**SonarQube**

* SonarQube is an open-source platform used for continuous code quality management.
* It provides a set of powerful tools and features to analyze, measure, and monitor the quality of source code.
* SonarQube helps developers and development teams to identify and address issues related to code quality, code duplication, security vulnerabilities, and adherence to coding standards.

Key features of SonarQube:

1. **Code Quality Analysis**: SonarQube performs static code analysis to identify bugs, code smells, and vulnerabilities. It analyzes various programming languages such as Java, C/C++, C#, JavaScript, Python, and more.
2. **Code Metrics and Statistics:** SonarQube collects and presents metrics and statistics related to code quality, including code complexity, code duplication, test coverage, and maintainability.
3. **Continuous Inspection:** SonarQube integrates with development tools and build systems, allowing it to perform code analysis automatically during the development process. It provides feedback to developers in real-time, helping them identify and fix issues early on.
4. **Security Vulnerability Detection:** SonarQube includes security plugins that can identify and report common security vulnerabilities in the code, such as SQL injections, cross-site scripting (XSS), and insecure cryptographic algorithms.
5. **Customizable Quality Profiles:** SonarQube allows you to define and customize quality profiles to enforce coding standards and best practices specific to your project or organization.
6. **Integration with CI/CD Pipelines:** SonarQube integrates seamlessly with continuous integration and continuous delivery (CI/CD) pipelines. It can be incorporated into build processes to automatically analyze code and provide quality reports as part of the build pipeline.
7. **Issue Tracking and Management:** SonarQube provides a dashboard and issue tracker to manage identified code issues. It allows developers and teams to prioritize, assign, and track the progress of fixing code quality issues.

By utilizing SonarQube, development teams can maintain high code quality standards, improve software reliability, reduce technical debt, and enhance overall maintainability of their codebase.

**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 / Standards

**More about SonarQube:**

SonarQube will generate code review report in html format / pdf format

It is a web based tool and it supports 25+ Programming Languages

It will support multi OS platform.

It will support multiple databases (MySQL, Oracle, SQL Server, PostGres SQL...).

It supports multiple browsers.

SonarQube will identify below category of issues in project source code:

1) Duplicate Code

2) Coding Standards

3) Unit Tests

4) Code Coverage

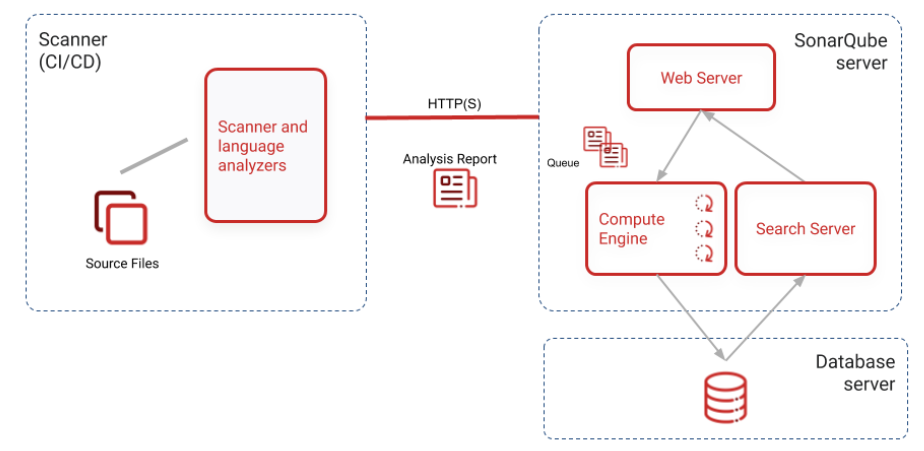
4) Complex Code

5) Commented Code

6) Potential Bugs

Initially SonarQube was developed only for Java Projects. Today SonarQube is supporting for 25+ Languages

**Sonarqube Architecture:-**



SonarQube follows a client-server architecture, where the SonarQube server acts as the central component responsible for code analysis and reporting, and the clients or analysis tools interact with the server to submit code for analysis and retrieve analysis results. Here is an overview of the SonarQube architecture:

1. **SonarQube Server:**

The SonarQube server is the core component of the architecture. It is responsible for coordinating the code analysis process, storing analysis results, managing plugins, and serving the web-based user interface. The server communicates with the underlying database and provides REST APIs for clients to interact with.

1. **Analysis Clients:**

Analysis clients are tools or plugins that interact with the SonarQube server to submit code for analysis. They perform the code analysis locally or within the build environment and send the analysis data to the SonarQube server. Some common analysis clients include SonarScanner (official command-line tool), build system plugins (e.g., Maven, Gradle), and integrated development environment (IDE) plugins (e.g., SonarLint).

