**Week 7 Design a Continuous Build and Integration System**

for

Master of Science

Information and Communication Technology

Ammireddy Jonnala

University of Denver University College

May 14, 2025

Faculty: Nathan Braun

Director: Cathie Wilson, M.S

Dean: Michael J. McGuire, MLS

**Continuous Build and Integration System for Parking System Application**

Implementing a Continuous Build and Integration (CI/CD) system for the Parking System Application is crucial to enhancing its reliability, efficiency, and maintainability. This system, designed to ensure the application's stability, will streamline the development, testing, and deployment processes, ensuring that changes to the application are managed and deployed efficiently and with minimal risk. The setup will allow developers to focus on writing code, knowing that the application remains stable and functional. In this scenario, we will explain the setup of a CI/CD pipeline for the Parking System Application, covering the required systems, developer interactions, and expected outputs.

Components of CI/CD Pipeline

**Version Control System (VCS):** Version Control Systems (VCS) like Git are essential CI/CD system components. They are crucial in managing the source code for the Parking System Application, including tracking changes, collaborating with team members, and maintaining different versions of the codebase. The significance of VCS cannot be overstated in the context of our proposed CI/CD system.

**Continuous Integration Server:** A continuous integration server, such as Jenkins or Travis CI, is a tool that automates the process of building and testing code changes. When developers, the key players in this process, add new code to the version control system, the CI server automatically triggers a build process, which compiles the code, runs tests, and generates reports. This helps ensure that code changes are thoroughly tested and integrated into the project smoothly, which can prevent issues down the line.

**Build Automation Tools:** Build automation tools like Maven or Gradle are recommended to manage dependencies, compile code, and generate executable artifacts such as JAR or WAR files.

**Automated Testing Framework:** The CI/CD pipeline comprises various test environments, including development, staging, and production environments. Each environment replicates different stages of the application lifecycle, allowing developers to test changes in isolated environments before deploying them to production.

**Code Quality and Static Analysis Tools:** Integrating source code analyzers like SonarQube and PMD into the Continuous Integration pipeline can help identify potential issues, enforce coding standards, and improve code quality.

**Artifact Repository:** Once the code is built and tested successfully, the CI server produces deployable artifacts like executable binaries or Docker images. These artifacts are then stored in an artifact repository, such as Nexus or Artifactory, which keeps track of build artifacts, dependencies, and libraries generated during the CI/CD procedure. This ensures that the artifacts are well-organized, versioned, and easily accessible for deployment across various environments.

**Deployment Pipeline:** The deployment pipeline is a sequence of stages that code alterations go through, starting from development and ending at production. It involves automated tasks, such as deploying staging environments, running additional tests, and promoting artifacts to production. The CI server manages the deployment pipeline and ensures that code changes are deployed safely and efficiently.

**Infrastructure as Code (IaC) Tools:** Tools like Ansible, Puppet, or Chef can automate the deployment process, ensuring consistent and repeatable deployments across different environments. These tools can automate infrastructure setup and configuration, including servers, databases, and networking components.

Developer Interactions

**Code Changes:** Developers locally change their code and push it to a central Git repository. They follow best practices for maintainability, documentation, and code organization to ensure seamless integration with the existing codebase.

**Continuous Integration:** After pushing code changes to the version control system, the continuous integration and delivery server detects the new commits and triggers a build job. The build job compiles the code, runs unit tests, and performs a static code analysis to identify any issues or errors.

**Automated Testing:** The CI/CD pipeline consists of automated tests, which include unit tests, integration tests, and end-to-end tests. These tests are used to verify that the Parking System Application functions correctly and performs well. The developers create test cases and set up the CI/CD server to run the tests automatically during the building process.

**Code Review:** Developers participate in code reviews to ensure code quality, consistency, and adherence to coding standards before merging changes into the main branch. Code reviews include feedback, constructive criticism, and improvement suggestions from peers.

Output Expectations

**Build Artifacts:** The CI/CD pipeline generates output artifacts, including executable binaries, libraries, and configuration files. These artifacts are versioned, tagged, and stored in an artifact repository for traceability and reproducibility.

**Test Reports:** Once the automated tests are run, the CI/CD server generates test reports that give a summary of the test results. The report includes the number of passed tests, failed tests, code coverage metrics, and any identified issues or errors. By analyzing test reports, one can evaluate the quality and reliability of the Parking System Application.

**Deployment Packages:** After the build and test stages are completed, the CI/CD pipeline generates deployment packages that include both the application code and configuration files. These packages are then deployed into different target environments, such as development, staging, and production, with the help of automated deployment scripts or Infrastructure as Code (IaC) tools.

**Continuous Deployment:** In a mature CI/CD pipeline, continuous deployment automates the deployment of approved changes to production environments without manual intervention. It ensures that validated changes are delivered to end-users quickly and efficiently, minimizing downtime and accelerating time to market.

Processes of a CI/CD System

**Continuous Integration:** Developers work on feature branches and regularly merge their changes into the main branch (e.g., primary or develop). The CI server monitors the version control system for new commits and triggers automated builds for each commit.

**Automated Testing:** As part of the software development process, a Continuous Integration (CI) server performs a series of automated tests to ensure the code is functional and high-quality. These tests comprise unit tests to validate individual components, integration tests to test the interactions between different parts of the system, and end-to-end tests to simulate user interactions and verify the overall performance of the software.

