script {

    sh '''

    docker stop $(docker ps -q) || true

    docker rm $(docker ps -a -q) || true

    docker rmi $(docker images -q) || true

    '''

  }

}

stage('docker build') {

  steps {

    script {

      sh 'docker build -t ragavmuthukumar/java-spring .'

    }

  }

}

stage('docker push') {

  steps {

    script {

      withDockerRegistry(credentialsId: 'docker-hub-credential', toolName: 'docker'){

        sh 'docker push ragavmuthukumar/java-spring'

      }

    }

  }

}

stage('docker container') {
```

```

steps {
    script {
        sh 'docker run -itd -p 8081:8080 java-spring'
    }
}

}

post {
    always {
        echo 'slack Notification.'

        slackSend(
            channel: '#ci-cd',

            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

sudo chmod 777 /var/run/docker.sock

Step 7: Dockerfile for Java Application

Use the following Multi stage build Dockerfile for the Java application:

```
FROM amazonlinux AS file
```

```
RUN yum install git -y
```

```
WORKDIR /app
```

```
RUN git clone https://github.com/RagavMuthukumar/java-spring-boot.git /app
```

```
FROM maven AS build
```

```
WORKDIR /source
```

```
COPY --from=file /app /source
```

```
RUN mvn clean install
```

```
FROM openjdk:17-alpine
```

```
WORKDIR /test
```

```
COPY --from=build /source/target/app-0.0.1-SNAPSHOT.war /test
```

```
CMD ["java", "-jar", "app-0.0.1-SNAPSHOT.war"]
```

```
EXPOSE 8080
```