UDC 004.8: 004.94 https://doi.org/10.20535.2786-8729.5.2024/318795

CI/CD INTEGRATION TOOLS FOR AUTOMATED CODE DEPLOYMENT AND VERIFICATION FOR TRAINING PURPOSES

Viktoriia Babenko *
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
https://orcid.org/0009-0008-1762-6671

Viktoriia Taraniuk
GlobalLogic Sweden AB, Gothenborg, Sweden
https://orcid.org/0000-0001-9044-1499

Valentyna Tkachenko
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine
https://orcid.org/0000-0002-1080-5932

Iryna Klymenko
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine
https://orcid.org/0000-0001-5345-8806

*Corresponding author: babenko.viktoriia@lll.kpi.ua

The article is devoted to the study and application of modern tools for Continuous Integration and Continuous Deployment (CI/CD) in the educational field. Automating the processes of software deployment and testing is a relevant task for both improving the educational process and developing DevOps skills among students. Significant attention is given to studying the core principles of CI/CD, including automated testing, code quality monitoring, and integration with source code repositories.

Popular CI/CD platform such as Jenkins is utilized to automate the educational process and train students. This tool enables the creation and deployment of applications using Docker technologies, which allow real-world scenarios to be modeled. A significant emphasis is placed on the scalability and adaptability of solutions, which enhance the efficiency of resource usage.

A methodology for implementing CI/CD into an educational course is proposed, including integration with project management platforms and version control systems such as Git, with Gitea as an example. The main stages include setting up automated builds, testing, and deployment, which enable students to practice the principles of continuous integration and delivery. From the perspective of improving the efficiency of the educational process, the proposed methodology allows for the automation of assignment verification. The problems of Gitea and Jenkins integration are considered. A way for integrating these tools through locally installed Jenkins and Gitea with private code repositories has been proposed. Recommendations are provided for organizing the educational process through practical and laboratory work focused on real-world scenarios of software deployment and test automation.

The results of the study confirm the effective use of CI/CD tools for educational purposes, ensuring the development of competencies required for working in modern IT teams. The use of CI/CD increases awareness of cybersecurity and optimizes DevOps processes.

Key words: CI/CD, DevOps, Jenkins, Git, Gitea, automated testing, pipeline.

1. Introduction

In today's world, automation permeates all spheres of activity, including software development. CI/CD technologies such as Gitlab CI/CD, Github Actions, Jenkins, and TeamCity make it much easier for developers to ensure continuous integration and deployment of software. This allows you to automate the deployment and code verification processes, which reduces the risk of errors and increases the quality of the product. The relevance of the article lies in addressing the growing demand for automation tools in education to prepare students for real-world challenges in software development.

The introduction of these technologies into the educational process not only shortens the time of checking tasks, but also contributes to the development of relevant skills among students of IT specialties. CI/CD tools help automate routine tasks such as building, testing, and deploying software that develop DevOps skills. Visualization of results with the help of dashboards allows you to evaluate achievements more effectively, and working with the Git version control system teaches you how to collaborate effectively in a team.

The problems of Gitea and Jenkins integration are considered. The article presents the architectural concept of automated task verification in the Gitea remote repository and proposes a training system for automated software code verification. The purpose of the development is to create an infrastructure for teamwork on educational projects, to ensure professional competence in the use of CI/CD tools, as well as quick and objective access to the results of educational tasks, especially in the distance learning format.

2. Analysis of literary sources and statement of the problem

Integrating Gitea and Jenkins is an urgent task in today's DevOps environment, where automation of software development and deployment processes becomes critical. As a lightweight and easy-to-use code storage and management service, Gitea is popular among small teams and projects. Jenkins, in turn, is one of the most common tools for automating CI/CD processes, providing flexibility and scalability. The problem is that without proper integration of these two systems, it is difficult to achieve a smooth and continuous development process that includes code version control, automated testing and deployment.

One of the main challenges is to ensure a reliable and uninterrupted data exchange between Gitea and Jenkins, which will allow CI/CD processes to be automatically started when changes are made to the code. This requires setting up webhooks, properly managing authentication and authorization, and taking into account the specifics of each tool. Improper integration can lead to disruptions in the development process, delays in deployment, and an increased risk of code errors, highlighting the importance of proper planning and execution of this process.

Automated testing is a key element of modern software development practices. In a significant number of modern works [1, 2, 3] the wide use of CI/CD tools for solving various automation tasks is considered. In general, the technology involves the use of specialized tools to automate the testing process, including execution, verification and analysis of results [4].

Regardless of the used methodologies (Agile, Scrum, Kanban, etc.) and development technologies, CI/CD integration allows useful adapting to any workflow, while ensuring high flexibility and team efficiency. It also improves collaboration between developers, testers, and other project participants, as everyone has access to up-to-date information on the project's status and can make the necessary adjustments in real-time.

This study will further review various architectural concepts of automated commit verification systems and various hosting services for git. In each system, the main concepts and principles of automation will be analyzed in detail and revealed. Such automation systems would be Jenkins, Gitlab CI/CD, and Github Actions. Regarding hosting services - Github, Gitea. Gitlab CI/CD and Github Actions offer built-in automation solutions described in [5, 6, 7] allowing easy configuration, integration and deployment processes directly within the hosting platform. This reduces the need for additional infrastructure and makes it easier to set up automated workflows.

Jenkins and TeamCity are powerful tools that can be integrated separately using plugins. They offer a wide range of possibilities for customization and expansion of functionality, making them ideal for complex and scalable projects. These tools allow creating customized pipelines for building, testing and deploying software, providing high flexibility and adaptability to different project requirements.

Unit testing improves code quality, simplifies development, and improves software stability. Test automation gives developers the freedom to integrate new technologies. The Git version control system is widely used for code management, and services such as GitHub, GitLab, and others provide convenient publishing and collaboration tools, as described in [8, 9].

Gitea is a cross-platform service based on Git, created on the basis of the Gogs project. It supports bug tracking, wiki usage, and code review, the use of which is described in the article [10].

There are many frameworks for unit testing in different programming languages, such as CTest, Google Test for C++. Automated testing increases efficiency, repeatability, functional coverage, reliability and reduces future testing costs.

