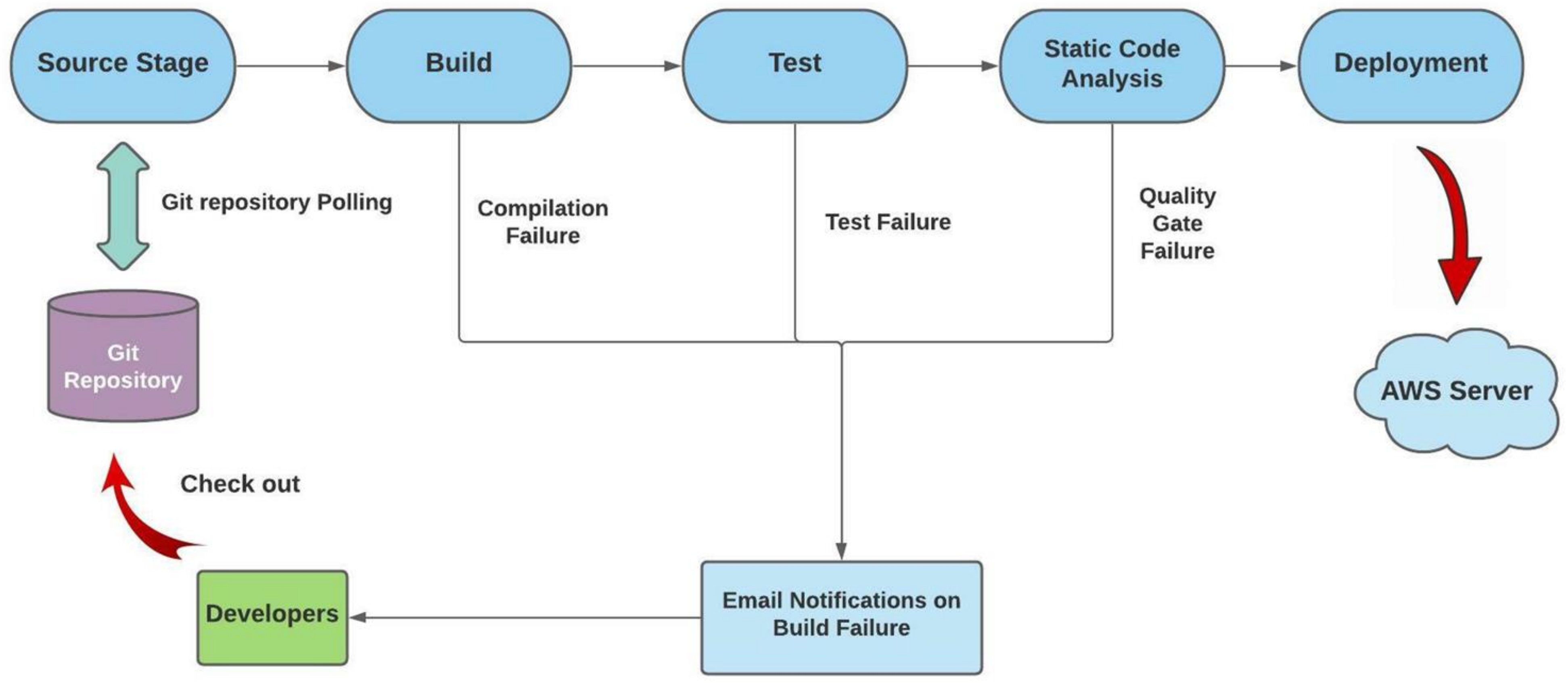
Automation of CI / CD Pipeline for Deployment of Spring Boot Application

Hot Desking Admin Module

Stages

- **Source Stage**
- In this stage, CI/CD pipeline is triggered by a code repository. Any change in the program triggers a notification to the CI/CD tool that runs an equivalent pipeline. Other common triggers include user-initiated workflows, automated schedules.
- **Build Stage**
- This is the second stage of the CI/CD Pipeline in which you merge the source code and its dependencies. It is done mainly to build a runnable instance of software that you can potentially ship to the end-user. Failure to pass the build stage means there is a fundamental project misconfiguration, so it is better that you address such issue immediately.
- **Test Stage**
- Test Stage includes the execution of automated tests to validate the correctness of code and the behaviour of the software. This stage prevents easily reproducible bugs from reaching the clients. It is the responsibility of developers to write automated tests.
- **Static Code Analysis Stage**
- Involves static analysis of the code base against parameters like code coverage , vulnerabilities , bugs etc by enforcing quality gates to detect issues like programming errors, coding standard violations, syntax violations, security vulnerabilities in the early stage.
- **Deployment**
- Involves deployment of the artifacts and executables generated to the server.

Overview


Source Stage
Build
Test
Static Code Analysis
Deployment
Git repository Polling
Compilation Failure
Test Failure
Quality Gate Failure
Git Repository
AWS Server
Check out
Developers
Email Notifications on Build Failure

Tools Used

- Operating System
  - Jenkins Host - Windows 10 Home
  - AWS Server – Amazon Linux 2
- Source Code Management – GitLab
- Java JDK – Open JDK 11. OpenJDK Runtime Environment 18.9
- Build tool – Apache Maven 3.8.1
- CI/CD Pipeline - Jenkins 2.289.1
- Static Code Analysis – SonarQube version 9.0

Resources

- OpenJDK 11
- https://jdk.java.net/java-se-ri/11
- Maven 3.8.1
- https://maven.apache.org/download.cgi
- Jenkins 2.289.1
- https://www.jenkins.io/download/
- Sonarqube v9.0
- https://www.sonarqube.org/downloads/
- Ngrok
- https://ngrok.com/download

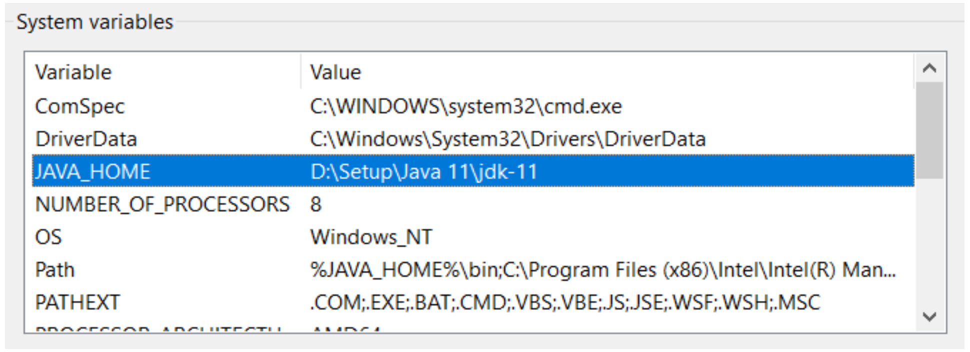
Installation

- How to install OpenJDK11 on Windows ?
- Git Installation Windows
- How to Setup Maven on Windows ?
- Jenkins Windows Installation Start on Port 8080 (default)
- SonarQube Installation (Server or Docker) Start on port 9000 (default)

Environment Variables on Jenkins Host

Go to Control Panel -> System -> Advanced System Settings -> Environment Variables

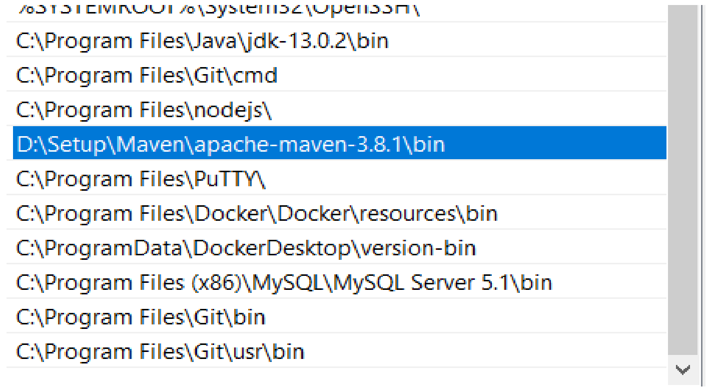
- JAVA_HOME



System variables

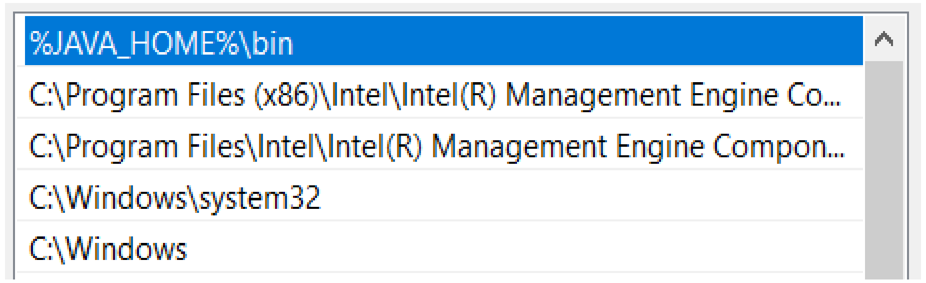
Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	D:\Setup\Java 11\jdk-11
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	%JAVA_HOME%\bin;C:\Program Files (x86)\Intel\Intel(R) Man...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU	AMD64

- In Path Set Maven Home



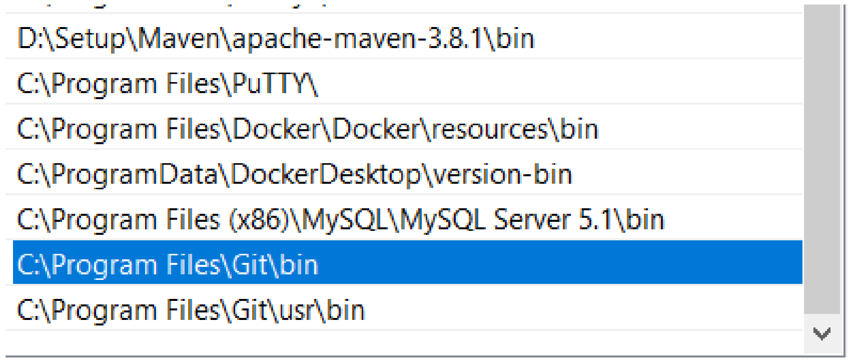
```
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Java\jdk-13.0.2\bin
C:\Program Files\Git\cmd
C:\Program Files\nodejs\
D:\Setup\Maven\apache-maven-3.8.1\bin
C:\Program Files\PuTTY\
C:\Program Files\Docker\Docker\resources\bin
C:\ProgramData\DockerDesktop\version-bin
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin
C:\Program Files\Git\bin
C:\Program Files\Git\usr\bin
```

- In Path Add \bin



```
%JAVA_HOME%\bin
C:\Program Files (x86)\Intel\Intel(R) Management Engine Co...
C:\Program Files\Intel\Intel(R) Management Engine Compon...
C:\Windows\system32
C:\Windows
```

- In Path Set Git and Git usr path



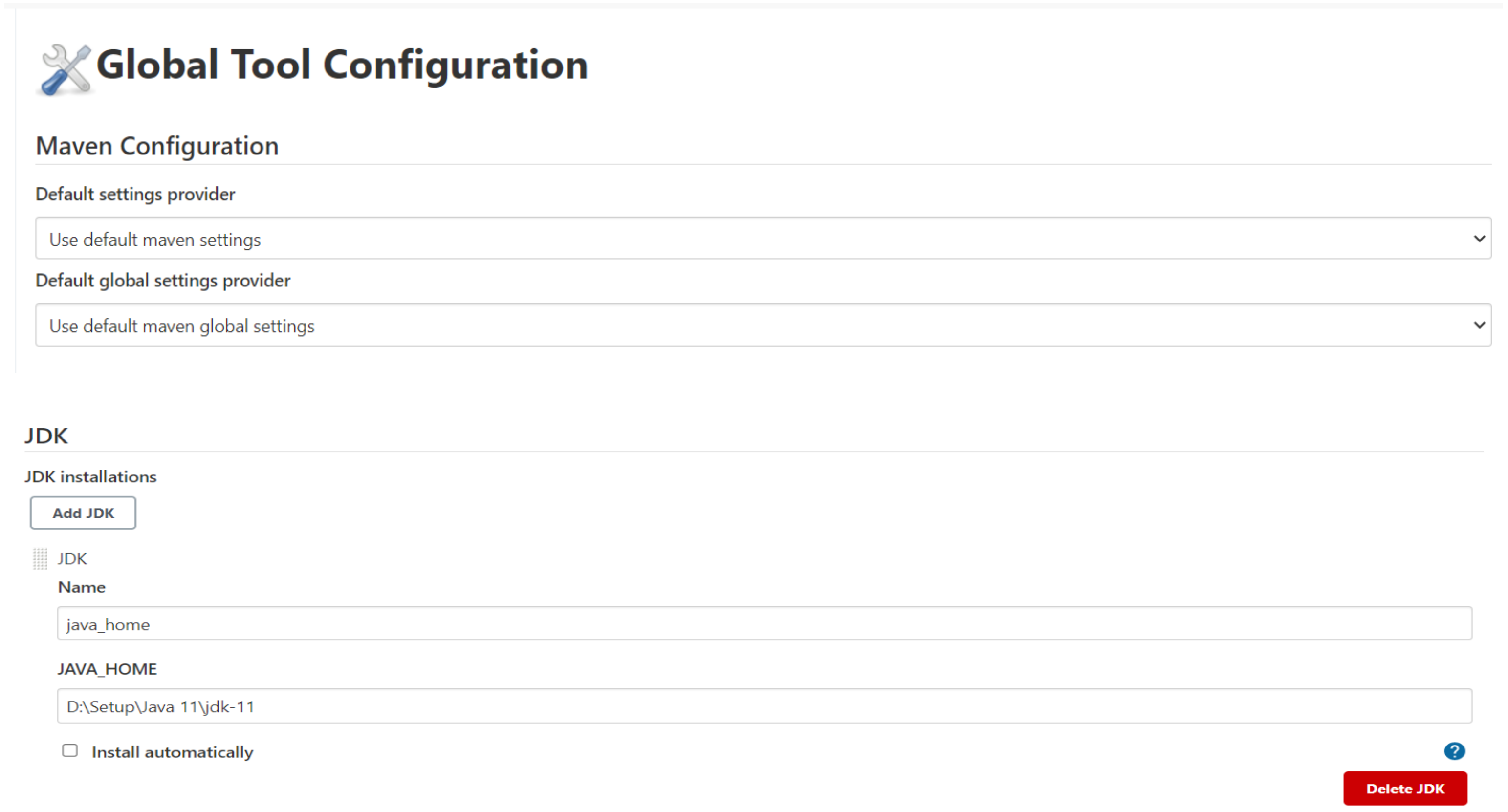
```
D:\Setup\Maven\apache-maven-3.8.1\bin
C:\Program Files\PuTTY\
C:\Program Files\Docker\Docker\resources\bin
C:\ProgramData\DockerDesktop\version-bin
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin
C:\Program Files\Git\bin
C:\Program Files\Git\usr\bin
```

Jenkins Plugins

- Default plugins
- Email Extension
- Gitlab plugin
- Gitlab API plugin
- JUnit Attachments
- SonarQube Scanner plugin
- Sonar Quality Gate Plugin
- Pipeline plugin
- SSH Build Agents

Jenkins Configuration

Go to Manage Jenkins -> Global Tool Configuration.



Global Tool Configuration

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK installations

Add JDK

JDK

Name

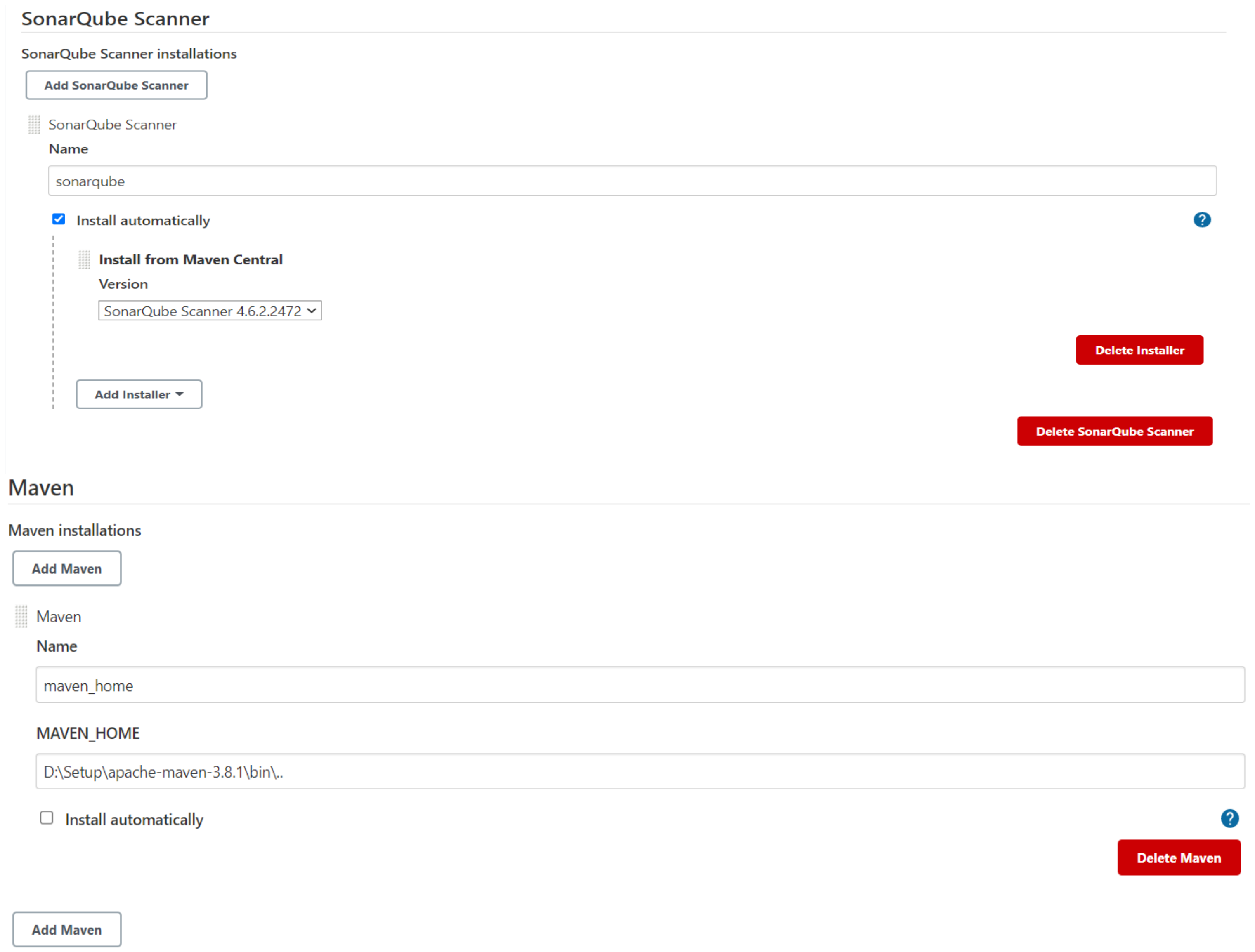
java_home

JAVA_HOME

D:\Setup\Java 11\jdk-11

☐ Install automatically

Delete JDK



SonarQube Scanner

SonarQube Scanner installations

SonarQube Scanner

Name

sonarqube

☑ Install automatically

Install from Maven Central

Version

SonarQube Scanner 4.6.2.2472

Maven

Maven installations

Maven

Name

maven_home

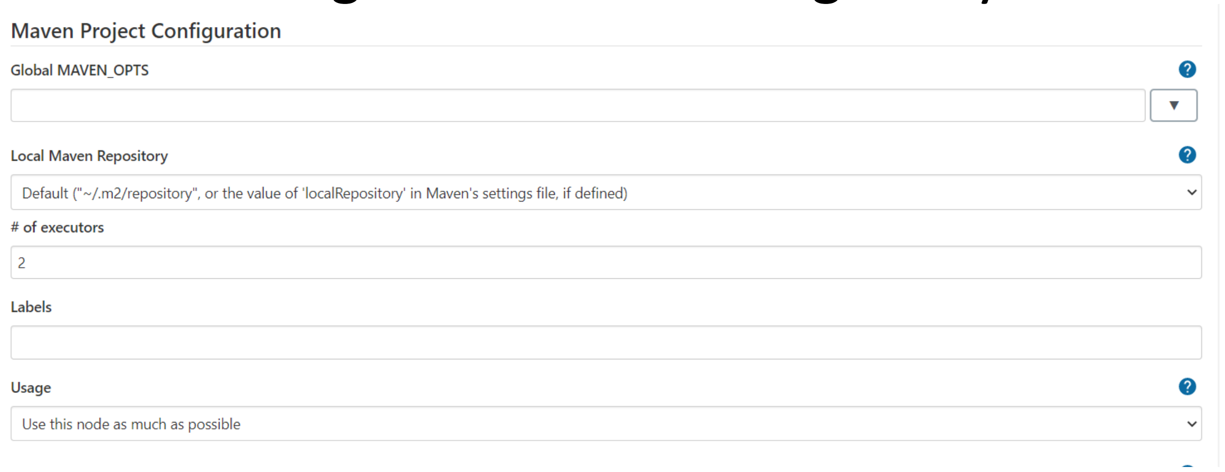
MAVEN_HOME

D:\Setup\apache-maven-3.8.1\bin\..

