sonarqube

# Introduction to SonarQube

Chris Vogel
cjvogel1972 cjvogel1972

## About Me

- Technical Architect @ Edward Jones
- Java developer for 22 years
- Background:
  - Application Architecture
  - Mentoring
  - Frameworks
  - Developer tools
  - Infrastructure

## Agenda

- What is SonarQube
- Architecture
- Installation
- Rules/Quality Profiles/Issues
- Quality Gates
- Run Scans
- SonarLint

sonarqube

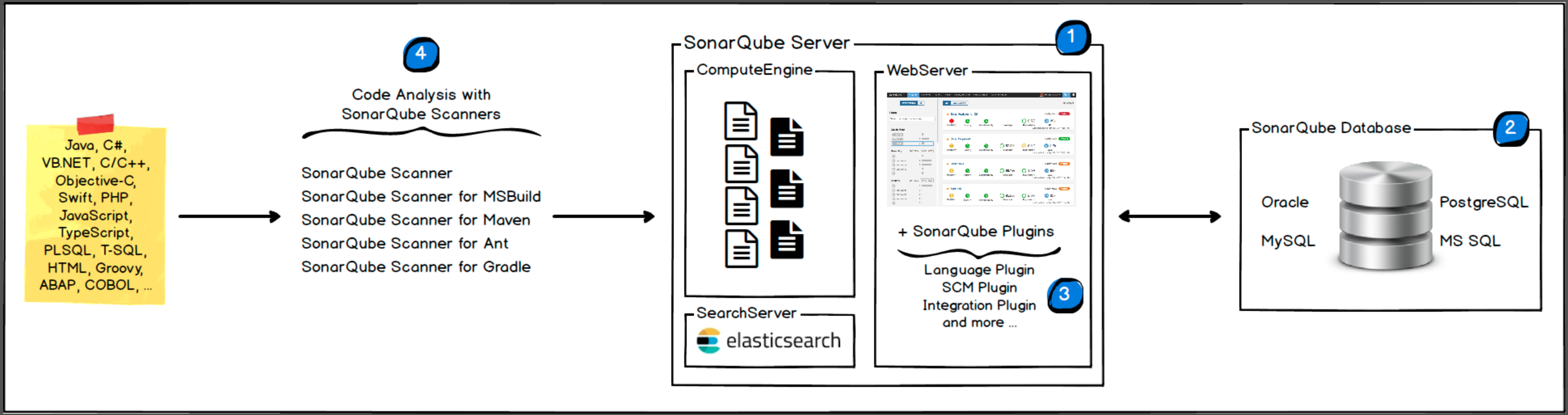
## SonarQube

- Open-source code quality tool
- Performs static code analysis
- Understands 15 languages
- Extensible by plugins (60+)
- Works with CI tools
- Integrates with IDEs

## SonarQube

- Company SonarSource
- Licensed products
  - Developer Edition
  - Enterprise Edition
  - Data Center Edition
- Additional languages and features

