International Journal Of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

NO TOUCH CODE SCANNING EXPERIENCE IN CICD PIPELINE

Kamalakar Reddy Ponaka
DevOps - Application Security
kamalakar.ponaka@gmail.com

Abstract

In modern software development, continuous integration and continuous deployment (CI/CD) pipelines play a crucial role in automating the build, test, and deployment processes. Integrating code scanning tools into CI/CD pipelines is essential to ensure code quality and security. However, traditional methods of code scanning often introduce significant overhead and friction, slowing down development cycles. This white paper explores the implementation of a low-touch code scanning experience in CI/CD pipelines, highlighting strategies, best practices, and tools to achieve an efficient, automated, and developer-friendly workflow. Additionally, we discuss simplified onboarding using ServiceNow CMDB application ID, pipeline gating, automated repository scanning, and traceability using application IDs across code scans.

Index Terms – Low Touch, CICD, AppSec, ServiceNow

I. INTRODUCTION

Code quality and security are paramount in delivering robust software. Code scanning tools help identify potential bugs, vulnerabilities, and coding standard violations early in the development cycle. Integrating these tools into CI/CD pipelines ensures continuous monitoring and remediation. A low-touch approach minimizes manual interventions, allowing developers to focus on writing code while maintaining high standards of quality and security [1] [2] 3]

II. OBJECTIVES

Objective of the white paper is to provide guidance around no touch scanning experience.

- **A. Automate Code Scanning:** Ensure that code scanning is seamlessly integrated into CI/CD pipelines with minimal manual steps.
- **B. Reduce Developer Overhead:** Minimize the friction and overhead associated with code scanning, enabling developers to focus on core tasks.
- **C. Enhance Code Quality and Security:** Continuously monitor and improve code quality and security through automated, low touch scanning processes.
- **D. Simplified Onboarding:** Use ServiceNow CMDB (Configuration Management Database) application ID for streamlined integration and management of code scanning tools
- **E. Pipeline Gating:** Implement pipeline gating to ensure that only code meeting predefined quality and security standards is allowed to proceed through the pipeline

International Journal Of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

F. **Automated Repository Scanning:** Implement automated repository scanning to continuously monitor repositories for vulnerabilities and code quality issues.
G. **Traceability:** Use application IDs to achieve traceability across code scans, ensuring a clear audit trail and accountability.

III. SELECTING RIGHT TOOLS

Choosing the appropriate tools is critical to achieving a low-touch experience. Tools should integrate seamlessly with CI/CD systems, provide actionable feedback, and require minimal configuration. Some recommended tools include:

A. **SCA:** Focuses on identifying and fixing vulnerabilities in open-source dependencies and provides excellent integration with CI/CD tools.
B. **SAST:** Focuses on identifying and fixing vulnerabilities in open-source dependencies and provides excellent integration with CI/CD tools.
C. **Advanced Security Tools:** Provides built-in code scanning capabilities for repositories.
D. **Code Coverage:** Allows for powerful static analysis and custom query definitions to identify security issues.

IV. AUTOMATING CODE SCANNING IN PIPELINE

Automating code scanning significantly reduce the time spend by developers. By leveraging following techniques, we can improve the speed.

A. **Pre-Commit Hooks:** Implementing pre-commit hooks ensures basic static analysis checks are performed before code is committed. Tools like pre-commit can run these checks locally, catching issues early.
B. **CI/CD Integration:** Integrate code scanning as a dedicated stage in the CI/CD pipeline.
C. **Configuration Management:** Use configuration files to automate tool setups, such for SCA/SAST tool. Store these files in the repository to maintain consistency and ease of use.

V. CONFIGURING TOOLS FOR LOW TOUCH OPERATION

A. **Thresholds and Rules:** Define thresholds for acceptable issue levels and configure tools to fail the build only on critical issues. This reduces noise and focuses on significant problems.
B. **Automatic Updates:** Enable automatic updates for tools and their vulnerability databases to ensure the latest checks are applied without manual intervention.

VI. INTEGRATING WITH DEVELOPER WORKFLOW

A. **Merge Request Checks:** Integrate code scanning results directly into merge requests. This ensures issues are identified and discussed before merging.
B. **Actionable Feedback:** Provide feedback that is clear, actionable, and directly within the development environment. Inline comments in merge requests help developers

understand and address issues promptly.[4]

VII. SIMPLIFIED ONBOARDING

ServiceNow CMDB (Configuration Management Database) helps organizations manage their IT infrastructure by maintaining a single source of truth for configuration items (CIs). By leveraging ServiceNow CMDB, you can streamline the onboarding process for code scanning tools, ensuring consistent and efficient management.

A. Using CMDB Application ID:

To simplify the onboarding of code scanning tools, utilize the ServiceNow CMDB application ID. This approach involves registering code scanning tools as configuration items within the CMDB, allowing for better tracking, management, and integration.

B. Steps to Onboard Using CMDB Application ID

- **Register Code Scanning Tools**: Create entries for each code scanning tool in the ServiceNow CMDB with unique application IDs.
- **Automate Configuration**: Use the CMDB entries to automate the configuration and setup of code scanning tools within the CI/CD pipeline. For example, store configuration details and API keys in the CMDB.
- **Integrate with CI/CD**: Modify the CI/CD pipeline scripts to fetch configuration details from the CMDB using the application ID, ensuring that each tool is correctly configured without manual intervention.

