

How to - Gitlab-sonar integration

Sonar integration for scala code:

Configuration at Jenkins

1.Install *SonarQube plugin* to Jenkins

Manage Jenkins >> Manage Plugins >> Available Tab and select “SonarQube Scanner for Jenkins” plugin and install.

Jenkins ▾ > Plugin Manager		
<input checked="" type="checkbox"/>	This plugin integrates with Gerrit code review.	2.30.0
<input checked="" type="checkbox"/>	Maven Integration plugin This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...).	3.4
<input checked="" type="checkbox"/>	Oracle Java SE Development Kit Installer Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	1.0
<input checked="" type="checkbox"/>	Pipeline: Supporting APIs Common utility implementations to build Pipeline Plugin	3.3
<input checked="" type="checkbox"/>	Plain Credentials Plugin Allows use of plain strings and files as credentials.	1.5
<input checked="" type="checkbox"/>	Sonar Gerrit Plugin This plugin allows to submit issues from SonarQube to Gerrit as comments directly.	2.3
<input checked="" type="checkbox"/>	Sonar Quality Gates Plugin Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")	1.3.1
<input checked="" type="checkbox"/>	SonarQube Scanner for Jenkins This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	2.9

And install Sbt for scala program

<input checked="" type="checkbox"/>	Matrix This plugin allows you to configure email notifications for build results	1.27	Uninstall
<input checked="" type="checkbox"/>	Matrix Authorization Strategy Offers matrix-based security authorization strategies (global and per-project).	1.1	Uninstall
<input checked="" type="checkbox"/>	Matrix Project Plugin Multi-configuration (matrix) project type.	1.14	Uninstall
<input checked="" type="checkbox"/>	Oracle Java SE Development Kit Installer Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	1.0	Uninstall
<input checked="" type="checkbox"/>	OWASP Markup Formatter Uses the OWASP Java HTML Sanitizer to allow safe-seeming HTML markup to be entered in project descriptions and the like.	1.1	Uninstall
<input checked="" type="checkbox"/>	PAM Authentication Adds Unix Pluggable Authentication Module (PAM) support to Jenkins	1.1	Uninstall
<input checked="" type="checkbox"/>	sbt plugin This plugin allows running SBT empowered scala projects in Hudson.	1.5	Uninstall
<input checked="" type="checkbox"/>	WMI Windows Agents Allows you to connect to Windows machines and start slave agents on them.	1.0	Uninstall

Go to Manage Jenkins >>configuring the system, Search SonarQube servers section,

Jenkins ▾ » configuration Delete

Add

☐ Tool Locations
SonarQube servers
 Environment variables
 SonarQube installations

☐ 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.

Name
 Server URL
Default is http://localhost:9000
 Server authentication token Add
SonarQube authentication token, mandatory when anonymous access is disabled.

Advanced

Delete SonarQube

Add SonarQube

List of SonarQube installations

Configure sonarquabe scanner in jenkins








manageJenkins > Global Tool Configuration

Jenkins ▾ » ENABLE AUTO-UPDATE

Improve user account validation

Bitbucket Approve Plugin 1.0.3

Bitbucket Approve Plugin stores credentials in plain text


-  **Configure System**
Configure global settings and paths.
-  **Configure Global Security**
Secure Jenkins, define who is allowed to access/use the system.
-  **Configure Credentials**
Configure the credential providers and types
-  **Global Tool Configuration**
Configure tools, their locations and automatic installers.
-  **Reload Configuration from Disk**
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
-  **Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
▲ There are updates available
-  **System Information**

movartis.devops.altimetrik.io:9000

SonarQube Scanner


SonarQube Scanner installations

Add SonarQube Scanner

 SonarQube Scanner

Name

☒ Install automatically ?