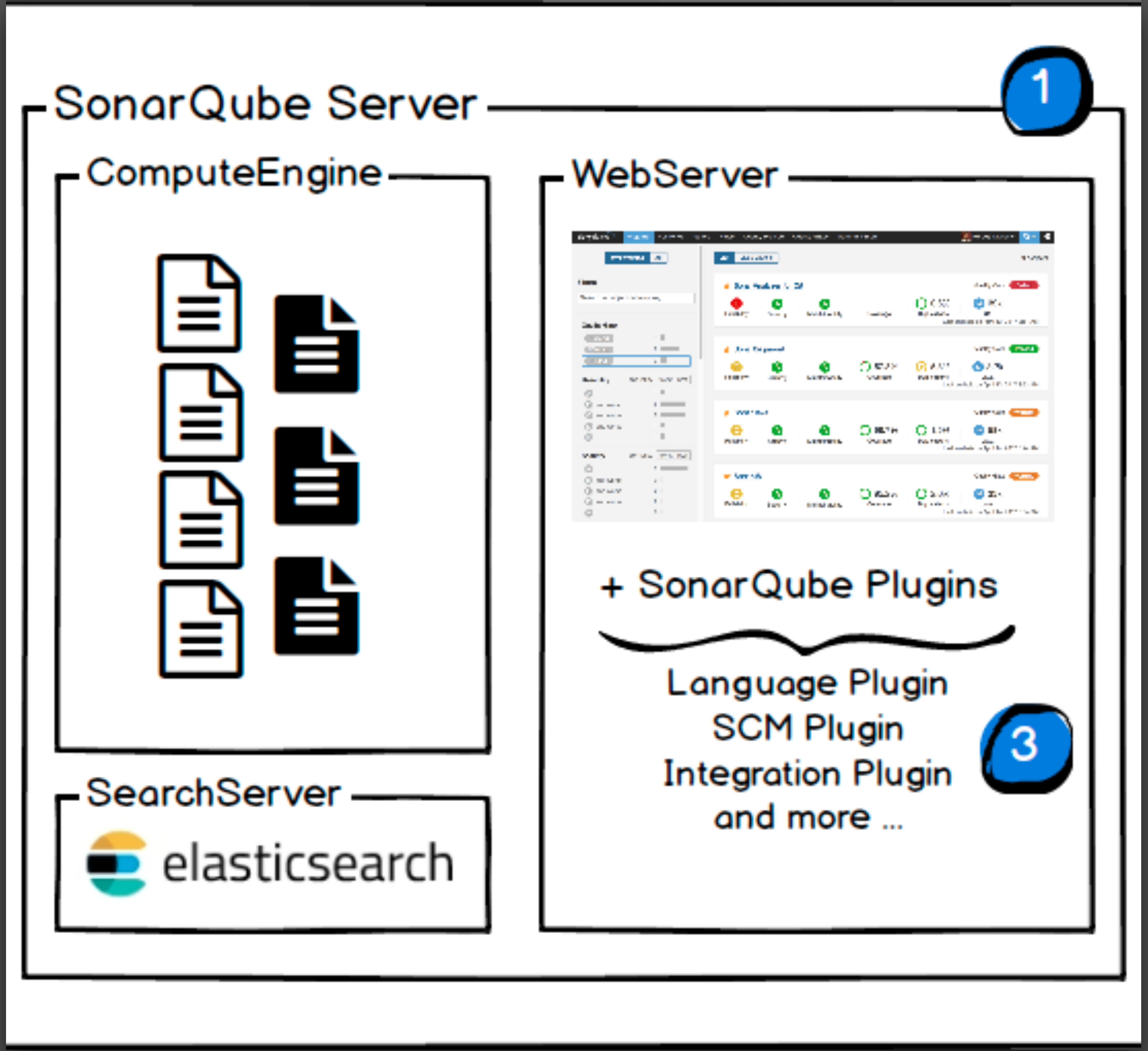
# Architecture


Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JavaScript, TypeScript, PLSQL, T-SQL, HTML, Groovy, ABAP, COBOL, ...
4
Code Analysis with SonarQube Scanners
SonarQube Scanner
SonarQube Scanner for MSBuild
SonarQube Scanner for Maven
SonarQube Scanner for Ant
SonarQube Scanner for Gradle
SonarQube Server
1
ComputeEngine
WebServer
+ SonarQube Plugins
Language Plugin
SCM Plugin
Integration Plugin
and more ...
3
SearchServer
elasticsearch
SonarQube Database
2
Oracle
MySQL
PostgreSQL
MS SQL

SonarQube Architecture

## Architecture - SonarQube Server

1. One SonarQube Server starting 3 main processes
   - Web Server for developers, managers to browse quality snapshots and configure the SonarQube instance
   - Search Server based on Elasticsearch to back searches from the UI
   - Compute Engine Server in charge of processing code analysis reports and saving them in the SonarQube Database

