SONARQUBE

It Will support more than 20+ Languages.

DefaultPort: 9090.

configuration files:

sonar.properties (file) --- loglevel, databasesconnections,

Below Sonar Cube Metrics:

1. **Common anti-patterns and coding flaws that can lead to bugs:** These SonarQube metrics are similar to what static code analysis tools, such as PMD and FindBugs, typically report.
2. **Breaches of coding standards and conventions:** These SonarQube metrics are similar to what might be generated by the [Maven CheckStyle Plugin](https://www.theserverside.com/video/Use-Maven-Checkstyle-Plugin-to-enforce-Java-quality-rules).
3. **Copy and pasting of code:** This causes needless bloat and duplication.
4. **A lack of**[**unit tests**](https://searchsoftwarequality.techtarget.com/definition/unit-testing)**and less-than-satisfactory unit test code coverage:** This is similar to the type of metrics generated by the JaCoCo Maven Plugin.
5. **Excessive code complexity:** This is indicated by a [McCabe cyclomatic complexity](https://www.theserverside.com/feature/How-to-calculate-McCabe-cyclomatic-complexity-in-Java) calculation performed on the source code.
6. **A spaghettilike design:** This makes tracing application flow difficult.
7. **A lack of comments or too many comments:** Nobody is quite sure if it's good or bad to comment code. Whichever side of the fence you sit on, there's a SonarQube metric to help you quantify what is going on in the code base.

**What is a Quality Gate?**

**Quality Gates** are the best way to ensure that standards are met and regulated across all the projects in your organization.Quality Gates can be defined as a set of threshold measures set on your project like Code Coverage, Technical Debt Measure, Number of Blocker/Critical issues, Security Rating/ Unit Test Pass Rate and more.

To pass the Quality Gates, the project should pass through each of the thresholds set.

When SonarQube runs it will identify if the code meets all the quality thresholds you have set – else it will fail the Quality Gate and will not allow you to check in code to source control. This is a very powerful feature since it enforces code quality in your projects and automates the process.

Jenkins:

configuration files:

1) /var/lib/jenkins/jobs ---------------------> It will store all jobs

2) /var/lib/jenkins/config.xml---------------------> Basic configuration file

3) ./jenkins.sh-------------------------------------> To start jenkins

4) Major Plugins:

1) Git plugin -------Useful to connect git(scm)

2) SSh copy plugin -------------It will copy files to destinnation

3) Blueu Ocean ------------------It will change view of the jenkins

4) Git Parameter Plug-in------------It will provide git credentials as parameters.

5) Extended Mail Plug-in------------It will send mail to respected person when specific event occurse

6) RoleBack plugin------------------

7) Maven Plugin

9) Sonar Scanner

10) Backupplugin --------------To take a backup

11) AnsiblePlugin --------------TO contact ansible server

12) DockerPlugin -------------To connect docker container

13) PublishOverSSH -------------Sending and running a commands in target machine

14) EmbededBuildStatusPlugin-----It will display the job status.

15) BigBucket-------------------To integrate with bigbucket

16) GreenBalls -----------------To change color of succes blue to grean

17) Junit-----------------------To publish junit testcases.

18) Rebuild---------------------With out entering parameters again when build rerun this plugin will enter paremeters again

19) JobGeneratorPlugin--------This plugin gibves flexiblity to define templat4es,

Developers will create own jobs

20) Role Based authentication --------To give authrization to user

21) Configuration Slicing Plugging--------TO change bulk changes in multiple jobs.