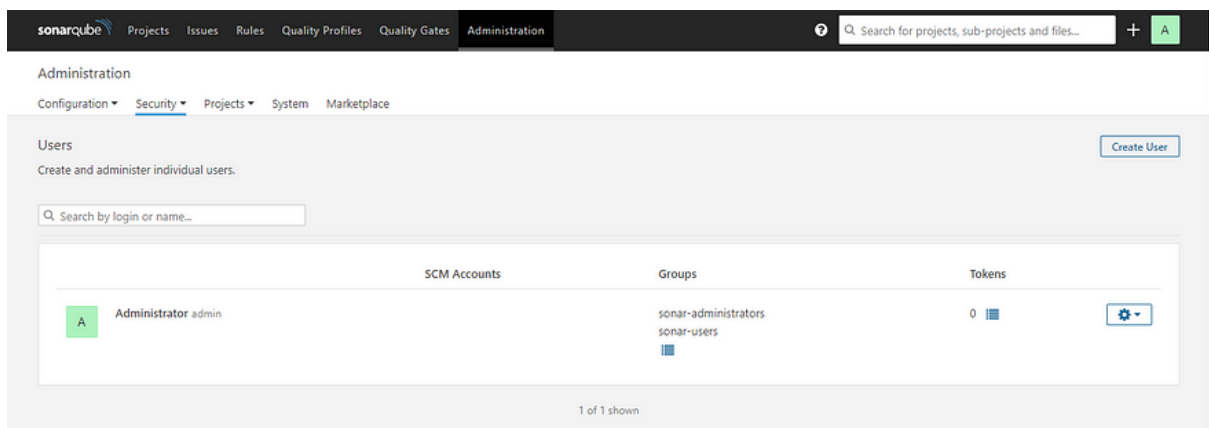


Integrate SonarQube with Jenkins

Step 1. Open SonarQube server- Go to Administration > click on Security > Users > Click on Tokens (image 1)> Generate token with some name > Copy the token (image 2), it will be used in Jenkins for Sonar authentication.



Tokens

Generate Tokens

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

42a18614f821bc940fdd77cfdad00477295026e6

Name	Created	
Jenkins	November 6, 2018	<input type="button" value="Revoke"/>

Step 2. Setup **SonarQube** with **Jenkins**- Go to *Manage Jenkins > Configure system > SonarQube* server section > *Add SonarQube* > Name it, provide Server Url as ***http://<IP>:<port>*** > and authentication token copied from SonarQube Server > *Apply* and **Save**

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	Server URL	Server authentication token
LocalSonar	http://localhost:9000 <small>Default is http://localhost:9000</small> <small>SonarQube authentication token. Mandatory when anonymous access is disabled.</small>

Step 3. Install **SonarQube plugin** to Jenkins. Go to *Manage Jenkins > Manage Plugins > Available > Search for SonarQube Scanner > Install*.

Filter:

Install ↓	Name	Version
<input type="checkbox"/>	CodeSonar A plugin that integrates with GrammaTech Codesonar.	2.0.7
<input checked="" type="checkbox"/>	SonarQube Scanner This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	2.8.1
<input type="checkbox"/>	Sonargraph Integration This plugin integrates Sonargraph functionality into Jenkins	2.1.2

Download *SonarScanner* if you don't have <https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>

Configure **Sonar Scanner** in *Jenkins* : Go to *Mange Jenkins > Global Tool Configuration > Scroll for SonarQube Scanner > Add sonar scanner > name it, uncheck if you already have sonar else it will automatically download for you and your sonar scanner setup will be done(in my case I already have) > provide path to **sonar runner home** as in below image*

The screenshot shows the Jenkins 'Global Tool Configuration' page. It is divided into two main sections: 'SonarScanner for MSBuild' and 'SonarQube Scanner'. The 'SonarScanner for MSBuild' section has a header 'SonarScanner for MSBuild installations' and a button 'Add SonarScanner for MSBuild'. Below it is a link 'List of SonarScanner for MSBuild installations on this system'. The 'SonarQube Scanner' section has a header 'SonarQube Scanner installations' and a button 'Add SonarQube Scanner'. Below this is a table for existing installations. The table has columns for 'Name' and 'Path'. The first row shows 'LocalSonarScanner' with the path 'D:\sonar-scanner-3.2.0.1227-windows'. There is a checkbox 'Install automatically' which is unchecked. A red button 'Delete SonarQube Scanner' is next to the first row. At the bottom of the table is another 'Add SonarQube Scanner' button and a link 'List of SonarQube Scanner installations on this system'.

Name	Path
LocalSonarScanner	D:\sonar-scanner-3.2.0.1227-windows

Step 4. Create a Job- *New Item* > Name and select a project type (in my case I am selecting *Maven* project you can opt for freestyle as well)

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours

☒ Clean before checkout

Step 5. Set *Git* under *SCM* section and use * * * * * for *Poll SCM* under *Build Trigger* section. Under *Build Environment* section add pre-build step > select ***Execute SonarQube Scanner***

☐ With Ant

Pre Steps

Add pre-build step ▾

- Execute SonarQube Scanner
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit
- SonarScanner for MSBuild - Begin Analysis
- SonarScanner for MSBuild - End Analysis