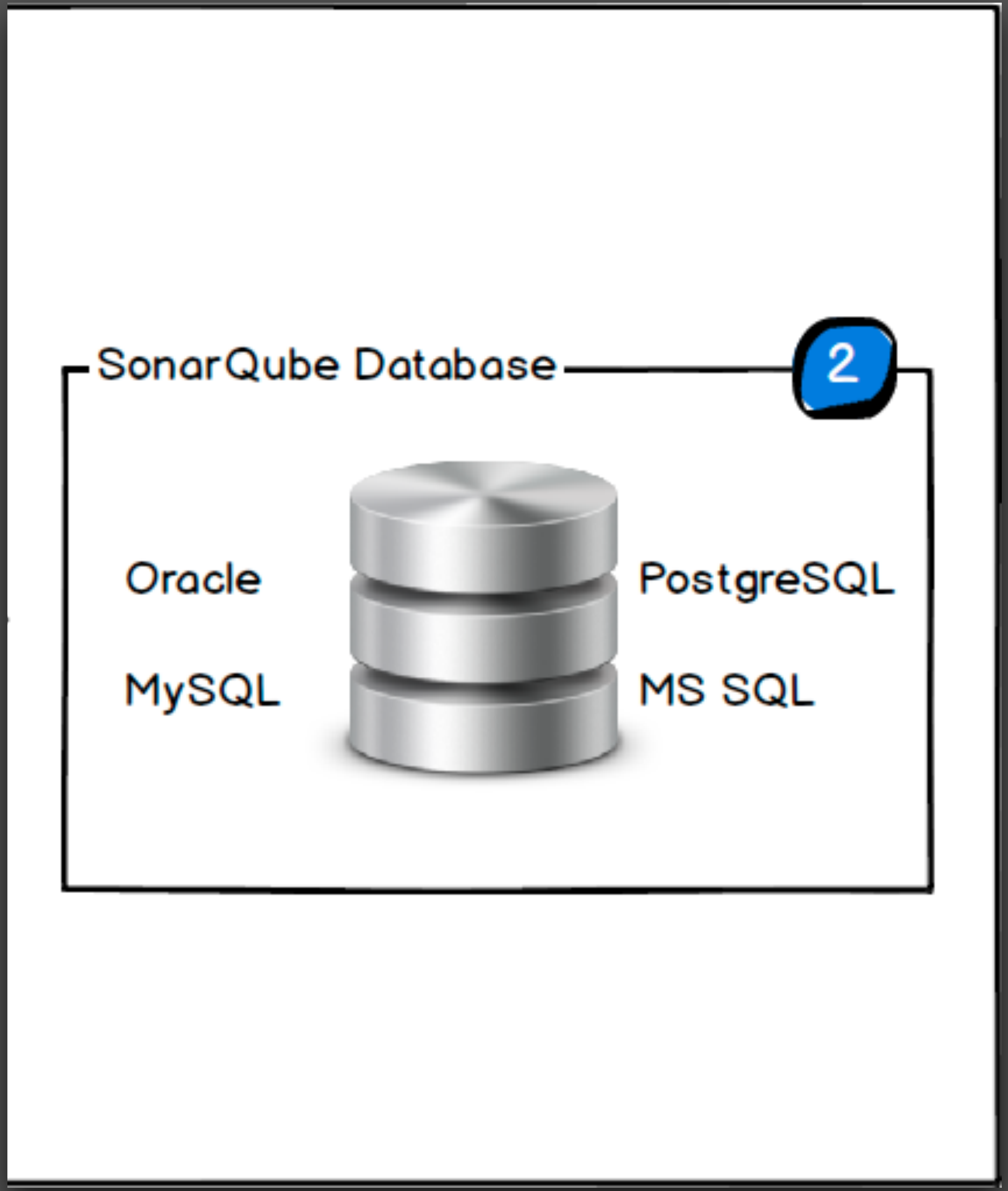


## Architecture - SonarQube Database

2. One SonarQube Database to store

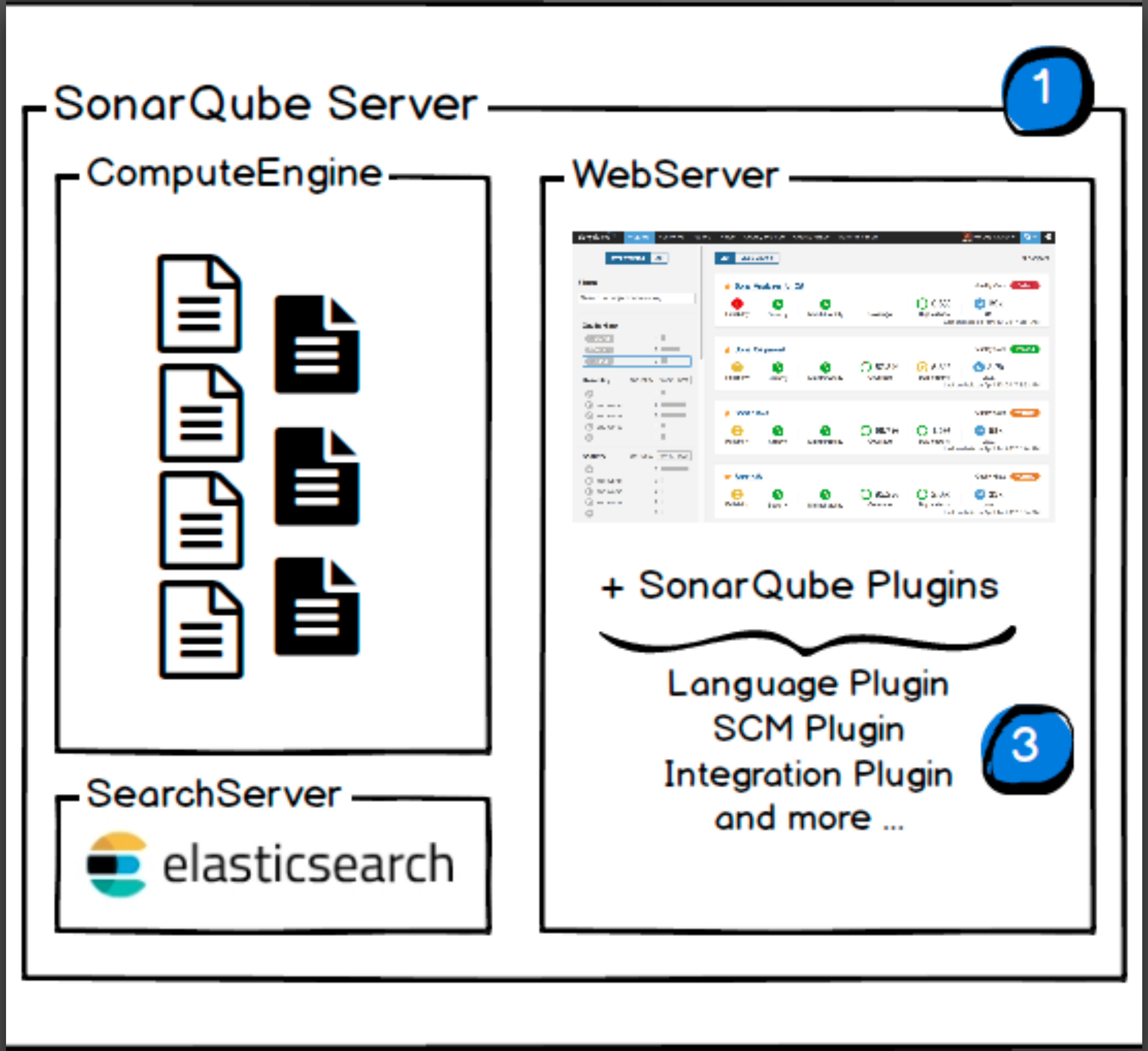
    - the configuration of the SonarQube instance (security, plugins settings, etc.)

    - the quality snapshots of projects, views, etc.



