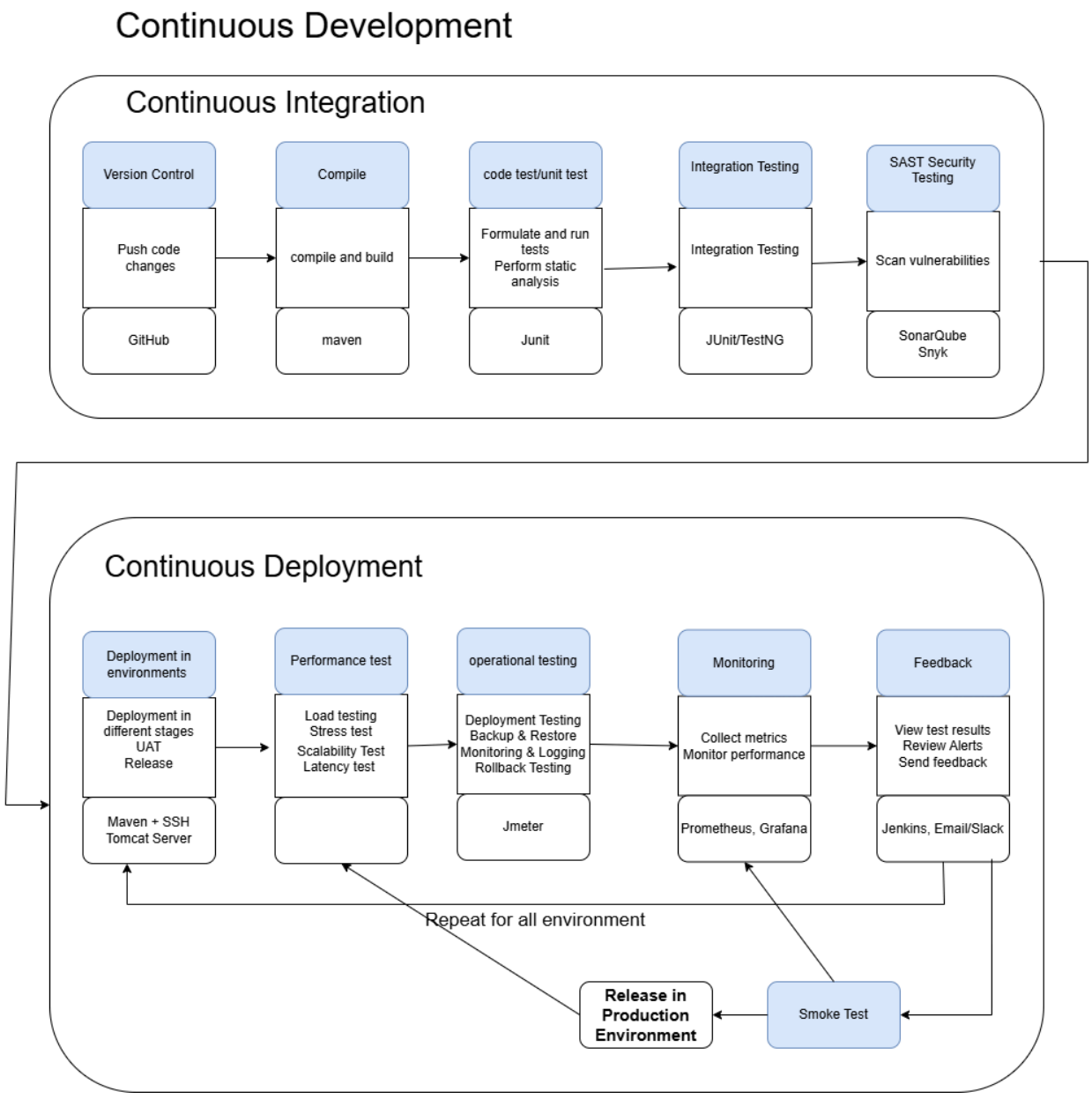
1. Explain a detailed **CI/CD workflow** using **Jenkins** as the CI/CD server to **build, test, secure, and deploy a Maven-based Java project**, integrated with GitHub for version control. This workflow also includes **JUnit for unit testing**, **Snyk for static application security testing (SAST)**, **JMeter for load testing**, and **Prometheus/Grafana for onitoring and feedback**.
   Sol:



