Experiment – 9

**Aim:** Create a Jenkins CI/CD pipeline with SonarQube/GitLab Integration to perform a static analysis of the code to detect bugs, code smells and security vulnerabilities on a sample Web/Java/Python application.

**Theory:**

**Overview of the Components**

**1. Jenkins:**

- **Definition:** Jenkins is an open-source automation server that facilitates continuous integration and continuous deployment (CI/CD).

- **Key Features:** Extensible via plugins, supports multiple languages, and integrates with various version control systems.

**2. SonarQube:**

- **Definition:** SonarQube is an open-source platform for continuous inspection of code quality. It provides detailed insights into code quality by identifying bugs, vulnerabilities, and code smells.

- **Key Features:** Multi-language support, integration with CI/CD tools, customizable quality gates, and dashboards.

**3. GitLab:**

- **Definition:** GitLab is a web-based DevOps lifecycle tool that provides a Git repository manager with CI/CD pipeline features.

- **Key Features:** Version control, CI/CD pipelines, issue tracking, and integrated code review.

**Benefits of the Integration**

- **Automated Quality Checks:** Integrating SonarQube into the CI/CD pipeline ensures that code quality is evaluated with every commit.

- **Immediate Feedback:** Developers receive instant feedback on code quality, allowing for quicker fixes and improvements.

- **Comprehensive Analysis:** SonarQube's analysis includes not only bugs but also security vulnerabilities and maintainability issues (code smells).

**Architecture Overview**

**1. GitLab:** The source code repository where developers push their changes.

**2. Jenkins:** The CI/CD server that pulls code from GitLab, triggers builds, and runs tests.

**3. SonarQube:** The analysis server that reviews the code for quality metrics.

**4. Deployment Target:** An environment where the application is deployed (could be on-premises, cloud, etc.).

**Setting Up the CI/CD Pipeline**

**Prerequisites**

**1. Install Jenkins:** Make sure you have a Jenkins server running.

**2. SonarQube Server:** Set up a SonarQube instance, either locally or on a cloud service.

**3. GitLab Repository:** A GitLab repository containing the sample application code.

**4. Jenkins Plugins:** Ensure the following plugins are installed in Jenkins:

- GitLab Plugin

- SonarQube Scanner Plugin

- Pipeline Plugin

**Step-by-Step Implementation**

**Step 1: Configure SonarQube**

**1. Install SonarQube:** Set up SonarQube on your server.

**2. Create a Project:** In SonarQube, create a new project for your application.

**3. Generate Tokens:** Generate an authentication token for the project to be used by Jenkins.

**Step 2: Configure Jenkins**

**1. Jenkins Configuration:**

- Navigate to **Manage Jenkins > Configure System.**

- Add SonarQube servers under the **SonarQube** section, including the server URL and token.

**2. Set Up GitLab Integration:**

- Go to **Manage Jenkins > Configure System.**

- Under the **GitLab** section, configure the GitLab connection (URL, token).

- Enable the **GitLab webhook integration**.

**Step 3: Create a Jenkins Pipeline Job**

**1. Create a New Pipeline Job:**

- Go to **Jenkins Dashboard** and click on **New Item**.

- Select **Pipeline** and give it a name.

**2. Configure the Pipeline:**

- In the pipeline configuration, set up the stages using the pipeline syntax.

**3. Save and Build:**

- Save the pipeline configuration and trigger a build to test the setup.

**Step 4:** Set Up GitLab Webhook

**1. Configure Webhook in GitLab:**

- Go to your GitLab repository settings.

- Navigate to **Webhooks** and add a new webhook pointing to your Jenkins server (e.g., `http://your-jenkins-url/gitlab/build\_now`).

- Select triggers like **Push events** to trigger the Jenkins pipeline on code changes.

**Running the Pipeline**

**1. Push Code Changes:** Make a change in your GitLab repository and push it.

**2. Trigger Jenkins:** The webhook should trigger the Jenkins pipeline automatically.

**3. Monitor the Pipeline:** Watch the stages execute in Jenkins. Code will be analyzed by SonarQube, and the results will be displayed in the SonarQube dashboard.

**4. Review Results:** Review SonarQube’s output for any bugs, vulnerabilities, or code smells.

**Experiment:**