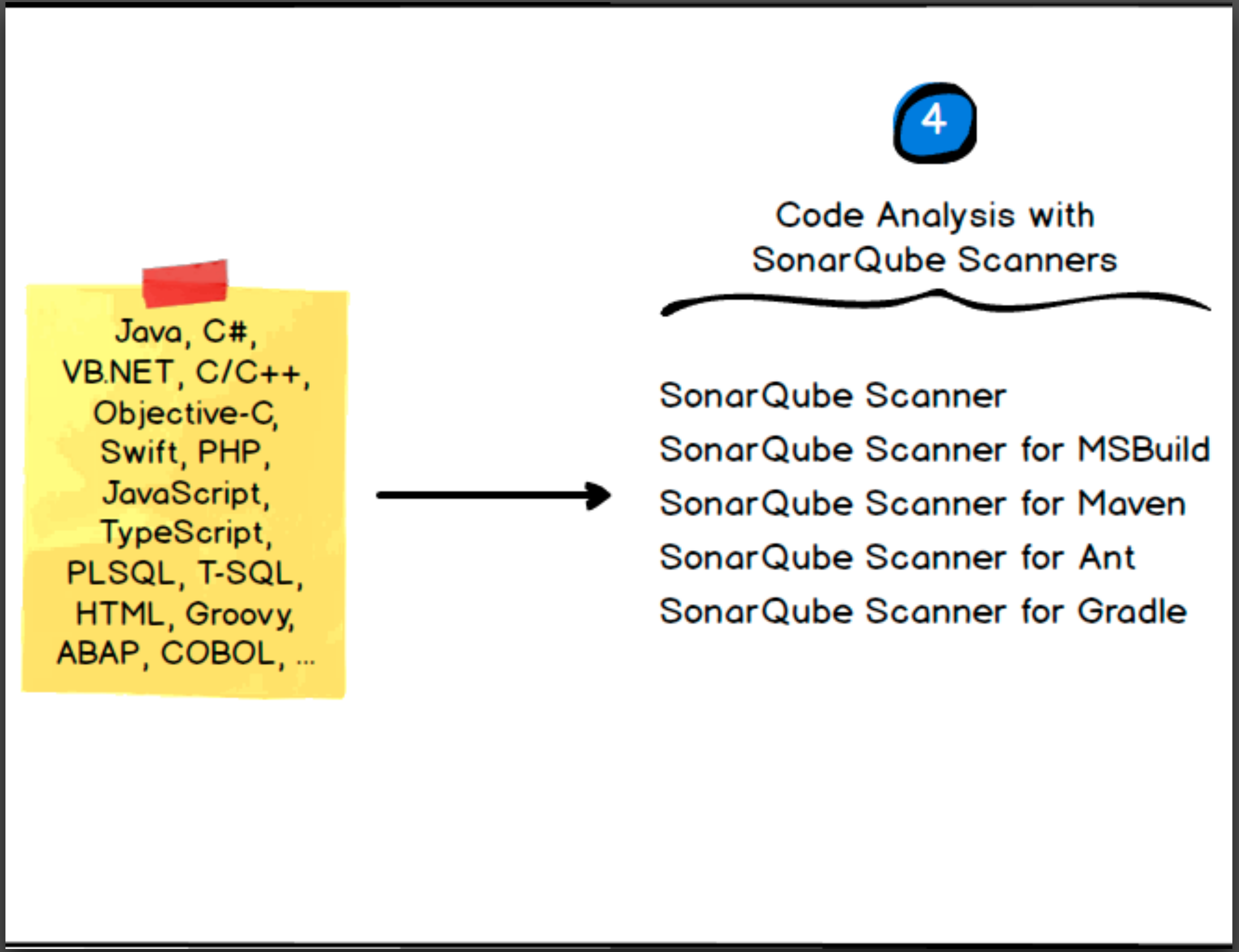
## Architecture - SonarQube Plugins

3. Multiple SonarQube Plugins installed on the server, possibly including language, SCM, integration, authentication, and governance plugins