CI/CD toolkit – Jenkins. Jenkins is one of the most popular services for integration with repository hosts. It has an extensive plugin system, making it a versatile and widely used software development tool. According to the above, it is necessary to investigate the possibilities of introducing automated testing into the educational process to improve the quality of the software being developed. The relevance of using CI/CD is described in [11, 5, 9]. The problem of ensuring the security and integrity of educational projects in the context of integrated local repositories is especially relevant.

Based on the performed review of modern technologies and the analysis of publications in literary sources on the use of these technologies, a general conclusion can be drawn that the use of automation technologies, such as GitLab CI/CD, GitHub Actions, Jenkins and TeamCity, contributes to increasing the efficiency of developers' work, ensuring a smooth process software development, testing and deployment. The use of these technologies in integration with joint project management systems requires the development of methods of integration and adaptation to specific target tasks.

Jenkins provides the most functionality among all tools and has a clear interface that makes it possible to easily integrate services and conduct testing.

In the context of this study, the problem of ensuring the security and integrity of data in local repositories of educational institutions was identified, which requires the development of special tools for the integration of access control and project management in student teamwork. In the context of the advantages of use in educational institutions, a conclusion should be drawn about the relevance and expediency of developing tools for testing automation and version control, which will increase the quality of the educational process and contribute to better preparation of students for the real conditions of software development.

Thus, the identified problem area indicates the need to develop an educational system architecture for automating task verification processes in Gitea repositories, which will provide a convenient tool for effective interaction between teachers and students. Main goals:

– improving communication;
– acceleration of verification;
– reduction of the teacher's workload;
– development DevOps-skills.

3. The purpose and objectives of the study

The purpose of the study is to develop an approach and tools for automated testing of software code for a remote repository storage system, using Gitea as an example. The implementation of the research goal is aimed at increasing the effectiveness of the educational process in various formats of the educational process, expanding the range of technologies that students use in their work, accelerating the updating of monitoring of completed tasks and saving time resources.

To achieve the goal, the following tasks were set:

– to develop an architectural concept of a system of automated verification of software code and a set of software tools with the possibility of integration into educational and methodical complexes of educational institutions of higher education that train specialists in the field of modern

IT technologies, which is capable of providing effective training in the use of CI/CD tools and automation of monitoring the performance of educational tasks;

– to implement the architectural concept and means of integration of automated testing of completed tasks based on Gitea, develop a set of software tools that will ensure continuous integration and delivery of completed tasks in the remote Gitea repository to Jenkins.

4. Development of the architectural concept of the automated software code verification system

4. 1. Justification of the choice of means for joint project management

Git was used as the primary tool for version control and project collaboration. Git provides efficiency through collaboration, version management, branch creation and merging, and merge conflict resolution. In addition, Git allows working on versions of a project locally without a constant Internet connection, which is especially useful for educational projects. This makes it ideal for university environments where data security and copyright control are important, as open repositories can increase the risk of academic misconduct.

Gitea is chosen as a stand-alone Git service that provides local hosting and code control. Its advantages over other hosting platforms include:

– Full control: Gitea allows universities to create their own private repositories to protect copyright and avoid plagiarism.

– Simplicity and ease: Gitea consumes minimal resources and is easily configurable across platforms.

– Self-hosted: The ability to install the system on its own server allows the university to have full control over storage and data.

– Integration with CI/CD: Although Gitea does not have built-in CI/CD tools, the ability to integrate via web hooks allows connecting external services for automation, such as GitLab CI, Jenkins, or GitHub Actions.

For Windows, Git Bash was used as a lightweight tool for working with Git on the command line, and for Linux and macOS, Git was installed through the terminal, which ensured a unified operation regardless of the operating system.

Gitea is a versatile and user-friendly Git hosting solution that allows hosting Git repositories and collaborate on code with team. It offers many of the features available in the larger Git hosting platforms, but is lightweight and easy to set up.

As for continuous integration tools support, there are no built-in plugins. But the advantages of using it are quite significant, so the article has developed a way of interaction between Jenkins and Gitea.

4.2. Rationale for choosing tools for CI/CD integration

Tools such as Gitea and Jenkins were used to implement the continuous integration and delivery system within the project. These tools provide efficient automation of software development, testing and deployment processes. Let's consider in more detail how exactly they were applied. Gitea – a stand-alone platform for hosting Git repositories – was chosen for repository management and code version control. It was used to store student projects, which provides:

– Data privacy: Gitea allows creating private repositories, which is important to preserve copyright and protect information.

– Ease of use: Students can easily upload their work, and teachers can easily track progress and review changes through version control.

After creating the repository, the student must initialize the local repository and upload the project to the remote repository on Gitea. After creating a repository, user need to generate a repository access token in Jenkins to integrate with Jenkins. This token entitles continuous integration tools to monitor repositories.

To automate the continuous integration (CI) and delivery (CD) processes, Jenkins was used, which allowed to create an automated process of code verification and testing:

– Integration with Gitea: Using web hooks enabled tests to be automatically run in Jenkins after each commit in the Gitea repository. This speeds up project review and allows teachers to get results faster.

– Automated testing: Jenkins automatically ran tests after changes were made to the code. This made it possible to detect errors at an early stage and ensure high quality code before deployment.

– Automatic deployment: After successful tests, Jenkins automatically deployed the software, which greatly simplified the process and reduced the number of errors during manual deployment.

As a result, the following architectural concept, see Figure 1, of the system was developed:

1. Users work on their projects locally and store code in Gitea repositories.
2. Gitea sends web hooks to Jenkins for every new commit or change in the repository.
3. Jenkins runs a pipeline that executes:
4. Automatic assembly of the project.
5. Automatic testing.
6. Generation of reports.
7. If the build and tests pass, Jenkins automatically deploys to a server or cloud platform.