☐ Install automatically

Jenkins Configure System

Go to Manage Jenkins -> Configure System



Maven Project Configuration

Global MAVEN_OPTS

Local Maven Repository

Default ("~/.m2/repository", or the value of 'localRepository' in Maven's settings file, if defined)

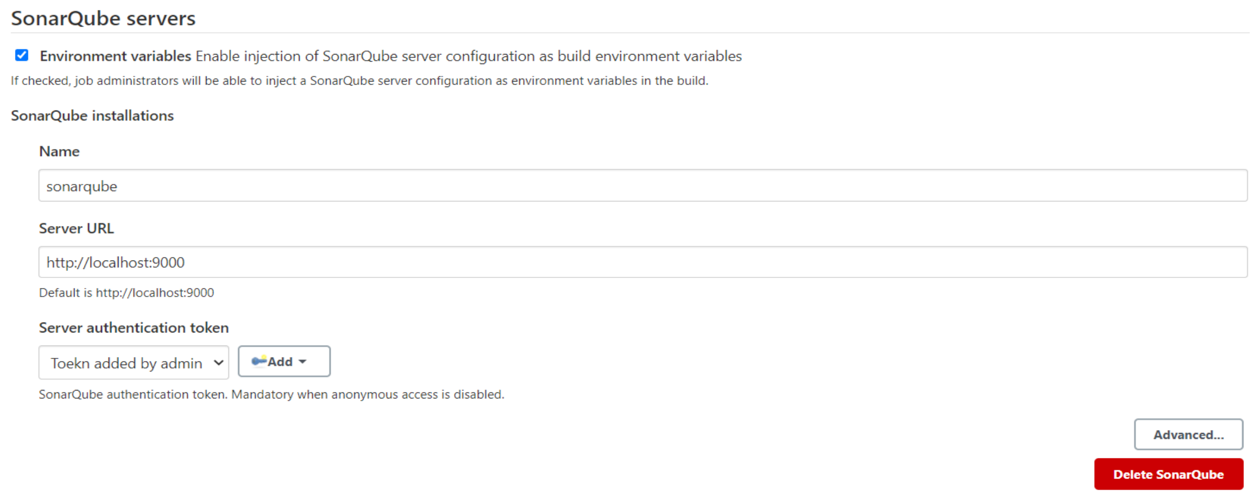
# of executors

2

Labels

Usage

Use this node as much as possible



SonarQube servers

☑ **Environment variables** Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

sonarqube

Server URL

http://localhost:9000

Default is http://localhost:9000

Server authentication token

Toekn added by admin

SonarQube authentication token. Mandatory when anonymous access is disabled.

Details of Configuration related to each stage explained later

Stage 1
Source Stage

Stage 1 – Source Stage

- This stage involves checking the git repository for any changes and configuring the pipeline by using suitable trigger mechanism to start the build.
- SCM tool Used – Gitlab
- Repository – Hotdesking Admin
- We can use multiple build triggers to start the pipeline. Some of them are as follows -

  1)Periodic Triggers

  2) Git Webhooks

  3) Git Polling

For this pipeline Git Polling is used.

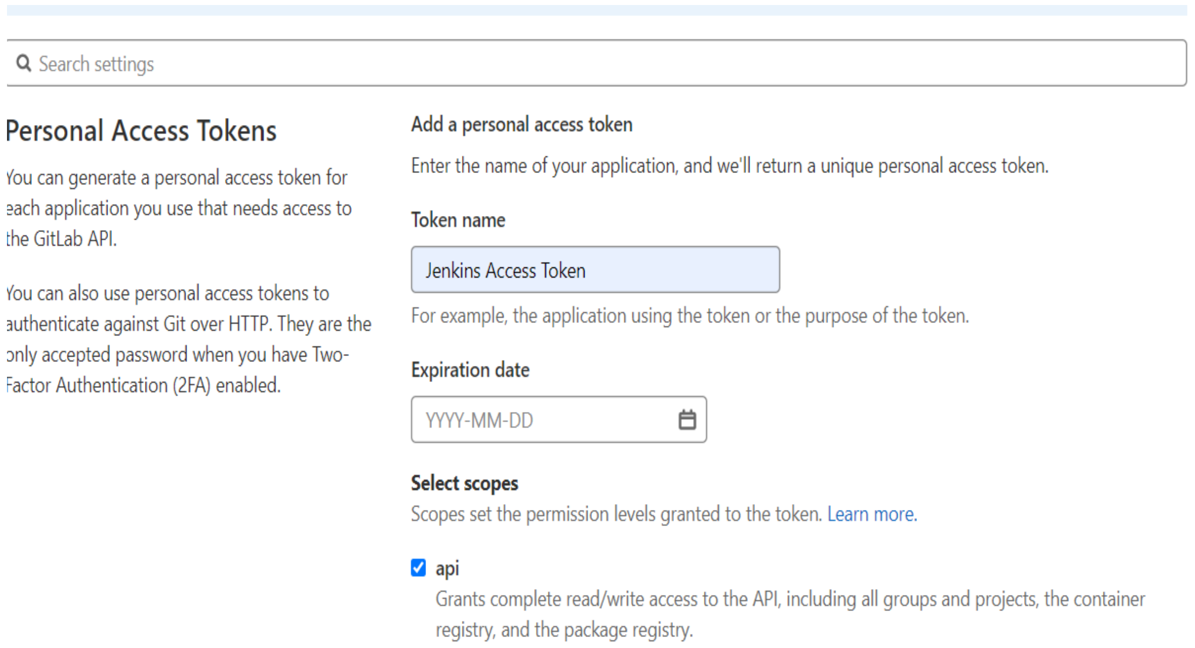
Git Polling - It is the trigger mechanism which involves checking the repository after certain interval of time and if any changes are made to the repository , triggering the execution of the pipeline.

Gitlab Configuration

- Creating a access token in gitlab
- Creating credential in Jenkins to access the gitlab repository.
- Creating Git SCM Stage in the pipeline

1) Creating Gitlab Access Token

- Inorder to have access to the private repository in our pipeline, we need to create a access token on the gitlab account.
- Go to your gitlab account -> User Settings -> Access Token
- Create a personal access token by providing name and scope as **api** and save the token somewhere securely.(This token will not be visible later).



Search settings

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Token name: Jenkins Access Token

For example, the application using the token or the purpose of the token.

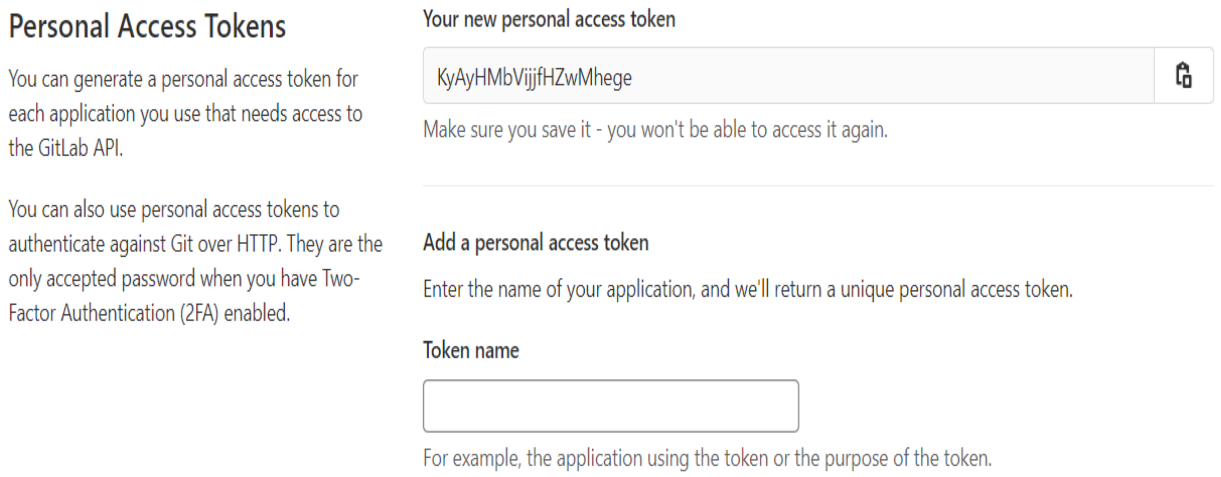
Expiration date: YYYY-MM-DD

Select scopes

Scopes set the permission levels granted to the token. Learn more.

[✓] api

Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.



Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Your new personal access token

KyAyHMbVijjfHZwMhege

Make sure you save it - you won't be able to access it again.

Add a personal access token

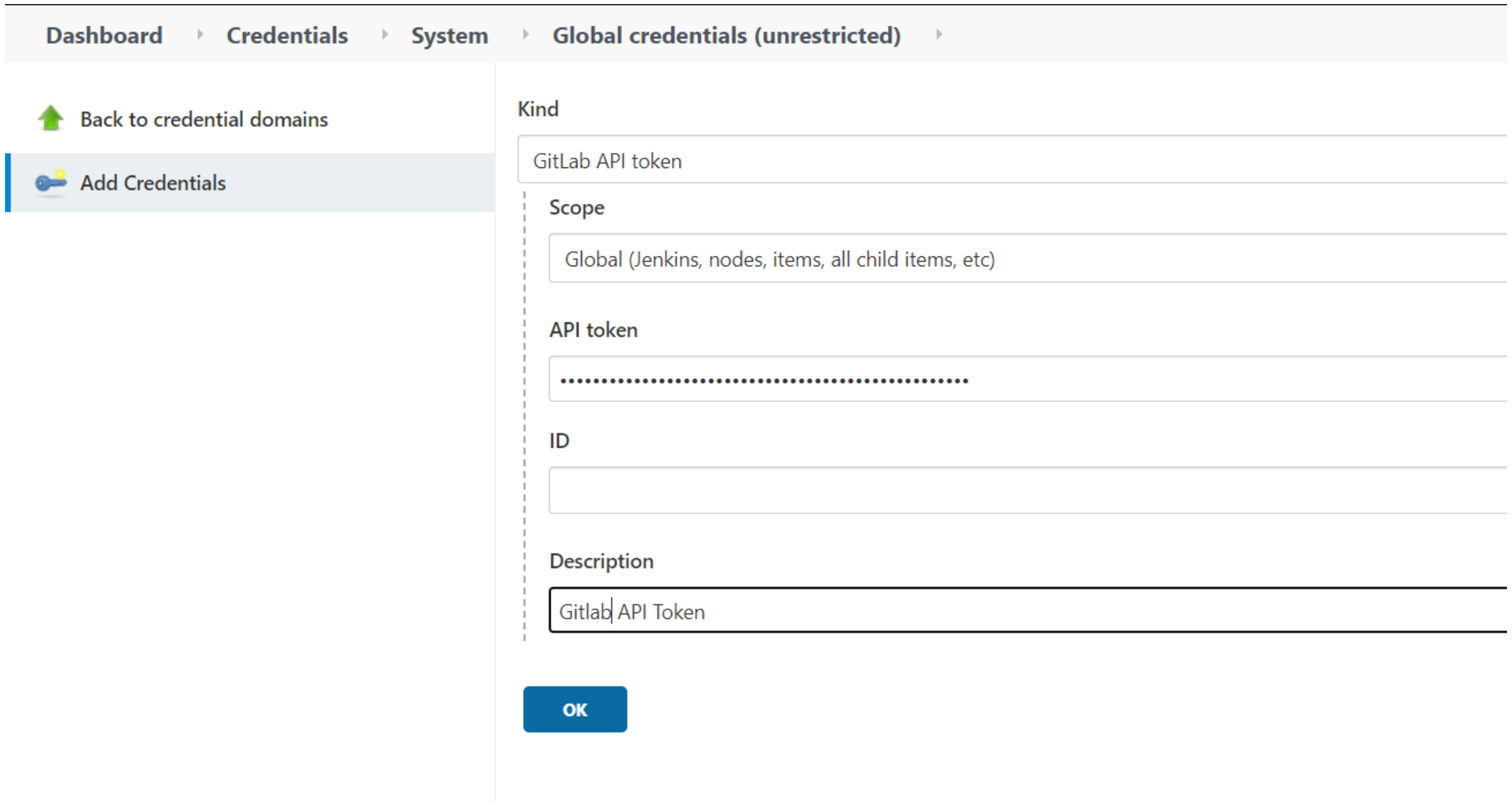
Enter the name of your application, and we'll return a unique personal access token.

Token name

For example, the application using the token or the purpose of the token.

2) Creating Credential for Access Token in Jenkins

Go to Manage Jenkins -> Manage Credentials -> Global Credentials -> Add Credential


Dashboard › Credentials › System › Global credentials (unrestricted) ›

Back to credential domains

Add Credentials

Kind: GitLab API token

Scope: Global (Jenkins, nodes, items, all child items, etc)

API token: ••

ID:

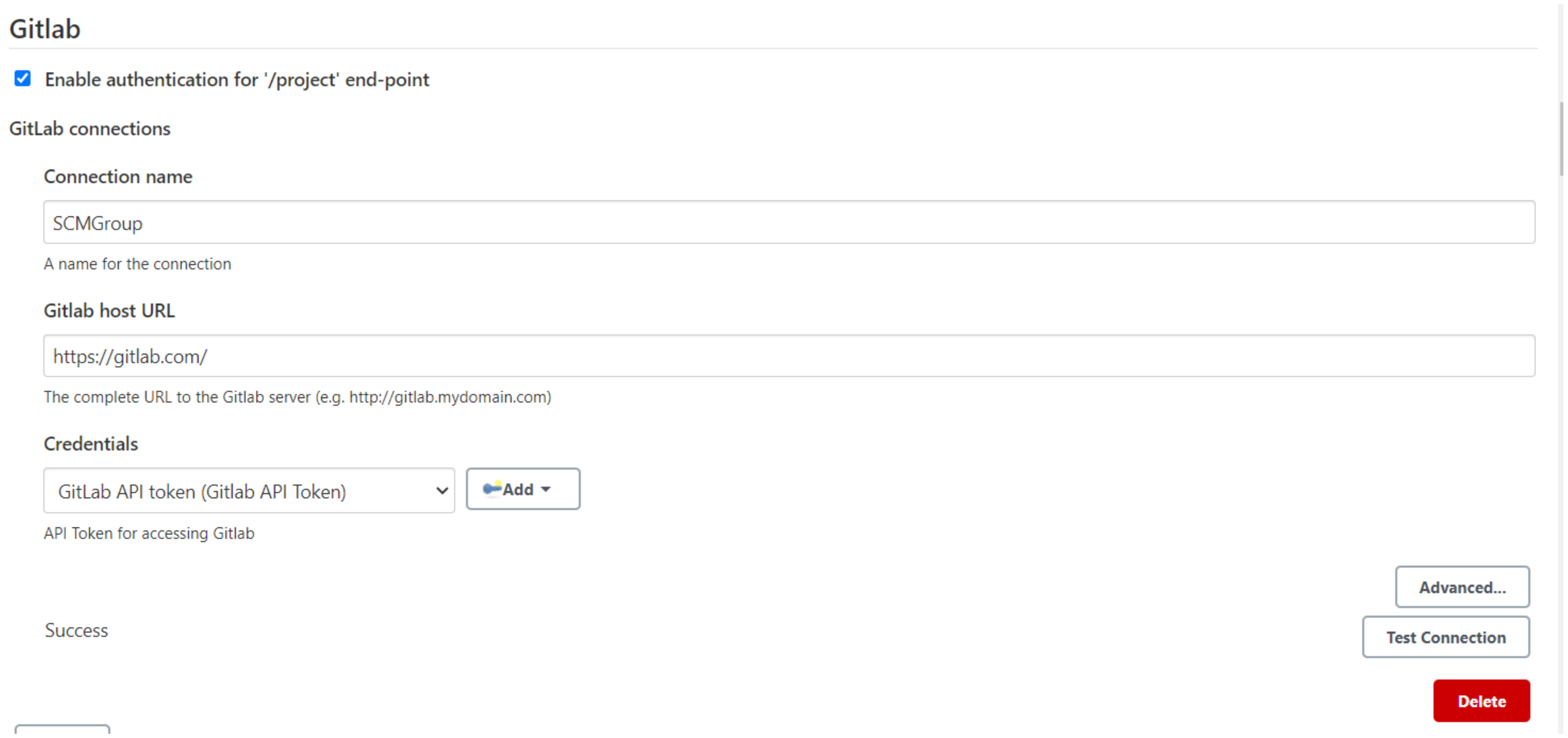
Description: Gitlab API Token

OK

Add Gitlab access token in configure system

Go to Manage Jenkins -> Configure System -> Gitlab

Add a connection with the host url and name , and select the access token we created in credentials. Test the connection, if the details are right , Success will be displayed.



Gitlab

☑ Enable authentication for '/project' end-point

GitLab connections

Connection name

SCMGroup

A name for the connection

Gitlab host URL

https://gitlab.com/

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials

GitLab API token (Gitlab API Token)

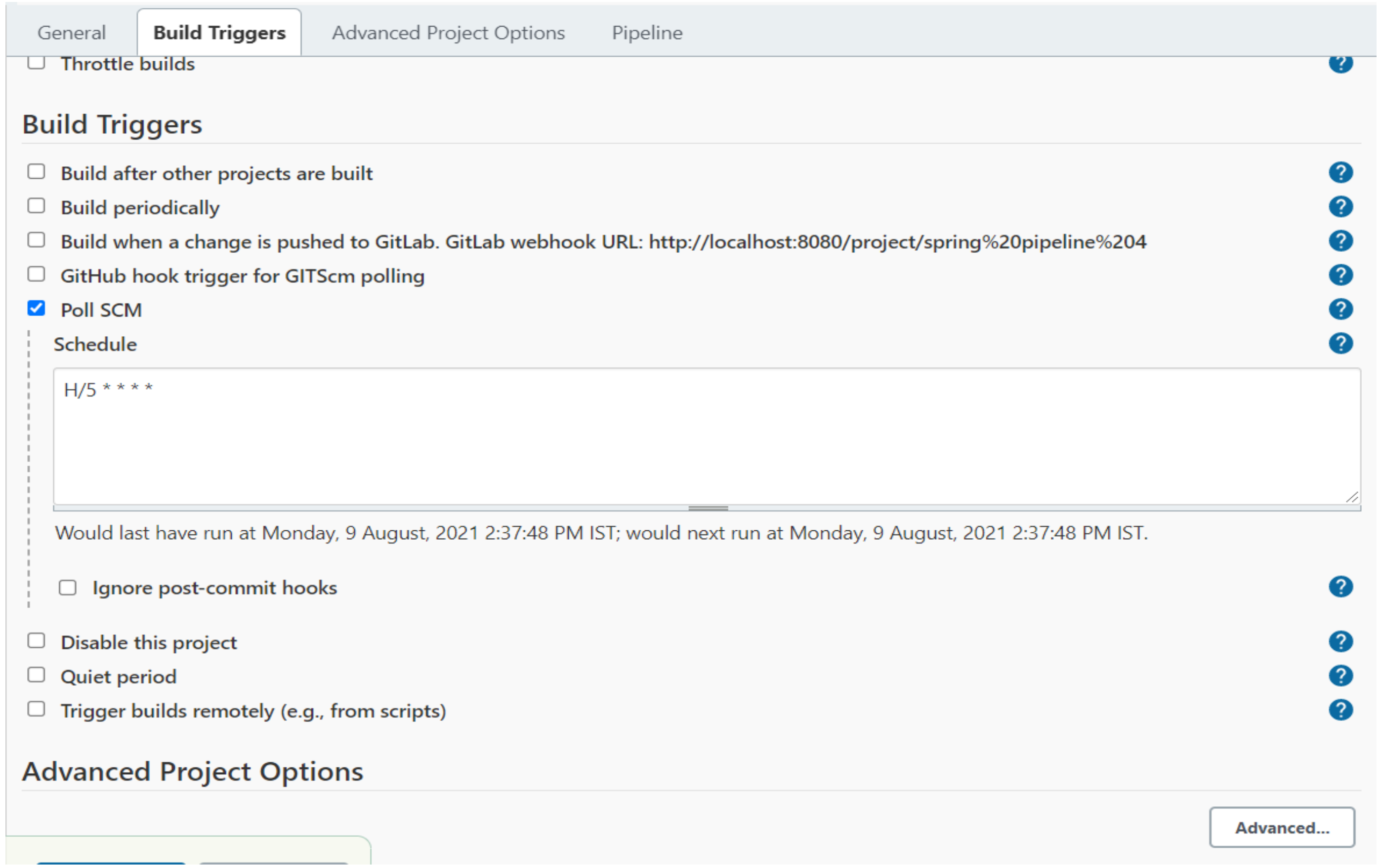
API Token for accessing Gitlab

Success

NOTE:
In case of Persistent Systems GitLab Repo, Host URL is: gitext.persistent.com

3) Create Git SCM Stage in pipeline

- Create a Pipeline Job in Jenkins by using Pipeline Script
- Add Poll SCM in the Build Job and mention the time (E.g Below it is 5 minutes). Jenkins uses cron-syntax to specify the time for polling.



☐ Throttle builds

Build Triggers

☐ Build after other projects are built
☐ Build periodically
☐ Build when a change is pushed to GitLab. GitLab webhook URL: http://localhost:8080/project/spring%20pipeline%204
☐ GitHub hook trigger for GITScm polling
☑ Poll SCM

Schedule

```
H/5 * * * *
```

Would last have run at Monday, 9 August, 2021 2:37:48 PM IST; would next run at Monday, 9 August, 2021 2:37:48 PM IST.

