Continuous Integration Testing: GitHub Actions

By Yumna Tasneem

Ocotber 2024

1 Introduction

GitHub Actions can be configured to respond to any event occurring in or related to a repository, such as Pull Requests, the addition of a new Contributor, the creation of Issues, or the merging of Pull Requests.

For instance:

- A welcome message can be automatically displayed when a new Contributor joins.
- Upon the creation of a new Issue, a workflow can be set up to sort, label, assign, and execute a script to reproduce the issue.
- Full CI/CD implementations can be managed through GitHub Actions.

A simple GitHub Actions workflow is demonstrated to run automated tests written in Cypress, with details and code provided in a GitHub repository.

2 Advantages of GitHub Actions Over JUnit5 for Continuous Integration

- **Comprehensive CI/CD Integration:** GitHub Actions is designed to automate the entire CI/CD pipeline, handling everything from building and testing to deployment. In contrast, JUnit5 is solely focused on unit testing.
- **Event-Driven Workflows:** GitHub Actions can trigger workflows based on various GitHub events (e.g., commits, pull requests, issues), allowing for immediate testing and deployment. JUnit5 lacks this automation for CI processes.
- **Parallel Job Execution:** GitHub Actions can run multiple jobs in parallel, significantly speeding up the overall CI process, especially for larger Android projects with numerous tests. JUnit5 primarily runs tests sequentially.
- **Extensive Community Actions:** The GitHub Marketplace offers a vast selection of reusable actions for various tasks, including deployment and notifications, which can be easily integrated into CI workflows. JUnit5 does not have a comparable ecosystem for automation beyond testing.
- **Customizable Workflows:** Users can create highly customizable workflows, tailoring the CI/CD process to specific project needs without modifying the testing framework. JUnit5 focuses on testing logic and lacks this flexibility.

- **Native GitHub Integration:** Being integrated directly into GitHub, Actions offers seamless connectivity with repositories, pull requests, and issues, enhancing collaboration and visibility. JUnit5 does not provide direct integration with CI/CD processes.
- **No Infrastructure Management:** GitHub Actions runs on GitHub's infrastructure, eliminating the need for managing build servers or resources. JUnit5 is a testing framework and requires an external CI system to run tests.
- **Multi-Platform Support:** GitHub Actions supports various programming languages and platforms, allowing for diverse project types beyond just Java/Android. JUnit5 is specialized for Java, making it less versatile for cross-platform projects.

3 Why Use GitHub Actions for Android Integration Testing

- Automated CI/CD Pipelines
- Customizable Workflows
- Parallel Execution
- Reusable Actions
- No Infrastructure Management
- Native GitHub Integration
- Scalable and Secure

4 Disadvantages of GitHub Actions

While GitHub Actions is powerful and flexible, it does have some common disadvantages:

- Limited Free Minutes
- Steeper Learning Curve
- Longer Build Times
- Limited Build Resources
- Debugging Complexity
- Integration with Non-GitHub Platforms

5 Comparison of GitHub Actions and JUnit5 for Continuous Integration Testing

Aspect	GitHub Actions (CI/CD Focus)	JUnit5 (Testing Focus)
End-to-End CI/CD Automation	Automates the entire lifecycle, from code building to testing and deployment. Integrates seamlessly with GitHub repositories.	Focuses on writing and running unit tests for code, ensuring early detection of bugs and maintaining code quality.
Platform Agnosticism & Flexibility	Supports any platform or toolchain, including Android development environments. Custom workflows can be tailored to project needs.	Integrates easily with Android-specific testing frameworks like AndroidJUnitRunner, covering unit and instrumentation testing.
Parallel Job Execution	Can run multiple tasks or tests in parallel, reducing the time needed for testing and builds.	Primarily focuses on sequential test execution within the Java ecosystem. Parallelism can be achieved but is limited to test scope.
Community-Powered Actions	Offers thousands of reusable actions from the GitHub Marketplace to automate tasks such as code coverage reporting and running emulators.	JUnit5 allows custom annotations and extensions but is more limited to testing logic and does not integrate with external tool repositories.
Extending Beyond Testing	Goes beyond testing by automating deployment tasks like building APKs, code signing, and publishing to beta tracks or the Play Store.	Focused on testing. Does not handle build, deployment, or other post-test activities.
Designed for Unit Testing	Primarily focused on CI/CD tasks, though it can run unit tests as part of the workflow.	Purpose-built for unit testing Java code, offering robust tools for testing individual units like classes and methods.
Integration with Testing Frameworks	Supports integration with testing tools but requires external actions for setup (e.g., Android emulators).	Directly integrates with Android-specific testing frameworks, enabling comprehensive testing of app components such as UI and activities.

Continued on next page

Aspect	GitHub Actions (CI/CD Focus)	JUnit5 (Testing Focus)
Test-First Development (TDD)	Primarily used for automating workflows and post-development pipeline tasks.	Supports test-driven development, encouraging test creation alongside code to ensure features are fully tested as they are developed.
Custom Annotations & Extensions	Supports customizable workflows via YAML files but relies on external actions for complex testing behaviors.	Provides extensive support for custom annotations and extensions, allowing deep customization of test logic.
Easily Integrated with CI Tools	Works as a CI/CD tool that can automate JUnit5 tests, offering a comprehensive workflow for development, testing, and deployment.	Can be integrated within GitHub Actions for running tests as part of a larger CI/CD pipeline.
Scope	Broader, handling the entire DevOps lifecycle (build, test, deploy).	Narrower, specializing in unit testing and code quality assurance.
Focus on Code Quality	Automates tasks but doesn't focus specifically on unit testing.	Designed for ensuring code quality through detailed unit tests.

Table 1: Comparison of GitHub Actions and JUnit5

---

6 Learn More

For more information on GitHub Actions, visit the following link: `https://youtu.be/mFFXuXjVgkU?si=6GP4LEzGZ9-3ozuv`