SonarQube

SonarQube is an **open-source platform** for **continuous inspection of code quality**.
It analyzes **bugs, vulnerabilities, code smells, security hotspots, and test coverage**, and integrates with CI/CD pipelines.

Key SonarQube Concepts

Term	Meaning
Bug	Code defect that can cause incorrect behavior.
Vulnerability	Security risk in the code.
Code Smell	Maintainability issue that doesn't break functionality but makes code harder to manage.
Security Hotspot	Code that might be vulnerable and needs manual review.
Quality Gate	Rules that decide if the code passes or fails the quality check.
Rule	A guideline for writing clean code (e.g., no unused imports).
Profile	Collection of rules for a language.
Project Key	Unique identifier for your project in SonarQube.

SonarQube Architecture

- **Server** – Hosts the dashboard and analysis reports.
- **Database** – Stores analysis results (PostgreSQL, MySQL, etc.).
- **Scanner** – CLI tool that runs analysis.
- **Plugins** – Extend functionality (e.g., for languages, security rules).
- **Authentication** – Local users, LDAP, or SSO.

Integrating SonarQube in CI/CD (Jenkins Example)

```
pipeline {
  agent any
  tools {
    jdk 'JAVA_HOME'
    maven 'maven3'
  }
  environment {
    SCANNER_HOME = tool 'sonar'
  }
  stages {
    stage('Checkout') {
      steps {
        git branch: 'main', url: 'https://github.com/org/repo.git'
      }
    }
    stage('SonarQube Analysis') {
      steps {
        withSonarQubeEnv('sonar') {
          sh '''$SCANNER_HOME/bin/sonar-scanner \
           -Dsonar.projectKey=myproject \
           -Dsonar.sources=. \
           -Dsonar.java.binaries=target/classes'''
        }
      }
    }
    stage('Quality Gate') {
      steps {
        timeout(time: 2, unit: 'MINUTES') {
          waitForQualityGate abortPipeline: true
```

```
            }
          }
        }
    }
}
```

Quality Gates

A **quality gate** is a set of conditions for passing the analysis:

- No new bugs/vulnerabilities
- Test coverage ≥ 80%
- Maintainability rating = A

Failing the quality gate should block deployments in CI/CD.

Real-World DevOps SonarQube Workflow

1. **Developer pushes code** to GitHub.
2. **CI pipeline** triggers SonarQube scan.
3. **Scanner** uploads results to SonarQube server.
4. **Quality gate** checks results.
5. If passed → Deploy to staging.
6. If failed → Stop pipeline, assign issues to developers.

Installation (on Docker or VM)

Option 1: Install SonarQube using Docker

```
# Pull the SonarQube image
sudo docker pull sonarqube:latest

# Run the container
sudo docker run -d \
  --name sonarqube \
  -p 9000:9000 \
  sonarqube:latest
```

- Access SonarQube on http://localhost:9000
- Default credentials: admin / admin

Option 2: Install SonarQube on VM (Ubuntu Example)

```
# Install Java 11 or higher
sudo apt update && sudo apt install openjdk-11-jdk -y

# Create SonarQube user
groupadd sonar && useradd -c "SonarQube" -d /opt/sonarqube -g sonar sonar

# Download and extract SonarQube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-<version>.zip
unzip sonarqube-<version>.zip -d /opt/
chown -R sonar:sonar /opt/sonarqube

# Start SonarQube
cd /opt/sonarqube/bin/linux-x86-64/
./sonar.sh start
```

Hands-on Lab: Install and Explore SonarQube Dashboard

Step-by-Step Lab:

1. Install SonarQube using Docker or on a local VM.
2. Access http://<your-ip>:9000 in a browser.
3. Login with admin credentials.
4. Create a new project manually.
5. Generate and configure the token.
6. Run the scanner from CLI:

```
sonar-scanner \
  -Dsonar.projectKey=myproject \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=<your_token>
```

7. View project quality overview, bugs, code smells, coverage, and duplications.

Real-time Use Case: During CI/CD execution, if SonarQube reports the Quality Gate has failed due to a newly introduced critical vulnerability, the deployment is halted. The developer is notified to fix the issue before re-submitting the code.

Why Sonar Token?

- A Sonar Token is required for authenticating SonarScanner or CI/CD tools (e.g., Jenkins) to interact with SonarQube securely.

Steps to Generate Token:

7. Log in to SonarQube dashboard (URL like http://<sonarqube-server>:9000).
8. Go to **My Account** (top-right corner).
9. Navigate to the **Security** tab.
10. Under **Generate Tokens**, enter a token name (e.g., jenkins-access-token).
11. Click **Generate**, then **copy and save the token** securely (you won't see it again).

****