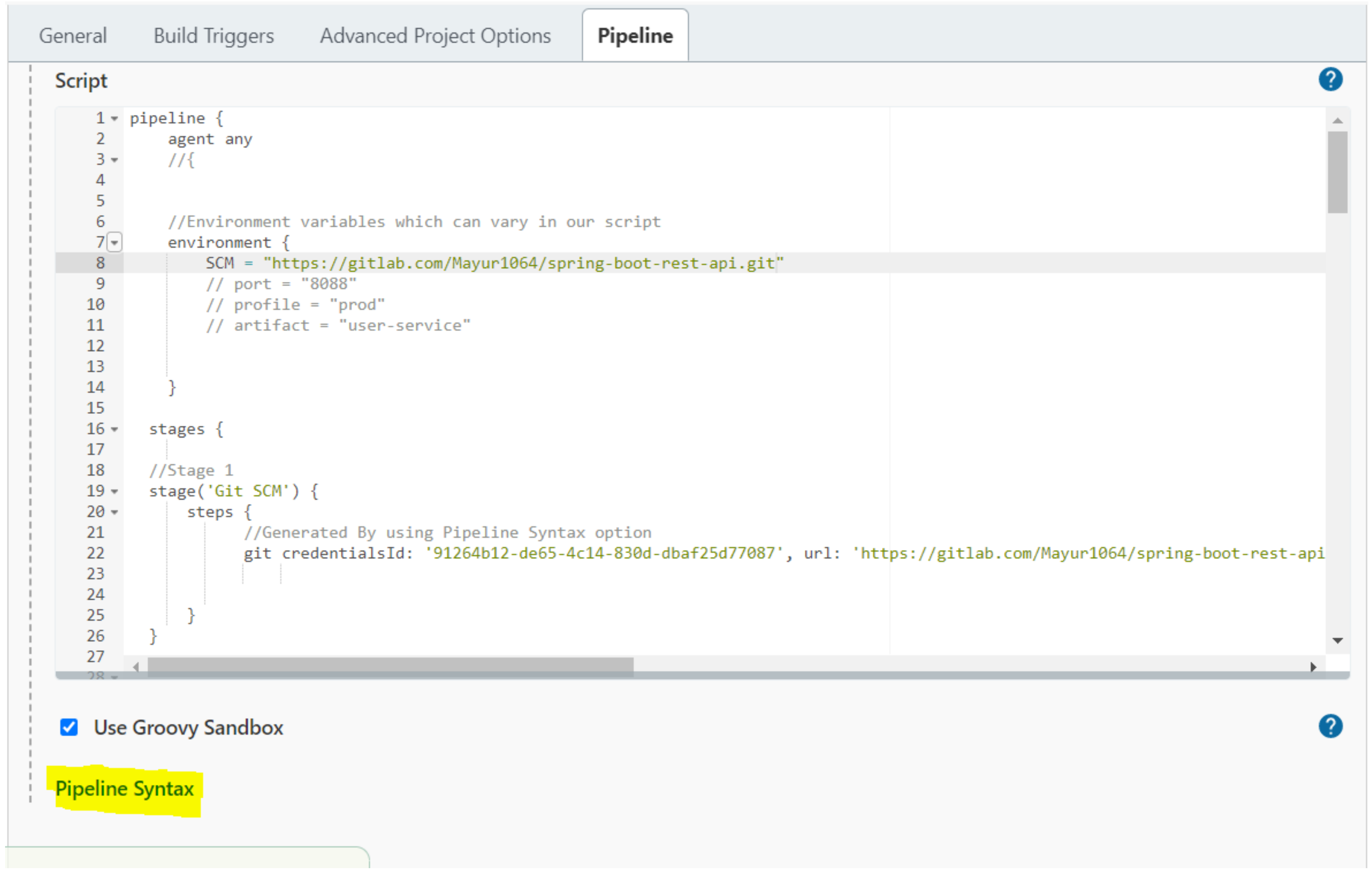
☐ Ignore post-commit hooks

☐ Disable this project
☐ Quiet period
☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Add SCM Stage in the pipeline

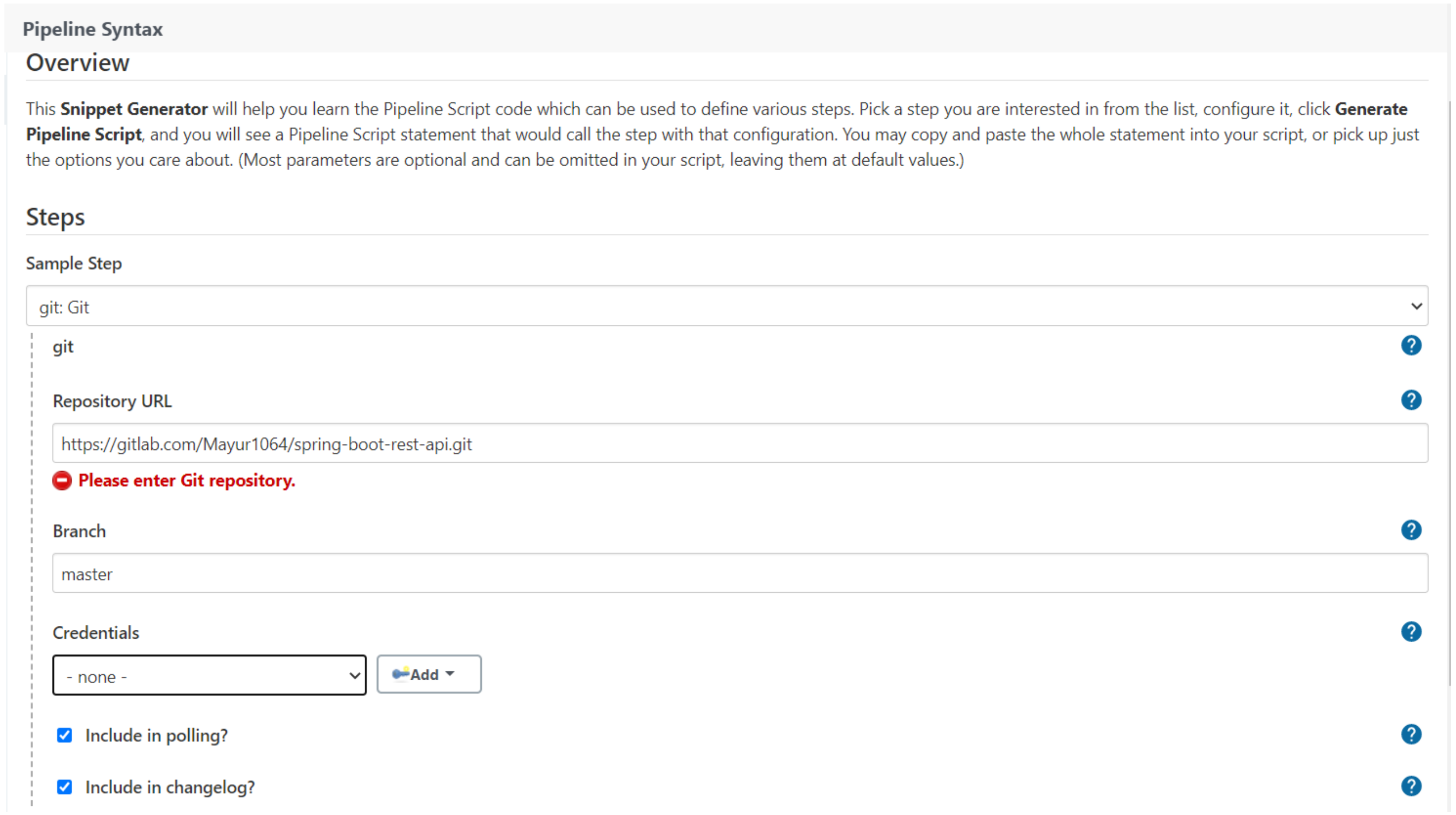
Use Pipeline Syntax option at bottom to generate pipeline script in required



```
pipeline {
    agent any
    //{


    //Environment variables which can vary in our script
    environment {
        SCM = "https://gitlab.com/Mayur1064/spring-boot-rest-api.git"
        // port = "8088"
        // profile = "prod"
        // artifact = "user-service"


    }

  stages {

  //Stage 1
  stage('Git SCM') {
      steps {
            //Generated By using Pipeline Syntax option
            git credentialsId: '91264b12-de65-4c14-830d-dbaf25d77087', url: 'https://gitlab.com/Mayur1064/spring-boot-rest-api


      }
  }

```

You can fork this repository for reference Click Here

Go to Pipeline Syntax option and specify the git repository url and branch name. Select Add -> Jenkins to create a credential



Pipeline Syntax

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: git: Git

git

Repository URL: https://gitlab.com/Mayur1064/spring-boot-rest-api.git

Please enter Git repository.

Branch: master

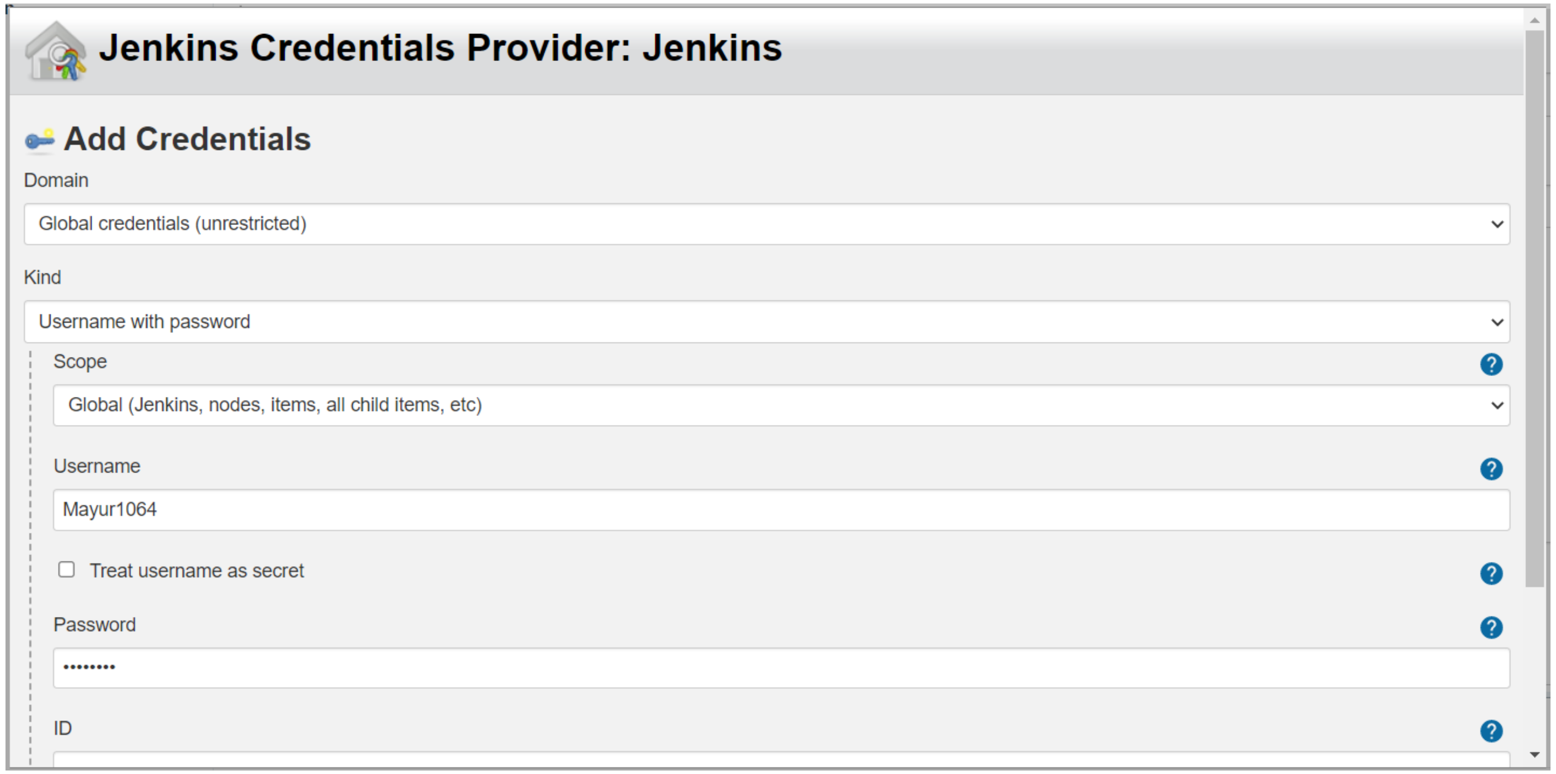
Credentials: - none -

☑ Include in polling?

☑ Include in changelog?

Create a credential for Gitlab access

Username and password Authentication



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: Mayur1064

☐ Treat username as secret

Password: ••••••••

ID

Note:
Don't try SSH with Username. Its blocked in case of Persistent Git repo. Stick to username and password mode of authentication

Provide the Gitlab username and password of your account and create a credential.

Using this credential generate the pipeline script and add it to the SCM stage of the pipeline

Stage 2
Build Stage

Compile Test Package Install

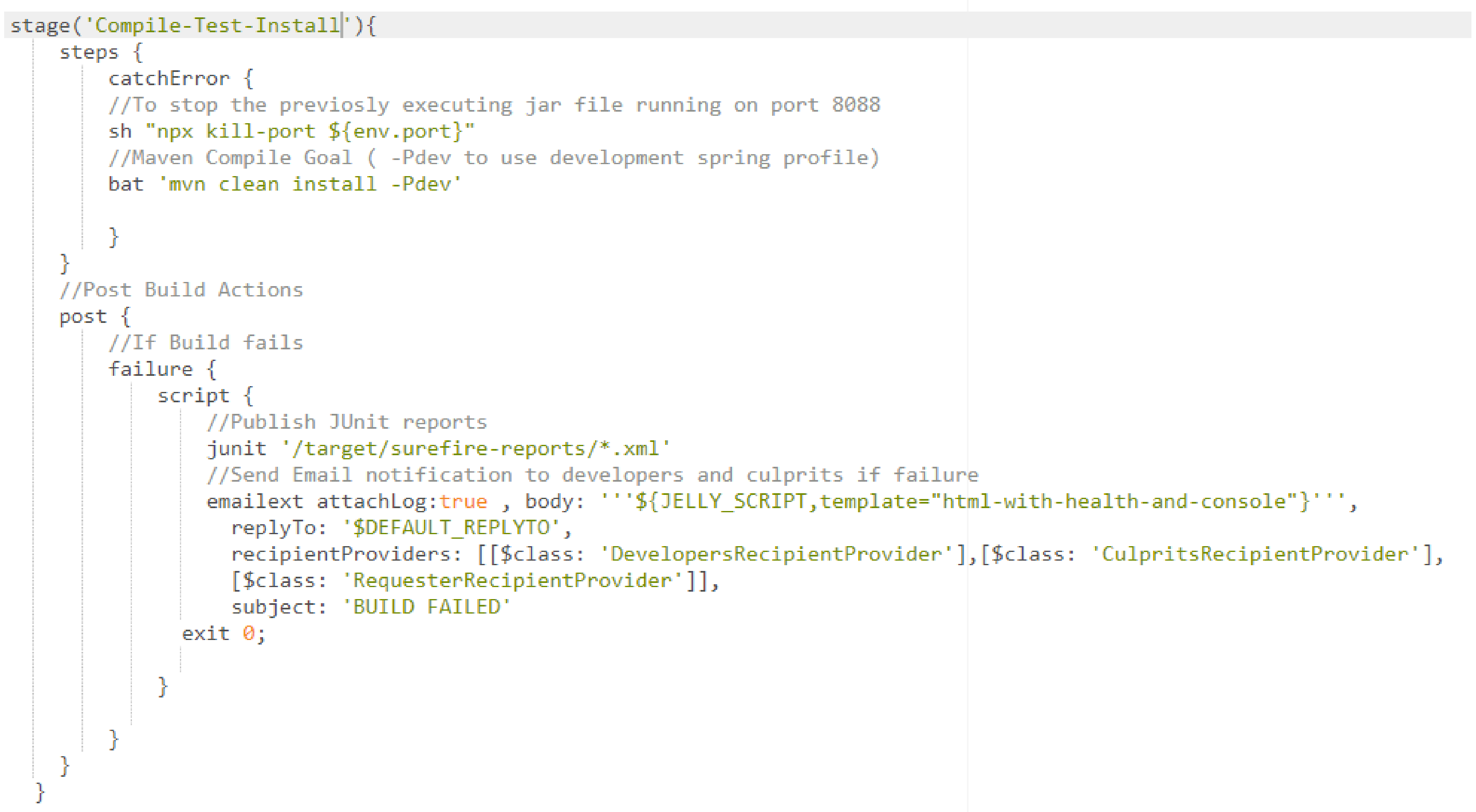
Build Stage

- This stage involves the execution of the phases in maven lifecycle.

  This includes
- Compiling the codebase to generate the necessary class files.
- Testing the code against Junit test cases defined by the developers/quality engineers.
- Packaging it to generate the necessary artifacts and executables(jar/war etc)
- Installing the package and the necessary plugins in local/remote maven repository.
- These phases can be executed in order by using single maven goal i.e **mvn install**

Compile – Test – Install Stage

Make assure maven is configured in Jenkins.

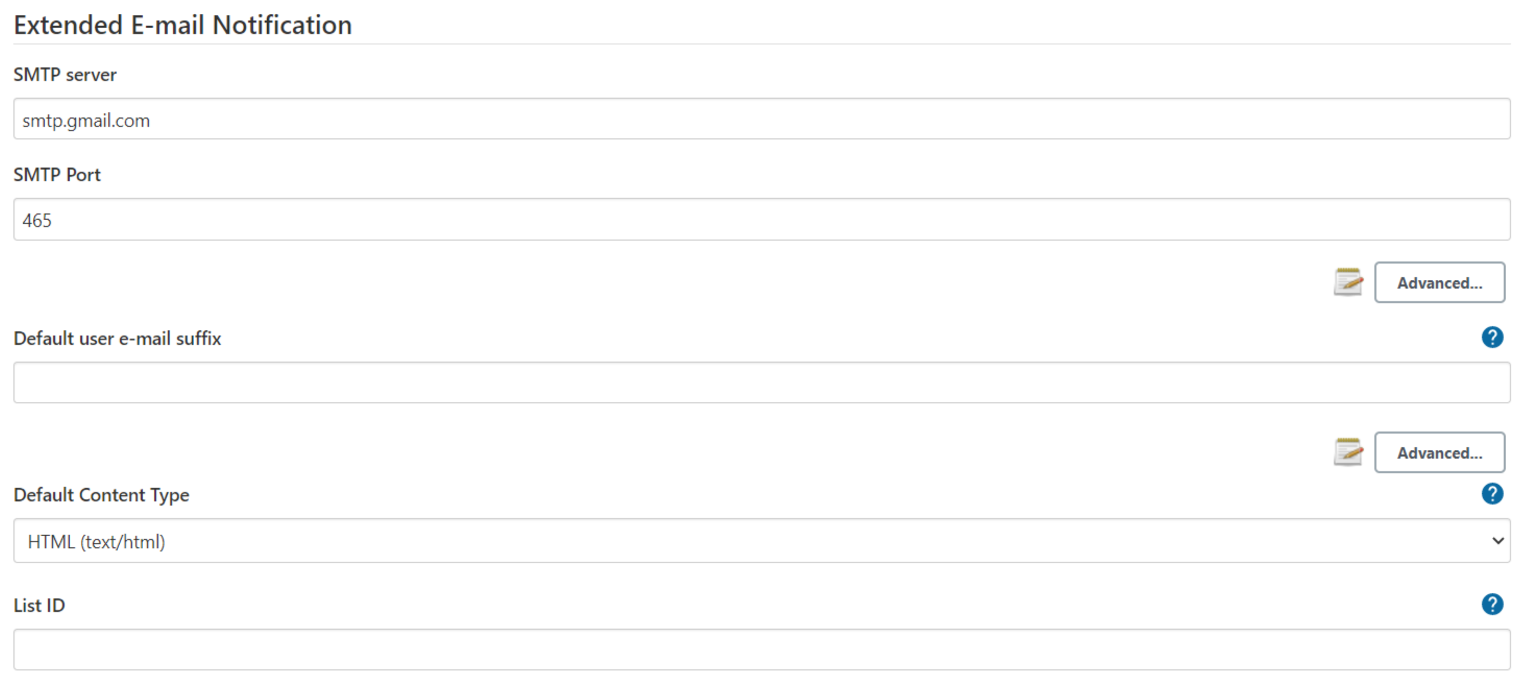


```
stage('Compile-Test-Install'){
    steps {
        catchError {
        //To stop the previosly executing jar file running on port 8088
        sh "npx kill-port ${env.port}"
        //Maven Compile Goal ( -Pdev to use development spring profile)
        bat 'mvn clean install -Pdev'

        }
    }
    //Post Build Actions
    post {
        //If Build fails
        failure {
            script {
                //Publish JUnit reports
                junit '/target/surefire-reports/*.xml'
                //Send Email notification to developers and culprits if failure
                emailext attachLog:true , body: '''${JELLY_SCRIPT,template="html-with-health-and-console"}''',
                  replyTo: '$DEFAULT_REPLYTO',
                  recipientProviders: [[$class: 'DevelopersRecipientProvider'],[$class: 'CulpritsRecipientProvider'],
                  [$class: 'RequesterRecipientProvider']],
                  subject: 'BUILD FAILED'
              exit 0;

            }

        }
    }
  }
```

More on Spring Profiles Later

Extended Email Notifications

- This plugin is used to send editable email notifications to the developer , culprits or any other recipients in case of build failure.
- Make sure Email extension plugin is installed and configured
- Go to Manage Jenkins -> Configure System -> Extended Email Notification
- Below is the configuration to send emails from Gmail SMTP server using SSL



Extended E-mail Notification

SMTP server: smtp.gmail.com

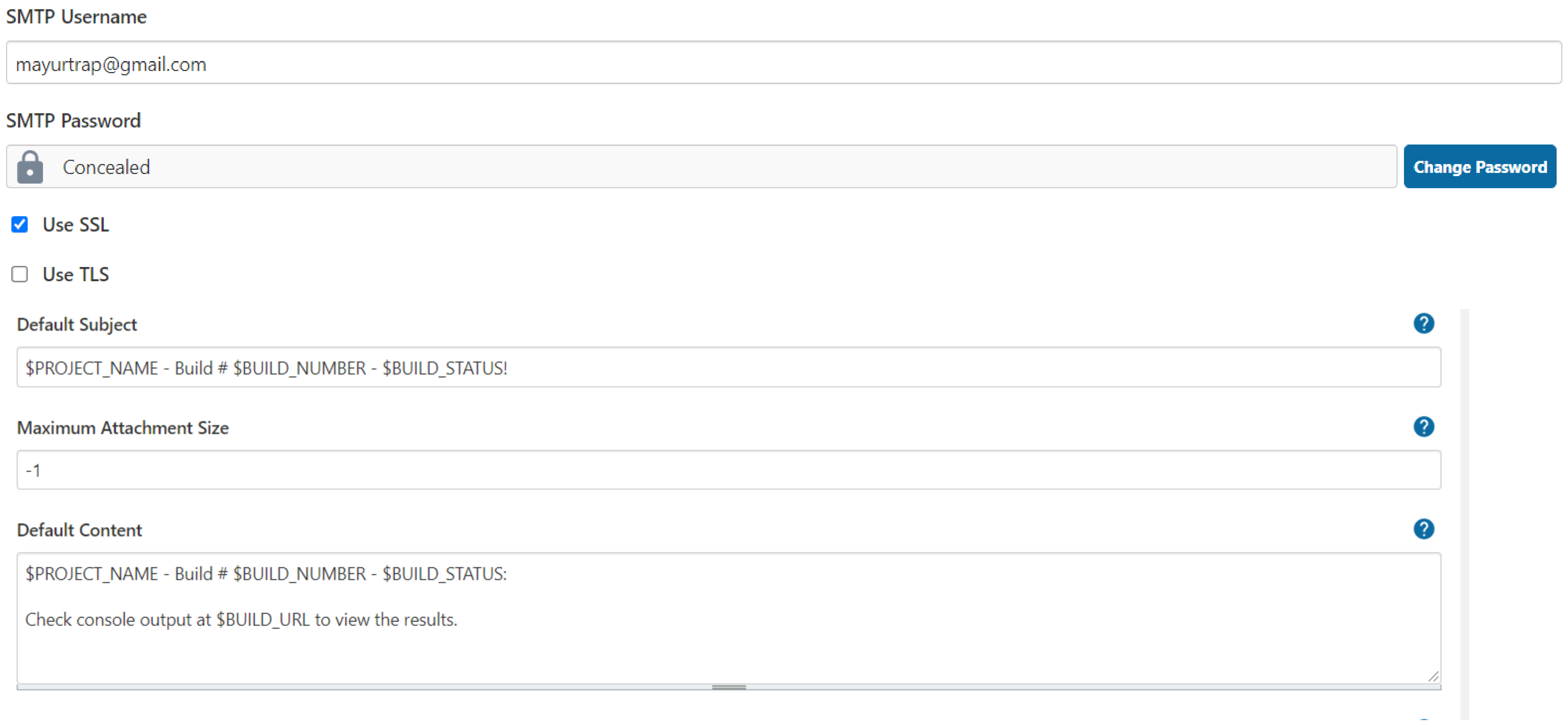
SMTP Port: 465

Default user e-mail suffix:

Default Content Type: HTML (text/html)

List ID:

Provide Username and password for the gmail account and make sure less secured apps is enabled for your google account. SMTP port for SSL is 465



SMTP Username

mayurtrap@gmail.com

SMTP Password

Concealed

[✓] Use SSL

[ ] Use TLS

Default Subject

$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS!

