Static Code Analysis using SonarQube:

What is SonarQube: SonarQube collects and analyses source code, measuring quality and providing reports for your projects. It combines static and dynamic analysis tools and enables quality to be measured continuously over time. Everything that affects our code base, from minor styling details to critical design errors, is inspected and evaluated by SonarQube, thereby enabling developers to access and track code analysis data ranging from styling errors, potential bugs, and code defects to design inefficiencies, code duplication, lack of test coverage, and excess complexity. The Sonar platform analyses source code from different aspects and hence it drills down to your code layer by layer, moving from the module level down to the class level. At each level, SonarQube produces metric values and statistics, revealing problematic areas in the source that require inspection or improvement.

Why SonarQube?

- As of now, CI tools does not have a plugin which would make all these plays together
- As of now, CI tools does not have plugins to provide nice drill-down features as SonarQube does
- CI plugins does not talk about overall compliance value
- CI plugins does not provide managerial perspective
- As of now there is no CI plugin for Design/Architecture issues
- It does not provide a dashboard for overall projects quality

What is provides?

1. Whether the coding has been done following a specific convention?
2. Whether well-known/established good practices have been followed and well-known/established bad practices have been avoided?
3. Are there any potential bugs and performance issues, security vulnerabilities?
4. Is there any duplicate code?
5. Is the code logic very complex?
6. Whether the public API has good documentation and comments?
7. Whether the code has unit tests?
8. Whether the code follows good design and architecture principles?

Sample Jenkinsfile for source code: https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/

Sonar-Quality-Gate Jenkinsfile stage: https://plugins.jenkins.io/sonar-quality-gates/

Code Coverage: Code coverage is a metric that many teams use to check the quality of their tests, as it represents the percentage of production code that has been tested. Jacoco and cobertura plugin can be used to generate a code coverage report.

Bugs and Vulnerabilities: findbug tool

Static Code Analysis:

Static code analysis is a collection of algorithms and techniques used to analyse source code in order to automatically find potential errors or poor coding practices**.**

1. Install plugins: Sonar Gerrit**,** Sonargraph**,** Sonar Quality Gates **,** SonarQube Scanner**,** Sonargraph Integration , SonarQube Generic Coverage**, and** Sonargraph Integration, jacoco, cobertura

2. Create **t2 medium ec2 linux** instance and install docker (minimum of 2GB of RAM, that's why we can't install sonar on t2.micro)

   ```
   yum install -y docker
   service docker start
   systemctl enable docker
   ```

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

run: create and start docker container

--name: name of the container

-p 9000:9000 : map 9000 VM port with sonar container port (default sonar port:9000)

Sonarqube: docker image name of sonar

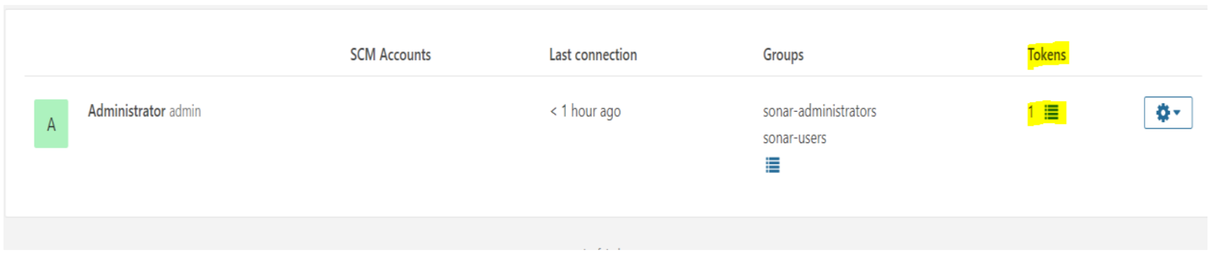
3. **login to sonarqube (user: admin password: admin)**

Access SonarQube Dashboard:

