УДК 621.326

Петришин Я. – ст. гр. СПс-21, Марцинюк Я.
Тернопільський національний технічний університет імені Івана Пулюя

РОЛЬ CI/CD У ПІДВИЩЕННІ ЕФЕКТИВНОСТІ ТА НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Науковий керівник: Кравчук Г.Б.

Petryshyn Y., Martsynyuk Y.
Ternopil Ivan Puluj National Technical University

ROLE OF CI/CD IN IMPROVING SOFTWARE EFFICIENCY AND RELIABILITY

Supervisor: Kravchuk H.B.

Ключові слова: CI/CD, ефективність, надійність, якість.
Keywords: CI/CD, efficiency, reliability, quality.

CI/CD, or Continuous Integration/Continuous Deployment, has become a necessary practice for modern software development [1, 2]. Coordinated CI/CD processes help ensure efficiency, reliability and high product quality [3]. Continuous Integration (CI) is a process that allows developers to integrate their changes into the core code of an application on a regular basis. This creates the ability to quickly identify and resolve conflicts between different versions of code, and to automatically run tests to verify the correctness and integrity of new code. This approach allows developers to respond more quickly to changes in market requirements and make the necessary changes to the software. After successfully passing the CI phase, the code is ready to be deployed in the production environment. Both Continuous Delivery and Continuous Deployment consist of the following stages: Acceptance test, Deploy to staging, Deploy to production, Smoke test [4].

The only difference is that Deploy to production stage is done manually in Continuous delivery. Continuous Delivery/Continuous Deployment (CD) provides a way to implement software changes. Continuous Delivery involves the automated implementation of changes in the production environment according to certain guidelines or criteria. This allows developers to maintain control over the deployment process and choose when to implement changes. On the other hand, Continuous Deployment involves fully automated implementation of changes immediately after successful completion of all tests and checks. This approach ensures the fastest introduction of new features and fixes to the product.

Consider a specific UI project (Fig. 1) and its CI/CD process. At the initial stage, code testing is performed. Dependencies are installed and tests run to check that the code works as expected and is bug-free. Static code testing is also performed at this stage to detect errors in the early stages of development, which allows efficient use of time and resources. This includes code reviews, style checks, system requirements testing, and other methods aimed at evaluating the quality of code and documentation without actually executing it.

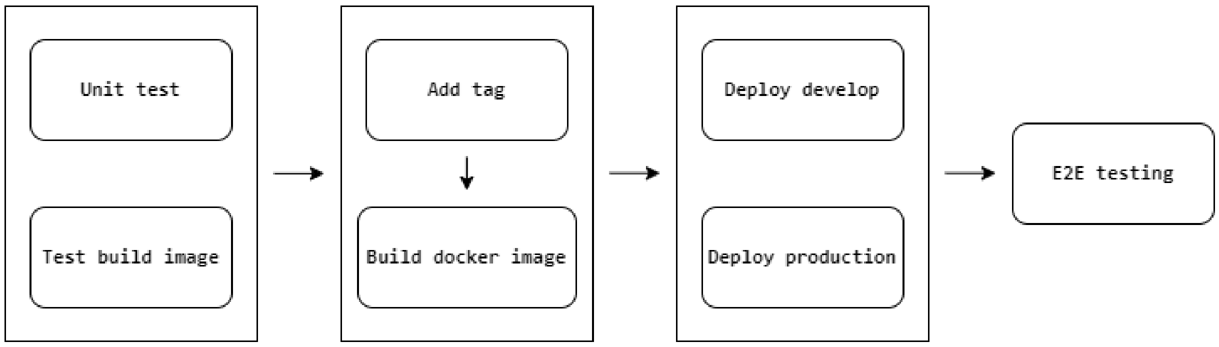

Figure 1 – CI/CD of UI project

Building the Docker image to be used for further deployment is done as the following step, based on the code. After the tests pass successfully, a new tag is added to the Git repository to indicate the version of the application and help track changes. This stage is started manually. Next, the Docker image is built using the generated tag. This means that each Docker image will be tagged according to the version of the application, which helps to track releases and determine which version of the application is being used. Next comes the deployment stage. The first step is to deploy to a test environment. At this stage, the tag in the configuration files is replaced with the one specified in the current repository. Finally, the updated code is deployed to the production server which also requires a manual start. At the final stage, end-to-end (E2E) tests are conducted automatically to check the operation of the application as a whole, from frontend to backend. This allows identifying problems that may arise due to the interaction of various system components. Running E2E tests helps ensure testing is complete. E2E testing completes the pipeline.

The above steps demonstrate how the CI/CD process helps improve the efficiency, reliability and quality of software development, allowing for the execution of repetitive tasks without human intervention. This not only saves time but also reduces the likelihood of human errors. Automation and standardization of processes allow developers to respond more quickly to changes in market requirements and ensure product stability and reliability [5]. This approach helps reduce time to implementation of changes, increases code quality and provides competitive advantages in the software market.

References

1. ISO/IEC/IEEE 12207:2017, IDT - Systems and software engineering. Software life cycle processes. URL: https://www.iso.org/standard/63712.html.

2. ISO/IEC/IEEE 29148-2018, IDT - International Standard. Systems and software engineering. Life cycle processes. Requirements engineering. URL: https://ieeexplore.ieee.org/document/8559686.

3. Software Engineering Body of Knowledge (SWEBOK). URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering.

4. Continuous integration vs. delivery vs. deployment. URL: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs - delivery-vs-deployment.

5. Constant Readiness or What is CI/CD in DevOps. URL: https://dataforest.ai/blog/constant-readiness-or-what-is-ci-cd-in-devops.