**Install SonarQube in ubuntu and integrate with Jenkins**

* **whats is SonarQube**
* **why static code analysis in the list**
* **customize the rules**
* **static code analysis tools is in market list**
* **sonar supports programming list**
* **code quality checks: potentials, rewritten code, excess complexity**
* **its integration with CICD etc. Jenkins**
* **components of SonarQube-**
* **server & scanner**
* **SonarQube best practices - bugs -**
* **SonarQube setup with Jenkins is one time for Devops engineer**
* **Developer & access to developer**
* **Database for SonarQube supported:**

**What is SonarQube?**

SonarQube is an open-source tool for managing the code quality of an application. It manages seven axes of code quality, such as architecture and design, duplications, unit tests, potential bugs, complexities, coding rules, and comments.

**WHY is SonarQube?**

SonarQube is the leading tool for continuously inspecting the Code Quality and Security of your codebases, and guiding development teams during Code Reviews. Covering 27 programming languages, while pairing-up with your existing software pipeline, SonarQube provides clear remediation guidance for developers to understand and fix issues, and for teams overall to deliver better and safer software. With over 225,000 deployments helping small development teams as well as global organizations, SonarQube provides the means for all teams and companies around the world to own and impact their Code Quality and Security.

1. **Importance of Static Code Analysis:**
   * Explore why static code analysis is crucial in the development process.
   * Identify and fix issues early in the development lifecycle.
   * Enhance code maintainability and readability.
   * Ensure adherence to coding standards.
2. **Customizing Rules in SonarQube:**
   * Learn how to tailor SonarQube's rules to fit your project's specific needs.
   * Explore customization options for coding standards, security, and other aspects.
3. **Static Code Analysis Tools:**

Static Code Analysis Tools are used to automatically analyze source code without executing the program. They help developers understand their code base and ensure that it is compliant, safe, and secure1. These tools can identify potential vulnerabilities and weaknesses in the source code1.

Here are some notable static code analysis tools:

Jetbrains , AdaControl Apache Yetus, Astrée Axivion Bauhaus Suite: A static code analysis tool suite that performs various analyses such as architecture checking, interface analyses, MISRA checking, and clone detection2.

Clang: An open-source compiler that includes a static analyzer2.

CodeQL: A code searching tool with an emphasis on finding software bugs2

* + Brief overview of popular static code analysis tools available in the market.
  + Highlight key features and use cases.

1. **Programming Languages Supported by SonarQube:**

* It covers programming languages and formats such as ABAP, C/C++, C#, COBOL, CSS, Erlang, Flex/ActionScript, Groovy, Java, JavaScript, JSON, Objective-C, PHP, PL/I, PL/SQL, Puppet, Python, RPG, Swift, VB.NET, Visual Basic 6, and XML. One of the most striking features is its extensibility. It is easy to cover new languages and add rule engines using an extension mechanism in the form of plugins.
  + List of programming languages supported by SonarQube.
  + Ensure your project's tech stack aligns with SonarQube's capabilities.

1. **Code Quality Checks with SonarQube:**
   * Understand how SonarQube performs code quality checks.
   * Identify potentials, review rewritten code, and manage excess complexity.
2. **Integration with CI/CD (Jenkins):**
   * Explore the seamless integration of SonarQube with CI/CD pipelines, focusing on Jenkins.
   * Set up automated code quality checks within your CI/CD workflow.
3. **Components of SonarQube:**
   * Break down SonarQube into its core components: the server and the scanner.
   * Understand the roles each component plays in maintaining code quality.
4. **SonarQube Best Practices:**
   * Dive into best practices for using SonarQube effectively.
   * Tips for identifying and resolving bugs, enhancing code quality, and ensuring optimal performance.
5. **Setting Up SonarQube with Jenkins: A One-Time Task:**
   * Step-by-step guide on setting up SonarQube integration with Jenkins.
   * Discuss how this one-time configuration benefits DevOps engineers.
6. **User Access and Developer Collaboration:**
   * Explore how developers can leverage SonarQube for code improvement.
   * Discuss user access control and collaboration features.
7. **Database Support in SonarQube:**
   * List databases supported by SonarQube.
   * Ensure compatibility with your preferred database technology.