Maximum Attachment Size

-1

Default Content

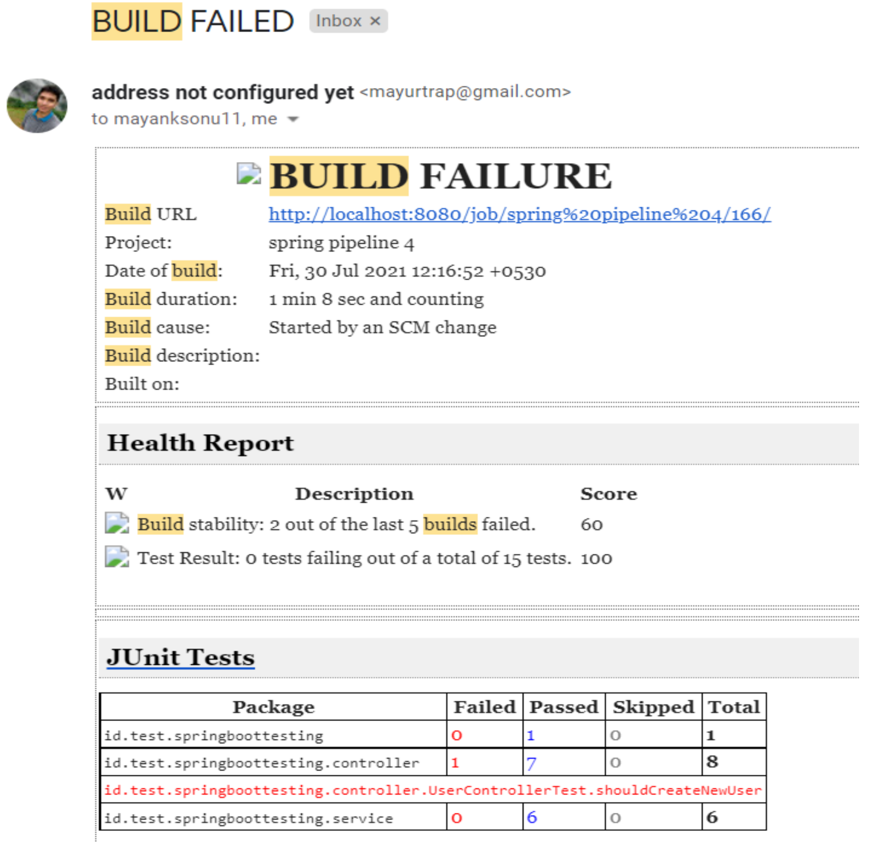
$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS:

Check console output at $BUILD_URL to view the results.

Email Configuration for persistent mail server later

Email Templates for Notification

- There are multiple templates we can use to make the email notification more intuitive. Some of them can be found below. These templates are written either using jelly or groovy script.
- Click Here   Save these templates under **email-templates** directory in the Jenkins Home.
- The template used for this pipeline is html-with-health-and-console and slight variations in it for different stages. It is rendered in the email as below –



BUILD FAILED Inbox ×

address not configured yet <mayurtrap@gmail.com>
to mayanksonu11, me

BUILD FAILURE

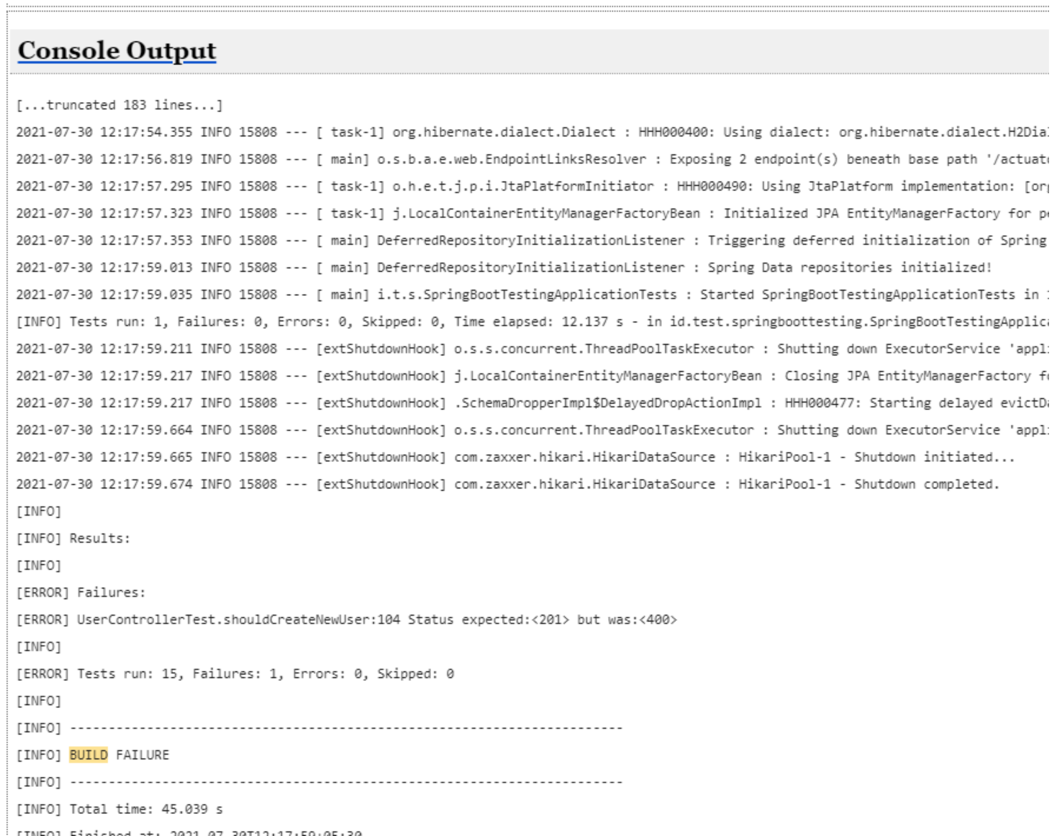
Build URL	http://localhost:8080/job/spring%20pipeline%204/166/
Project:	spring pipeline 4
Date of build:	Fri, 30 Jul 2021 12:16:52 +0530
Build duration:	1 min 8 sec and counting
Build cause:	Started by an SCM change
Build description:	
Built on:	

Health Report

W	Description	Score
	Build stability: 2 out of the last 5 builds failed.	60
	Test Result: 0 tests failing out of a total of 15 tests.	100

JUnit Tests

Package	Failed	Passed	Skipped	Total
id.test.springboottesting	0	1	0	1
id.test.springboottesting.controller	1	7	0	8
id.test.springboottesting.controller.UserControllerTest.shouldCreateNewUser				
id.test.springboottesting.service	0	6	0	6

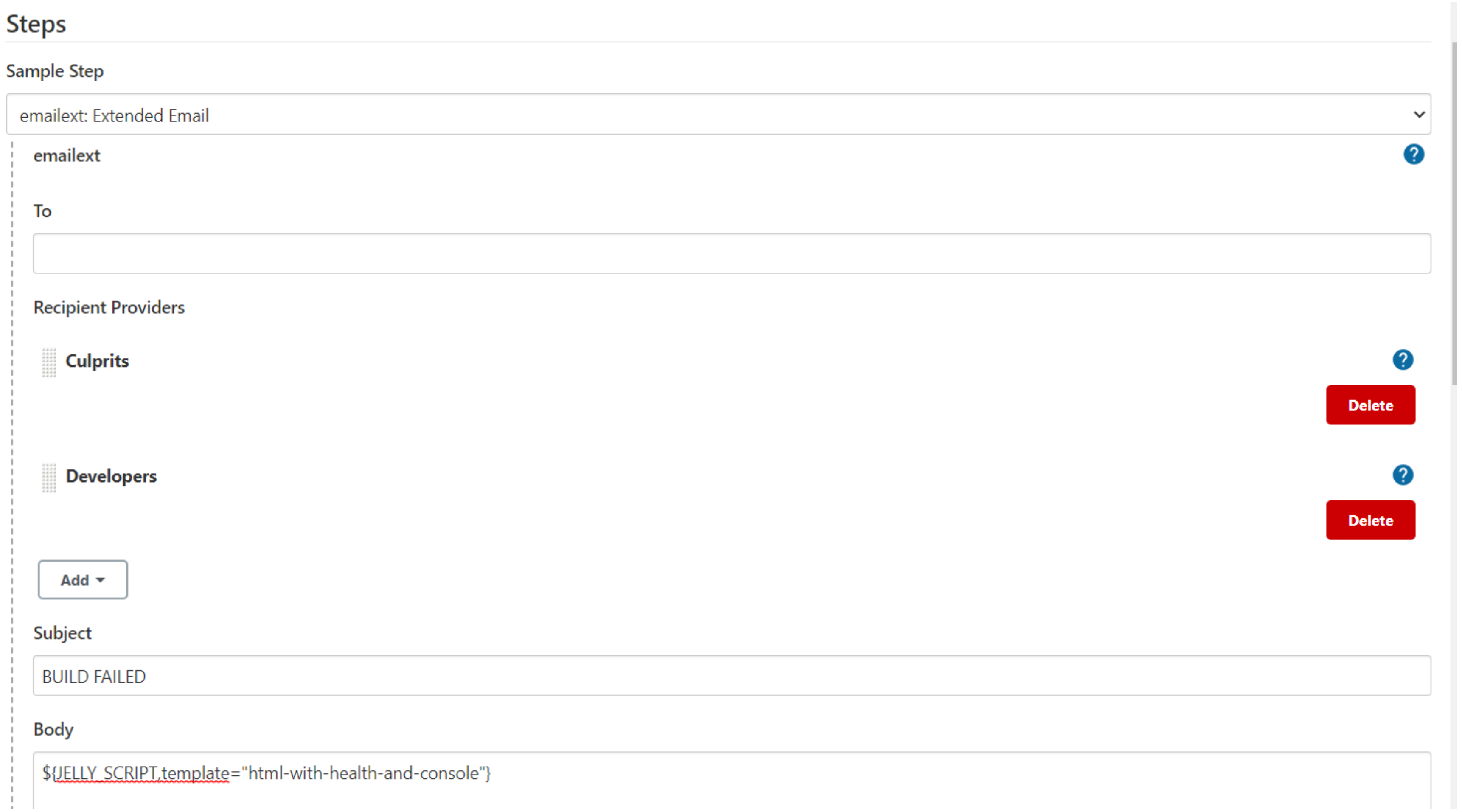


Console Output

```
[...truncated 183 lines...]
2021-07-30 12:17:54.355 INFO 15808 --- [ task-1] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.H2Dia
2021-07-30 12:17:56.819 INFO 15808 --- [ main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuat
2021-07-30 12:17:57.295 INFO 15808 --- [ task-1] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [or
2021-07-30 12:17:57.323 INFO 15808 --- [ task-1] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for p
2021-07-30 12:17:57.353 INFO 15808 --- [ main] DeferredRepositoryInitializationListener : Triggering deferred initialization of Spring
2021-07-30 12:17:59.013 INFO 15808 --- [ main] DeferredRepositoryInitializationListener : Spring Data repositories initialized!
2021-07-30 12:17:59.035 INFO 15808 --- [ main] i.t.s.SpringBootTestingApplicationTests : Started SpringBootTestingApplicationTests in
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 12.137 s - in id.test.springboottesting.SpringBootTestingApplic
2021-07-30 12:17:59.211 INFO 15808 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'appl
2021-07-30 12:17:59.217 INFO 15808 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2021-07-30 12:17:59.217 INFO 15808 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Starting delayed evictD
2021-07-30 12:17:59.664 INFO 15808 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'appl
2021-07-30 12:17:59.665 INFO 15808 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2021-07-30 12:17:59.674 INFO 15808 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
[INFO]
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR] UserControllerTest.shouldCreateNewUser:104 Status expected:<201> but was:<400>
[INFO]
[ERROR] Tests run: 15, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 45.039 s
[INFO] Finished at: 2021-07-30T12:17:59+05:30
```

Adding Email Notification as a post build action

By using the Pipeline Syntax option generate script for extended email and add it as a post build action to the build stage


Steps

Sample Step

emailext: Extended Email

emailext

To

Recipient Providers

Culprits

Delete

Developers

Delete

Add

Subject

BUILD FAILED

Body

```
${JELLY_SCRIPT,template="html-with-health-and-console"}
```

Add Developers and Culprits as the recipients and the jelly script template in the body

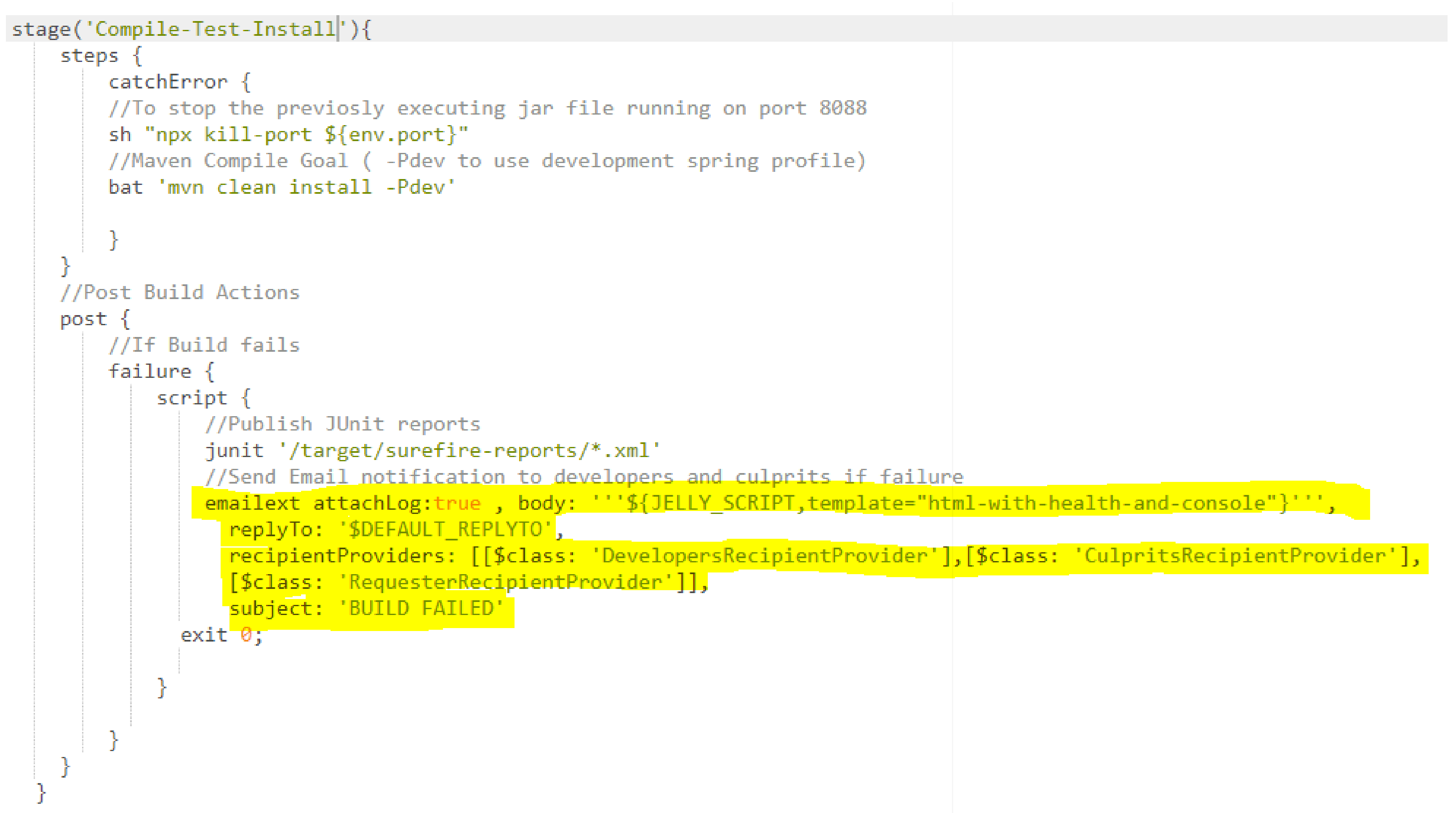
Who are Developers and Culprits in Recipients ?

- Developers - Sends email to anyone who checked in code for the last build. The developer whose commit triggered the pipeline comes in this category.

- Culprits - Sends email to the list of users who committed a change since the last non-broken build till now. This list at least always include people who made changes in this build, but if the previous build was a failure it also includes the culprit list from there.

- For email notification to work for these categories , the developers should have their email address configured in the git config.

Post Build Action Added to the Build Stage

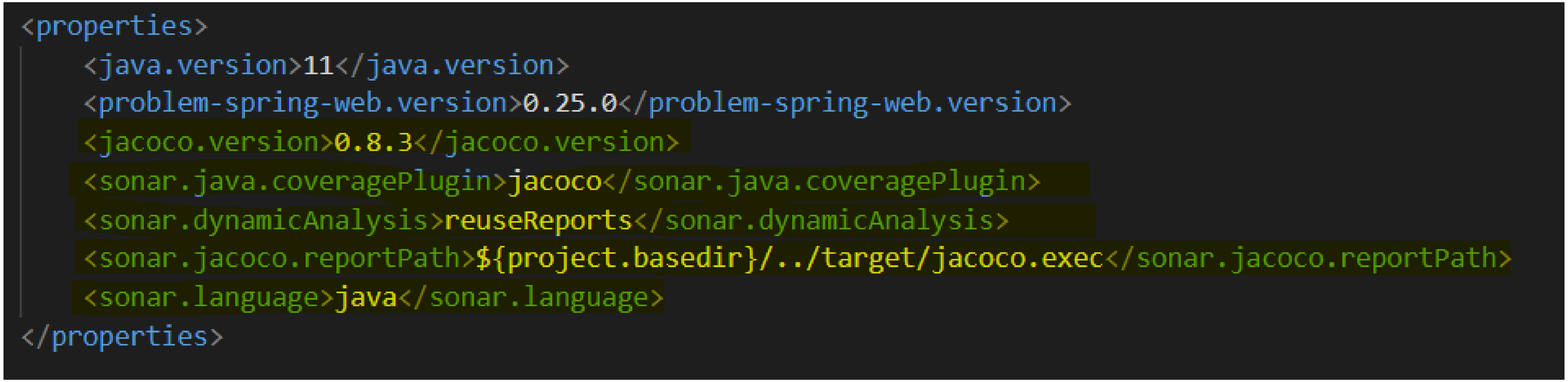


```
stage('Compile-Test-Install'){
    steps {
        catchError {
        //To stop the previosly executing jar file running on port 8088
        sh "npx kill-port ${env.port}"
        //Maven Compile Goal ( -Pdev to use development spring profile)
        bat 'mvn clean install -Pdev'

        }
    }
    //Post Build Actions
    post {
        //If Build fails
        failure {
            script {
                //Publish JUnit reports
                junit '/target/surefire-reports/*.xml'
                //Send Email notification to developers and culprits if failure
                emailext attachLog:true , body: '''${JELLY_SCRIPT,template="html-with-health-and-console"}''',
                  replyTo: '$DEFAULT_REPLYTO',
                  recipientProviders: [[$class: 'DevelopersRecipientProvider'],[$class: 'CulpritsRecipientProvider'],
                  [$class: 'RequesterRecipientProvider']],
                  subject: 'BUILD FAILED'
               exit 0;

            }

        }
    }
 }
```

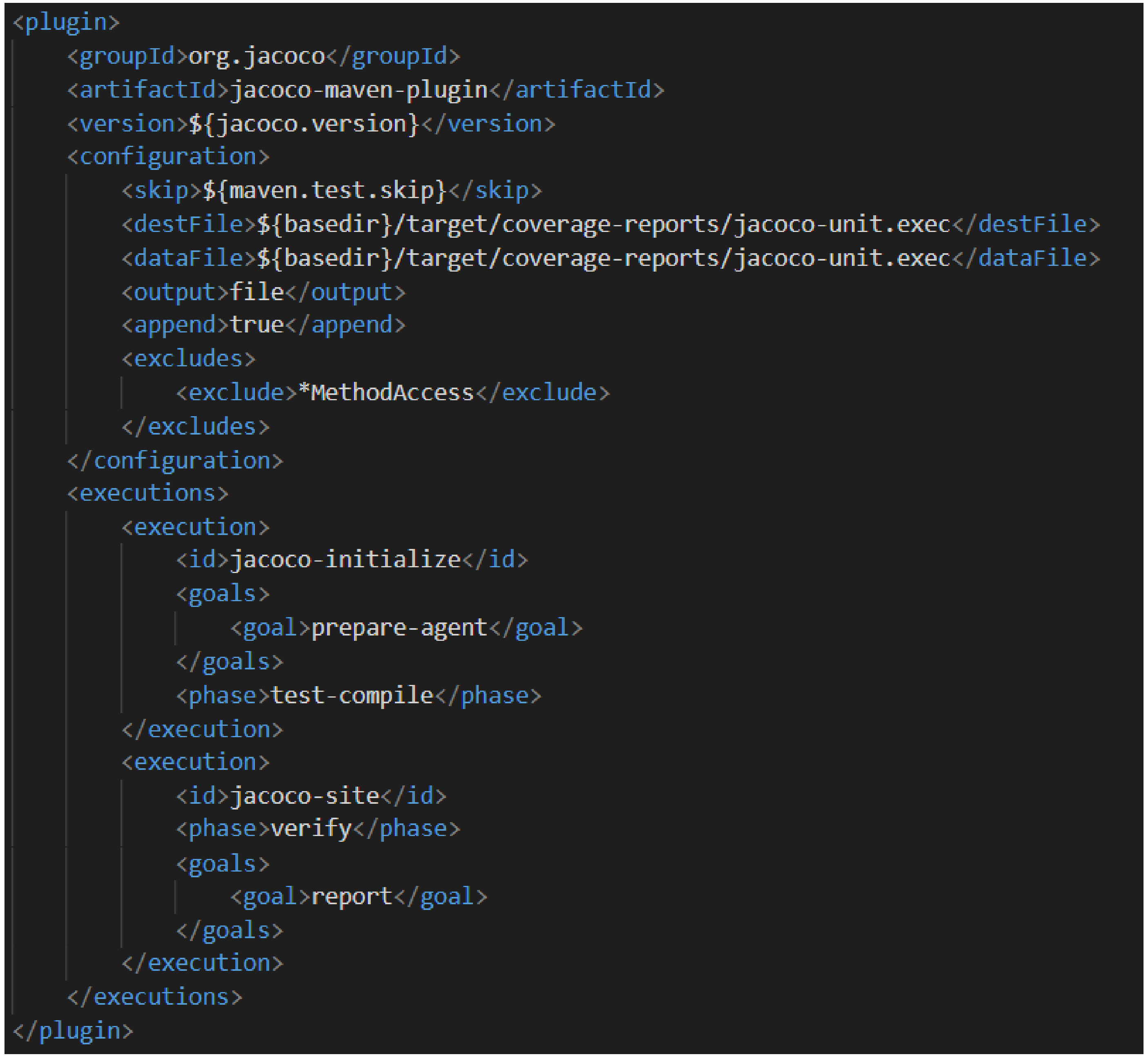
Code Coverage and JaCoCo Plugin

- **Code Coverage** - It is a measure of how much of the application’s code has been executed in testing. Essentially, it's a metric that many teams use to check the quality of their tests, as it represents the percentage of the production code that has been tested and executed.
- This gives development teams reassurance that their programs have been broadly tested for bugs and should be relatively error-free.
- In order to generate the coverage reports during the sonarqube analysis in stage 3 , **jacoco plugin** must be included in the pom.xml of the application before **mvn install**.
- **JaCoCo** is a free code coverage library for Java.
- Include the appropriate version of JaCoCo Plugin in pom.xml. **Jacoco 0.8.3** is used in this project.

