

**МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

CI/CD

ISNS-1909R

**Lysenko Nikita**

CI/CD is one of the DevOps practices. It also applies to agile practices: automation of deployment allows developers to focus on the implementation of business requirements, on code quality and security.

**CI/CD Goals:**

providing a consistent and automated way to assemble, package, and test products or applications;

automation of deployment in different environments;

minimizing errors and problems.

**CI/CD Principles**

There are four guiding principles for CI/CD:

**Division of responsibility.** Each of the participants in the process shares responsibility for certain stages of the product lifecycle. Business logistics is being designed, end-to-end functions are being implemented, acceptance tests are being conducted and code logistics is being organized.

**Risk reduction.** Each team involved in product development strives to reduce risks — the correctness of business logistics is monitored, user experience is checked, data storage and processing are improved, and so on.

**Shortening the feedback loop.** The developer and the client should strive to increase the speed of making changes and approving edits. Code assembly and testing can be automated. And for situations where human participation is required, it is possible to minimize the number of information intermediaries.

**Implementation of the environment.** Developers should have a common workspace with the main and auxiliary branches for version control and quality, acceptability, fault tolerance and other criteria.

**CI/CD Stages**

The CI/CD methodology involves dividing the development process into seven stages:

**Writing code.** Developers write the code of their module and conduct manual testing. After that, the result of the work is connected in the main branch with the current version of the project. After all the module codes are published in the main branch, the second stage begins.

**Assembling**. The selected version control system initiates the automatic build and subsequent testing of the project. Triggers for activating the build can be configured independently. Jenkins or another tool is used to automate the assembly.

**Manual testing.** After the CI system checks the operability of the test version, the code is transmitted for manual investigation.

**Release**. After manual testing, corrections are made to the assembly. This is followed by the release of the code version for customers.

**Deployment**. At this stage, the current (working) version of the code is hosted on the developer's production servers. The client can interact with the program and learn its functions.

**Support and monitoring.** The product begins to be used by end users. At the same time, the developers continue to support it and analyze the user experience.

**Planning.** Based on user experience, new functionality is being developed and a plan of improvements is being prepared. After that, the developer starts writing code — and the cycle closes.

**Tools for CI/CD**

In software development, various tools can be used to automate testing processes and deliver code to end users. Here are some of them:

**GitLab.** The environment makes it possible to manage project repositories, document test results (improvements) and functionality, as well as track bugs.

**Docker.** An automatic project deployment system that supports containerization, which makes it possible to package a project with the entire environment and dependencies.

**Travis-CI.** A tool capable of connecting to repositories in GitHub with the least settings.

Travis-CI is a cloud—based tool, so its installation is not required.

**Jenkins.** A popular DevOps tool that allows you to work with different plug-ins to flexibly configure processes for specific tasks of the product being developed.

**PHP Censor**. A continuous integration server designed to automate the assembly of PHP projects. It works with GitLab, GitHub, Mercurial and other repositories, as well as with Atoum, PHP Spec, Behat testing libraries. The tool is documented, but requires self-configuration and hosting.

**Conclusion**

CI/CD is not just a methodology, but also a very good tool for the development team. The longer it is planned to develop and maintain a project, the more benefit it will bring to having a well-designed CI/CD.