 Install from Maven Central

Version

Sbt

Sbt installations

Add Sbt

 Sbt

name

sbt launch arguments

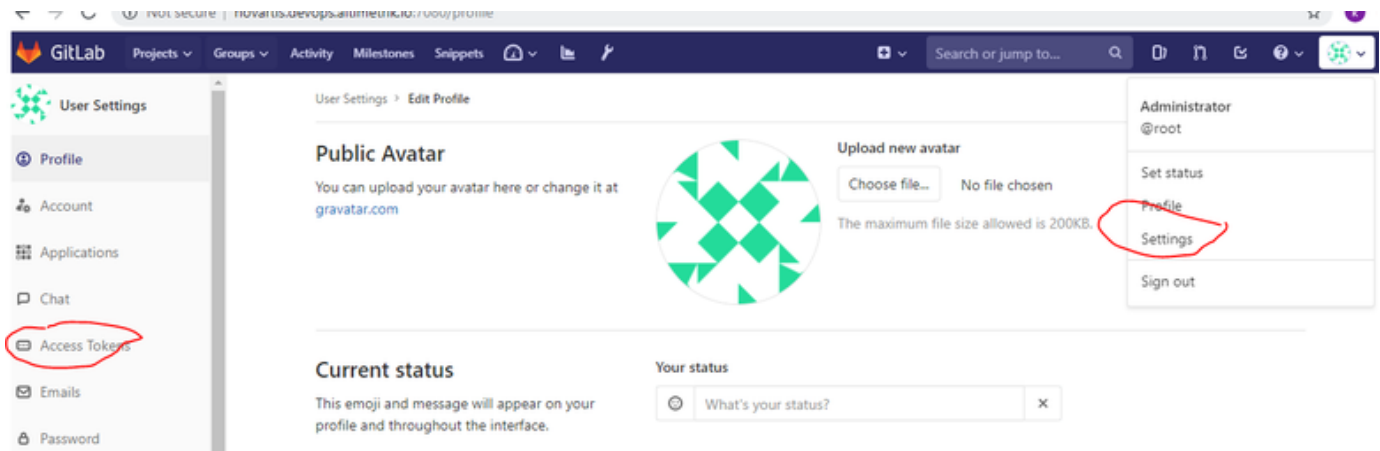
Save

Apply

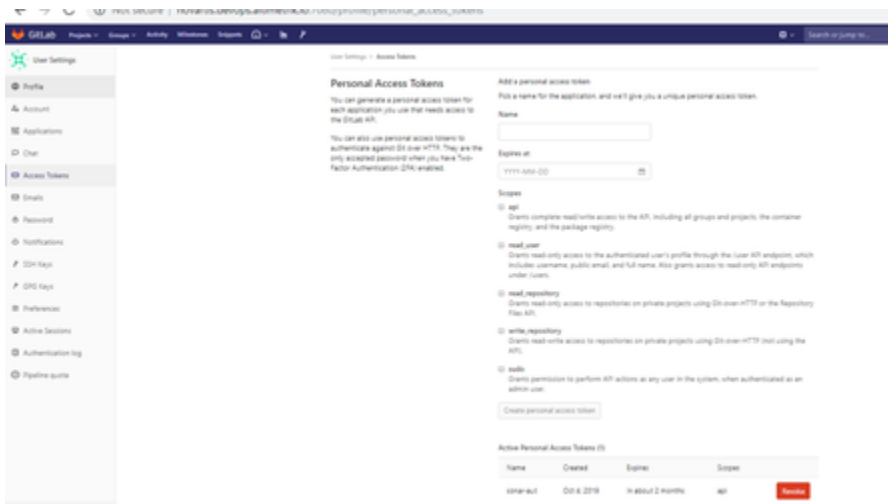
Save it.

Configurations at Git Lab:

- Go to **User Settings** form Settings menu.



- Go to **Access Tokens**




- Create a personal access token by adding any unique name(**Name**) and token expiry date(**Expires at**). Also set the **Scopes** to api- Full access.

Configure GitLab and SonarQube at Jenkins


Go to manage jenkins configure system

Go to GitLab tab and add your GitLab Server URL at **GitLab host URL**



GitLab Server

Display Name 


A unique name for the server

Server URL 


The url to the GitLab server

Credentials  

The Personal Access Token for GitLab API access

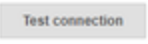

Web Hook ☐ **Manage Web Hooks** 

Do you want to automatically manage GitLab Web Hooks on Jenkins Server?

System Hook ☐ **Manage System Hooks** 

Do you want to automatically manage GitLab System Hooks on Jenkins Server?

Credentials verified for user root






Adding a project to Jenkins



Go to Jenkins Dashboard -> **New Item** > Select **Freestyle Project**.

Jenkins > All

Enter an item name

* This field cannot be empty, please enter a valid name

-  **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
-  **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
-  **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
-  **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

  **Project Name**

Add gitlab URL under SCM

Source Code Management

☐ None
☒ Git

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser:

Now, scroll to **Build Triggers**, select the GitLab webhook URL checkbox.