Including JaCoCo Plugin in POM



```xml
<properties>
    <java.version>11</java.version>
    <problem-spring-web.version>0.25.0</problem-spring-web.version>
    <jacoco.version>0.8.3</jacoco.version>
    <sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
    <sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>
    <sonar.jacoco.reportPath>${project.basedir}/../target/jacoco.exec</sonar.jacoco.reportPath>
    <sonar.language>java</sonar.language>
</properties>
```


```xml
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>${jacoco.version}</version>
    <configuration>
        <skip>${maven.test.skip}</skip>
        <destFile>${basedir}/target/coverage-reports/jacoco-unit.exec</destFile>
        <dataFile>${basedir}/target/coverage-reports/jacoco-unit.exec</dataFile>
        <output>file</output>
        <append>true</append>
        <excludes>
            <exclude>*MethodAccess</exclude>
        </excludes>
    </configuration>
    <executions>
        <execution>
            <id>jacoco-initialize</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
            <phase>test-compile</phase>
        </execution>
        <execution>
            <id>jacoco-site</id>
            <phase>verify</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Stage Result

- If the Build Stage is successful , target directory will be generated consisting of the class files, JUnit reports and the artifacts (executable jar file)
- If the Build Stage fails , an email notification will be sent to the developer and culprits specifying the issue (compilation failure, test failure etc) along with JUnit reports, build logs attached and the pipeline will be aborted.

Stage 3
Static Code Analysis

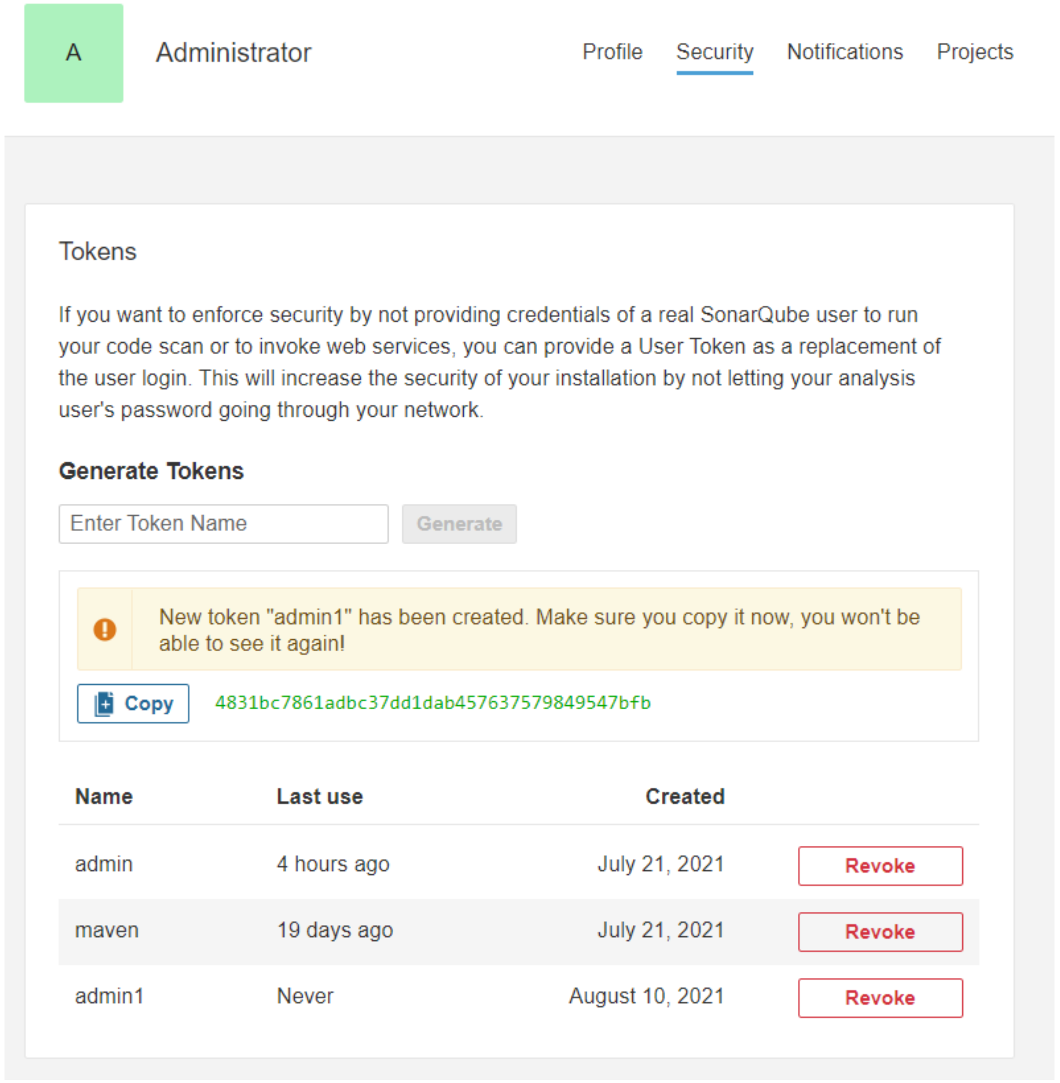
SonarQube Analysis

What is SonarQube ?

- **Static Code Analysis** – Involves analysing the code base against parameters like code coverage , vulnerabilities , bugs etc by enforcing quality gates to detect issues like programming errors, coding standard violations, syntax violations, security vulnerabilities in the early stage**.**
- **SonarQube** is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
- It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.
- Make sure that the sonarqube is installed , configured in Jenkins and running on the default port (port 9000) as mentioned in the installation section.

Create a Access Token in SonarQube

Go to sonarqube account -> security -> Tokens. Generate a new Token and save it.



Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

New token "admin1" has been created. Make sure you copy it now, you won't be able to see it again!

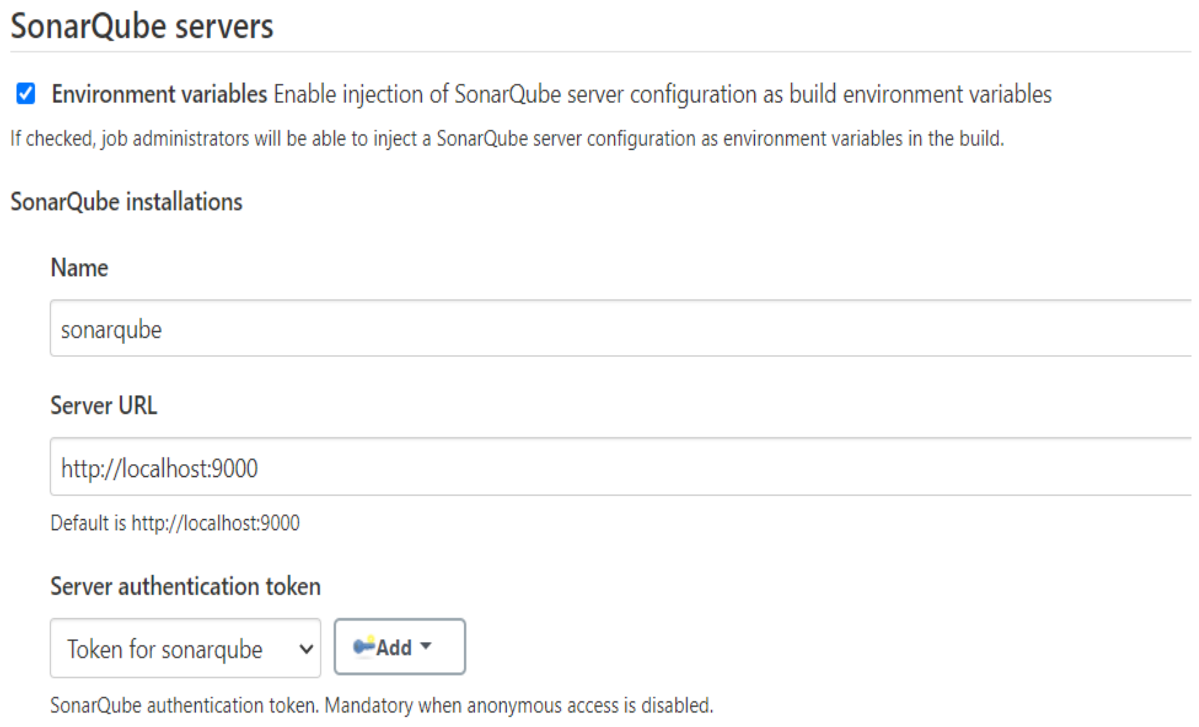
```
4831bc7861adbc37dd1dab457637579849547bfb
```

Name	Last use	Created	
admin	4 hours ago	July 21, 2021	Revoke
maven	19 days ago	July 21, 2021	Revoke
admin1	Never	August 10, 2021	Revoke

Copy and Save the Token somewhere

Add Sonarqube server and access token in Configure System

Go to Configure System -> SonarQube server. Specify the sonar server url and add the access token to the server authentication token as a secret text.



SonarQube servers

☑ **Environment variables** Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

sonarqube

Server URL

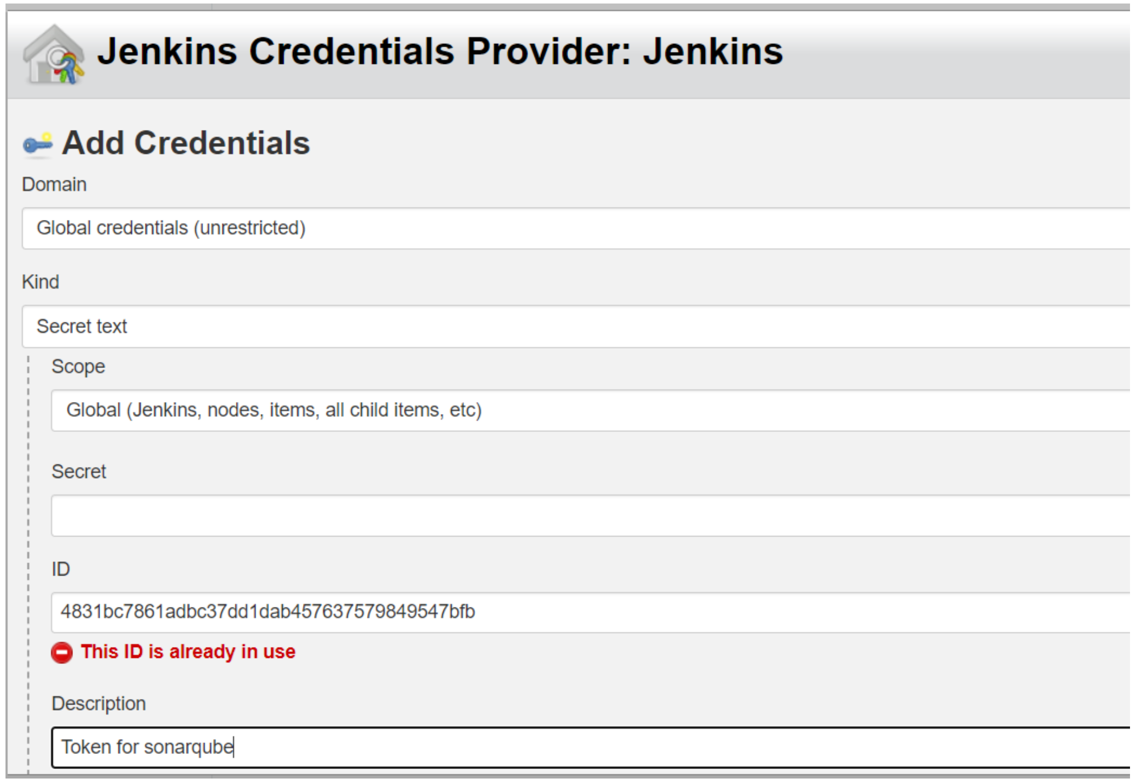
http://localhost:9000

Default is http://localhost:9000

Server authentication token

Token for sonarqube | Add

SonarQube authentication token. Mandatory when anonymous access is disabled.



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Secret text

Scope

Global (Jenkins, nodes, items, all child items, etc)

Secret

ID

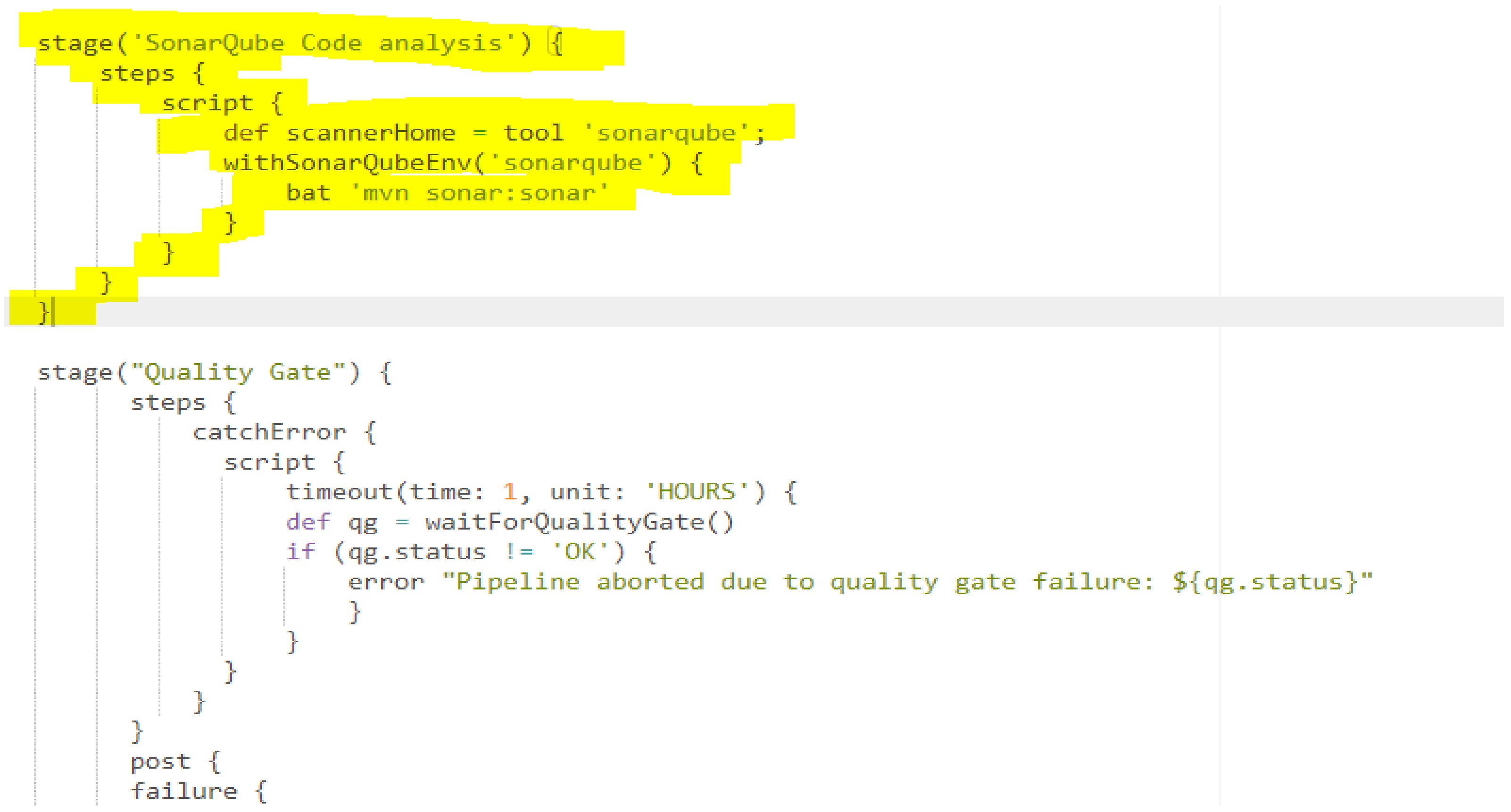
4831bc7861adbc37dd1dab457637579849547bfb

This ID is already in use

Description

Token for sonarqube

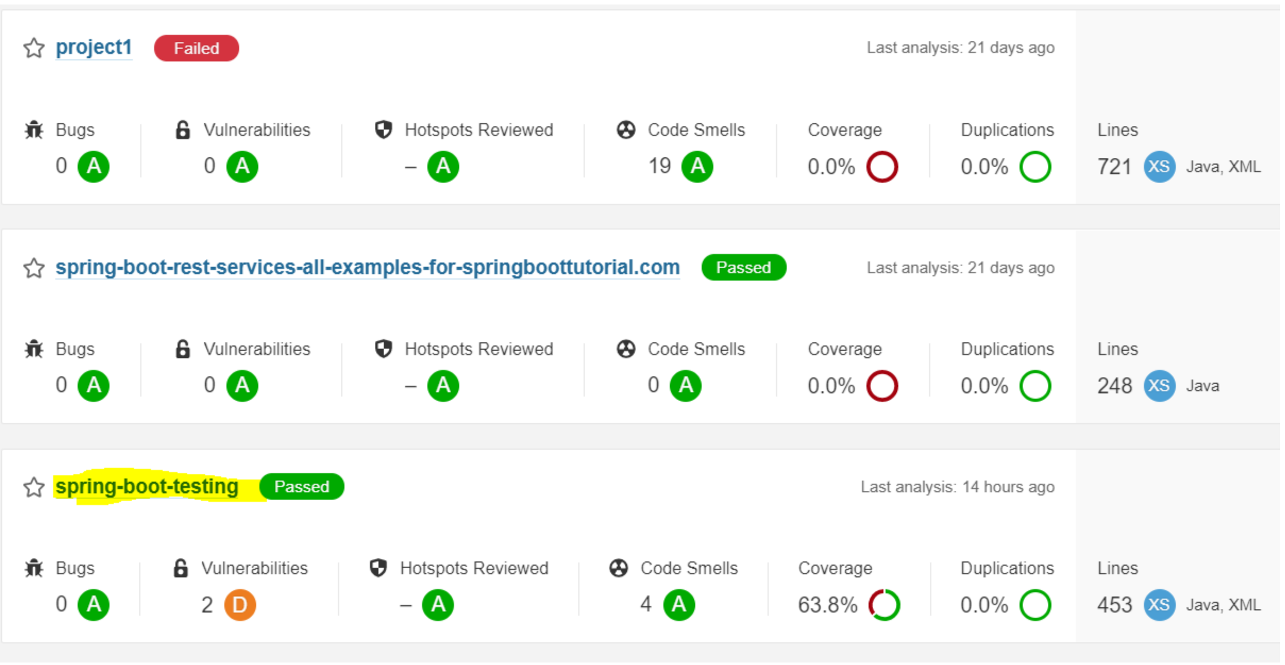
Adding the Sonarqube Stage in the pipeline