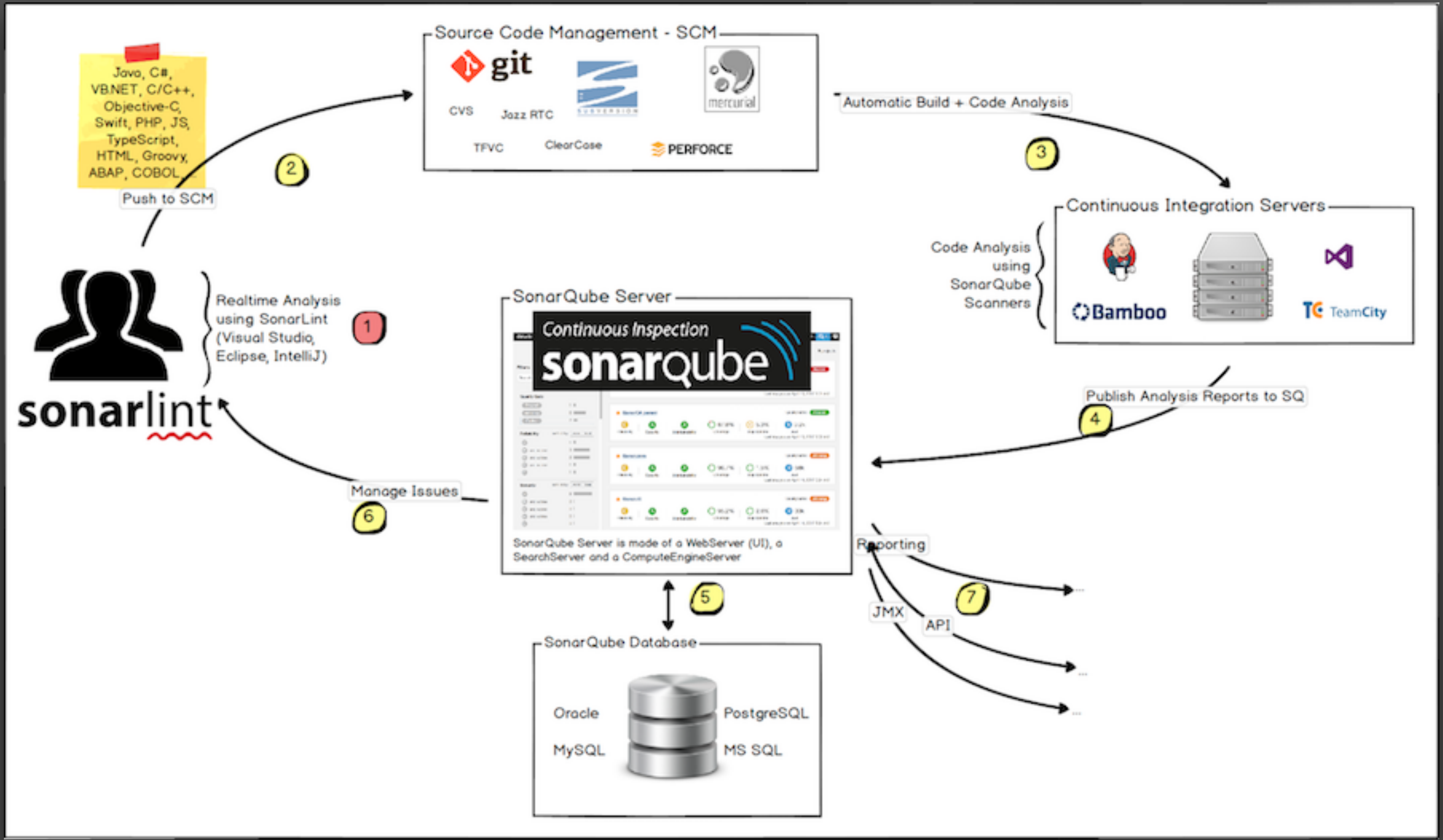
## Architecture - SonarScanners

4. One or more SonarScanners running on your Build / Continuous Integration Servers to analyze projects


Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JavaScript, TypeScript, PLSQL, T-SQL, HTML, Groovy, ABAP, COBOL, ...
4
Code Analysis with SonarQube Scanners
SonarQube Scanner
SonarQube Scanner for MSBuild
SonarQube Scanner for Maven
SonarQube Scanner for Ant
SonarQube Scanner for Gradle

# Integration

1. Developers code in their IDEs and use SonarLint to run local analysis.
2. Developers push their code into their favorite SCM : git, SVN, TFVC, ...
3. The Continuous Integration Server triggers an automatic build, and the execution of the SonarScanner required to run the SonarQube analysis.
4. The analysis report is sent to the SonarQube Server for processing.
5. SonarQube Server processes and stores the analysis report results in the SonarQube Database, and displays the results in the UI.
6. Developers review, comment, challenge their Issues to manage and reduce their Technical Debt through the SonarQube UI.
7. Managers receive Reports from the analysis. Ops use APIs to automate configuration and extract data from SonarQube. Ops use JMX to monitor SonarQube Server.


Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JS, TypeScript, HTML, Groovy, ABAP, COBOL, ...
Push to SCM
2
Source Code Management - SCM
git
CVS
Jazz RTC
mercurial
TFVC
ClearCase
PERFORCE
Automatic Build + Code Analysis
3
Continuous Integration Servers
Code Analysis using SonarQube Scanners
Bamboo
TeamCity
Publish Analysis Reports to SQ
4
Realtime Analysis using SonarLint (Visual Studio, Eclipse, IntelliJ)
1
sonarlint
SonarQube Server
Continuous Inspection
sonarqube
SonarQube Server is made of a WebServer (UI), a SearchServer and a ComputeEngineServer
Manage Issues
6
5
SonarQube Database
Oracle
PostgreSQL
MySQL
MS SQL
Reporting
JMX
API
7
...
...
...