Continuous Development – Continuous Integration Phase

1. Version Control

- **Tool**: GitHub
- **Purpose**: Store code; detect changes (e.g., `git push`) to trigger Jenkins build
- **Jenkins Configuration**:
    - Use **Git plugin**
    - Configure webhook from GitHub → Jenkins

2. Compile

- **Tool**: Maven
- **Purpose**: Compile the codebase into `.class` or `.jar`
- **Jenkins Configuration**:
    - Add Maven tool in Jenkins Global Tool Config
    - Use `mvn clean compile`

3. Code Test / Unit Test

- **Tool**: JUnit
- **Purpose**: Run automated tests and perform static analysis
- **Jenkins Configuration**:
    - JUnit Plugin for visualizing results
    - Static analysis via Checkstyle or SpotBugs

4. Integration Testing

- **Tool**: JUnit / TestNG
- **Purpose**: Validate how modules interact
- **Jenkins Implementation**:

    - Use Maven `verify` phase for integration tests

5. SAST – Static Application Security Testing

- **Tools**: SonarQube, Snyk
- **Purpose**: Scan code for vulnerabilities
- **Jenkins Implementation**:
    - Use **SonarQube plugin**
    - Configure **Snyk CLI** with credentials

Continuous Deployment Phase

1. Deployment in Environments

- **Tools**: Maven + SSH, Tomcat
- **Purpose**: Deploy to UAT, QA, and Prod

2. Performance Testing

- **Tool**: JMeter
- **Purpose**: Load, stress, and latency tests
- **Jenkins Implementation**:
    - Run JMeter via shell
    - Use Performance plugin for reports

3. Operational Testing

- **Tool**: JMeter (scripts), custom shell scripts
- **Purpose**: Test rollback, backup, restore, and infra readiness
- **Jenkins Implementation**:
    - Include shell scripts to simulate backup/restore

4. Monitoring

- **Tools**: Prometheus, Grafana
- **Purpose**: Collect metrics, generate alerts
- **Jenkins Implementation**:
    - Install Prometheus plugin to expose metrics
    - Use Prometheus/Grafana separately to visualize

Feedback

- **Tools**: Jenkins, Email/Slack Plugins
- **Purpose**: Notify developers about pipeline results and system health

Comprehensive CI/CD Testing Taxonomy

I. Development-Level Testing

Testing Type	Description
Unit Testing	Test individual functions/methods in isolation.
Code Linting / Static Analysis	Detect code quality issues and bugs at compile time.
Developer Smoke Testing	Quick manual checks before committing to verify app doesn't crash

Tools: JUnit, pytest, ESLint, Checkstyle, SonarQube

Integration Testing: Verifies data flow and interactions between modules/services.
Regression Testing: Re-checks previously working features to ensure no breakages
System Testing: Full end-to-end testing of the integrated system. Ex UAT, performance testing, functional testing , non performance testing

III. Functional Testing (Subset of System Testing)

Testing Type	Description
API Testing	Verifies API endpoints, responses, and behavior.
UI Testing	Validates visual components and user interactions.
End-to-End (E2E) Testing	Tests user workflows from start to finish.
Acceptance Testing (UAT)	Business validation that system meets user needs.

Tools: Selenium, Postman, Cypress, Cucumber

IV. Non-Functional Testing (Subset of System Testing)

Testing Type	Description
Security Testing (SAST/DAST)	Finds vulnerabilities in code and live apps.
Compatibility Testing	Tests across OS, browsers, and devices.
Usability Testing	Evaluates UI design and ease of use.
Scalability Testing	Assesses the system's ability to scale.

Tools: Snyk, OWASP ZAP, BrowserStack, Burp Suite

V. Performance Testing (Non-Functional Category, CI/CD Phase)

Testing Type	Description
Load Testing	Evaluates performance under expected traffic/load.
Stress Testing	Tests stability when pushed beyond normal limits.
Spike Testing	Observes behavior during sudden traffic surges.
Endurance Testing	Measures system performance under continuous load over time.
Latency Testing	Tests response delays in system operations.
Throughput Testing	Measures number of successful requests per second.

Tools: Apache JMeter, Gatling, Locust, k6, BlazeMeter

VI. Smoke Testing (CI/CD Stage Gate)

Testing Type Description

Smoke Testing Basic test to confirm whether the app is stable enough for further testing or deployment.

Sanity Testing Verifies basic functionality after small changes or bug fixes.

Tools: Custom scripts, Selenium, Jenkins build stages