**LABS**

**Installation steps in aws machine:**

Download Sonar from <http://www.sonarqube.org/downloads/>

* Docker installation on Ubuntu EC2

1. install docker in ubuntu

sudo apt install docker.io -y

2. docker version add permission to docker to ubuntu

sudo usermod -aG docker ubuntu

sudo usermod -aG docker $USER

refresh the group

newgrp docker

to validate the all the permission is successfully done

docker ps

go to the directory 7 file docker.sock provide permissions to all

sudo chmod 777 /var/run/docker.sock

sudo systemctl restart docker

docker install successfully

* After the docker installation, we will create a Sonarqube container (Remember to add 9000 ports in the security group).

<https://www.sonarsource.com/products/sonarqube/downloads/success-download-community-edition/>

Run this command on your EC2 instance to create a SonarQube container:

SonarQube installation on same server with docker images

[SonarQube - Official Image | Docker Hub](https://hub.docker.com/_/sonarqube/)

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

Once Sonar server is up and Running use url to access sonar server dash board.

<http://IP:9000/> or http://<ip\_addr>:9000/

1. Login the sonar dash board using default username and password. The default username and password is admin/admin.

login with admin & admin default username & password

login : admin

sonarqube – aseemadmin

Go to profile- then administrator

sqa\_abed585d20e9b8bfb0ae576c4672c47fb40d371c

sonar proporties

sonar.projectKey=website

jenkins file

node {

stage('SCM') {

checkout scm

}

stage('SonarQube Analysis') {

def scannerHome = tool 'SonarScanner';

withSonarQubeEnv() {

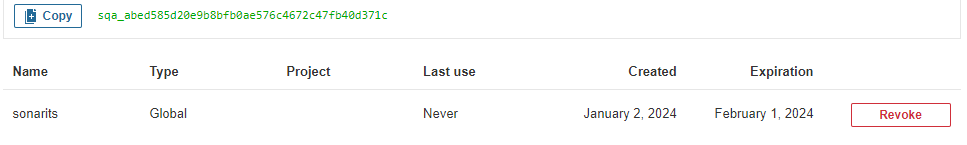
sh "${scannerHome}/bin/sonar-scanner"

}

}

}

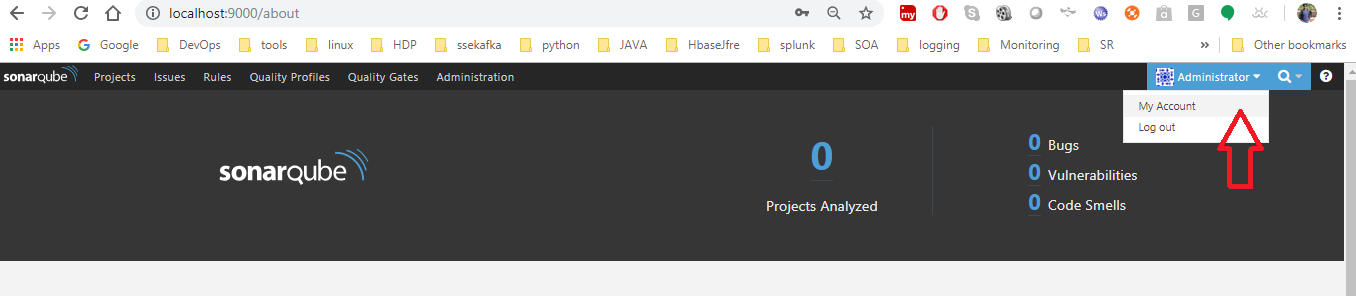
sqa\_85e34a489106004670a610a3a6c5623ac002dd98



sonarqube - itsadmin

update your password & set new password

1. If we want to integrate with Jenkins, we need to create security token trough sonar dash board. Goto administration 🡪 MyAccount 🡪 Security 🡪 enter the name of the token 🡪Generate the token.



Check the updated token list goto Administration 🡪 select security 🡪 Users.

