Continuous Integration Testing Example Article

In the current age of information, ensuring the reliability of software is crucial for cybersecurity practices. When code repositories are not maintained, vulnerabilities will cause the code to be compromised and an accomplice in a cyber-attack. To mitigate risk, Continuous Integration (CI) testing is recommended to confirm the effectiveness of software.

What is Continuous Integration Testing?

Continuous Integration Testing is the practice of automatically testing and integrating code when pushed to a repository. This practice was first created to solve the issue of multiple contributors committing new code changes at different intervals to the same repository. CI Testing has become increasingly important within the DevOps workflow as it allows collaboration and increased communication between the development and operations teams.

Instead of committing changes to the main repository, contributors will merge their changes into a centralized integration and testing branch of the main repository. After the changes are made, an automated testing process will run on the merged code. The cadence of this automation will vary from organization to organization. Some organizations prefer to have the automation run on each commit, where others prefer the process to be run nightly after all of the commits have been made.

Best Practices for Continuous Integration Testing

Like traditional coding and software development, CI testing also has best practices that should be implemented to ensure reliability. Below are a few of the best practices when utilizing CI testing.

- Maintain a Code Repository

  This step may seem redundant, but it can go overlooked in the CI testing process. The first step of CI testing is the repository that the developers merge their commits and where the automation testing occurs. Maintaining the repository allows the CI/CD pipeline to operate efficiently. Frequent commits to this repository will ensure that it is on the most recent version for developers and testers to utilize. To properly maintain a repository, you will need to have organization. Code that is categorized and labeled in a way that reflects the software is key. Without organization, commits will be ineffective as the tests automation will require more manpower to account for misplaced code.

- Use Failures to Improve

  It is inevitable that the automated testing will fail. The unit tests may need adjusted to account for an updated feature of the software or removed entirely. In major releases of software, prior unit tests may become obsolete or incorporated with other tests. In any case, failure is part of the process and allows the CI testing to constantly be improved and updated. The failures are to be weak points that can be strengthened and improved upon.

- Test Mirrors Production

  Developers often do not have direct access to production in software environments. This is done for security reasons but also to limit untested code. The closest a developer should be to production is a test environment that mirrors the current release. If the developer is conducting a bug fix on an outdated version, that fix could cause more issues due to the version difference. Ensuring that the test environment mirrors the production environment will also verify the developer is working with the most recent release and not using outdated code.

- Automated Tests Are Fast and Accurate

  CI testing is designed to streamline the testing process while ensuring the code will not cause bugs and other issues when merged. Accuracy is important, but if the tests take weeks and months, then the tests need to be reevaluated to reduce the scope. The scope of the tests, whether unit, regression, or another unique type, should have a narrow scope to verify the functionality of a specific portion of the code. Each line does not necessarily need to be tested as the overall function is the important factor. The timeframe will vary between organization and what is best should be evaluated to best fit your specific use case.

How is CI/CD Testing Done?

At the highest level, CI testing involves automated testing to merges made to a specified repository. The steps below outline the steps in how CI testing is done.

1. Understand the scope of CI testing for your software. It is at this point that the automated testing pipeline is developed and refined based on your workflow.
2. Create an integration/testing repository for the development team of the software.
3. Contributors commit changes into the integration repository.
4. The software changes are compiled into a build used for testing.
5. Unit, integration, and regression testing are performed by the automated process.
6. The build will either fail or pass the testing. A new update will be created if the code passes the tests. If the tests fail, then the owner of the CI testing process will be alerted to review further.

How Company X Enhances the CI Testing Process

CI Testing should not be a suggestion, but instead a requirement for any organization serious about software development. There are many points of failure during the software development lifecycle; continuous integration testing helps mitigate those failure points. Company X provides organizations with an enhanced CI Testing Process that will allow you to focus on creating software as Company X focuses on the CI testing pipeline and security policies. Using security practices, including End-to-end encryption (E2EE) and Risk Analysis, Company X allows for a more secure CI/CD testing pipeline for software development organizations.