1. **SonarQube Database:**

SonarQube utilizes a relational database to store the analysis results and configuration data. By default, it uses an embedded database (H2) for evaluation or small setups, but for production environments, it is recommended to use an external database such as PostgreSQL, MySQL, or Oracle.

1. **Web Interface and Reporting:**

The SonarQube server provides a web-based user interface that allows users to view analysis results, metrics, and reports. It offers dashboards, project overviews, code issue listings, trend charts, and other visualizations to help users understand and improve code quality.

**Environment Setup:**

To install SonarQube, you need to ensure that your system meets the following requirements:

* **Operating System:** 
  1. SonarQube is compatible with various operating systems, including Windows, Linux, and macOS. Supported Linux distributions include Red Hat Enterprise Linux, CentOS, Debian, Ubuntu, and Oracle Linux.
* **Hardware Requirements:**
  1. SonarQube requires a machine with a minimum of 2 CPU cores and 2 GB of RAM.
  2. For better performance, it is recommended to have at least 4 CPU cores and 8 GB of RAM.
  3. Disk space requirements depend on the number of projects and the size of the codebase being analyzed. A minimum of 1 GB of free disk space is recommended.
  4. Java Development Kit (JDK):
     1. SonarQube requires Java to be installed on the machine.
     2. SonarQube 9.0 and later versions require Java 11 or later.
     3. **SonarQube 8.x versions support Java 11**
     4. Make sure you have a compatible JDK installed and configured properly.
* **Database:**
  1. SonarQube requires a database to store analysis results and other data.
  2. Supported databases include PostgreSQL, Microsoft SQL Server, Oracle, and MySQL.
  3. SonarQube also supports the use of an embedded database (H2) for evaluation or small setups, but it is not recommended for production environments.
* **Network Connectivity:**
  1. SonarQube requires network connectivity to download plugins, updates, and dependencies during installation and runtime.

Ensure that the server has access to the internet or the necessary network resources.

* **Create EC2 instance with 4 GB RAM (t2.medium) (Amazon Linux AMI)**
* **Connect with EC2 instance using MobaXterm**
* **Check space (free -h)**

**SonarQube Install in Amazon Linux 2023:**

**1. Switch to root user**

$ sudo su -

**2. Install JDK 11**

$ yum install java-11-amazon-corretto -y

**3. Download SonarQube 8.9 LTS**

$ wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.10.61524.zip> -P /tmp

**4. Unzip to /opt location**

$ unzip /tmp/sonarqube-8.9.10.61524.zip -d /opt

**5. Rename to sonarqube (remove version for better naming)**

$ mv /opt/sonarqube-8.9.10.61524/ /opt/sonarqube

**6. Create user sonar (we cannot run sonarqube as root user)**

$ useradd sonar

**7. Modify Sudoer file to use sonar user without password**

$ visudo

**# add below line after root line**

$ sonar ALL=(ALL) NOPASSWD: ALL

**8. Modify owner and file permissions for sonarqube directory**

$ chown -R sonar:sonar /opt/sonarqube/

$ chmod -R 775 /opt/sonarqube/

**9. Switch to sonar user**

su - sonar

**10. create sonarqube as service**

**$** sudo nano /etc/systemd/system/sonarqube.service

**# add below lines then (ctrl+O , Press Enter, ctrl+X)**

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar

Group=sonar

Restart=always

[Install]

WantedBy=multi-user.target

**11. Enable and Start SonarQube, then check status**

$ sudo systemctl enable sonarqube

$ sudo systemctl start sonarqube

$ sudo systemctl status sonarqube

**12. Default port number is 9000 (Allow in EC2 Instance Security Group)**

**$** ls -l /opt/sonarqube/conf/

$ cat /opt/sonarqube/**conf/sonar.properties** | grep "port="

**\*) Notes: "/opt/sonarqube/logs/" Log files location incase of any issues in install/start**

**13. Access Sonar Server in Browser**

URL : http://EC2-VM-IP:9000/

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

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

**INTEGRATE SONAR SERVER WITH JAVA MAVEN APP:**

**(come back to ec2-user, execute exit command twice from sonar user and then)**

1. Install Git and Maven

$ sudo yum install git -y

$ sudo wget https://dlcdn.apache.org/maven/maven-3/3.8.8/binaries/apache-maven-3.8.8-bin.zip -P /tmp