```
stage('SonarQube Code analysis') {
    steps {
        script {
            def scannerHome = tool 'sonarqube';
            withSonarQubeEnv('sonarqube') {
                bat 'mvn sonar:sonar'
            }
        }
    }
}

stage("Quality Gate") {
      steps {
          catchError {
            script {
                timeout(time: 1, unit: 'HOURS') {
                def qg = waitForQualityGate()
                if (qg.status != 'OK') {
                    error "Pipeline aborted due to quality gate failure: ${qg.status}"
                    }
                }
            }
          }
      }
      post {
      failure {
```

This will generate the sonar analysis of the codebase along with code coverage if JaCoCo plugin is configured. This Analysis can be viewed in SonarQube Interface.
(A separate directory will be created in target named coverage-reports)

Analysis generated in SonarQube



☆ project1 Failed — Last analysis: 21 days ago

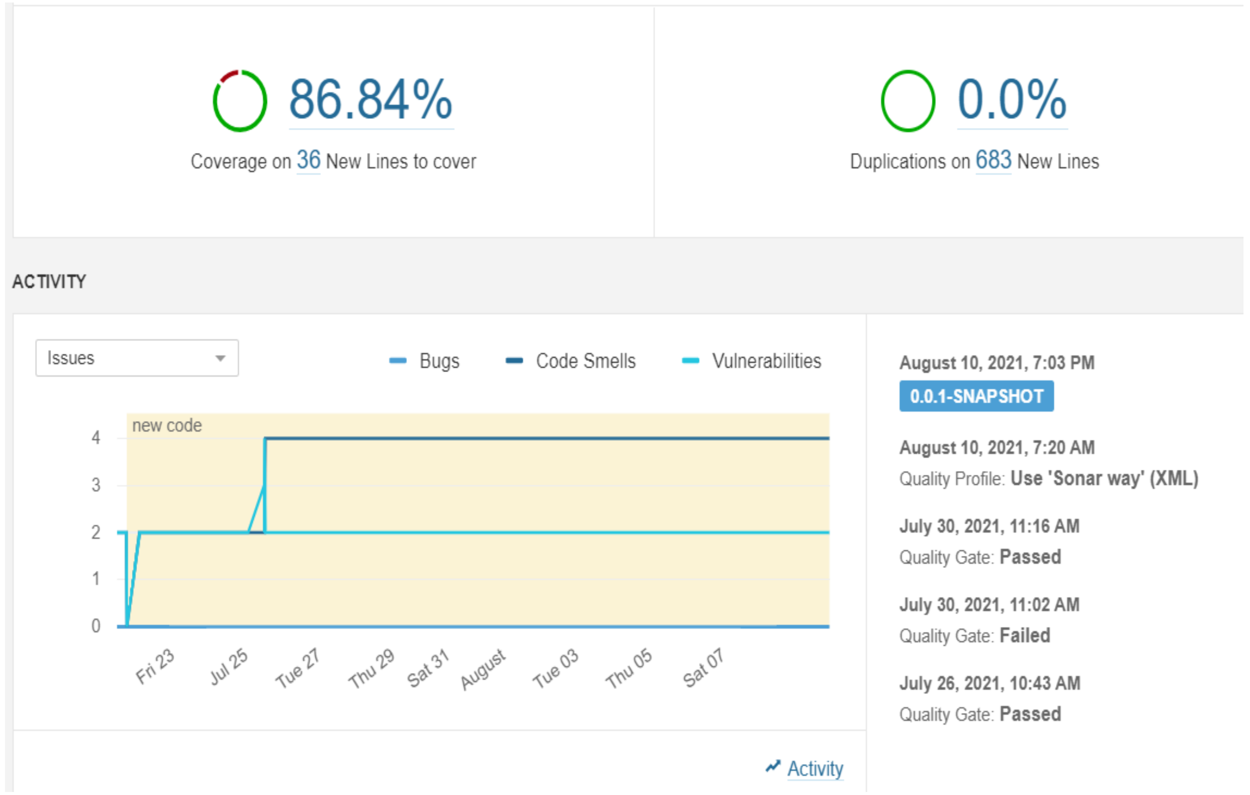
Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	19 A	0.0%	0.0%	721 XS Java, XML

☆ spring-boot-rest-services-all-examples-for-springboottutorial.com Passed — Last analysis: 21 days ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	0 A	0.0%	0.0%	248 XS Java

☆ spring-boot-testing Passed — Last analysis: 14 hours ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	2 D	– A	4 A	63.8%	0.0%	453 XS Java, XML



86.84%
Coverage on 36 New Lines to cover

0.0%
Duplications on 683 New Lines

ACTIVITY

August 10, 2021, 7:03 PM
0.0.1-SNAPSHOT

August 10, 2021, 7:20 AM
Quality Profile: Use 'Sonar way' (XML)

July 30, 2021, 11:16 AM
Quality Gate: Passed

July 30, 2021, 11:02 AM
Quality Gate: Failed

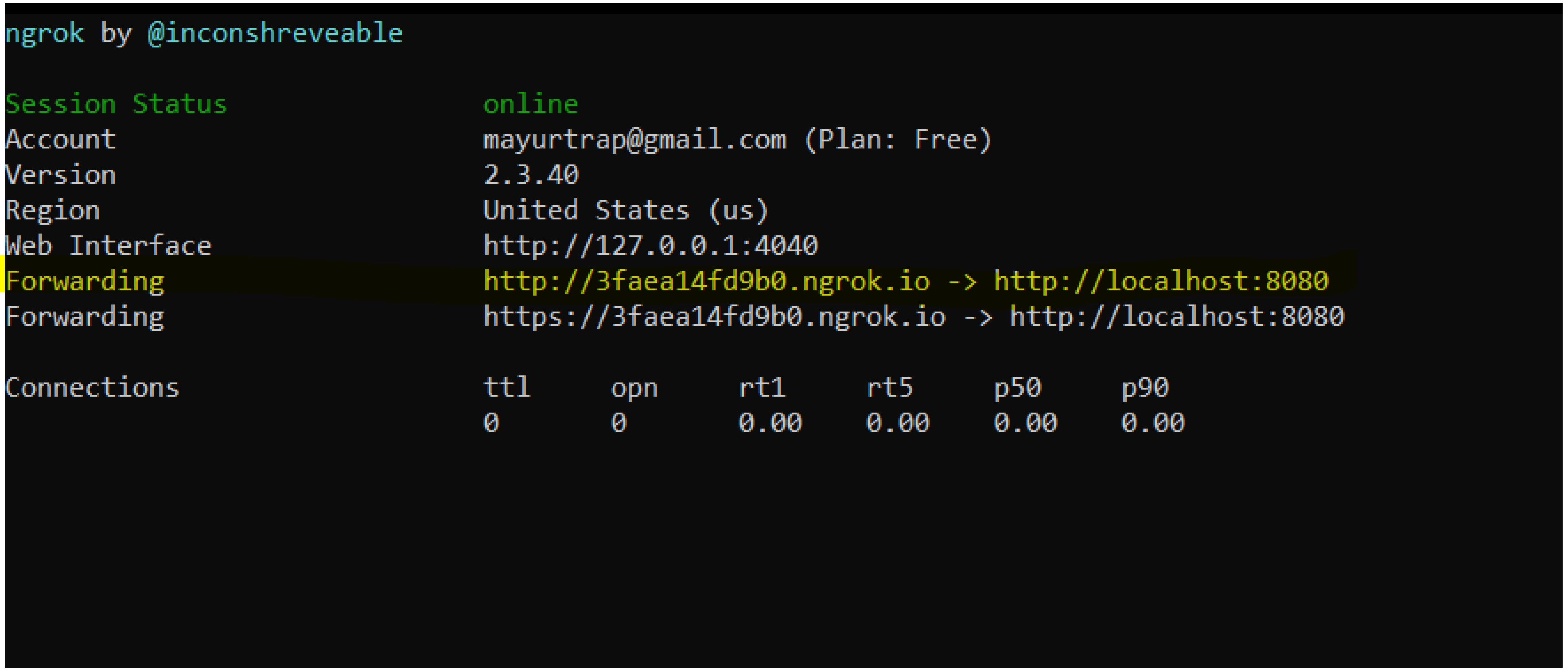
July 26, 2021, 10:43 AM
Quality Gate: Passed

Note - spring-boot-testing is the name of the application used for this demo.

Introduction to Ngrok

- Before moving ahead to creating custom quality gate for our project, lets get familiar with ngrok.
- For adding a quality gate in our Jenkins pipeline, we have to create a webhook for the Jenkins and add that to the project in sonarqube.
- Webhooks are not supported for the localhost urls , so we need to use ngrok for the local development purpose.
- **Ngrok** is a cross-platform application that enables developers to expose a local development server to the Internet with minimal effort.
- By using this , we can create https/http urls for a specific port on internet which will be forwarded to the localhost.

Download the Ngrok application and run the following command where it is downloaded.
Command - **ngrok http 8080** (8080 is the port on which Jenkins is running)



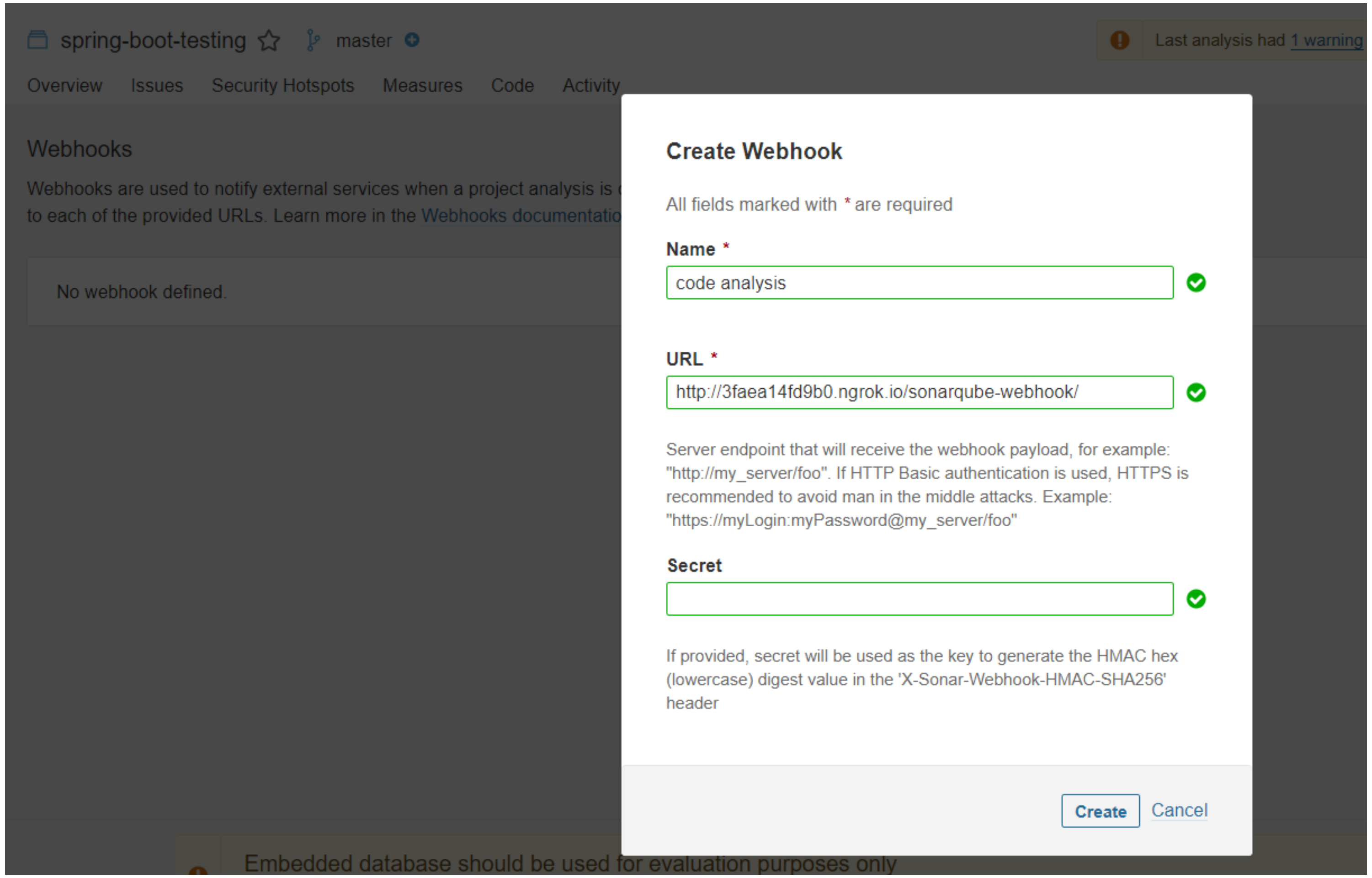
```
ngrok by @inconshreveable

Session Status                online
Account                       mayurtrap@gmail.com (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://3faea14fd9b0.ngrok.io -> http://localhost:8080
Forwarding                    https://3faea14fd9b0.ngrok.io -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

The http url which is forwarded to the localhost will be used to create the webhook for the quality gate in sonarqube

Creating Webhook for SonarQube Project

In your project in SonarQube Go to Overview -> Project Settings -> Webhooks.

Create a new webhook by providing the name and the url



Webhooks

Webhooks are used to notify external services when a project analysis is

to each of the provided URLs. Learn more in the Webhooks documentatio

No webhook defined.

Create Webhook

All fields marked with * are required

Name *

code analysis

URL *

http://3faea14fd9b0.ngrok.io/sonarqube-webhook/

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header

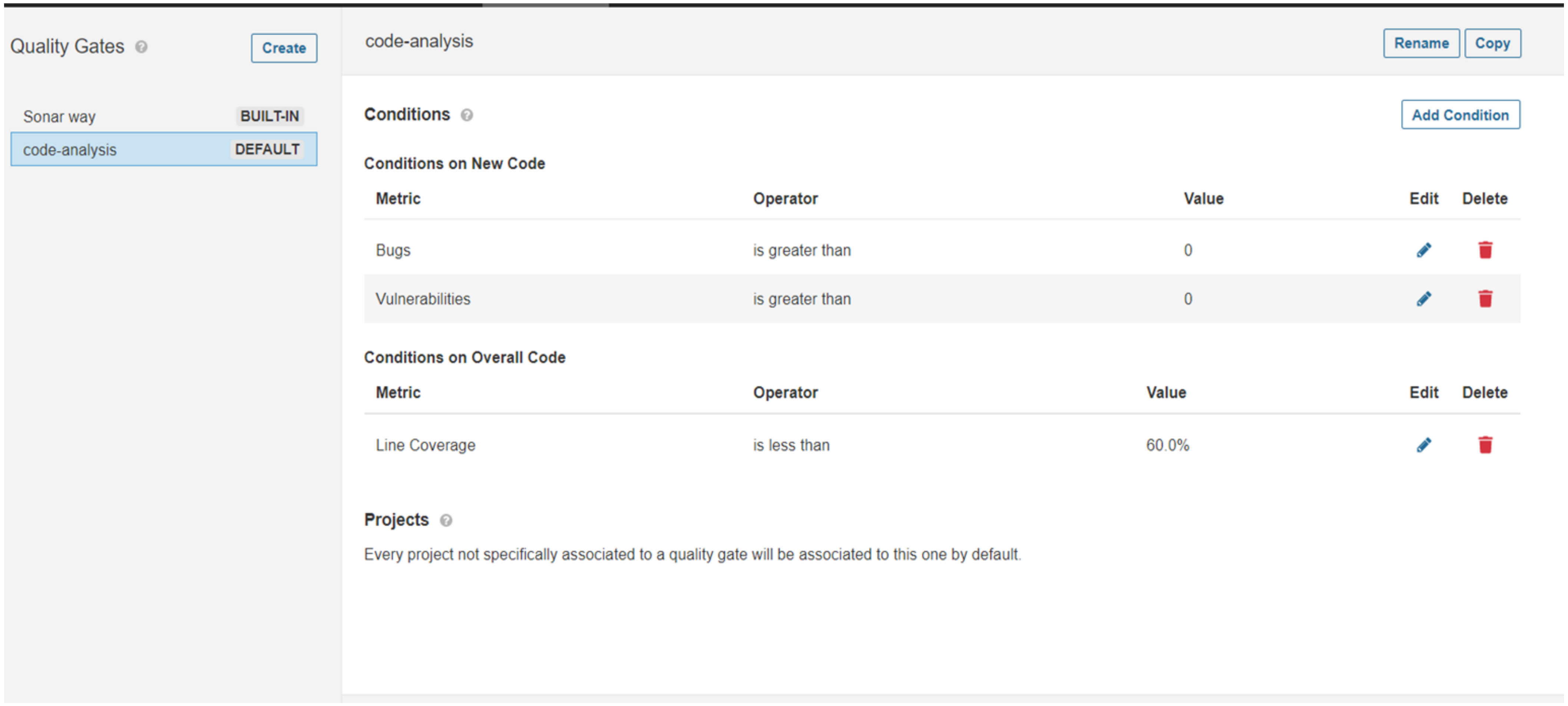
Embedded database should be used for evaluation purposes only

- The url format is **[ngrok-url]/sonarqube-webhook/**
- Make sure ngrok is active

Configuring Quality Gates

- Quality Gates are the set of conditions a project must meet before it should be pushed to further environments. Quality Gates considers all of the quality metrics for a project and assigns a *passed* or *failed* designation for that project.
- Sonarqube provides a default Quality Gate which will be applied to all projects not explicitly assigned to some other gate. But we can create project specific custom quality gates as well.
- We can add multiple criteria to the custom quality gate based on code coverage, bugs, vulnerabilities on new code as well as on overall code.
- **Quality Profiles** are a core component of SonarQube where you define sets of rules that, when violated, raise issues on your codebase (example: Methods should not have a Cognitive Complexity higher than 15). Each individual language has its own Quality Profile.
- Read more on Quality profiles n sonarqube here. Click Here

Go to Quality Gates at top and create a new quality gate. Add required conditions to it. Select a project for this quality gate for make it the default gate for all projects



Conditions on New Code

Metric	Operator	Value	Edit	Delete
Bugs	is greater than	0		
Vulnerabilities	is greater than	0		

Conditions on Overall Code

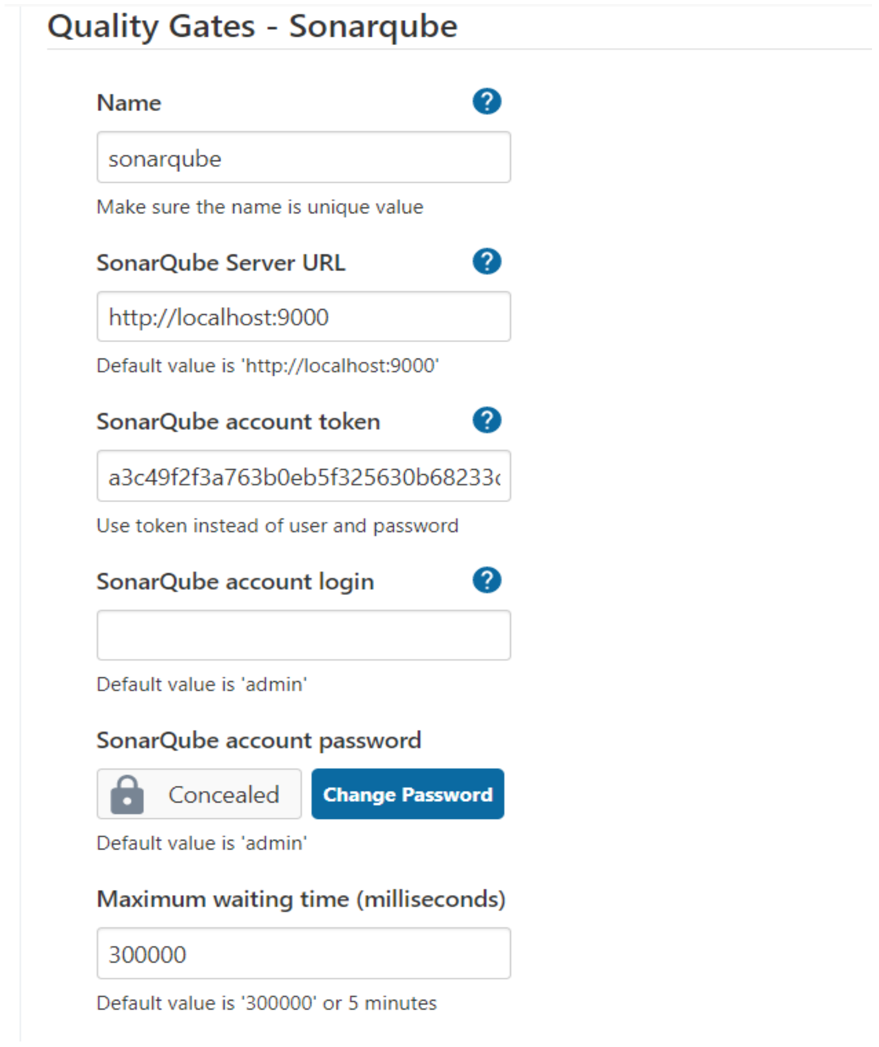
Metric	Operator	Value	Edit	Delete
Line Coverage	is less than	60.0%		

Projects

Every project not specifically associated to a quality gate will be associated to this one by default.

Configuring Quality Gate in Jenkins

Go to Manage Jenkins -> Configure System -> Quality Gates.