1. Once we can security token with handy now, we can integrate with Jenkins.

Jenkins Installation on EC2 instance via shell script

[Linux (jenkins.io)](https://www.jenkins.io/doc/book/installing/linux/#debianubuntu)

Create a Shell Script to run all instructed command to install Jenkins on Ubuntu jenkins.sh

sudo apt update -y

sudo apt install fontconfig openjdk-17-jre

java -version

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status jenkins

Provide permission to shellscript the read write execute permit

sudo chmod 777 jenkins.sh

sh jenkins.sh

jenkins credentials – to be used

adminsuper

pass – adminsuper@qst

go to jenkins & install sonar-scanner plugin

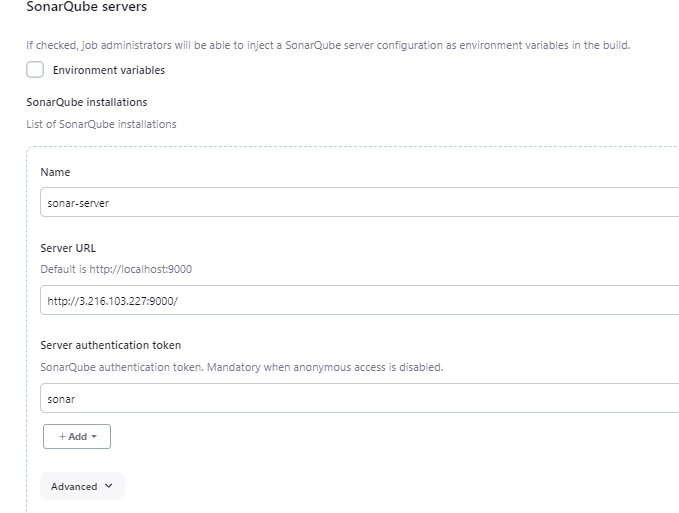
1. Goto Jenkins dashboard Jenkins Home 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Available tab 🡪 search with

**SonarQube Scanner**

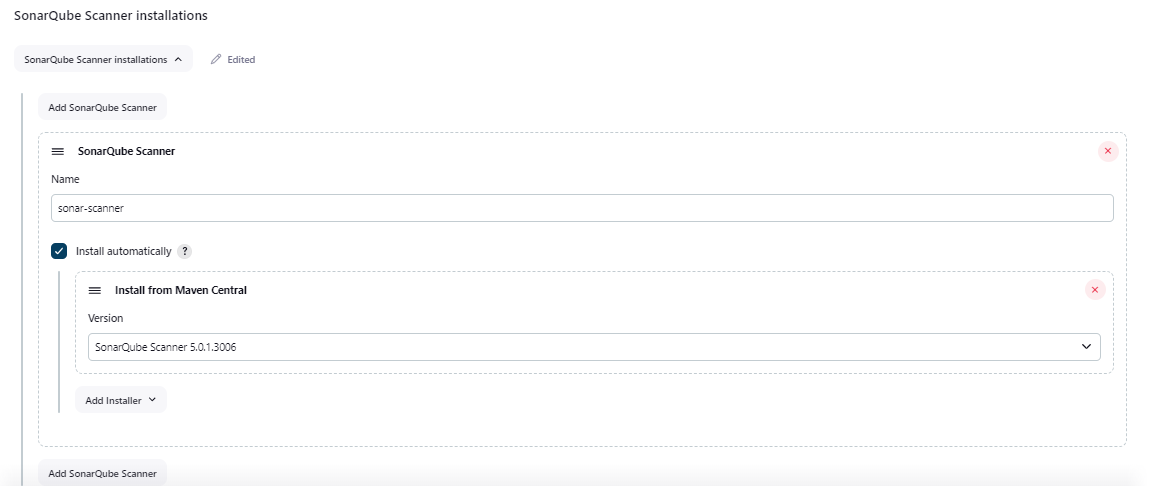
sonar-scanner plugin Pluginselect install without restart.