Copy the GitLab webhook URL. We need to setup **webhook** at GitLab again, using this url.

☒ Build when a change is pushed to GitLab. GitLab webhook URL:

Enabled GitLab triggers

Push Events	<input checked="" type="checkbox"/>
Opened Merge Request Events	<input checked="" type="checkbox"/>
Accepted Merge Request Events	<input type="checkbox"/>
Closed Merge Request Events	<input type="checkbox"/>
Rebuild open Merge Requests	<input type="text" value="Never"/>
Approved Merge Requests (EE-only)	<input checked="" type="checkbox"/>
Comments	<input checked="" type="checkbox"/>
Comment (regex) for triggering a build	<input type="text" value="Jenkins please retry a build"/>

Secret token:

Click on Add build step build using sbt

Build

Build using sbt

sbt launcher

sbt

JVM Flags

sbt Flags

-Dsbt.log.noformat=true

Actions

coverage coverageReport sonarScan

Advanced...

Add build step

Save it.

Setup webhook at GitLab.

1. login to gitlab
2. Go to **Your Projects** at **Project** menu.

GitLab

Projects

Groups

Activity

Milestones

Snippets

Search or jump to...

Projects

New project

Your projects 6

Starred projects 0

Explore projects

Filter by name...

Last updated

All

Personal

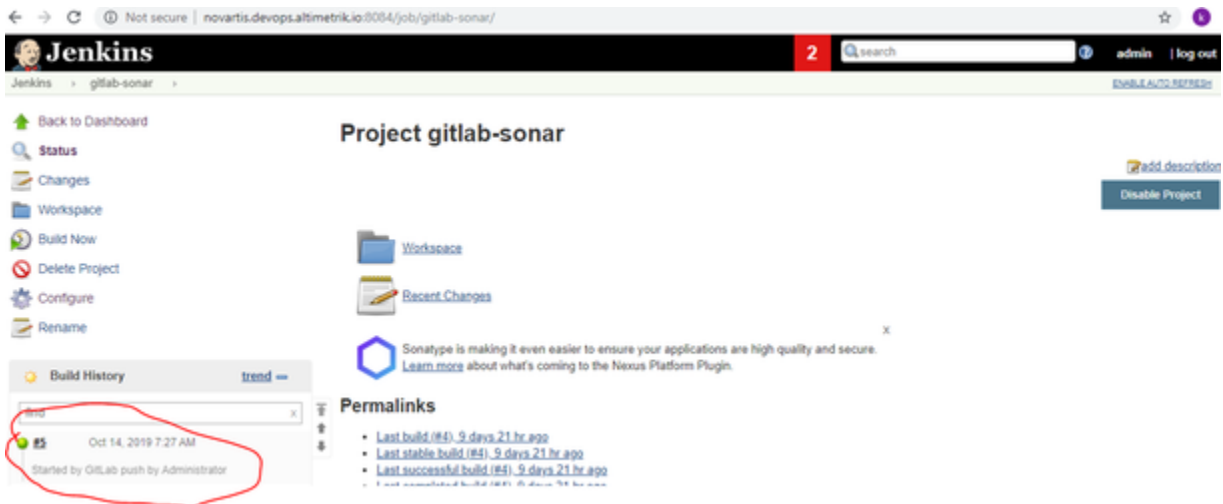
N	Administrator / novartis_new	Maintainer	★ 0	Y 0	I 0	D 0	Updated 1 week ago
G	Administrator / GitLab_Sonar Sonar integration with gitlab	Maintainer	★ 0	Y 0	I 0	D 0	Updated 1 week ago
S	Administrator / Scala_Sample_Project	Maintainer	★ 0	Y 0	I 0	D 0	Updated 2 weeks ago

3. Go to **Settings > Integrations**





After test is successful, Project starts building at Jenkins. Login to Jenkins and verify the Project builds.



Run sonar with build.sbt

Now add the `sbt-sonar` plugin dependency to your scala project .

```
addSbtPlugin("com.github.mwz" % "sbt-sonar" % "2.0.0")
```

This sbt plugin can be used to run sonar-scanner launcher to analyze a Scala project with SonarQube.