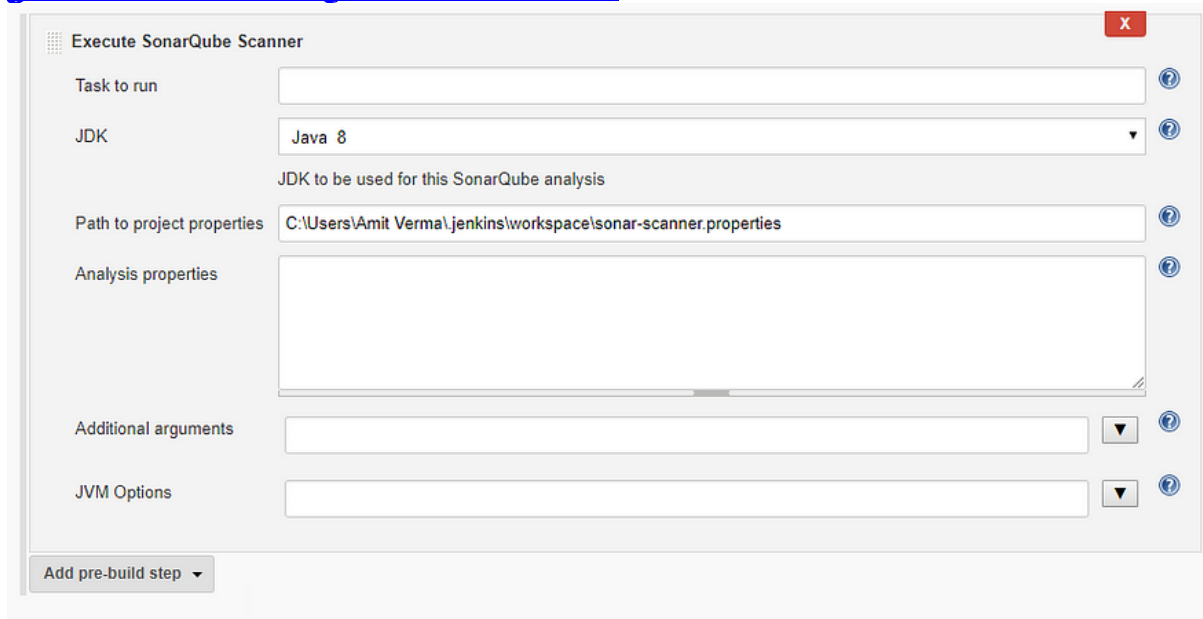
☐ Run only if build succeeds
 ☐ Run only if build succeeds or is unstable
 ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

harQube/configure#

Step 6. Create a **.properties** file at any location and provide path on the task as below(I have created it in Jenkins workspace folder). This property file will be **project specific**. It contains certain sonar properties like which folder to scan, which folder to exclude in scanning, what is the project key and many more you can see it from <https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>

The image shows a Jenkins configuration page titled "Execute SonarQube Scanner". It contains several input fields: "Task to run" (empty), "JDK" (set to "Java 8"), "Path to project properties" (set to "C:\Users\Amit Verma\jenkins\workspace\sonar-scanner.properties"), "Analysis properties" (empty text area), "Additional arguments" (empty), and "JVM Options" (empty). There are help icons (question marks) next to the "JDK", "Path to project properties", "Analysis properties", "Additional arguments", and "JVM Options" fields. At the bottom left, there is a button labeled "Add pre-build step" with a dropdown arrow.

Inside sonar-scanner.properties write below code —

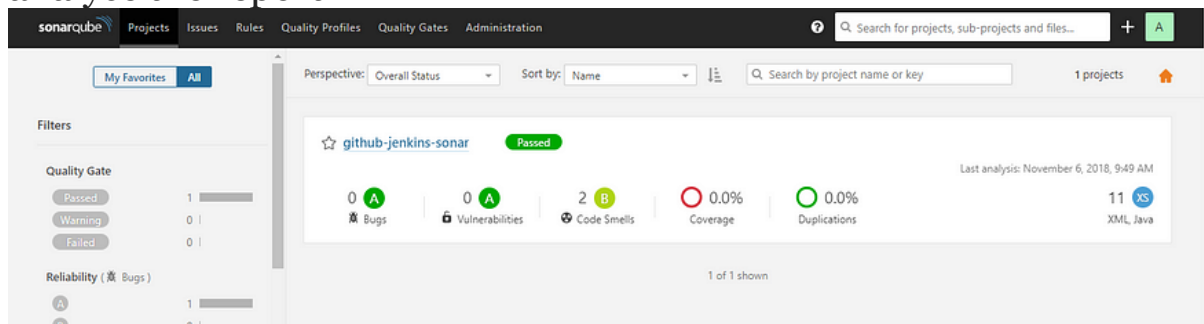
```
sonar.projectKey=github-jenkins-sonar  
sonar.sources=./src
```

To keep it simple I have used only two properties(as above), **sonar.projectKey** property will create a project inside your *SonarQube server* with the same name if project don't exist else it will append analysis to it, **sonar.sources** defines that which folder to scan. You can provide either relative path from your Jenkins Job workspace or actual path to the folder you want to scan.

Since I have used **./src** (use / for windows path) that means that I am currently on my Job workspace i.e. on **C:\Users\Amit Verma\.jenkins\workspace\Jenkins-GitHub-SonarQube** location and from here I am providing the path to the folder(src) I want to scan.

Step 7. Build the job. After successful build if you can see build logs it will show you the files and folder it has scanned and after scanning it has posted the analysis report to *SonarQube Server* you have integrated.

Step 8. From job dashboard, click on sonar icon or navigate to *Sonar server* click on **Projects** (on header) you will see a new project with same project key you have given in **sonar-scanner.properties** file. Now you can go inside your project and analyse the report



Example using declarative pipeline:

```
pipeline {
```

```
agent none
stages {
  stage("build & SonarQube analysis") {
    agent any
    steps {
      withSonarQubeEnv('My SonarQube Server') {
        sh 'mvn clean package sonar:sonar'
      }
    }
  }
  stage("Quality Gate") {
    steps {
      timeout(time: 1, unit: 'HOURS') {
        waitForQualityGate abortPipeline: true
      }
    }
  }
}
```