**SONARQUBE**

**Sonar Qube**

* Sonar Qube is Continuous Code Quality Checking Tool
* We can do Code Review using Sonar Qube tool

**What is Code Coverage & Code Review?**

Code Coverage: **How many lines of source code is tested by unit test cases**

Note: **Industry standard Code Coverage is 80 %**

Code Review: **Checking Coding Conventions**

* Sonar Qube is an open source software quality management tool
* It will continuosly analyze and measures quality of the source code
* It will generate code review report in html format / pdf format
* It is a web based tool and it supports 29 Programming Languages
* It will support multi OS platform
* It will support multiple databases (MySQL, Oracle, SQL Server, PostGres SQL...)
* It supports multiple browsers
* Sonar Qube will identify below category of issues in project source code

**1) Duplicate Code**

**2) Coding Standards**

**3) Unit Tests**

**4) Code Coverage**

**4) Complex Code**

**5) Commented Code**

**6) Potential Bugs**

* Initially Sonar Qube was developed only for Java Projects
* Today Sonar Qube is supporting for 29 Languages

**Open source:** Java, Java Script, C#, Web Technologies (HTML, JSP), XML, Python, Groovy, PHP, Puppet etc...

**Commercial:** ABAP, C, C++, COBOL, PL/SQL, Visual Basic, VB.Net, Swift etc..

**Environment Setup**

Java is the pre-requisite software

7.6 --> Java 1.8v

7.8 - 8.x --> Java 11v

Note: We can check this compatibility in official sonar website

**Hardware Requirements**

**Minimum RAM: 2 GB**

**t2.micro ---> 1 GB RAM**

**t2.medium ---> 4 GB RAM**

**Sonar Qube Architecture**

1) SonarQube Scanner

(it will take source code as input and it will generate report)

2) SonarQube Server

SonarQube Server contains 3 components

1) Compute Engine

2) Database

3) WebServer

4) Search Engine

**Compute Engine**

- Vulnerabilities

- Bugs

- Code Smells

Note: Compute Engine will store report into H2 DB (we can configure external db also)

**WebServer** (it will display report in webserver)

Note: Developers will see sonar report and will fix the issues

**Search Engine** (it will give search results)

**Sonarqube set up**

Create EC2 instance with 4 GB RAM (t2.medium)

Connect with EC2 instance

Check space (free -h)

* $ sudo su
* $ yum install wget -y
* $ cd /opt
* $ sudo yum install java-1.8.0-openjdk
* $ java -version
* $ yum install wget unzip -y
* $ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.8.zip
* $ unzip sonarqube-7.8.zip

**Note: SonarQube server will not run with root user**

**Create new user**

* $ useradd sonar
* $ visudo

**Configure sudo user without pwd**

* sonar ALL=(ALL) NOPASSWD: ALL

**Change ownership for sonar folder**

* $ chown -R sonar:sonar /opt/sonarqube-7.8/
* $ chmod -R 775 /opt/sonarqube-7.8
* $ su - sonar (switching to sonar user)

**Goto bin directory then goto linux directory and run sonar server**

* $ sh sonar.sh start

**Check sonar server status**

* $ sh sonar.sh status

**Note: Sonar Server runs on 9000 port number by default**

Enable port number in EC2 VM - Security Group

Access Sonar Server in Browser

URL: http://VM-IP:9000/

Note: We can change sonar port at this location (/opt/sonarqube-7.8/conf)

**Default Credentials of Sonar User is admin & admin**

After login, we can go to Security and we can enable Force Authentication.

Note: Once your work got completed then stops your EC2 instance because we have t2.medium so bill will be generated**.**

**Start sonar server**

**Go to sonar server bin folder**

**Switch to sonar user**

* $ sh sonar.sh start

**Note: Search Engine can't run with root user**

* $ sh sonar.sh status

**Note: If sonar not started, then go to log file and see**

* $ sudo rum -rf /opt/sonar-folder/temp/
* $ cd ../bin/
* $ sh sonar.sh start
* $ sh sonar.sh status

**Access sonar server in browser and login into that**

**After shut down when we want to connect again then use following commands**

* $ sudo su
* $ su - sonar
* $ cd /opt
* $ /sonarqube-7.8/bin/linux-x86-64
* $ sh sonar.sh start

**Integrate Sonar server with Java Maven App**

* Configure Sonar Properties under <properties/> tag in pom.xml

**<sonar.host.url>sonar-server-url</sonar.host.url>**

**<sonar.login>username</sonar.login>**

**<sonar.password>pwd</sonar.password>**

* Go to project pom.xml file location and execute below goal

**$ mvn sonar:sonar**

* After build success, goto sonar dashboard and verify that

**Note: Instead of username and pwd we can configure token also in “pom.xml”**

**Working with sonar token**

* Goto sonar-> Login -> click on profile -> Adminstrator -> Myaccount -> Security -> Generate Token
* Copy the token and configure that token in pom.xml