C. Example Integration

Here is an example of how you might integrate ServiceNow CMDB with a CI/CD pipeline for a SAST/SCA Tool and reference taken from [5]

International Journal Of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

```
# Example Snyk script for GitLab CI/CD Pipeline with Node.js project

dependency_scanning:
  image: node:latest
  stage: test
  script:
    # Install npm, snyk, and snyk-to-html
    - npm install -g npm@latest
    - npm install -g snyk
    - npm install snyk-to-html -g
    # Run snyk help, snyk auth, snyk monitor, snyk test to break build and
out report
    - snyk --help
    - snyk auth $SNYK_TOKEN
    - snyk monitor --project-name=$CI_PROJECT_NAME
    - snyk test --json | snyk-to-html -o snyk_results.html

  # Save report to artifacts
  artifacts:
    when: always
    paths:
      - snyk_results.html
```

VIII. SIMPLIFIED ONBOARDING

A. What is Pipeline Gating

Pipeline gating is a mechanism that ensures only code meeting predefined quality and security standards is allowed to progress through the CI/CD pipeline. This helps prevent issues from reaching production, thereby maintaining high standards of code quality and security.

B. Implementing Pipeline Gating

- **Define Gate Criteria:** Establish criteria for code quality and security that must be met for code to progress. This could include passing all tests, meeting code coverage thresholds, and having no critical vulnerabilities.
- **Configure Gates in CI/CD Pipeline:** Integrate gating logic into the CI/CD pipeline stages to automatically enforce these criteria.

International Journal Of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

C. Example CI/CD Pipeline with Gating

Sample gating script below[5].

```
# TODO Make stage success/failure based on vulnerability severity
snyk-dependency-analysis:
  image: awesomeaiden/custom_snyk
  stage: static_analysis
  only:
    changes:
      - boot-kotlin-svc/**/*
  before_script:
    - cd boot-kotlin-svc
    - export GRADLE_USER_HOME=`pwd`/.gradle
    - export SNYK_TOKEN=$secret_snyk_token
  after_script:
    - cd ..
  variables:
    SNYK_TOKEN: $secret_snyk_token
  script:
    - snyk test
  cache:
    key: "$CI_COMMIT_REF_NAME"
    policy: pull
    paths:
      - build
      - .gradle
  allow_failure: true
```

IX. AUTOMATED REPOSITORY SCANNING

Automated repository scanning involves continuously monitoring code repositories for vulnerabilities, code quality issues, and compliance with coding standards. This process helps identify and address issues early in the development lifecycle, ensuring that code in repositories remains secure and high-quality.

A. Implementing Automated Repository Scanning

- **Enable Repository Scanning Tools:** Use SCA/SAST tools to enable automated scanning of repositories.
- **Schedule Regular Scans:** Configure tools to perform regular scans of the repository, such

as daily or weekly, to ensure continuous monitoring.

- **Integrate with CI/CD:** Ensure that the results of automated scans are integrated into the CI/CD pipeline, so issues are surfaced and addressed promptly.

X. TRACEABILITY USING APPLICATION ID ACROSS CODE SCANS

A. Traceability

Traceability in the context of CI/CD pipelines refers to the ability to track and link different stages of the development lifecycle, ensuring that all changes and their impacts are recorded and auditable. This includes tracking code changes, scans, builds, and deployments back to their source.

B. Implementing Traceability with Application ID

Using an application ID from ServiceNow CMDB, you can ensure consistent tracking of code changes and scan results across different stages of the CI/CD pipeline. This application ID serves as a unique identifier that links all activities and artifacts related to a specific application.

C. Steps to Implement Traceability

- **Assign Application ID:** Assign a unique application ID from the ServiceNow CMDB to each project or repository.
- **Integrate Application ID in CI/CD Pipelines:** Include the application ID in all relevant stages of the CI/CD pipeline, ensuring that scan results, build logs, and deployment records are tagged with this ID.
- **Store Scan Results and Logs:** Store scan results, build logs, and other artifacts in a centralized system, tagged with the application ID for easy retrieval and auditing.

XI. CONCLUSION

Implementing a low-touch code scanning experience in CI/CD pipelines significantly enhances code quality and security while minimizing the overhead for developers. By automating code scanning, integrating with developer workflows, leveraging ServiceNow CMDB for simplified onboarding, implementing pipeline gating, enabling automated repository scanning, and ensuring traceability with application IDs, organizations can ensure a streamlined, efficient, and secure software development process.

REFERENCES

1. https://www.servicenow.com/products/servicenow-platform/configuration-management-database.html
2. https://docs.gitlab.com/ee/ci/pipelines/
3. https://docs.snyk.io/scan-using-snyk/snyk-code
4. L. Davis, "DevOps Best Practices for Enterprise Scalability," Software Engineering Review , vol. 22, no. 6, pp. 789-798, Jun. 2021.
5. https://github.com/snyk-labs/snyk-cicd-integration-examples/tree/master/GitLabCICD