Quality Gates - Sonarqube

Name

sonarqube

Make sure the name is unique value

SonarQube Server URL

http://localhost:9000

Default value is 'http://localhost:9000'

SonarQube account token

a3c49f2f3a763b0eb5f325630b68233(

Use token instead of user and password

SonarQube account login

Default value is 'admin'

SonarQube account password

Concealed

Default value is 'admin'

Maximum waiting time (milliseconds)

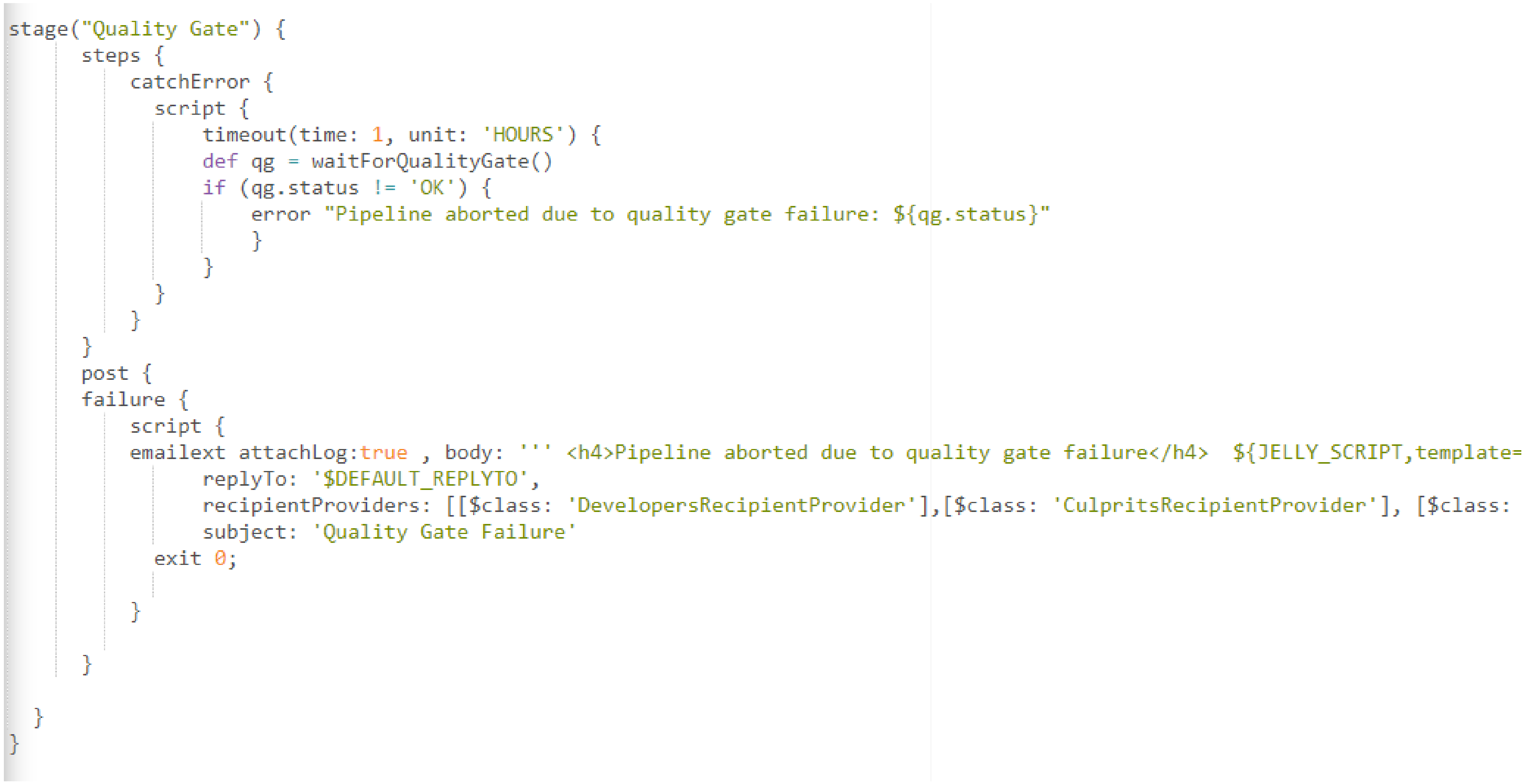
300000

Default value is '300000' or 5 minutes

Add the SonarQube account token created before or username and password can also be used as an alternative

Maximum waiting time is the time for which Jenkins waits for the quality gate success, otherwise the build fails.

Adding Quality Gate stage in Pipeline script

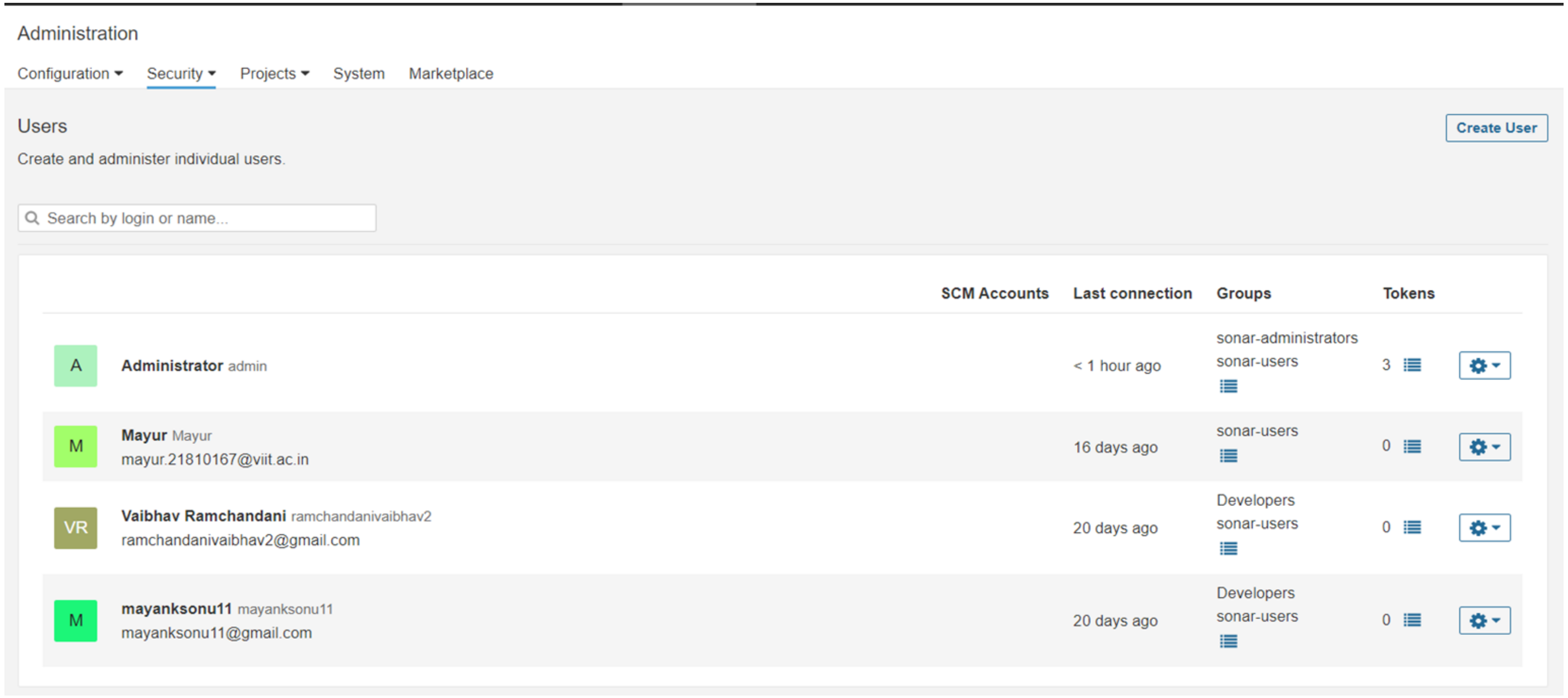


```
stage("Quality Gate") {
      steps {
          catchError {
            script {
                timeout(time: 1, unit: 'HOURS') {
                def qg = waitForQualityGate()
                if (qg.status != 'OK') {
                    error "Pipeline aborted due to quality gate failure: ${qg.status}"
                    }
                }
            }
          }
      }
      post {
      failure {
          script {
          emailext attachLog:true , body: ''' <h4>Pipeline aborted due to quality gate failure</h4>  ${JELLY_SCRIPT,template=
                replyTo: '$DEFAULT_REPLYTO',
                recipientProviders: [[$class: 'DevelopersRecipientProvider'],[$class: 'CulpritsRecipientProvider'], [$class:
                subject: 'Quality Gate Failure'
            exit 0;

          }

      }

  }
}
```

Email notification is added as the post build step , to notify the developer in case if the quality gate fails

Adding Notification in SonarQube

- We can also add email notifications in sonarqube for more specific notification about reason for quality gate failure.
- Go to SonarQube -> Administrator ->Users
- Create a User for the developer by specifying the email address.

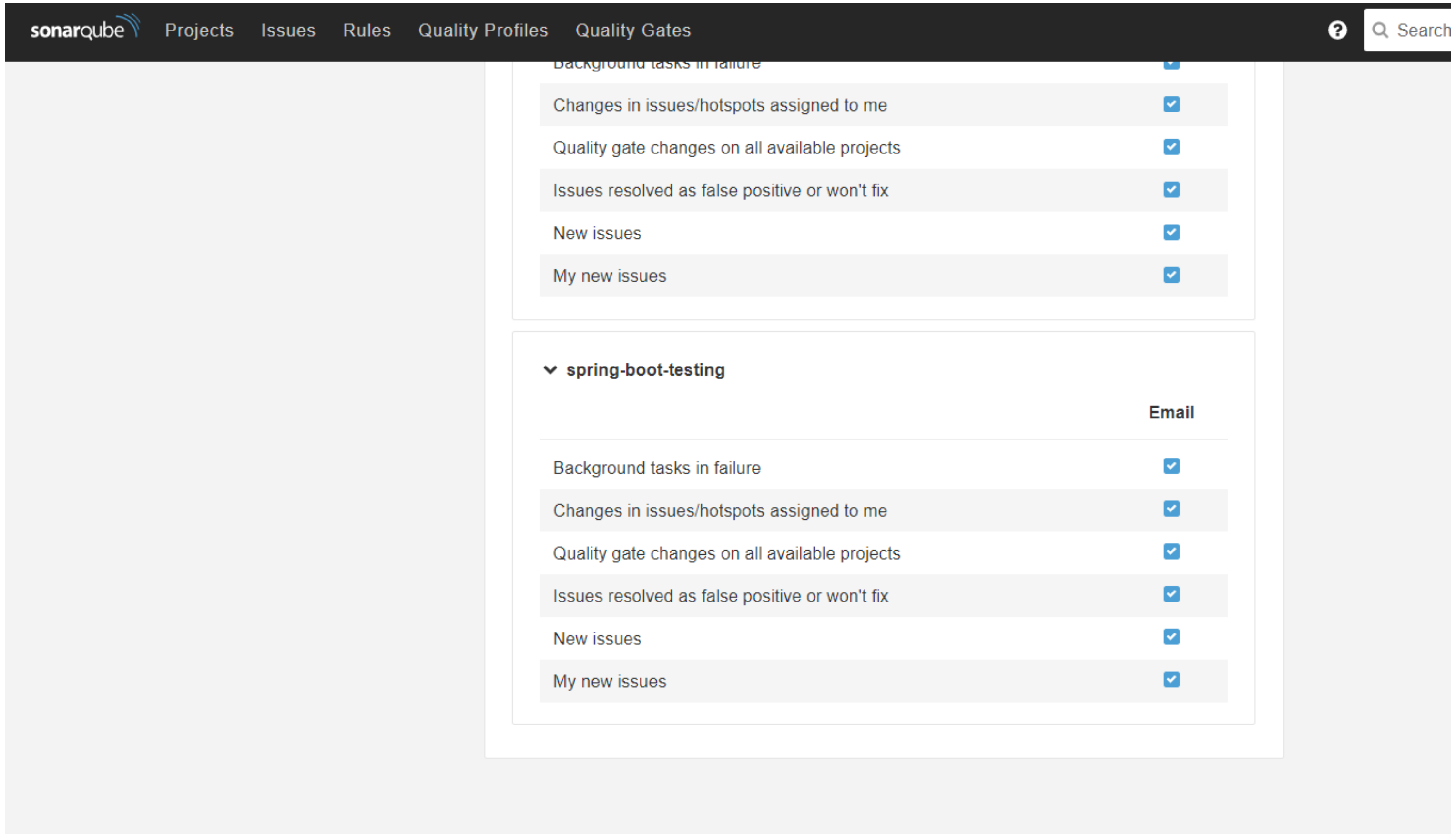


Users

Create and administer individual users.

	SCM Accounts	Last connection	Groups	Tokens
Administrator admin		< 1 hour ago	sonar-administrators sonar-users	3
Mayur Mayur mayur.21810167@viit.ac.in		16 days ago	sonar-users	0
Vaibhav Ramchandani ramchandanivaibhav2 ramchandanivaibhav2@gmail.com		20 days ago	Developers sonar-users	0
mayanksonu11 mayanksonu11 mayanksonu11@gmail.com		20 days ago	Developers sonar-users	0

The developer has to subscribe to the notifications for the project.
For developer . Log In -> Account -> Notifications



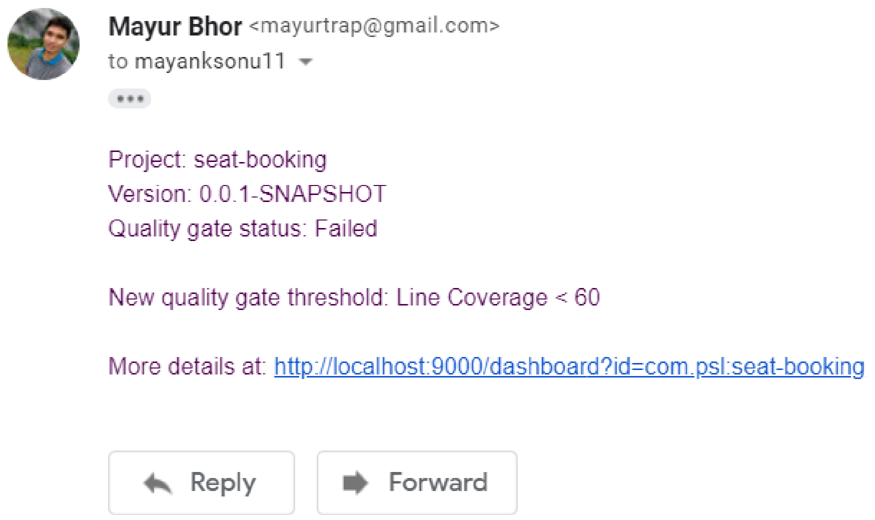
Changes in issues/hotspots assigned to me	✓
Quality gate changes on all available projects	✓
Issues resolved as false positive or won't fix	✓
New issues	✓
My new issues	✓

spring-boot-testing

	Email
Background tasks in failure	✓
Changes in issues/hotspots assigned to me	✓
Quality gate changes on all available projects	✓
Issues resolved as false positive or won't fix	✓
New issues	✓
My new issues	✓

Developer(User) can select various scenarious in which the notifications should be received for the project

Email Notification sent by sonarQube.

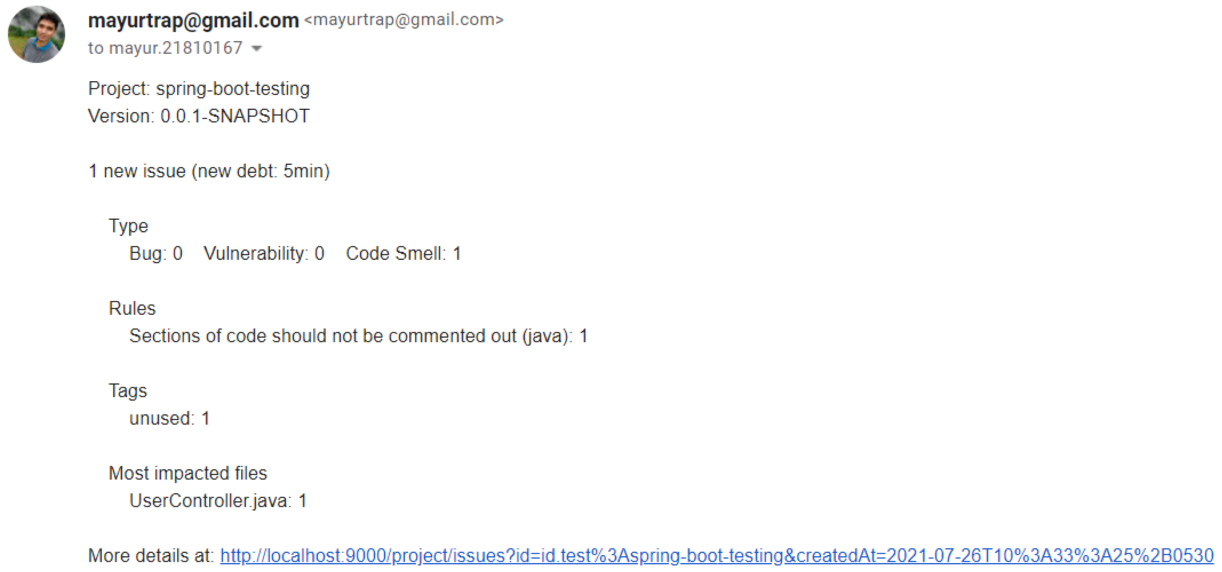


Mayur Bhor <mayurtrap@gmail.com>
to mayanksonu11

Project: seat-booking
Version: 0.0.1-SNAPSHOT
Quality gate status: Failed

New quality gate threshold: Line Coverage < 60

More details at: http://localhost:9000/dashboard?id=com.psl:seat-booking



mayurtrap@gmail.com <mayurtrap@gmail.com>
to mayur.21810167

Project: spring-boot-testing
Version: 0.0.1-SNAPSHOT

1 new issue (new debt: 5min)

Type
Bug: 0 Vulnerability: 0 Code Smell: 1

Rules
Sections of code should not be commented out (java): 1

Tags
unused: 1

Most impacted files
UserController.java: 1

More details at: http://localhost:9000/project/issues?id=id.test%3Aspring-boot-testing&createdAt=2021-07-26T10%3A33%3A25%2B0530

Note – SonarQube will only send notification to the developer in case if any new issue is encountered. So we can add email notification in Jenkins as a post build action as well , to make sure notification is sent every time there is quality gate failure.

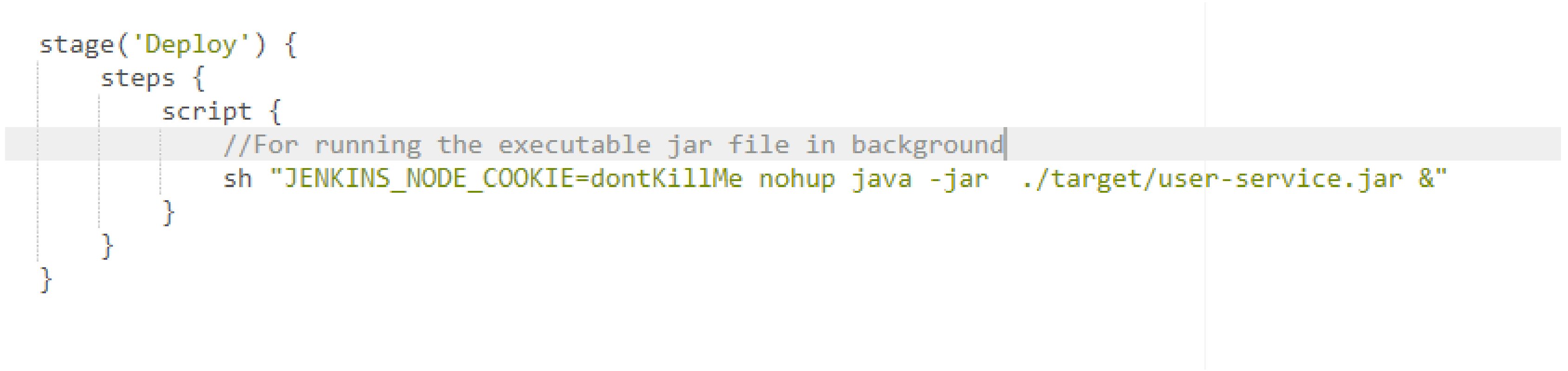
Stage Result

- SonarQube Analysis is generated along with code coverage (if Jacoco plugin is included) which can be seen in the sonarqube interface.
- If all the conditions in the quality fate are passed , the pipeline will move to the later stage.
- If the quality gate fails , notification is sent to the developer about the same , and the pipeline is aborted.

Stage 4 - Deployment

- This stage involves starting the execution of the artifacts generated (jar/war) and running them either locally or on a external server.
- While running the jar file generated locally, make sure we run it in background so that it is not terminated by Jenkins when the stage is over.
- For running the executable in background , **nohup** is used in this pipeline. So make sure Git Bash in installed in the system.(Since bat will not work , sh has to be used)

1) Deploying the Jar Locally



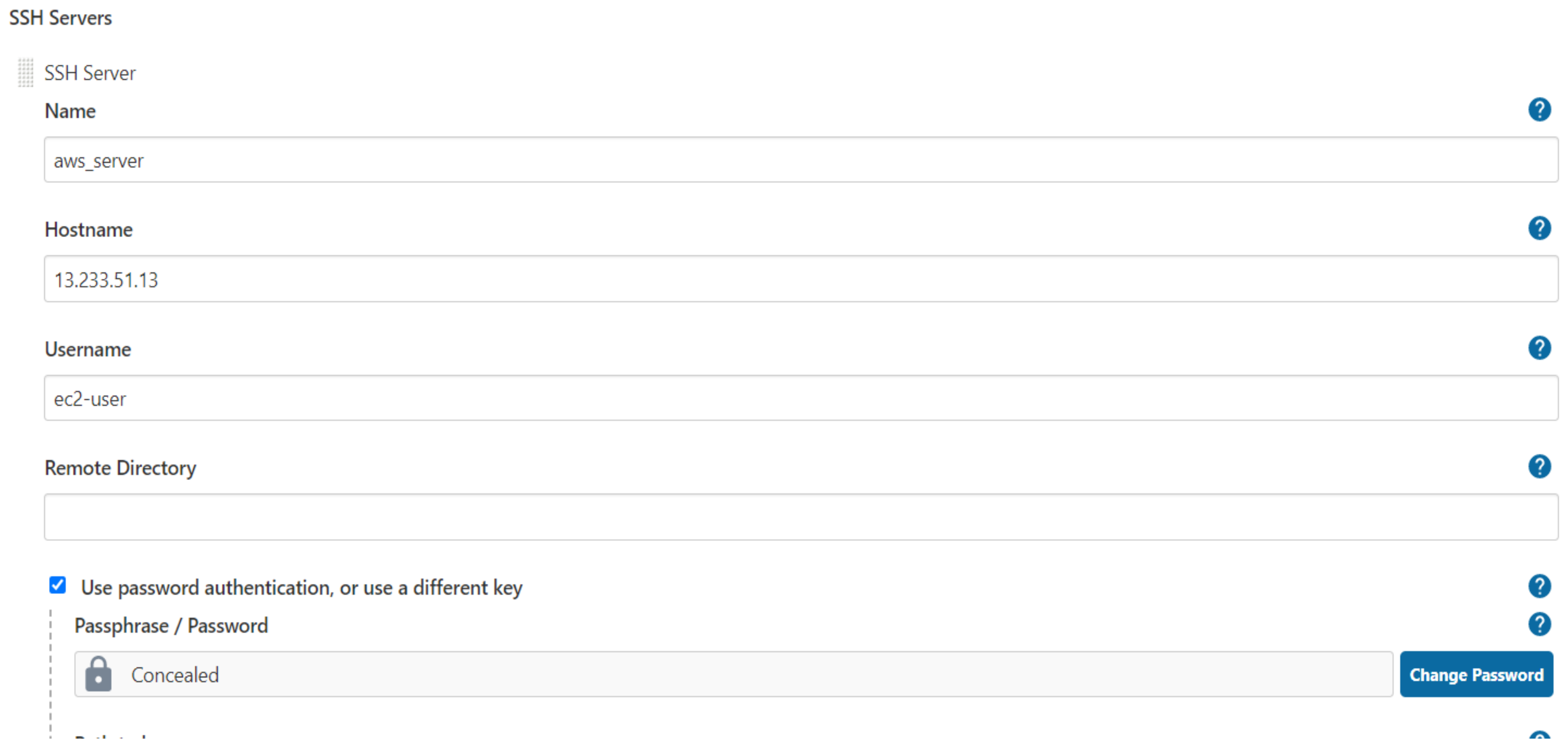
```
stage('Deploy') {
    steps {
        script {
            //For running the executable jar file in background
            sh "JENKINS_NODE_COOKIE=dontKillMe nohup java -jar  ./target/user-service.jar &"
        }
    }
}
```

- The executable jar files can be located in the target directory.
- Make sure the port on which the application is going to run is free, if not get the port changed in application.properties of the spring boot application.
- JENKINS_NODE_COOKIE=dontKillMe – for not killing the application process when Jenkin job is finished.
- nohup -- & – for running the application in background

2) Deploying to the AWS server

- This involes publishing the artifacts generated(jar) to the AWS server and running the application on the server.
- Make sure Java (with the same version as used in development) is installed and configured on the server.
- Install the Publish over SSH plugin in Jenkins.