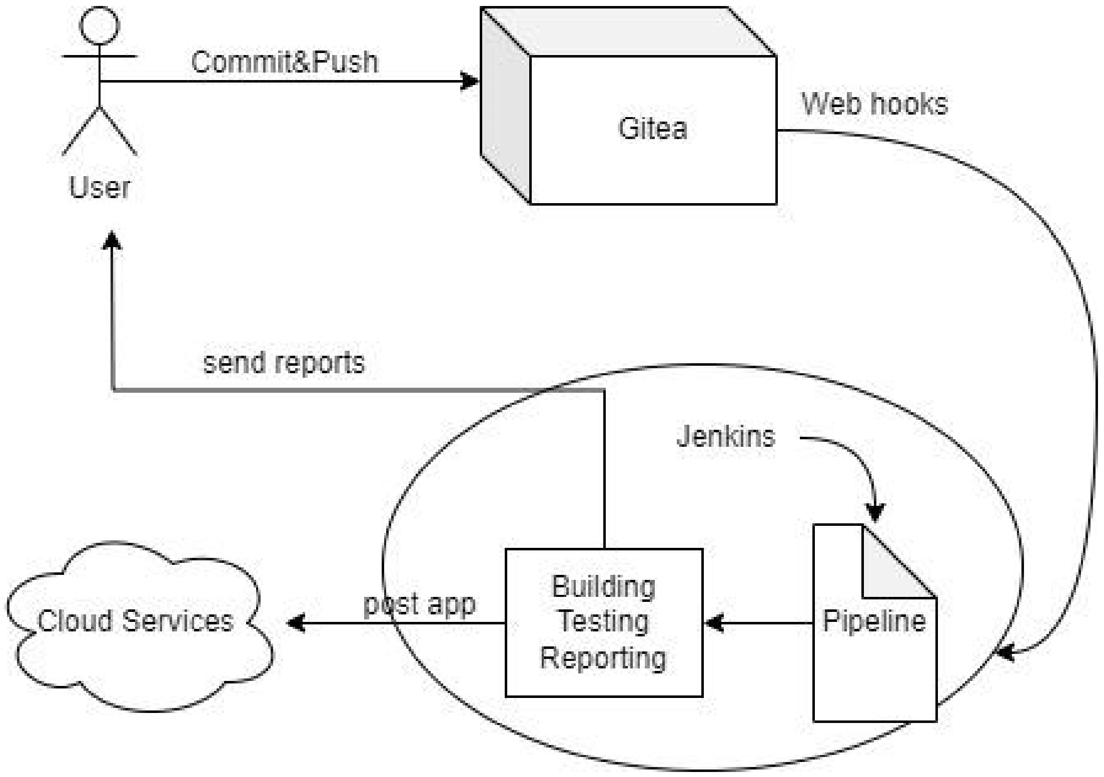


Fig. 1. Architectural concept of the system

A pipeline, also known as a build script, is an automation mechanism in Jenkins that allows a developer to describe and manage the sequence of steps that must be performed during the development, testing, and deployment of software. Using Jenkins Pipeline allows creating structured and repeatable build processes.

Jenkins Pipeline syntax can be expressed in two formats: declarative and scripted (Scripted Pipeline) [12]. Both formats describe a sequence of steps to automate the process of building, testing, and deploying software.

xUnit is a Jenkins plugin that provides the ability to visualize test run reports on the Dashboard. Integrating a project build with this plugin is quite easy, as it supports the execution results of Google Test, Junit, MSTest, UnitTest++, Boost Test Library [13, 14] etc. The plugin provides an opportunity for creating a convenient infographic of completed tests and quickly processes report files, which improves the functionality of the system.

MSBuild plugin is a project build tool built into Visual Studio. The plugin uses XML-like project files to define project structure and build tasks. MSBuild is responsible for building, compiling, packaging, and deploying software.

The MSBuild plugin for Jenkins is a Jenkins extension that adds support for using MSBuild to build projects. This plugin usually uses MS Build to compile and build software during continuous integration/continuous deployment process in Jenkins. It provides the ability to configure build options, set the path to the MSBuild project file, and control the build process.

Using the MSBuild plugin for Jenkins, developers can integrate MSBuild project builds into their continuous integration and continuous deployment processes, automate software builds and testing, and import build information into Jenkins to analyse and monitor the build process.

5. The results of the research of CI/CD integration tools for the system of automated verification of software code

5.1. Implementation of the architectural concept of the system of automated verification of software code

In accordance with the developed materials and selected technological means for the implementation of the architectural concept, it is possible to make the transition to the creation of the software part of the automated task verification system in the Gitea remote repository. In order to make the automation architecture as flexible as possible, it is necessary to describe the main parts of the system's functionality - program blocks, the main scenarios of project assembly.

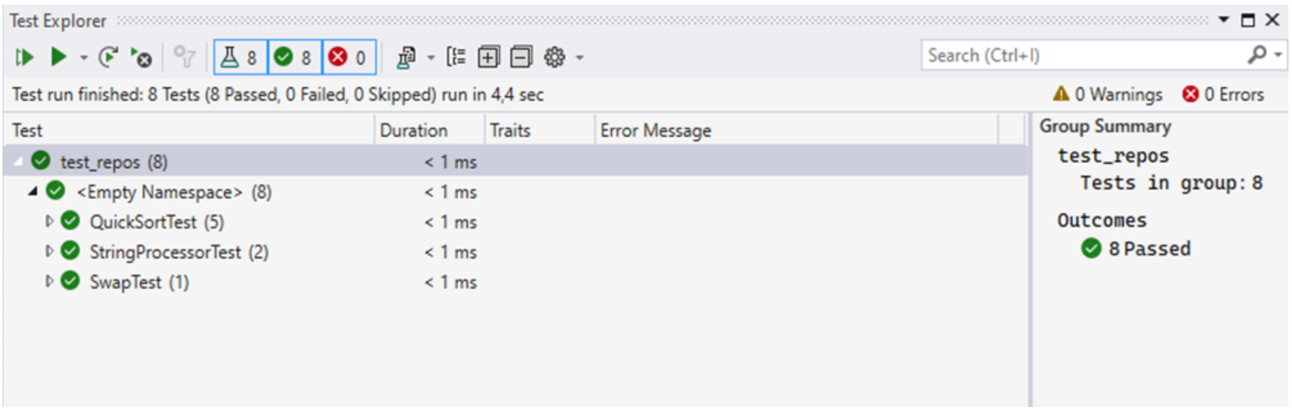
According to the given problem, the following implementation of the architectural concept was carried out in the form of a template repository for the fork and detailed instructions for using frameworks and Jenkins.