# Installation

- Download zip file (https://www.sonarqube.org/downloads/)
- Thorough documentation (https://docs.sonarqube.org/latest/)
- Requires Java 8
- Recommended 8 cores & 16GB RAM
- External database (Derby or H2 built-in)
  - Oracle
  - PostgreSQL
  - SQL Server

## Installation - The EASY way

- Docker image

```
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

- Only community edition
- “Suitable for demonstration and testing purposes only”

## Rules

- Analyzers contribute rules to be executed on code
- Four types of rules
  - Code Smell (Maintainability domain)
  - Bug (Reliability domain)
  - Vulnerability (Security domain)
  - Security Hotspot (Security domain)
- Provide compliant and non-compliant code examples
- Tags used to classify rules

## Rules

- Rules can be customized
    - Can add/remove tags
    - Can extend description for organization clarification
- Can create custom rules
- Templates provided to aid in rule creation

## Quality Profiles

- A grouping of rules for a language
- Profiles can be extended
- Severity of rules can be modified
- Built-in profiles, “Sonar Way”
- Encouraged to create own profiles
- Ideally, use one profile per language (the default profile)
- New projects are assigned the default profile
- Projects can use the non-default profile

## Issues

- Created when code breaks a coding rule
- Severities
  1. BLOCKER
     Bug with a high probability to impact the behavior of the application in production: memory leak, unclosed JDBC connection, .... The code MUST be immediately fixed.
  2. CRITICAL
     Either a bug with a low probability to impact the behavior of the application in production or an issue which represents a security flaw: empty catch block, SQL injection, ... The code MUST be immediately reviewed.
  3. MAJOR
     Quality flaw which can highly impact the developer productivity: uncovered piece of code, duplicated blocks, unused parameters, ...
  4. MINOR
     Quality flaw which can slightly impact the developer productivity: lines should not be too long, "switch" statements should have at least 3 cases, ...
  5. INFO
     Neither a bug nor a quality flaw, just a finding.

## Issues

- Statuses
  - Open - set by SonarQube on new issues
  - Confirmed - set manually to indicate that the issue is valid
  - Resolved - set manually to indicate that the next analysis should Close the issue
  - Reopened - set automatically by SonarQube when a Resolved issue hasn't actually been corrected
  - Closed - set automatically by SonarQube for automatically created issues.

## Issues

- Resolutions - Closed
  - Fixed - set automatically when a subsequent analysis shows that the issue has been corrected or the file is no longer available (removed from the project, excluded or renamed)
  - Removed - set automatically when the related rule is no longer available. The rule may not be available either because it has been removed from the Quality Profile or because the underlying plugin has been uninstalled.
- Resolutions - Resolved
  - False Positive - set manually
  - Won't Fix - set manually

## Quality Gates

- Enforce quality policy
- Answers the question:
    - Can I deliver my project today?
- Boolean conditions based on measured thresholds
- Ideally, all project use the same quality gate
- Built-in quality gate, “Sonar way”
    - Focus on keeping new code clean

# Quality Gates

- Can be notified when a quality gate fails
- Older versions would break CI builds
- Plugin, Sonar Build Breaker, was created to break CI builds
  - It hasn't been updated in 2 years
- Can use webhooks with Jenkins (https://blog.sonarsource.com/breaking-the-sonarqube-analysis-with-jenkins-pipelines/)

## Running Scans

- Use scanner appropriate for your project:
  - Gradle - SonarScanner for Gradle
  - MSBuild - SonarScanner for MSBuild
  - Maven - use the SonarScanner for Maven
  - Jenkins - SonarScanner for Jenkins
  - Azure DevOps - SonarQube Extension for Azure DevOps
  - Ant - SonarScanner for Ant
  - anything else (CLI) - SonarScanner

## Running Scans

- Analysis depends on the language
  - On all languages, "blame" data will automatically be imported from supported SCM providers. Git and SVN are supported automatically. Other providers require additional plugins.
  - On all languages, a static analysis of source code is performed (Java files, COBOL programs, etc.)
  - A static analysis of compiled code can be performed for certain languages (.class files in Java, .dll files in C#, etc.)
  - A dynamic analysis of code can be performed on certain languages.

## SonarLint

- IDE extension
- Detects quality issues as you write code
- IDEs supported
  - Eclipse
  - IntelliJ IDEA
  - Visual Studio
  - VS Code