1. Jdk   
   Eclipse termurin installer plugin

A. dashboard Jenkins Home 🡪 Manage Jenkins 🡪 system🡪



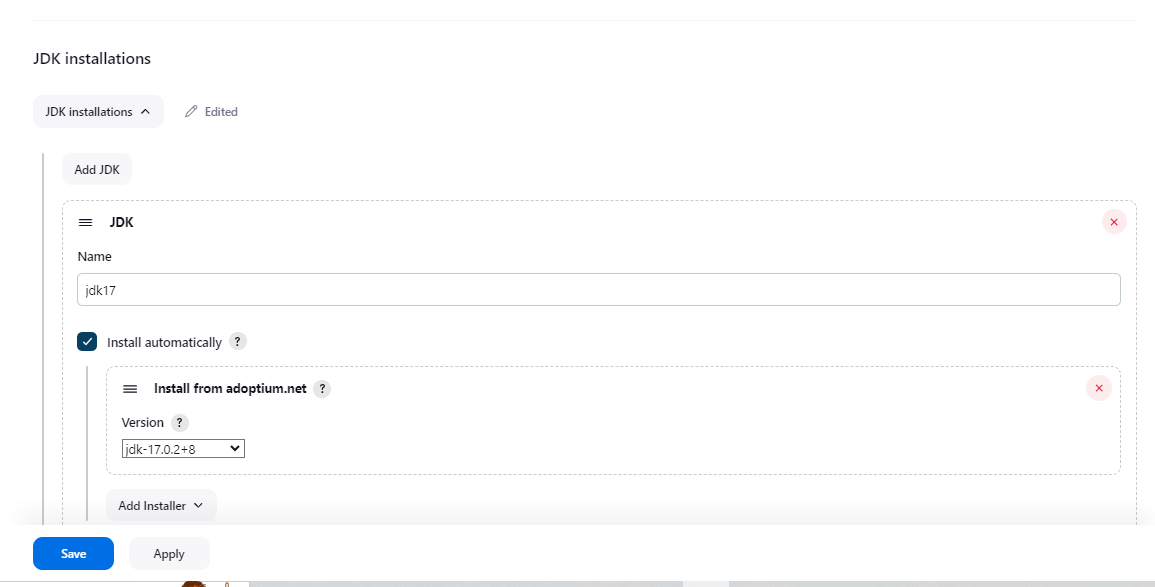
B. dashboard Jenkins Home 🡪 Manage Jenkins 🡪 tools 🡪



1. Jdk   
   Eclipse termurin installer plugin

Once plugin install successfully, we need to configure the sonar server properties. Go to Jenkins home 🡪 Manage Jenkins 🡪 configure system 🡪 SonarQube Servers and add the properties appropriate filed.

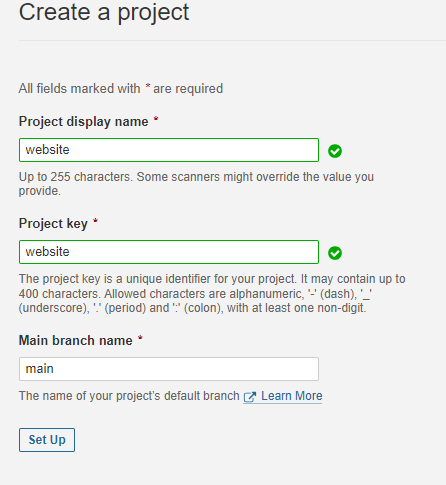
Jdk17 – install from adaptiuim



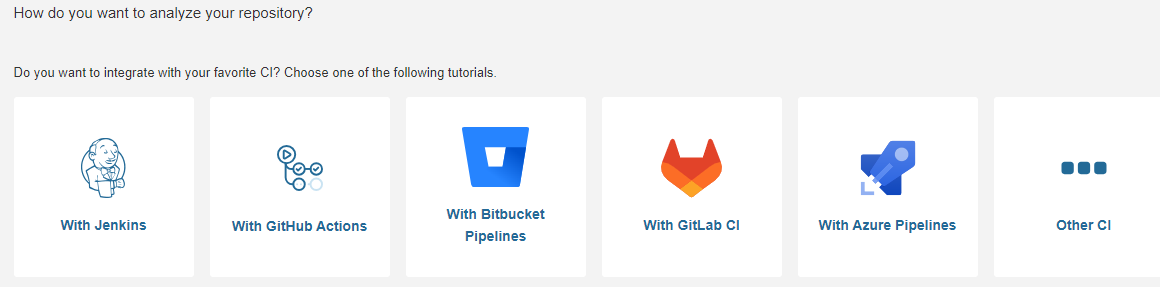
1. And we need to configure **SonarQube Scanner** installation tool in Jenkins 🡪 Global tool configurations 🡪 SonarQube Scanner
2. Once all Sonar-related installations and configurations are completed, we need to add a build step to execute **SonarQube Scanner**. Run the build job with these steps.