**<sonar.login>token</sonar.login>**

**<sonar.host.url>sonar-server-url</sonar.host.url>**

**<sonar.login>d0a49d314f5d169eccce35b158eb166e238f3fae</sonar.login>**

* Then build the project using "mvn sonar:sonar" goal
* Clone project repository

**git clone** [**https://github.com/ashokitschool/SB-REST-H2-DB-APP**](https://github.com/ashokitschool/SB-REST-H2-DB-APP)

**Quality Profile**

* For each language sonar qube provided one quality profile with set of rules
* Quality Profile means set of rules to perform code review
* We can create our own quality profile based on project requirement
* Create One Quality Profile

**- Name: SBI\_Project**

**- Language: Java**

**- Parent: None**

**Note: We can make our quality profile as default one then it will be applicable for all the projects which gets reviewed under this sonar server.**

**Note: If we have any common ruleset for all projects then we can create one quality profile and we can use that as parent quality profile for other projects.**

* We can configure quality profile to specific project

**- click on project name**

**- Go to administration**

**- Click on quality profile**

**- Select profile required**

**Quality Gate**

* Quality Gate represents set of metric to identify project quality is Passed or Failed
* Every Project Quality Gate should be passed
* In Sonar We have default Quality Gate
* If required, we can create our own Quality Gate also

**Note:**

* **If project quality gate is failed then we should not accept that code for deployment.**
* **If project is having Sonar issues then development team is responsible to fix those issues**
* **As a DevOps engineer, we will perform Code Review and we will send Code Review report to Development team (we will send sonar server URL to development team)**

**Sonar Server with Jenkins Integration**

**Pre-Requisites**

1) Sonar Qube Server

2) Jenkins Server

* On SonarQube Server Generate a Token
* On Jenkins Server

- install apache maven

- Install Sonar Plugin

- Configure SonarServer with Token

- Install Sonar Scanner

- Run Jenkins Pipeline Job

* Connect to Jenkins server VM using mobaxterm and start sonar server
* Access Sonar Server in browser & generate token
* Connect to Jenkins Server using MobaXterm
* Execute below commands in Jenkins Server VM

**$ sudo su**

**$ cd /opt**

**$ wget https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz**

**$ tar -xvf apache-maven-3.6.3-bin.tar.gz**

Manage Jenkins -> Plugins -> Available **-> Sonar Qube Scanner Plugin ->** Install it

Manage Jenkins **-> Configure System -> Sonar Qube Servers -> Add Sonar Qube Server**

- Name: Sonar-Server-7.8

- Server URL: http://52.66.247.11:9000/ (Give your sonar server url here)

- Add Sonar Server Token

**(Token we should add as secret text)** 31258f2c2e33b9826f4de365742e0d32543377ac

(Save it)

Manage Jenkins -> Global Tools Configuration **- > SonarQube Scanner**

**- Name: Sonar-Scanner-4.7**

**- Select Sonar Version ( SonarQube Scanner - 4.7)**

**(Save it)**

Goto jenkins dashboard -> click on 'New Item -> Enter Item name -> Select Pipeline -> Click on Next and Enter below pipeline script in given text box

**pipeline{**

agent any

environment {

PATH = "$PATH:/opt/apache-maven-3.6.3/bin"

}

stages{

stage('GetCode'){

steps{

git ''https://github.com/ashokitschool/maven-web-app.git"

}

}

stage('Build'){

steps{

sh 'mvn clean package'

}

}

stage('SonarQube analysis') {

// def scannerHome = tool 'SonarScanner 4.0';

steps{

withSonarQubeEnv('Sonar-Server-7.8') {

// If you have configured more than one global server connection, you can specify its name

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

sh "mvn sonar:sonar"

}

}

}

}

}

**MY SCRIPT**

**pipeline{**

agent any

environment {

PATH = "$PATH:/opt/apache-maven-3.6.3/bin"

}

stages{

stage('GetCode'){

steps{

git branch: 'main', credentialsId: 'f6dad8a3-84e7-45b1-9d9f-8a23cb51540e', url: 'https://github.com/Saisunil3012/maven-web-app.git'

}

}

stage('Build'){

steps{

sh 'mvn clean package'

}

}

stage('SonarQube analysis') {

steps{

withSonarQubeEnv('Sonar-Server-7.8') {

sh "mvn sonar:sonar"

}

}

}

}

}