The project directory allows students to work with an already configured framework with unit tests. Developer only need to work with the raw code in the main project directory. In order to create such a project, a project in Visual Studio Google Test for C++ needs to be created. After fully initializing the project and adding the main raw files, the student should get a standard project structure. To automate the process of checking tasks, a task for working with pointers has been selected.

In addition, the Google Test tool is already loaded in Visual Studio in this project configuration. This avoids errors when connecting the test tool directly. This framework is attached *via pch.cpp* and *pch.h files*.

For educational purposes, code blocks were replaced with comments for further educational work. Comments in these files can be replaced by student developers with their own solutions. In fact, by modelling such template files, the teacher provides the student with the opportunity for his own implementation of tasks with his creativity.

After running the unit tests in Visual Studio, the results of the Google Test handler are displayed as shown in Figure 2.

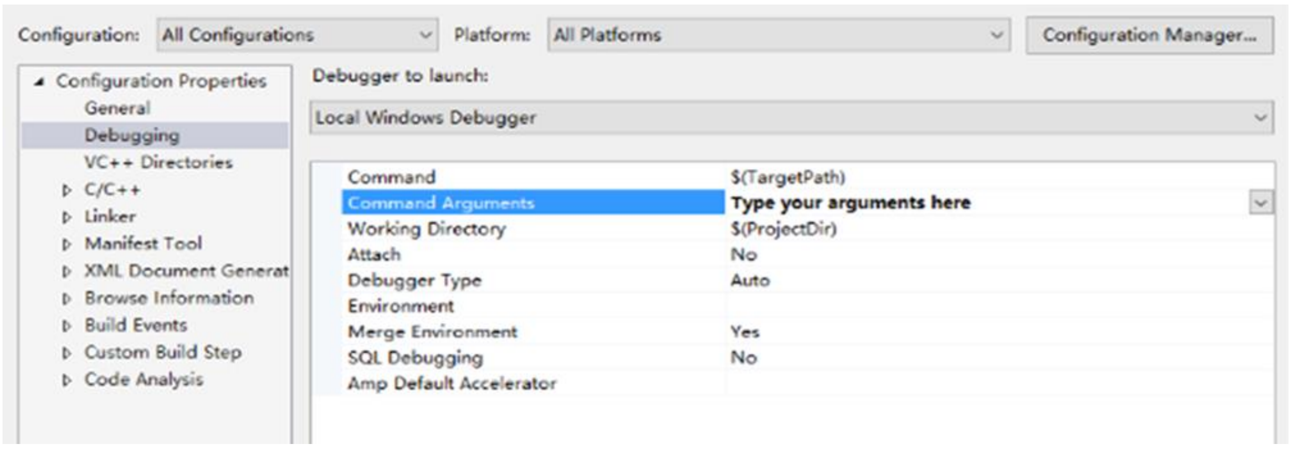


```
Test run finished: 8 Tests (8 Passed, 0 Failed, 0 Skipped) run in 4,4 sec      0 Warnings   0 Errors

Test                              Duration   Traits   Error Message     Group Summary
test_repos (8)                    < 1 ms                                 test_repos
  <Empty Namespace> (8)           < 1 ms                                   Tests in group: 8
    QuickSortTest (5)             < 1 ms                                 Outcomes
    StringProcessorTest (2)       < 1 ms                                   8 Passed
    SwapTest (1)                  < 1 ms
```

Fig. 2. Test Explorer

In order to receive an .xml file with the results of the test framework in the project, user need to adjust the Debug process. To do this, use Project Properties and set the appropriate environment variables, as in Figure 3.



Configuration: All Configurations
Platform: All Platforms

Debugger to launch:
Local Windows Debugger

Command	$(TargetPath)
Command Arguments	**Type your arguments here**
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	

Fig. 3. Fixing Debug process in Visual Studio

Debug provides the generation of a valid .xml file that can easily be processed by the xUnit plugin to publish test results to the server.

5.3. How to integrate Jenkins and Gitea

Jenkins was chosen as the continuous integration and continuous delivery tool. This tool provides an opportunity to openly monitor both private and public repositories, organizations, and various branches in the repositories. The architecture of the project is based on the fact that every student should have his own private repository - this will reduce cases of plagiarism.

To run Jenkins via localhost, the Jenkins service must first be started through Services on Windows. After the service is successfully started, Jenkins Server can be started on localhost. After completing the previous steps, the Jenkins Server would be able to work with Gitea repositories.

To work with Gitea as a repository, the Git plugin is installed in Jenkins

When building the project, the MSBuild plugin was used, in order for the plugin to work correctly, it needs to be configured. This option is performed by specifying the path to the directory with the MSBuild.exe executable file.

MSBuild provides the ability to build Visual Studio solutions on a Jenkins server.

The xUnit plugin is used to publish test results. xUnit works by parsing an *.xml file that describes the results of executing unit tests. Mandatory components of such a file should be <*testsuit*> and <*testcase*> tags.

Thanks to the use of xUnit, it is possible to configure the interface for displaying the test results in Jenkins, which is an advantage for the quick processing of the student's work results.

5.4. Build script configuration

In order for the project assembly to be executed serially-parallel on the Jenkins server, a script or pipeline should be attached to the repository to be monitored.

Each pipeline has its own "steps" of assembly according to the business logic of the project, one of the implementation methods is shown in the steps of the scenario, which can be considered in follow script:

```
pipeline {
   agent any
   stages {
      stage('Checkout') {
            steps {
                git url: 'https://git.comsys.kpi.ua/victoria_babenko/repos_test.git', credentialsId:
'jenkins-gitea-test'
            }
        }
        stage('Build') {
            steps {
                bat 'VS_pr_build test_repos.sln /p:Configuration=Release'
            }
        }
        stage('Test') {
            steps {
                bat 'x64/Debug/test_repos.exe --gtest_output="xml:x64/Debug/"'
            }
        }
        stage('Publish Test Results') {
            steps {
                junit 'x64/Debug/test_repos.exe.gta.testdurations'
            }
        }
    }
}
```

Ussually this file is called Jenkinsfile and is added to the root directory of the repository.

The main stages of the scenario are presented below:

– The first stage is responsible for performing a "Checkout" operation in the Jenkinsfile to clone the repository from the specified URL using Git

– The second stage in the project build – "stage" in the Jenkinsfile performs the "Build" operation for the project using the `bat' command (assuming that the job is running on a Windows platform).

