**Experiment No. 08**

**Practical Name:** Understand a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

**Aim :** To understand static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web

/ Java / Python application

# Theory:

* **What is SonarQube? Ans :**
* SonarQube (formerly Sonar) is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) platform developed by [SonarSource](https://en.wikipedia.org/wiki/SonarSource) for continuous inspection of [code quality](https://en.wikipedia.org/wiki/Software_quality) to perform automatic reviews with static [analysis of](https://en.wikipedia.org/wiki/Static_program_analysis) [code](https://en.wikipedia.org/wiki/Static_program_analysis) to detect [bugs,](https://en.wikipedia.org/wiki/Software_bug) [code smells](https://en.wikipedia.org/wiki/Code_smell), and security vulnerabilities on 20+ [programming](https://en.wikipedia.org/wiki/Programming_language) [languages.](https://en.wikipedia.org/wiki/Programming_language) SonarQube offers reports on [duplicated code](https://en.wikipedia.org/wiki/Duplicate_code), [coding standards](https://en.wikipedia.org/wiki/Programming_style), [unit tests](https://en.wikipedia.org/wiki/Unit_testing), [code](https://en.wikipedia.org/wiki/Code_coverage) [coverage,](https://en.wikipedia.org/wiki/Code_coverage) [code complexity](https://en.wikipedia.org/wiki/Cyclomatic_complexity), [comments](https://en.wikipedia.org/wiki/Comment_(computer_programming)), [bugs](https://en.wikipedia.org/wiki/Defensive_programming), and security vulnerabilities.
* SonarQube can record metrics history and provides evolution graphs. SonarQube provides fully automated analysis and integration with [Maven,](https://en.wikipedia.org/wiki/Apache_Maven) [Ant,](https://en.wikipedia.org/wiki/Apache_Ant) [Gradle](https://en.wikipedia.org/wiki/Gradle), [MSBuild](https://en.wikipedia.org/wiki/MSBuild) and [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) tools ([Atlassian](https://en.wikipedia.org/wiki/Bamboo_(software)) [Bamboo](https://en.wikipedia.org/wiki/Bamboo_(software)), [Jenkins](https://en.wikipedia.org/wiki/Jenkins_(software)), [Hudson](https://en.wikipedia.org/wiki/Hudson_(software)), etc.).
* SonarQube is a Code Quality Assurance tool that collects and analyses source code, and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. Everything from minor styling choices, to design errors are inspected and evaluated by SonarQube. This provides users with a rich searchable history of the code to analyse where the code is messing up and determine whether or not it is styling issues, code defeats, code duplication, lack of test coverage, or excessively complex code. The software will analyse source code from different aspects and drills down the code layer by layer, moving module level down to the class level, with each level producing metric values and statistics that should reveal problematic areas in the source code that needs improvement.
* SonarQube also ensures code reliability, Application security, and reduces technical debt by making your code base clean and maintainable. SonarQube also provides support for 27 different languages, including C, C++, Java, JavaScript, PHP, GO, Python, and much more. SonarQube also provides Ci/CD integration, and gives feedback during code review with branch analysis and pull request decoration.

# Why SonarQube Jenkins integration is important? Ans :

* SonarQube is an open-source tool for continuous inspection of code quality. It performs static analysis of code, thus detecting bugs, code smells and security vulnerabilities. In addition, it also can report on the duplicate code, unit tests, code coverage and code complexities for multiple programming languages. Hence, in order to achieve Continuous Integration with fully automated code analysis, it is important to integrate SonarQube with CI tools such as Jenkins.

# What is the role of Gitlab? Ans :

* SonarQube's integration with GitLab Self-Managed and GitLab.com allows you to maintain code quality and security in your GitLab projects.
* With this integration, you'll be able to:
  + Authenticate with GitLab - Sign in to SonarQube with your GitLab credentials.
  + Import your GitLab projects - Import your GitLab Projects into SonarQube to easily set up SonarQube projects.
  + Analyse projects with GitLab CI/CD - Integrate analysis into your build pipeline. Starting in Developer Edition, SonarScanners running in GitLab CI/CD jobs can automatically detect branches or merge requests being built so you don't need to specifically pass them as parameters to the scanner.
  + Report your Quality Gate status to your merge requests - (starting in Developer Edition) See your Quality Gate and code metric results right in GitLab so you know if it's safe to merge your changes.

# What is a CI/CD pipeline? Ans :

* A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps you to safely deploy a new version of the software. CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.
* CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product. It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as CI/CD pipeline.
* A CI/CD pipeline is a series of steps that must be performed in order to deliver a new version of software. [Continuous integration/continuous delivery (CI/CD](https://www.redhat.com/en/topics/devops/what-is-ci-cd)) pipelines are a practice focused on improving software delivery using either a [DevOps](https://www.redhat.com/en/about/videos/learn-cloud-native-series-what-is-devops) or [site reliability](https://www.redhat.com/en/topics/devops/what-is-sre) [engineering (SRE)](https://www.redhat.com/en/topics/devops/what-is-sre) approach.
* A CI/CD pipeline introduces monitoring and [automation](https://www.redhat.com/en/topics/automation) to improve the process of application development, particularly at the integration and testing phases, as well as during delivery and deployment. Although it is possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.
* Elements of a CI/CD pipeline :

The steps that form a CI/CD pipeline are distinct subsets of tasks grouped into what is known as a pipeline stage. Typical pipeline stages include:

* + **Build** - The stage where the application is compiled.
  + **Test** - The stage where code is tested. Automation here can save both time and effort.
  + **Release** - The stage where the application is delivered to the repository.
  + **Deploy** - In this stage code is deployed to production.
  + **Validation and compliance** - The steps to validate a build are determined by the needs of your organization. Image security scanning tools, like [Clair](https://www.redhat.com/en/topics/containers/what-is-clair), can ensure the quality of images by comparing them to known [vulnerabilities (CVEs)](https://www.redhat.com/en/topics/security/what-is-cve).

# What is bug? Ans :

* A software bug is an error, flaw or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. After a product is released or during public beta testing, bugs are still apt to be discovered. When this occurs, users have to either find a way to avoid using the "buggy" code or get a patch from the originators of the code. Although bugs typically just cause annoying computer glitches, their impact can be much more serious.
* Most bugs arise from mistakes and errors made in either a program's design or its source code, or in components and operating systems used by such programs. A few are caused by compilers producing incorrect code. A program that contains many bugs, and/or bugs that seriously interfere with its functionality, is said to be buggy (defective). Bugs can trigger errors that may have ripple effects. Bugs may have subtle effects or cause the program to crash or freeze the computer. Other bugs qualify as security bugs and might, for example, enable a malicious user to bypass access controls in order to obtain unauthorized privileges.

# What are code smells? Ans :

* Code smells are not bugs or errors. Instead, these are absolute violations of the fundamentals of developing software that decrease the quality of code. Having code smells does not certainly mean that the software won’t work, it would still give an output, but it may slow down processing, increased risk of failure and errors while making the program vulnerable to bugs in the future. Smelly code contributes to poor code quality and hence increasing the technical debt.
* Code smells indicate a deeper problem, but as the name suggests, they are sniffable or quick to spot. The best smell is something easy to find but will lead to an interesting problem, like classes with data and no behaviour. Code smells can be easily detected with the help of tools.

# Conclusion :

Hence, understood why integrating Jenkins with SonarQube is important and also what are bugs, code smells.