Add dependencies libraries

Source	Description
..	
project/target/config-classes	
target	
build.properties	sonar
Dependencies.scala	dependency added
plugin.sbt	plugin.sbt edited online with Bitbucket

```
import sbt._

object Dependencies
{
  lazy val scalaTest
= "org.scalatest" %%
  "scalatest" % "3.0.
5"
}
```

Configuration

Now, configure the sonar-properties in your project. This can be done in 2 ways

1. Use sonar-project.properties file This file has to be placed in your root directory
2. Configure Sonar-properties in build.sbt file
3. Configure plugins in plugin.sbt file

By default, the plugin expects the properties to be defined in the sonarProperties setting key in sbt.

```
import sbt.sonar.SonarPlugin.autoImport.sonarPro
  properties

import Dependencies._

ThisBuild / scalaVersion      := "2.12.8"
ThisBuild / version           := "0.1.0-SNAPSHOT"

lazy val root = (project in file("."))
  .settings(
    name := "demo-hello",
    libraryDependencies += scalaTest % Test
  )

sonarProperties ++= Map(
  "sonar.host.url" -> "http://novartis.devops.alti
metrik.io:9000/",
  "sonar.projectName" -> "demo-hello",
  "sonar.projectKey" -> "demo-hello",
  "sonar.sourceEncoding" -> "UTF-8",
  "sonar.java.source" -> "1.8",
  "sonar.sources" -> "src/main/scala",
  "sonar.tests" -> "src/test/scala",
)
```

Plugin.sbt

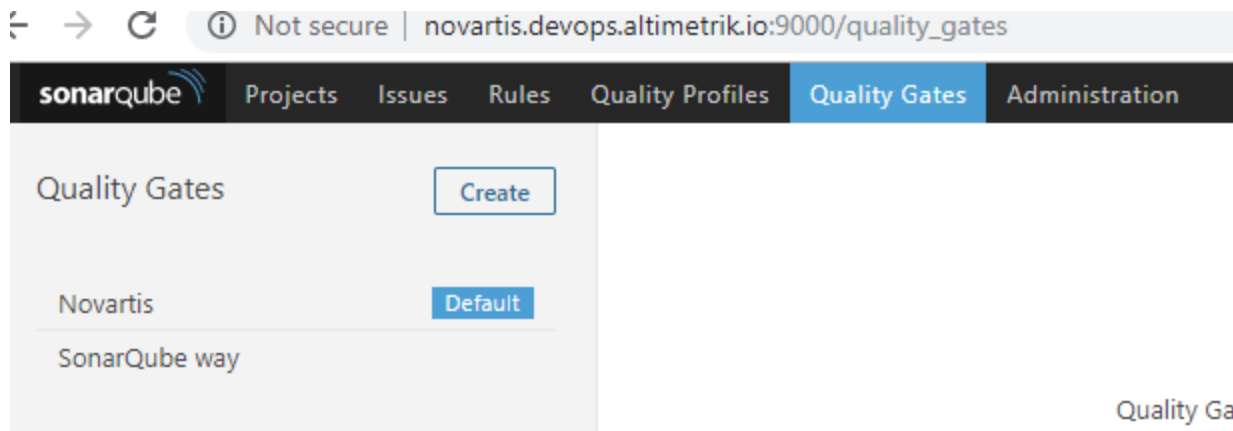
```
addSbtPlugin("com.github.mwz" % "sbt-sonar" % "2.0.0")
addSbtPlugin("org.scoverage" % "sbt-scoverage" % "1.6.0")
addSbtPlugin("com.sksamuel.scapegoat" %% "sbt-scapegoat" % "1.0.9")
```



Run the build and go to sonar dashboard.

Create quality gates

1. Go to sonar dashboard >Quality gates



Click on create option give the name

A screenshot of the 'Create Quality Gate' dialog box. It has a title bar 'Create Quality Gate'. Below the title, there is a label 'Name *' followed by an empty text input field. At the bottom right of the dialog, there are two buttons: 'Create' and 'Cancel'.

Add conditions and save it.

Quality Gates

Create

Novartis

Default

SonarQube way

Novartis

Rename

Copy

Unset as Default

Delete

Conditions

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. [More](#)

Metric	Over Leak Period	Operator	Warning	Error		
Conditions to Cover on New Code	Always	equals		5	Update	Delete
Coverage	<input type="checkbox"/>	is less than		80	Update	Delete
Skipped Unit Tests	<input type="checkbox"/>	is not		0	Update	Delete

Add Condition

Projects