– The third stage (stage) in the Jenkinsfile performs the "Test" operation for the project using the bat command. This stage allows developers to run the tests and save the results in XML format for further use in Jenkins, for example to analyse the results, generate reports or integrate with other Jenkins plugins;

– The fourth post stage in the Jenkinsfile uses the gtest/xUnit plugin to publish Google Test test results. In this post step, the command gtest (*testResults: 'test_report.xml'*) is executed, where *test_report.xml* is the path to the Google Test results file.

In order for Jenkins to run a build using a pipeline, a new item should be created.

After performing this operation, user need to name the item and choose the item – pipeline configuration. The next step is setting the item. Next, it is necessary to specify the pipeline source – a pipeline script from SCM (Source Control Management), where SCM is Git. In this case, the repository address and the access token configured in the Git plugin settings in the global settings must be specified.

After that, the branch in the project to execute the build and the path in the repository directory from the Jenkinsfile must be specified. Once these settings are completed, the project can be successfully built with completed tasks on Gitea.

6. Visualization and discussion of results

Demonstration of system operation. This section of the work will consider the visualization of the work of the system, which is built on the architectural concept of checking tasks in a remote repository.

The client-server part starts on port 8080 at the address *http://localhost:8080*. After going to this address, the main Jenkins interface is displayed. After that, after completing all the settings described in the previous section, the project builds from the repository. To do this, developer should click on the green triangle next to the required project or click on the project and get a window like in Figure 4 and click on Build now.

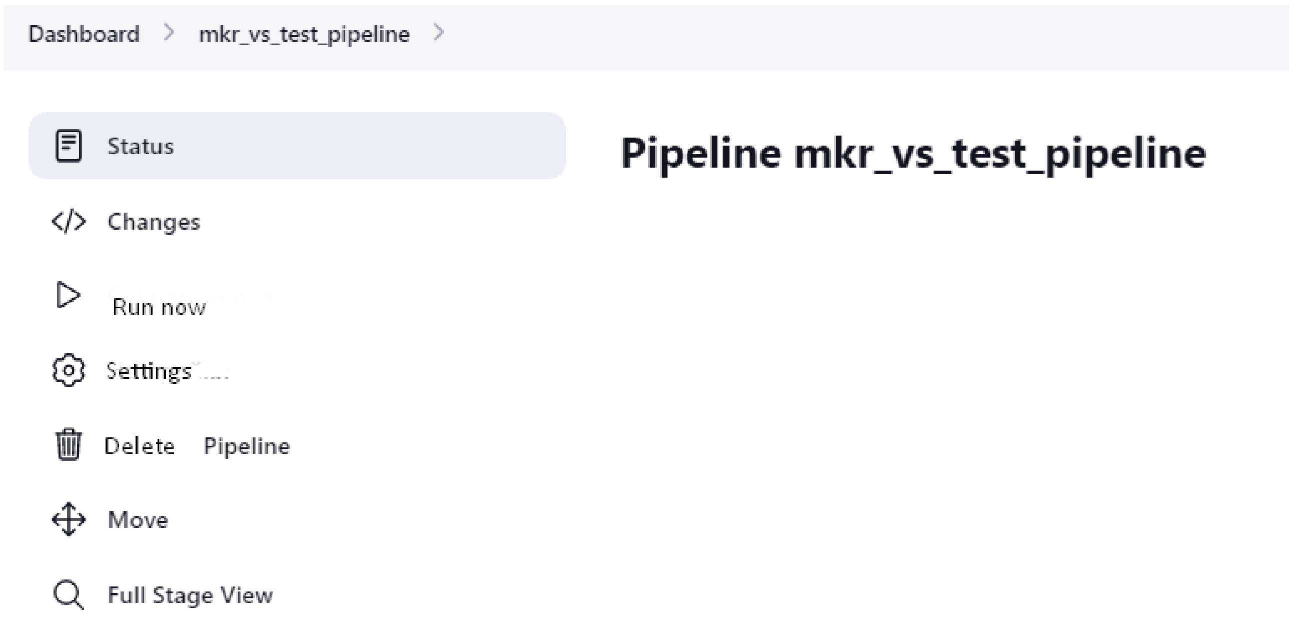
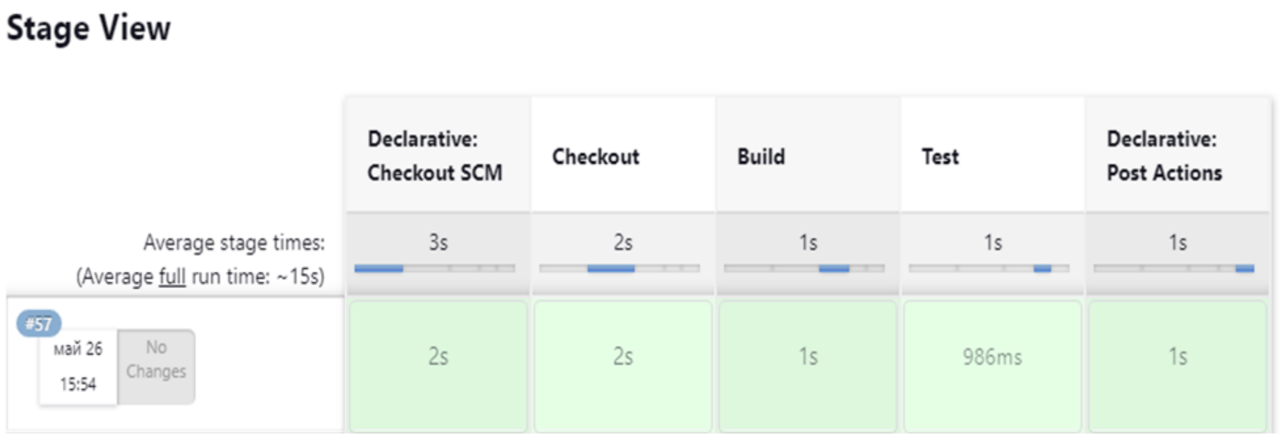

Dashboard > mkr_vs_test_pipeline >
Status
Changes
Run now
Settings
Delete Pipeline
Move
Full Stage View
Pipeline mkr_vs_test_pipeline

Fig. 4. Start project build

After running the build, in a view that is convenient for visual monitoring, each stage of the pipeline starts to run in the Jenkinsfile, as shown in Figure 5.


