University Parking System: Continuous Integration and Deployment

for

Master of Science

Information Technology

Kassandra Vega Lucero

University of Denver College of Professional Studies

May 18, 2025

Faculty: Nathan Braun, MBA

Dean: Michael J. McGuire, MLS

Table of Contents

Background

The university has been developing a parking system for its parking lot. The university allows community members along with students and staff members to register with the parking office. Registering with the school gives users a parking permit to use the parking lots. Although I have been working on the university's system individually, it is vital to consider the importance of continuous integration (CI) and deployment (CD) for future implementation at a greater scale. The automation surrounding CI and CD aims to discover and resolve bugs more efficiently, share code, and implement software enhancements quickly to increase software quality. I have begun implementing CI and CD basics through the current system development.

Existing Implementation

Using a repository to commit code is a fundamental component of CI and CD. So far, I have used GitHub to maintain my code for easy access and to have version control for the parking system. Having the system's code in the repository would allow other developers to integrate their portions of code and locate bugs sooner. The current system has only been tested using unit testing. This contributes to the overall continuous deployment of the university parking system. Continuous deployment happens after constant integration, which allows developers to test code in environments like production. Due to how extensive CI and CD are, a more detailed outline is presented below.

Implementation Outlook

Setting up a system that implements continuous integration and deployment requires several tools and technologies. As with any system, developers would begin by writing the code. The code is written and modified locally. Once the code is written, it is committed to a

repository. As mentioned before, the parking system would need a version control system. Currently, GitHub, GitLab, and Bitbucket are used. Systems such as these can host and track code changes, allowing changes to be reversed.

After the code is added to the repository, a pipeline is created to automatically run build and unit tests and store and review code. Using a pipeline is essential for integration as it allows for more efficient code integration due to trigger words that ignite the workflow. Some examples of pipeline tools are Sonar, Jenkins, GitHub Actions, GitLab CI/CD, and Circle CI. The system can be configured to pipelines whenever there is a merge or commit for the codebase. It is important to signal that there is a difference between the CI/CD and CI/CD pipelines. CI/CD is a set of processes where multiple changes are made to the code simultaneously to improve the software delivery, and the pipeline is a process that takes code from development to production automatically and smoothly (GeeksforGeeks 2025). Tools such as Maven and Gradle would be used during the build stage. After building the code, the pipeline would pull it to compile and package it after installing the dependencies.

The pipeline stage leads into the testing stage. During Testing, a series of tests are run through a combination of testing frameworks. As mentioned, unit testing has been completed using JUnit, but other testing frameworks, such as Selenium for UI Tests and Postman for API tests, must also be considered. Unit tests are used to verify the individual code components. The UI tests ensure that the interface components work correctly, and then the API tests test components on the backend for proper behavior. Running the parking through a series of tests ensures that the released product is good quality. When all tests pass, the pipeline will continue onto the next stage, the development stage; however, when the system fails any tests,

developers will be notified of the unsuccessful tests through tools such as GitHub Actions or Jenkins, as well as the selected communication options, such as a notification or an email. Developers then go back into the code and troubleshoot the errors to resolve the test failures. After the code is reviewed, developers will submit a pull request to get approval to merge the changes into the main branch. Once the test run is successful, the CD pipeline moves into the development stage.

The development stage is reached solely after passing the tests within the pipeline. The system is sent to a staging server within the pipeline for final checks. After approval, it goes into a production server where users can use it. The tools used in this phase can be Docker Hub and Kubernetes to aid in containerization and the automated orchestration of multiple containers. Tools such as Azure, Google Cloud, and AWS can be used if cloud hosting is in question. The development and deployment of the app using a CI/CD pipeline automates the building, testing, and deployment and, in turn, reduces any manual labor associated with the tasks, both in the setup and maintenance phases.

Automation and Testing

Since implementing continuous integration and deployment relies heavily on automatic progressions through the pipeline, the technicalities of monitoring the system remain a question. Triggers configured into the system automate the building and testing workflow. These tools can be set up to send developers messages or emails with the build results. Testing frameworks each have a result dashboard indicating successful and failed tests. The dashboard can then be reviewed, indicating a manual adjustment to the code for prompt resolution. During the building stages, the CI/CD system produces build artifacts or build

outputs of files and data generated during the build and testing phases. The output can be compiled code, test reports, and other files necessary for deploying and running the app. During deployment, developers can also configure the system to be notified of successful implementations and any rollback alerts.

The outputs and reports from the CD pipeline vary from test results to notifications or alerts for changes that need to be made. Since the system is being built and tested in an automated flow, the reports and alerts are meant to guide developers in modifying the code, which will maintain efficiency. Because the system is always running tests on the code, the code in the repository most often contains an operable version. When changes are updated, a new version of the working code is released since approval must be obtained before merging the changes into the existing code. Having the outputs listed, issues can be identified early on and before most of them get released to the users in the deployment phase.

Conclusion

Implementing a fully automated approach to building, testing, and deploying software enhancements required exploring additional testing approaches and understanding how to fully activate the triggers for push and pull requests. The current approach has already considered and implemented unit testing, but interface and API testing would ensure the release of high-quality software. Future tasks include adding the missing components to improve the quality of the software.

References

Brown, Jason. 2022. “Rollback, Revert, roll-forward, oh my!” Medium. Published March 29, 2022. https://medium.com/@jasonkingsley.brown/rollback-revert-roll-forward-oh-my-1753d8d2e079.

GeeksforGeeks. 2025. “What is CI/CD?” GeeksforGeeks. Updated April 14, 2025. https://www.geeksforgeeks.org/what-is-ci-cd/.

GeeksforGeeks. 2024. “7 Best Testing Frameworks for Java Developers.” GeeksforGeeks. Updated September 17, 2024. https://www.geeksforgeeks.org/7-best-testing-frameworks-for-java-developers/.

GitHub Docs. n.d. “About pull requests.” GitHub Docs. Accessed May 15, 2025. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests

GitHub Docs. n.d. “Events that trigger workflows.” GitHub Docs. Accessed May 14, 2025. https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows.

Susnjara, Stephanie. 2024. “What is Continuous Deployment?” IBM Think. Published August 28, 2024. https://www.ibm.com/think/topics/continuous-deployment.

Susnjara, Stephanie. 2024. “What is continuous integration/continuous delivery (CI/CD)?” IBM Think. Published September 24, 2024. https://www.ibm.com/think/topics/ci-cd-pipeline.

TatvaSoft. n.d. “Popular Java Build Tools for Developers.” TatvaSoft sculpting thoughts. Accessed May 14, 2025. https://www.tatvasoft.com/outsourcing/2024/04/java-build-tools.html.

Wanda. 2024. “The Top 15 API Testing Frameworks: Your Ultimate Guide.” DEV. Posted August 19, 2024. https://dev.to/apilover/the-top-15-api-testing-frameworks-your-ultimate-guide-27ok.