CLIENT NAME
Computer Science Department
Swinburne Vietnam Alliance Program HCMC

Usability Testing Documentation

Secure CI/CD Pipeline for Application Development

Version 1.0

Prepared by: GROUP 1

- Dang Vi Luan - 103802759
- Nguyen Linh Dan – 103488557
- Nguyen Duy Khang - 104056476
- Tran Bao Huy - 103505799

Table of Contents

1. USABILITY TESTING PLAN

1.1 Testing goals

- Evaluate the ease of setting up the CI/CD pipeline using the provided solution package.
- Assess the user's ability to understand and navigate the various components of the pipeline.
- Identify any pain points or areas of confusion in the setup process.
- Determine the effectiveness of the security implementations.
- Measure the user's ability to use the monitoring features effectively.

1.2 Testing methodology

- One-on-one moderated sessions: Participants interact with the product while observed by a moderator, enabling in-depth understanding of user experiences.
- Think-aloud protocol: Participants verbalize their thoughts, feelings, and actions, providing insight into their decision-making.
- Task completion time: Measuring task times identifies areas of inefficiency and benchmarks usability.
- Post-test feedback: Questionnaires and interviews gather qualitative user feedback to complement quantitative data.

1.3 Target demographic

- IT students with programming experience
- Software developers with basic knowledge of the CI/CD concepts

1.4 Performed scenarios

Scenario 1: Initial Setup and GitHub Integration

Acceptance criteria:

- The user successfully sets up the local environment.
- The user connects their GitHub repository to the CI/CD pipeline.
- The user correctly configures GitHub Secrets for credential security.

Scenario 2: CI Pipeline Configuration.

Acceptance criteria:

- The user successfully configures the CI pipeline in GitHub Actions.
- The user implements and runs the Software Composition Analysis.
- The user sets up automated testing in the pipeline.

Scenario 3: CD Pipeline Setup with ArgoCD.

Acceptance criteria:

- The user successfully installs and configures ArgoCD

- The user creates an application in ArgoCD linked to their GitHub repository.
- The user verifies that ArgoCD can sync changes between the repository and the Kubernetes cluster.

<u>Scenario 4:</u> Infrastructure Provisioning with Terraform

<u>Acceptance criteria:</u>

- The user writes basic Terraform code to provision infrastructure on AWS
- The user successfully runs the Terraform code through the GitHub Actions workflow.
- The user verifies the created infrastructure on AWS.

<u>Scenario 5:</u> Setting up Monitoring with Prometheus and Grafana

<u>Acceptance criteria:</u>

- The user installs Prometheus and Grafana on the Kubernetes cluster
- The user configures basic monitoring for the application.
- The user creates a simple dashboard in Grafana to visualize application metrics.

<u>Scenario 6:</u> Implementing Security Measures

<u>Acceptance criteria:</u>

- The user sets up and runs a SAST scan using SonarQube.
- The user configures and executes a DAST scan using OWASP ZAP.
- The user reviews and interprets the security scan results.

2. CONDUCT THE USABILITY TESTING

2.1. Participants profiles

- **Participant 1 – Le Thanh Dat:** Junior Developer with 2 years of experience.

<u>Background:</u> 02 years of experience, familiar with Git and basic CI/CD concepts.

- **Participant 2 – Le Hoang Phi:** IT Student - Final Year

<u>Background:</u> Internship experience in software development.

- **Participant 3 – Vo Thanh Tam:** IT Student, 2nd year

<u>Background:</u> Major in System Management, familiar with shell scripting.

- **Participant 4 – Phan Van Ky:** IT Student - Freshmen

<u>Background:</u> Basic knowledge of programming.

***Note**: All Participant have agreed to voluntarily participate in the testing and allow the project team to use their information for this report. (see <u>References</u>)*

2.2. Summary of testing result

- **Participant 1: Junior Developer (2 years of experience)**

Scenario 1 result: Completed successfully.

Scenario 2 result: Completed successfully.

Scenario 3 result: Needed some guidance with ArgoCD concepts but completed the task.

Scenario 4 result: Failed to do Terraform tests.

Scenario 5 result: Completed basic setup of Prometheus and Grafana.

Scenario 6 result: Completed SAST setup, had some difficulties interpreting DAST results.

Participant's feedback: The CI/CD setup was familiar, but I found the infrastructure provisioning and some security aspects challenging. More detailed guides for these areas would be helpful.

Tester's comment: Participant's CI/CD knowledge was good but there are gaps in infrastructure and advanced security concepts. Consider providing more in-depth resources for these topics.

Summary: The participant completed **5/6** tests.

- **Participant 2: IT Student (Final year, internship experience)**

Scenario 1 result: Completed with minor assistance.

Scenario 2 result: Completed successfully, showed good understanding of CI concepts.

Scenario 3 result: Needed explanation of ArgoCD but completed the task.

Scenario 4 result: Failed to do Terraform tests.

Scenario 5 result: Completed basic monitoring setup, struggled with creating custom dashboards.