Stage View

	Declarative: Checkout SCM	Checkout	Build	Test	Declarative: Post Actions
Average stage times: (Average full run time: ~15s)	3s	2s	1s	1s	1s
#57 май 26 15:54 No Changes	2s	2s	1s	986ms	1s

Fig. 5. Running pipeline stages

When hovering over each of the stages, the console log and stage status are displayed to user.

After successful completion of all stages, a graphic diagram of test execution is presented to developer. This diagram shows a graph of the dependence on the number of performed tests and the number of launched project builds. The diagram is shown in Figure 6.

With the help of the developed architecture, it is possible to view changes in the repository without monitoring directly in Gitea. The "Changes" tab should be opened, and a window with changes will appear.

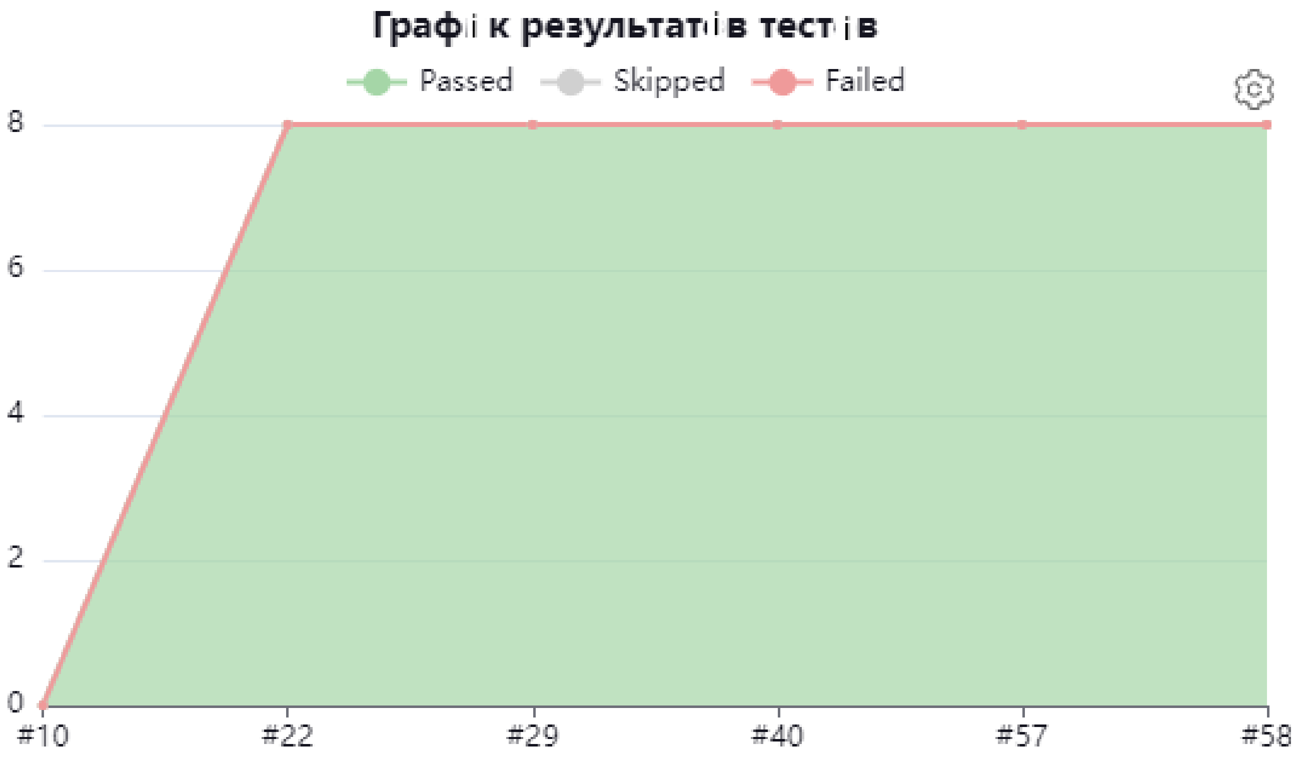
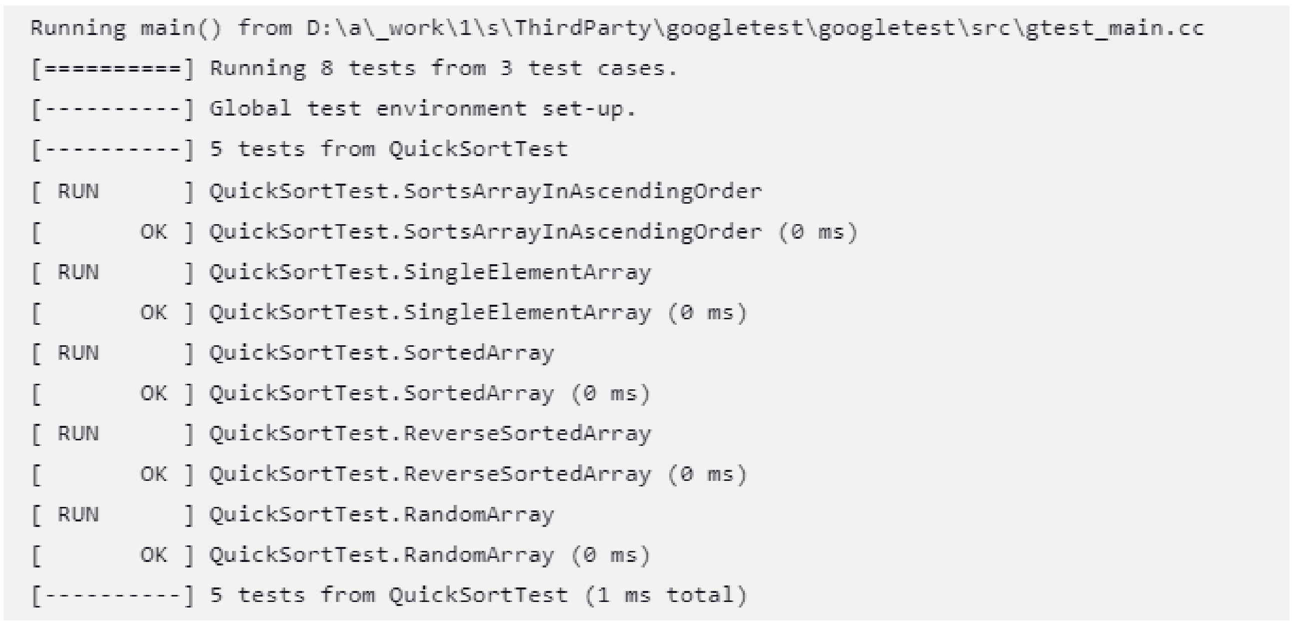


Fig. 6. Diagram of automated testing results

These commits are duplicated on the server through Gitea API interaction.