**Continuous Deployment:** After the code changes have passed all the necessary tests, the CI server generates deployable artifacts, which are then deployed to staging environments for further testing. In the staging environment, automated and user acceptance tests are carried out to verify the changes are valid before being promoted to production.

**Continuous Monitoring:** After deployment, the CI/CD system continuously monitors the performance and health of the Parking System Application in production. Metrics such as response time, error rate, and resource utilization are collected and analyzed to detect anomalies and troubleshoot issues proactively.

Benefits of Implementing a CI/CD System:

**Faster Time to Market:** CI/CD enables rapid delivery of new features and updates to the Parking System Application by automating the build, test, and deployment processes. This leads to shorter release cycles, faster time to market, and greater frequency of code changes.

**Improved Code Quality:** Continuous integration and automated testing help maintain the quality of code by detecting bugs and regressions early in the development cycle. This reduces the risk of introducing defects into production and enhances the application's overall reliability.

**Increased Collaboration:** Implementing a CI/CD (Continuous Integration/Continuous Deployment) system can promote teamwork and cooperation among development teams. The system offers a centralized platform for version control, code review, and issue tracking, which enables developers to work together more efficiently, share their knowledge, and resolve problems collectively. This approach leads to improved outcomes for the Parking System Application.

**Scalability and Efficiency:** Continuous Integration and Deployment help automate repetitive tasks and streamline the software delivery process. This also allows development teams to scale their operations efficiently. With CI/CD, organizations can handle larger codebases, onboard new developers more quickly, and easily adapt to changing requirements.

**Continuous Feedback Loop:** Continuous Integration and Continuous Delivery (CI/CD) is a software development approach that provides immediate feedback on the quality and performance of code changes. This feedback loop allows developers to iterate quickly and make informed decisions, leading to a culture of continuous improvement. Teams learn from their mistakes and strive for excellence in software development, resulting in high-quality, reliable software products.

Conclusion

Implementing a CI/CD pipeline for the Parking System Application is crucial to streamline the development lifecycle, improve code quality, and increase the efficiency of deployment processes. By automating build, test, and deployment processes, CI/CD allows development teams to innovate faster, collaborate effectively, and deliver high-quality software products that meet the needs of the University parking office and its customers. By leveraging automation, developers can concentrate on delivering value to end-users while maintaining confidence in the reliability and stability of the application. A well-designed CI/CD pipeline accelerates the feedback loop, enables rapid iteration, and encourages a culture of continuous improvement within the development team.

**References**

Fowler, Martin. 2011. “Continuous Integration.” Martinfowler.com. 2011. <https://martinfowler.com/articles/continuousIntegration.html>.

Humble, Jez, and David Farley. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education. <https://books.google.com/books?hl=en&lr=&id=6ADDuzere-YC&oi=fnd&pg=PT34&dq=Continuous+Delivery:+Reliable+Software+Releases+through+Build,+Test,+and+Deployment+Automation&ots=-xpsOQHaic&sig=EtRGzFOu77ONipKqZzqd45Fmr88#v=onepage&q=Continuous%20Delivery%3A%20Reliable%20Software%20Releases%20through%20Build%2C%20Test%2C%20and%20Deployment%20Automation&f=false>

Newman, Sam. Building microservices. 2021. " O'Reilly Media, Inc.", <https://books.google.com/books?hl=en&lr=&id=ZvM5EAAAQBAJ&oi=fnd&pg=PT8&dq=Building+Microservices:+Designing+Fine-Grained+Systems&ots=ui5jaB4HZp&sig=4wpwZSP3NiAM3d8sOo8e2D0HAXs#v=onepage&q=Building%20Microservices%3A%20Designing%20Fine-Grained%20Systems&f=false>

Bobrovskis, Sergejs, and Aleksejs Jurenoks. 2018. "A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment." In BIR workshops, pp. 314-322. <https://ceur-ws.org/Vol-2218/paper31.pdf>.

Nath, Mahendra, Jayashree Muralikrishnan, Kuzhanthaiyan Sundarrajan, and Madhu Varadarajanna. n.d. "Continuous integration, delivery, and deployment: a revolutionary approach in software development." *International Journal of Research and Scientific Innovation (IJRSI)* 5, no. 7 (2018): 185-190.

<https://d1wqtxts1xzle7.cloudfront.net/64077768/Continuous_Integration__Delivery_and_Deployment__A_revolutionary_%282%29-libre.pdf?1596379306=&response-content-disposition=inline%3B+filename%3DContinuous_Integration_Delivery_and_Depl.pdf&Expires=1715378040&Signature=U~PRqqWhplJmztAcbpFeyP9JcgowtDNr~UP12tntgg5Y9k-z1oubx7DJGY0iwQIv3y8Z8KbgUjlMbx08Y2lwXWWzB5vi7d9Ox6NexIr-42aAlNaAXNF5M1veWR5yWkjJIRYWYCbDECmk04MksHzBL1ORA0Y9w5JanH--I43laP7JWZ-UsyKB3krXgZY89iaB5gx7G6mxVR3mJpRNnwDZNZOUTOa-CTQxzou6SlAtTHhImYoAy3Ib2z6JxAXNr0wkg5iesE5zUIF4THD3sHYLjgwIcDWWrr~QI9s-4QZAiGyrET5CUZEhdG3asT-mh8TmLlq61Ieuf3UNUoGvdf0Trw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>