<Sonar instance public IP>:9000

Generate Token for Administrator user :-

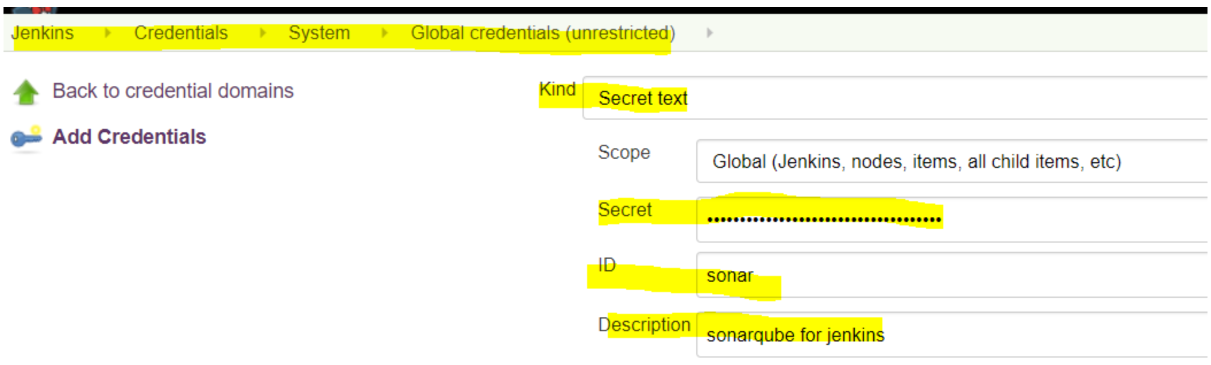
Administrator->Security->User->Token->Generate Token



	SCM Accounts	Last connection	Groups	Tokens	
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	1	

4. **Add Credential in Jenkins for SonarQube.**

Jenkins dashboard->Credentials->Jenkins under store scoped to Jenkins->**Global Credentials**->**Add Credentials**-> from dropdown select **Secret Text**->paste previously generated **sonar token**->Under ID section put as **sonar** -> under description put **sonarqube for Jenkins**



Jenkins ▸ Credentials ▸ System ▸ Global credentials (unrestricted) ▸

Back to credential domains

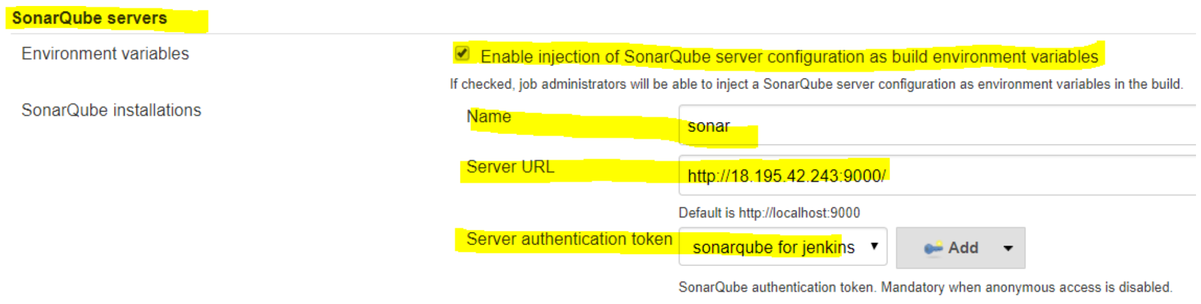
Add Credentials

Kind	Secret text
Scope	Global (Jenkins, nodes, items, all child items, etc)
Secret	••
ID	sonar
Description	sonarqube for jenkins

5. **Integrate Jenkins and SonarQube**

   Jenkins dashboard->Manage Jenkins->Configure System:

   Check for SonarQube servers "enjection option" and put name as "sonar”, sonar url and select server authentication token which you generated in previous step.



SonarQube servers

Environment variables

✓ Enable injection of SonarQube server configuration as build environment variables

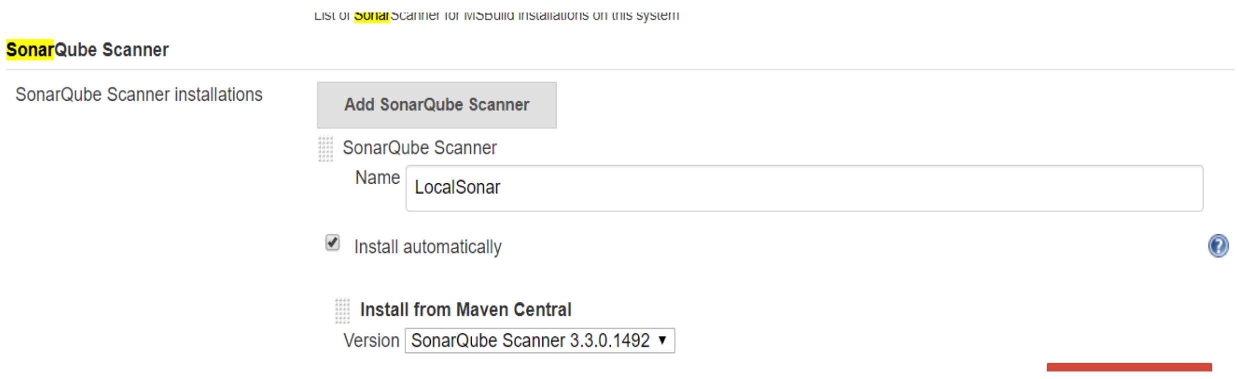
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	sonar
Server URL	http://18.195.42.243:9000/
	Default is http://localhost:9000
Server authentication token	sonarqube for jenkins ▼ Add ▼
	SonarQube authentication token. Mandatory when anonymous access is disabled.