It is important for any student to have information about exactly where the project "breaks", so for this they can look at the build console output, a fragment of which is shown in Figure 7. In fact, this project is built through a script, so the console output is generated in a sequential script.

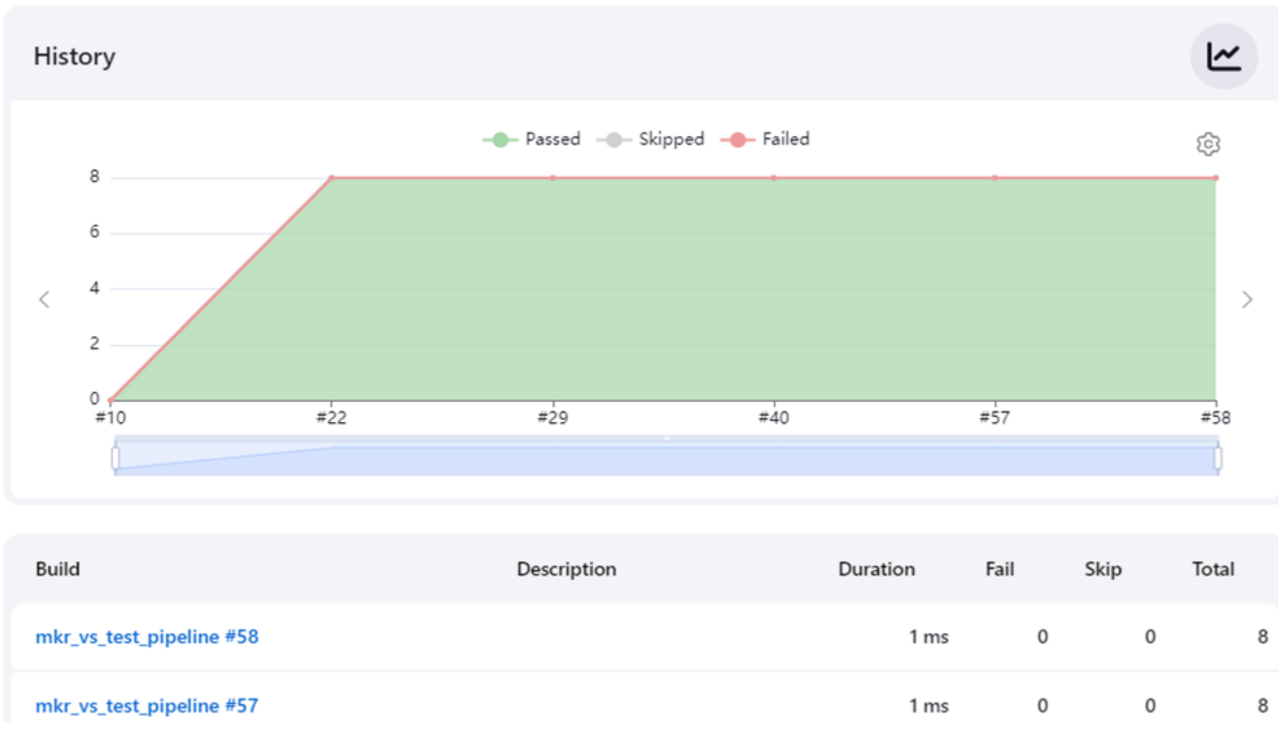


```
Running main() from D:\a\_work\1\s\ThirdParty\googletest\googletest\src\gtest_main.cc
[==========] Running 8 tests from 3 test cases.
[----------] Global test environment set-up.
[----------] 5 tests from QuickSortTest
[ RUN      ] QuickSortTest.SortsArrayInAscendingOrder
[       OK ] QuickSortTest.SortsArrayInAscendingOrder (0 ms)
[ RUN      ] QuickSortTest.SingleElementArray
[       OK ] QuickSortTest.SingleElementArray (0 ms)
[ RUN      ] QuickSortTest.SortedArray
[       OK ] QuickSortTest.SortedArray (0 ms)
[ RUN      ] QuickSortTest.ReverseSortedArray
[       OK ] QuickSortTest.ReverseSortedArray (0 ms)
[ RUN      ] QuickSortTest.RandomArray
[       OK ] QuickSortTest.RandomArray (0 ms)
[----------] 5 tests from QuickSortTest (1 ms total)
```

Fig. 7. Fragment of console output

If debugging or changes are made in the project, this system provides an opportunity to perform the assembly not from the beginning, but from a specifically specified stage. To do this, developer need to go to the Restart from Stage tab and select the appropriate stage from the pipeline and start the assembly.

The Test Results window usually shows the most detailed analysis of the test execution. This window shows a list of all builds, execution time, successful/failed test executions, as in Figure 8.



Build	Description	Duration	Fail	Skip	Total
mkr_vs_test_pipeline #58		1 ms	0	0	8
mkr_vs_test_pipeline #57		1 ms	0	0	8

Fig. 8. Infographic of tests execution

If a teacher or student wants to view the structural execution of a pipeline, open Pipeline Steps. In the specified window, the steps of the build script will be described in detail by the server and it will be clear where exactly the problems occurred if the student or teacher is not able to parse the console output.

By comparing the results of execution through the Jenkins server with the results shown in Figure 2, it can be concluded that the results coincide, indicating the successful execution of the project.

7. Conclusions

In the study, the main goal was to implement an architecture of automated task verification in the Gitea remote repository, which would speed up the code verification process for educational purposes, to develop a stable system with a clear interface for educational purposes.