We need **sonar-project.properties** to configure Sonar with a specific application. In our sample application, the **sonar-project.properties** file is already available, as shown here:

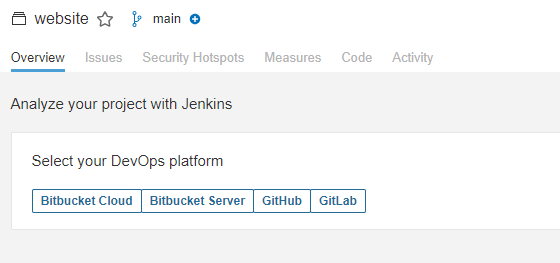
In the PROJECTS section, we can find project details available now. Click on the project name:

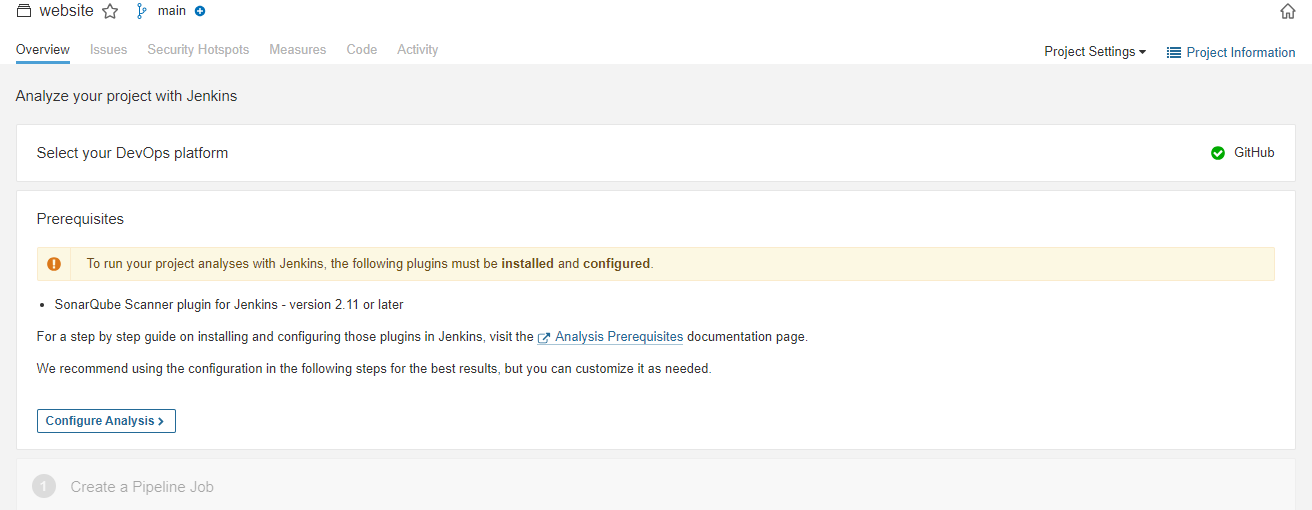


Select Jenkins as CICD tool

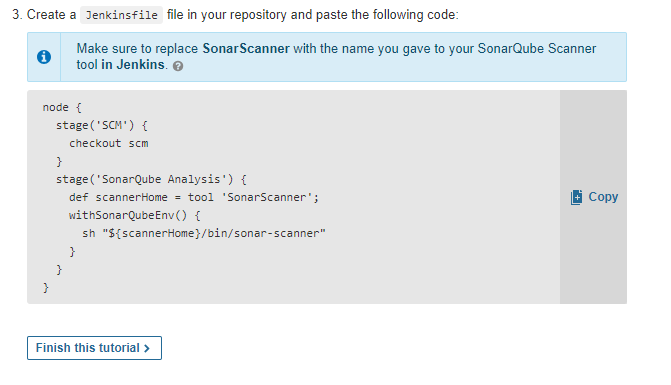


Click on github





Click on next & click on other project



sonar.projectKey=website

node {

stage('SCM') {

checkout scm

}

stage('SonarQube Analysis') {

def scannerHome = tool 'SonarScanner';

withSonarQubeEnv() {

sh "${scannerHome}/bin/sonar-scanner"

}

}

}

Install plugin