$ sudo unzip /tmp/apache-maven-3.8.8-bin.zip -d /opt/

$ sudo mv /opt/apache-maven-3.8.8/ /opt/maven

$ sudo nano /etc/profile.d/maven.sh

export M2\_HOME=/opt/maven

export PATH=${M2\_HOME}/bin:${PATH}

**CTRL+O, ENTER AND CTRL+X ENTER**

$ sudo chmod +x /etc/profile.d/maven.sh

$ source /etc/profile.d/maven.sh

$ mvn -version

**2. Clone GitHub Project:**

$ git clone <https://github.com/javabyraghu/SB-REST-H2-DB-APP.git>

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

<properties>

<sonar.host.url>http://ec2-public-ip:9000/</sonar.host.url>

<sonar.login>admin</sonar.login>

<sonar.password>**PASSWORD**</sonar.password>

</properties>

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

$ mvn sonar:sonar

5. After build success, go to sonar dashboard and verify that

Note: Instead of username and password we can configure sonar token in pom.xml

**WORKING WITH SONAR TOKEN:**

Go to Sonar -> Login -> Click on profile -> My Account -> Security -> Generate Token

**Copy the token and configure that token in pom.xml file like below**

<sonar.host.url>http:// ec2-public-ip:9000/</sonar.host.url>

<sonar.login>ff4d464eda3eccdea05d77b742767c777545863e</sonar.login>

Then build the project using "mvn sonar:sonar" goal

**SONARQUBE QUALITY PROFILE AND QUALITY GATEWAY:**

* **Rule:** In SonarQube, a rule represents a specific code quality issue or best practice that needs to be enforced. It defines a particular coding pattern or behaviour that is considered either a bug, vulnerability, or code smell. For example, a rule might check for the use of ***hardcoded passwords, the absence of unit tests, or the presence of unused variables.*** SonarQube provides a wide range of pre-defined rules, but you can also create custom rules to address specific requirements.
* **Quality Profile:** A quality profile in SonarQube is a ***collection of rules*** that are applied to analyse the source code of a project. It defines which rules should be active or inactive for a particular project or programming language. SonarQube supports multiple programming languages such as Java, C#, JavaScript, Python, and more. Each quality profile can be customized to include or exclude specific rules based on the project's requirements and coding standards. By associating a quality profile with a project, SonarQube ensures that the defined rules are enforced during code analysis.
* **Quality Gate:** A quality gate is a ***set of predefined conditions or thresholds*** that must be met for a project's code quality to be considered acceptable. It acts as a checkpoint during the software development process, helping to ensure that code quality remains within the defined limits. A quality gate is associated with one or more metrics, such as ***code coverage, code duplication, number of bugs, vulnerabilities, and maintainability issues***. When code is analysed in SonarQube, the results are evaluated against the defined quality gate conditions. If any of the conditions are not met, the quality gate fails, indicating that the code does not meet the desired quality standards.

**We can provide our custom Quality profile and Quality Gate details in maven project using pom.xml properties *[works for older version of SonarQube 6.1 or before]***

<properties>

….

<sonar.profile>**your-custom-profile-name**</sonar.profile>

<sonar.qualitygate>**your-custom-quality-gate-name**</sonar.qualitygate>

</properties>

**\*) In latest version we need to specify directly in sonar server under project settings.**

**\*) If Application is non-spring boot then it is recommended to add Sonar plugin inside pom.xml as**

<build>

<plugins>

<plugin>

<groupId>org.sonarsource.scanner.maven</groupId>

<artifactId>sonar-maven-plugin</artifactId>

<version>3.9.1.2184</version>

</plugin>

</plugins>

</build>

**CREATING NEW USER IN SONARQUBE:**

To add a user in SonarQube, you need to have administrative privileges.

1. Log in to your SonarQube server with an account that has administrative access.
2. Once logged in, click on the user icon (usually located in the top-right corner) and select "Administration" from the dropdown menu. This will take you to the SonarQube administration page.
3. In the left sidebar of the administration page, under the "Security" section, click on "Users." You will see a list of existing users.
4. To add a new user, click on the "Create User" button.
5. Fill in the required information for the new user in the provided fields. The mandatory fields usually include:

Login: The unique username for the user.

Name: The full name of the user.

Password: The password for the user. You can choose to generate a password or set one manually.

Confirm Password: Re-enter the password for confirmation.

Email: The email address of the user.

1. Optionally, you can assign the user to specific groups by selecting the desired groups from the "Groups" field. This determines the user's access rights and permissions within SonarQube.
2. Once you have filled in the required information, click on the "Create" button to add the user.

The new user will now be added to the SonarQube server with the specified login, name, and assigned groups. The user can use the provided credentials to log in to SonarQube and access the features based on their assigned permissions.