Configuring AWS server in Jenkins
Go to Manage Jenkins -> Configure System -> Publish over SSH



SSH Servers

SSH Server

Name: aws_server

Hostname: 13.233.51.13

Username: ec2-user

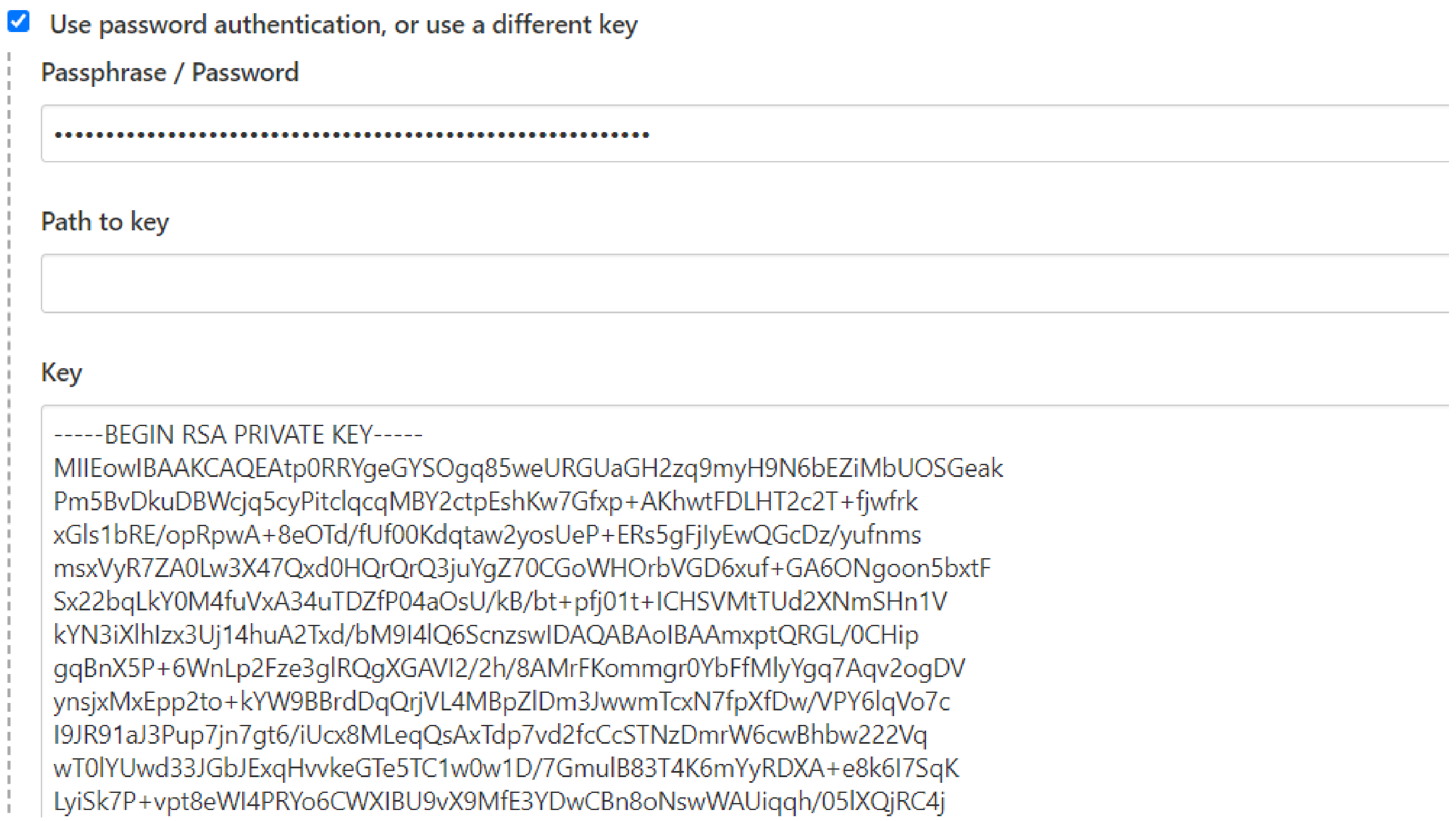
Remote Directory:

☑ Use password authentication, or use a different key

Passphrase / Password: Concealed

Add the hostname of the server(IP address of the aws server) , username and select password authentication.

Go to Advanced options and add the key for authentication . Key is the .pem file generated when creating the AWS instance .
Test the configuration to get Suceess.



☑ Use password authentication, or use a different key

Passphrase / Password

••

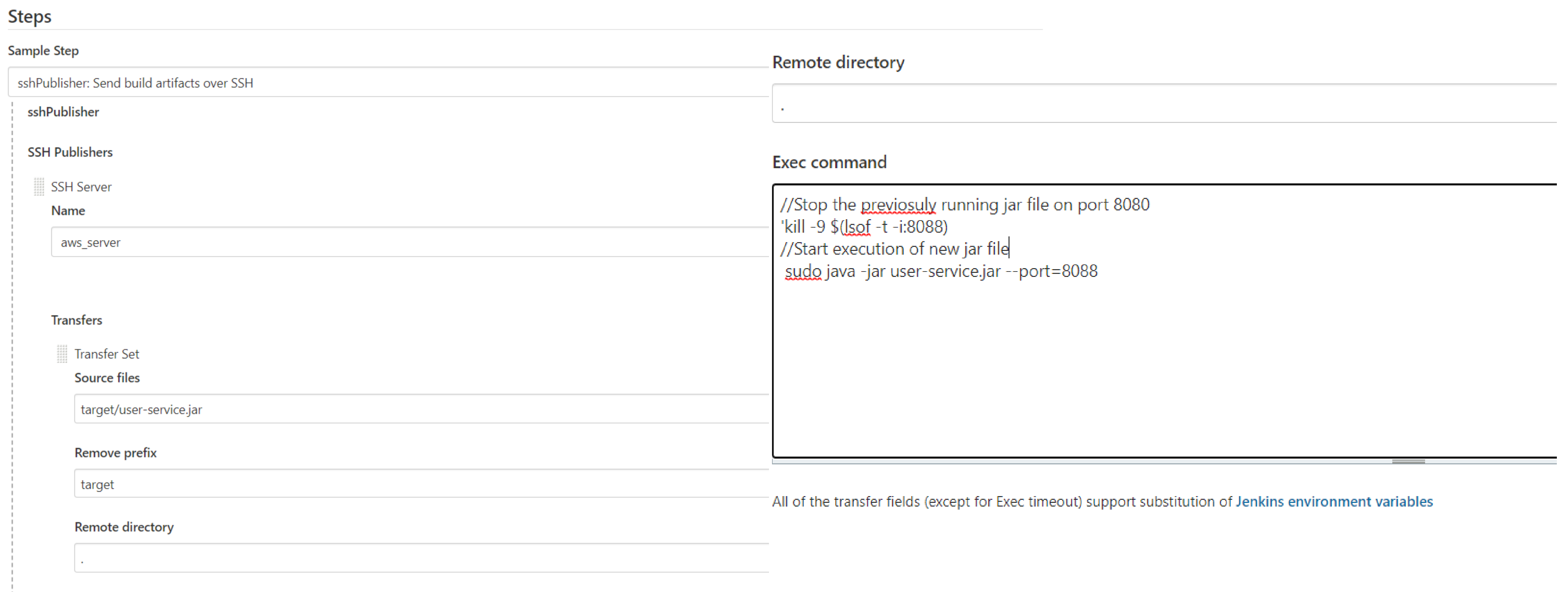
Path to key

Key

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAtp0RRYgeGYSOgq85weURGUaGH2zq9myH9N6bEZiMbUOSGeak
Pm5BvDkuDBWcjq5cyPitclqcqMBY2ctpEshKw7Gfxp+AKhwtFDLHT2c2T+fjwfrk
xGls1bRE/opRpwA+8eOTd/fUf00Kdqtaw2yosUeP+ERs5gFjlyEwQGcDz/yufnms
msxVyR7ZA0Lw3X47Qxd0HQrQrQ3juYgZ70CGoWHOrbVGD6xuf+GA6ONgoon5bxtF
Sx22bqLkY0M4fuVxA34uTDZfP04aOsU/kB/bt+pfj01t+ICHSVMtTUd2XNmSHn1V
kYN3iXlhIzx3Uj14huA2Txd/bM9I4lQ6ScnzswIDAQABAoIBAAmxptQRGL/0CHip
gqBnX5P+6WnLp2Fze3glRQgXGAVI2/2h/8AMrFKommgr0YbFfMlyYgq7Aqv2ogDV
ynsjxMxEpp2to+kYW9BBrdDqQrjVL4MBpZlDm3JwwmTcxN7fpXfDw/VPY6lqVo7c
I9JR91aJ3Pup7jn7gt6/iUcx8MLeqQsAxTdp7vd2fcCcSTNzDmrW6cwBhbw222Vq
wT0lYUwd33JGbJExqHvvkeGTe5TC1w0w1D/7GmuIB83T4K6mYyRDXA+e8k6I7SqK
LyiSk7P+vpt8eWI4PRYo6CWXIBU9vX9MfE3YDwCBn8oNswWAUiqqh/05lXQjRC4j
```

Generate the syntax for AWS deploy

- By using the pipeline syntax option, generate the pipeline script for sending the jar file to AWS server over ssh and executing it.
- Select the name of the server (just created) , source file(jar file), and the remote directory.
- Specify the command to be executed on the AWS server



Steps

Sample Step

sshPublisher: Send build artifacts over SSH

sshPublisher

SSH Publishers

SSH Server

Name

aws_server

Transfers

Transfer Set

Source files

target/user-service.jar

Remove prefix

target

Remote directory

.

Remote directory

.

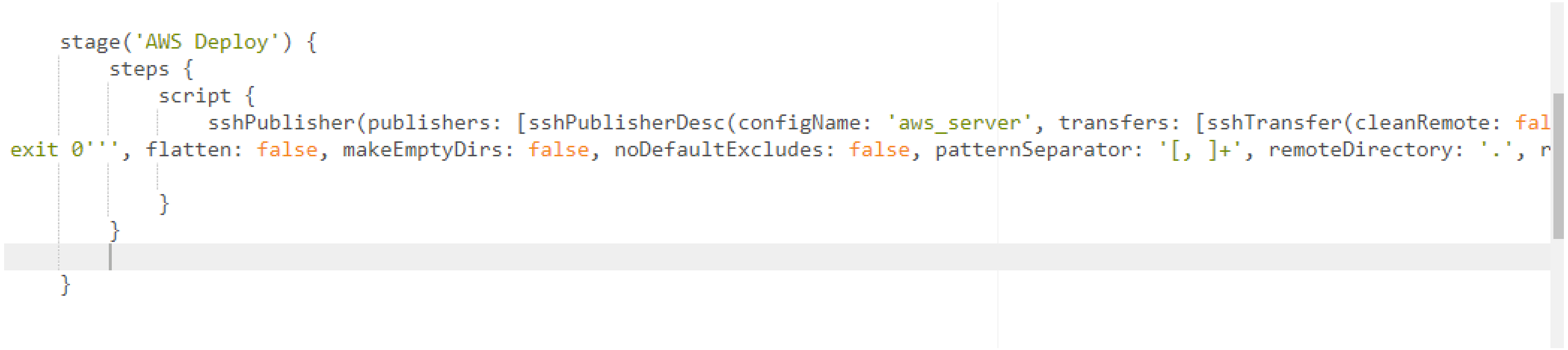
Exec command

```
//Stop the previosuly running jar file on port 8080
'kill -9 $(lsof -t -i:8088)
//Start execution of new jar file
 sudo java -jar user-service.jar --port=8088
```

All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables

Adding AWS deploy stage in pipeline

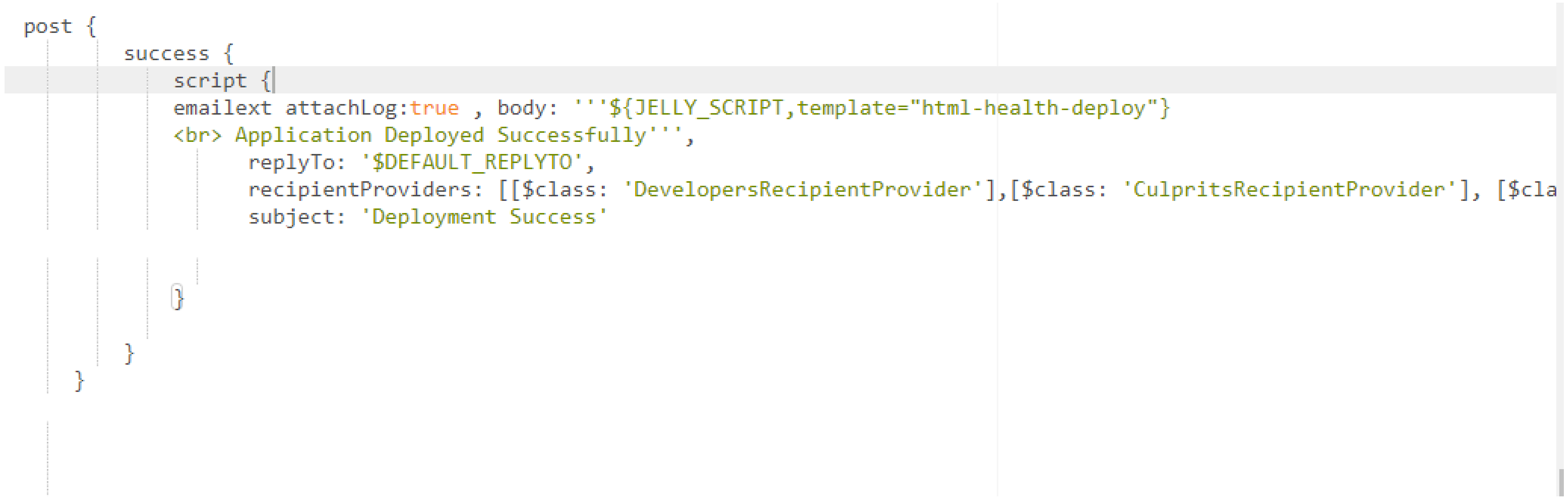
Add the generated script to the stage in pipeline



```
    stage('AWS Deploy') {
        steps {
            script {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'aws_server', transfers: [sshTransfer(cleanRemote: fal
exit 0''', flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '.', r

            }
        }

    }
```

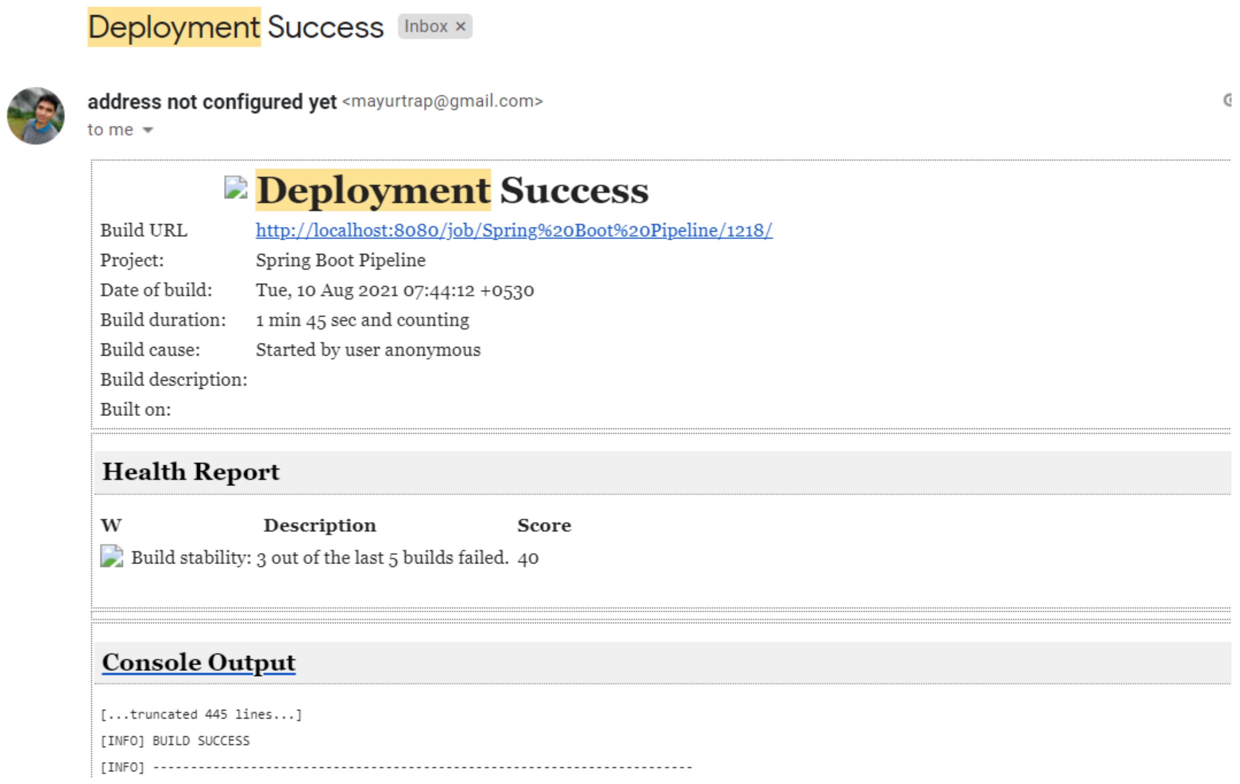
Post Declarative Actions

Finally whole pipeline is successfully finished , and the application is deployed to the server, a notification is sent to the developer about the successful deployment.



```
post {
        success {
            script {
            emailext attachLog:true , body: '''${JELLY_SCRIPT,template="html-health-deploy"}
            <br> Application Deployed Successfully''',
                  replyTo: '$DEFAULT_REPLYTO',
                  recipientProviders: [[$class: 'DevelopersRecipientProvider'],[$class: 'CulpritsRecipientProvider'], [$cla
                  subject: 'Deployment Success'


            }

        }
    }
```

Deployment Successful Notification



Deployment Success Inbox ×

address not configured yet <mayurtrap@gmail.com>
to me

Deployment Success

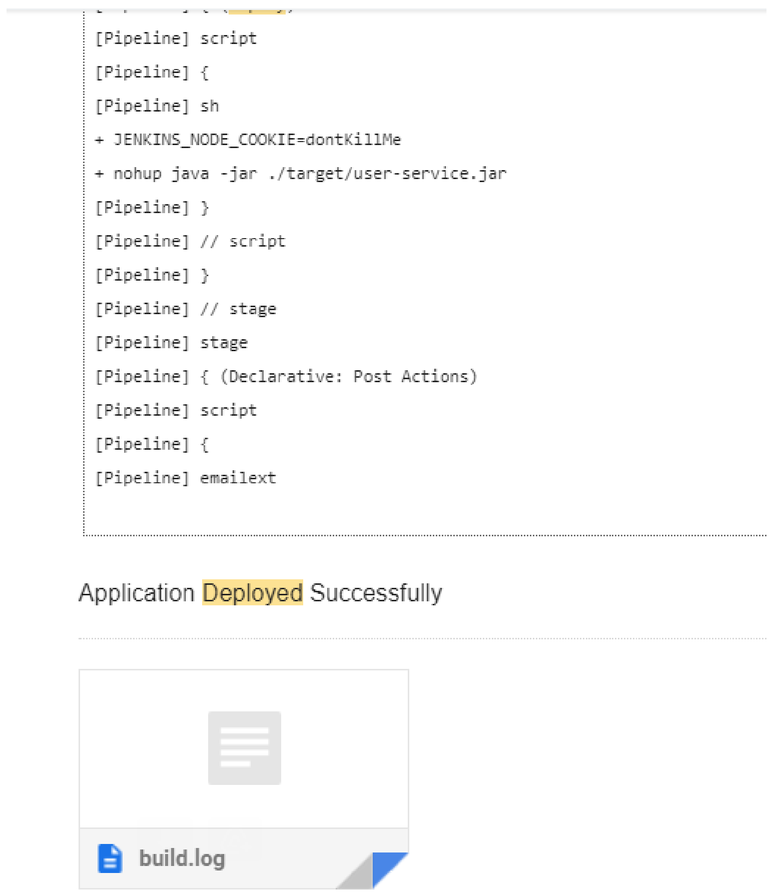
Build URL	http://localhost:8080/job/Spring%20Boot%20Pipeline/1218/
Project:	Spring Boot Pipeline
Date of build:	Tue, 10 Aug 2021 07:44:12 +0530
Build duration:	1 min 45 sec and counting
Build cause:	Started by user anonymous
Build description:	
Built on:	

Health Report

W	Description	Score
	Build stability: 3 out of the last 5 builds failed.	40

Console Output

```
[...truncated 445 lines...]
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```



```
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ JENKINS_NODE_COOKIE=dontKillMe
+ nohup java -jar ./target/user-service.jar
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] script
[Pipeline] {
[Pipeline] emailext
```

Application Deployed Successfully

build.log

Complete Pipeline Script

Complete Pipeline Script can be found here. Click here