Scenario 6 result: Completed SAST and DAST setups with some assistance, showed interest in security concepts.

Participant's feedback: The process was informative and aligned with what I have seen in my internship. The infrastructure and advanced monitoring parts were the most challenging.

Tester's comment: Participant's internship experience was evident in their approach to CI/CD. Additional focus on infrastructure-as-code and advanced monitoring concepts would be beneficial.

Summary: The participant completed **5/6** tests.

- **Participant 3: IT Student (2nd year, System Management major)**

Scenario 1 result: Completed with some difficulties, needed help with Git concepts.

Scenario 2 result: Failed to do the CI integration pipeline process.

Scenario 3 result: Failed to understand the concepts of Kubernetes and ArgoCD

Scenario 4 result: Showed better understanding due to familiarity with scripting, required additional help with Terraform.

Scenario 5 result: Completed basic Prometheus and Grafana setup with assistance.

Scenario 6 result: Failed to understand the concepts of SAST and DAST.

Participant's feedback: The shell scripting knowledge helped in some areas, but I found many concepts unfamiliar. A glossary of terms and more background information would be useful.

Tester's comment: Participant's system management background provided some advantages but lacked broader development and CI/CD knowledge. Consider creating a more comprehensive onboarding module for students with similar backgrounds.

Summary: The participant completed **3/6** tests.

- **Participant 4: IT Student (Freshman)**

Scenario 1 result: Failed GitHub account set up and have no knowledge about credentials storing.

Scenario 2 result: Failed CI test, don't know about CI implementation.

Scenario 3 result: Failed ArgoCD and Kubernetes tests, participant does not know these concepts.

Scenario 4 result: Failed to do Terraform tests.

Scenario 5 result: Failed to create dashboards

Scenario 6 result: Participant failed to understand the concept of SAST and DAST

Participant's feedback: The process was very overwhelming. I felt lost most of the time and could not connect the concepts to my current knowledge.

Tester's comment: The solution is too advanced for a freshman with only basic programming knowledge. We should consider creating a simplified, educational version of the pipeline for beginners, focusing on core concepts before introducing advanced topics.

Summary: The participant completed **0/6** tests.

3. USABILITY ANALYSIS REPORT

3.1. Summary of the results

- Experience level impact: The usability of the CI/CD pipeline solution varied significantly based on the participants' experience levels. While junior developer and final-year IT student can navigate through most scenarios with some assistance, early-year students failed all of the tests.
- Concept familiarity: Participants showed varying levels of familiarity with key concepts. CI/CD and Git were generally well-understood by more experienced participants, while topics like infrastructure-as-code, Kubernetes, and advanced security measures posed challenges across the board.
- Task completion: More experienced participants (junior developer and final-year student) were able to complete most tasks with minor to moderate assistance. The second year student completed tasks with significant help, while the freshman failed to do all the tasks.

- Learning curve: All participants, regardless of experience level, faced a steep learning curve in at least some areas of the pipeline setup, particularly in infrastructure provisioning and security implementation.
- Engagement: Despite difficulties, most participants showed interest in the process, particularly in security aspects and monitoring capabilities.

3.2. Identified problems

- Complexity overload: The solution package assumes a level of knowledge that exceeds that of early-stage IT students and even challenges some aspects of junior developers' expertise.
- Lack of prerequisite knowledge: Concepts in areas like Kubernetes, ArgoCD, and Terraform can cause delays.
- Steep learning curve: The jump from basic programming knowledge to implementing a full CI/CD pipeline with advanced features can be steep for less experienced users.
- Tool overwhelm: The number of different tools and platforms introduced (GitHub, ArgoCD, Terraform, Prometheus, Grafana, SonarQube, OWASP ZAP). Each tool is not at a complex level, but the number of tools can be overwhelming for less-experienced users.

3.3. Recommendations and improvements to be made

- Tiered learning approach: Develop a tiered system of tutorials and tasks, starting from basic concepts and progressively moving to more advanced topics. This allows users of all levels to engage with the material at an appropriate pace.
- Concept primers: Develop short, focused modules that introduce key concepts (e.g., Kubernetes, ArgoCD, Terraform) before diving into their implementation in the pipeline.
- Simplified starter version: Create a basic version of the pipeline that introduces core CI/CD concepts without the complexity of advanced features. This can serve as a starting point for beginners.
- Progress tracking: Implement a system that allows users to track their progress and clearly see which skills they've acquired and what's next in their learning journey.
- Peer learning support: Establish a community forum or chat where users can ask questions, share experiences, and learn from each other.
- Regular feedback loop: Implement a system for continuous user feedback to iteratively improve the learning experience and identify pain points quickly.

By implementing these recommendations, the usability of the CI/CD pipeline solution can be significantly improved, making it more accessible to a wider range of users while still providing value to more experienced developers. The key is to create a more scaffolded, flexible learning experience that can adapt to different skill levels and learning paces.

REFERENCES

General Information for Participants: Link

Consent form (4 participants): Link