6. **Configure SonarQube for Jenkins job**

   Manage Jenkins->Global tool Configuration->Sonarqube Scanner – add sonar


List of SonarScanner for MSBuild installations on this system

SonarQube Scanner

SonarQube Scanner installations

SonarQube Scanner

Name: LocalSonar

[✓] Install automatically

Install from Maven Central

Version: SonarQube Scanner 3.3.0.1492

Install JDK11 or higher on Jenkins:-

```
yum remove java-1.8.0-openjdk-devel -y
amazon-linux-extras install java-openjdk11

java -version
alternatives --config java
```

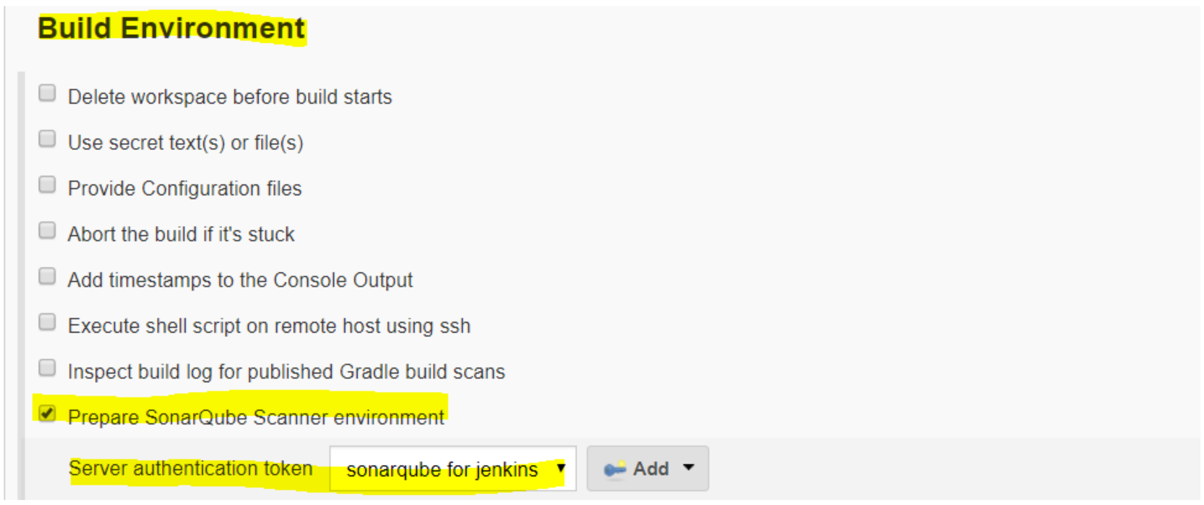
select jdk11 selection

Global tool configuration-> update jdk home path to

```
/usr/lib/jvm/java-11-openjdk-11.XXXXXXXXx.amzn2.0.2.x86_64/
```

Create Jenkins Job:

In Build Environment section: Select Prepare SonarQube Scanner:

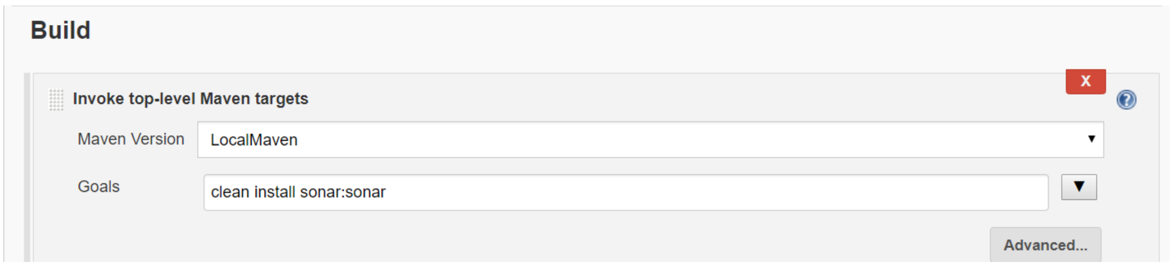


Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Provide Configuration files
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] Execute shell script on remote host using ssh
- [ ] Inspect build log for published Gradle build scans
- [x] Prepare SonarQube Scanner environment

Server authentication token: sonarqube for jenkins

In Build Section:



Build

Invoke top-level Maven targets

Maven Version: LocalMaven

Goals: clean install sonar:sonar

Build

Enjoy 😊

Pipeline Job with SonarQube:

Create Pipeline Job: https://github.com/prakashk0301/maven-sonar/

(Select Branch name: **ci-cd-pipeline-sonar)**

Build

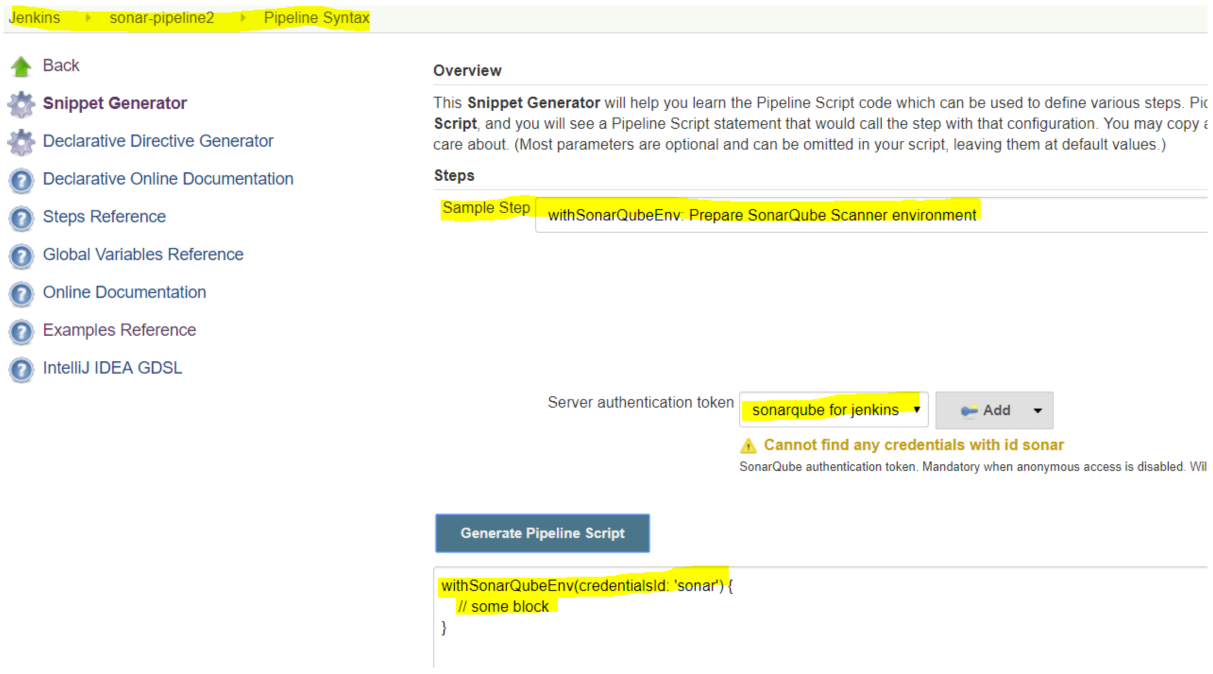
Done ☺

How to generate Jenkins file script for Sonarqube:

From **Pipeline Syntax**, look for **withSonarQubeEnv**

Choose your sonarqube **server authentication token**

Then generate Pipeline script.


Jenkins › sonar-pipeline2 › Pipeline Syntax

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pic
Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy a
care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: withSonarQubeEnv: Prepare SonarQube Scanner environment

Server authentication token: sonarqube for jenkins

Cannot find any credentials with id sonar

SonarQube authentication token. Mandatory when anonymous access is disabled. Wil

```
withSonarQubeEnv(credentialsId: 'sonar') {
    // some block
}
```

Then add stage in Jenkins file.

```
    stage ('build && SonarQube analysis') {
       steps {
                    withSonarQubeEnv('sonar') {
            withMaven(maven : 'LocalMaven' , installationName:  'sonar') {
                sh 'mvn clean package sonar:sonar'
}}}}
```

Some other projects for labs

https://github.com/prakashk0301/sonarqubemaven

Sample Jenkinsfile:

```
pipeline
{
agent any
stages
{

 stage('scm checkout')
 { steps { git branch: 'master', url: 'https://github.com/prakashk0301/maven-
project.git' } }


 stage ('run unit test framework')
 { steps { withMaven(jdk: 'JAVA_HOME', maven: 'MAVEN_HOME')
   {
    sh 'mvn test'
    }
 } }

  stage ('create package & sonar analysis')
 { steps { withMaven(jdk: 'JAVA_HOME', maven: 'MAVEN_HOME')
   {
     withSonarQubeEnv(credentialsId: 'sonar', installationName: 'sonar')
     {
         sh 'mvn package sonar:sonar'
     }
 } }}


}
}
```