An overview of existing solutions was carried out, a detailed description of their architectures was carried out, and a comparative description of the most popular CI/CD instruments was carried out. The selection of a set of technologies for the implementation of the architectural concept of automation for tasks written in the C language was analyzed and performed. Namely, the toolkit for each stage of software development was analyzed – work in a local repository, writing and local launch of unit tests, work in a remote repository and automation processes of building, testing and visualizing software results through continuous delivery and integration tools.

A detailed description of the implementation of the system of automated verification of tasks based on Gitea was carried out. The optimal development environment, plug-ins, project structure were selected, the debugging processes of the project assembly in the local repository were adjusted. Moved to remote repository on Gitea. Configured access through continuous delivery tools and integration with remote repositories. Processes of script development for project assembly and debugging, as well as test execution infographics, are shown. This concept makes it possible to speed up verification, reduce the cost of human resources, and provides students with the opportunity to acquire new skills in debugging their solutions to tasks.

A demonstration of all the capabilities of the system, which is built on this architectural concept, was performed. Functionality of the sequence of execution of the script at each stage is checked both through the visual version and in the console output. The performance of each stage of the project development, starting from the repository check, ending with the execution schedule of unit tests on the Jenkins server, is demonstrated. An analytical study of the further development of the project was carried out. The architectural concept can be modified for different development environments and programming languages. The toolkit used in the implementation of the system is available and long-term supported by the developers. System operations and details can be improved using a variety of technologies and tools.

References

[1] P. Shen, X. Ding, W. Ren and C. Yang, "Research on Software Quality Assurance Based on Software Quality Standards and Technology Management," 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 385–390, Busan, Korea (South), 2018, https://doi.org/10.1109/SNPD.2018.8441142.

[2] S. Ibarra and M. Muñoz, "Support tool for software quality assurance in software development," 2018 7th International Conference On Software Process Improvement (CIMPS), pp. 13–19, Guadalajara, Mexico, 2018, https://doi.org/10.1109/CIMPS.2018.8625617.

[3] Y. Zhao, Y. Hu and J. Gong, "Research on International Standardization of Software Quality and Software Testing, "2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), pp. 56–62, Xi'an, China, 2021, https://doi.org/10.1109/ICISFall51598.2021.9627426.

[4] L. Bhaskar, R. B. Natak and R. Ranjith, "Unit Testing for USB Module Using Google Test Framework," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–3, Kharagpur, India, 2020, https://doi.org/10.1109/ICCCNT49239.2020.9225528.

[5] C. Cowell, N. Lotz and C. Timberlake, "Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples," Packt Publishing Ltd, 2023, 328 p.

[6] R. Leszko, "Continuous delivery with Docker and Jenkins: Create secure applications by building complete CI", Packt Publishing Ltd, 2022, 374 p.

[7] L. Qiao "Continuous Delivery 2.0: Business-leading DevOps Essentials", CRC PRESS, 2021, 332 p.

[8] J. Fairbanks, A. Tharigonda and N. U. Eisty, "Analysing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab," 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)," pp. 176–181, Orlando, FL, USA, 2023, https://doi.org/10.1109/SERA57763.2023.10197778.

[9] R. Aiello, "Hands-On DevOps for Architects: Implementing continuous delivery through automation", Packt Publishing, 2018, 345 p.

[10] A. S. Manasa Venigalla and S. Chimalakonda, "DocMine: A Software Documentation-Related Dataset of 950 GitHub Repositories," 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), pp. 407–411, Melbourne, Australia, 2023, https://doi.org/10.1109/MSR59073.2023.00062.

[11] M. Labouardy, "Pipeline as Code: Continuous Delivery with Jenkins, Kubernetes, and Terraform," Manning, 2021, 141 p.

[12] MSBuild, Microsoft Documentation, 2023, [online] Available: https://learn.microsoft.com/ru-ru/visualstudio/msbuild/msbuild?view=vs-2022.

[13] xUnit, Jenkins Documentation, 2022, [online] Available: https://plugins.jenkins.io/xunit/.

[14] Sonya Moisset, "Open Source Software Security Handbook – Best Practices for Securing Your Projects", freecodecamp, 2023, [online] Available: https://www.freecodecamp.org/news/oss-security-bestpractices/.

УДК 004.8: 004.94

ЗАСОБИ CI/CD ІНТЕГРАЦІЇ ДЛЯ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ ТА ПЕРЕВІРКИ ПРОГРАМНОГО КОДУ ДЛЯ НАВЧАЛЬНИХ ЦІЛЕЙ

Вікторія Бабенко
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0009-0008-1762-6671

Вікторія Таранюк
GlobalLogic Sweden AB, Гетеборг, Швеція
https://orcid.org/0000-0001-9044-1499

Валентина Ткаченко
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0000-0002-1080-5932

Ірина Клименко
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0000-0001-5345-8806

Стаття присвячена вивченню та застосуванню в освітніх галузях сучасних засобів безперервної інтеграції та розгортання коду (CI/CD, Continuous Integration/Continuous Deployment). Автоматизація процесів розгортання та перевірки програмного забезпечення є актуальною задачею як для вдосконалення навчального процесу так і для розвитку навичок DevOps у студентів. Значна роль відводиться вивченню головних принципів CI/CD: автоматичного тестування, моніторингу якості коду та інтеграції зі сховищами вихідного коду.

Для автоматизації навчального процесу та навчання студентів використовується популярна платформа CI/CD, така як Jenkins. Цей інструментарій дозволяє створення та розгортання додатків за допомогою технологій Docker, що дозволяє моделювати реальні сценарії. Значний акцент надано питанням масштабованості та адаптивності рішень, які підвищують ефективність використання ресурсів.

Запропоновано методологія втілення CI/CD в освітній курс, включаючи інтеграцію з платформами управління проектами та контролем версій Git, зокрема на прикладі Gitea. Основні етапи якої включають налаштування автоматичної збірки, тестування та розгортання, що дозволяє студентам практикувати принципи безперервної інтеграції та доставки. З точки зору підвищення ефективності навчального процесу, запропонована методологія дозволяє автоматизувати перевірку навчальних завдань. Розглянуто проблеми інтеграції Gitea та Jenkins. Було запропоновано спосіб інтеграції цих інструментів через локально встановлені Jenkins і Gitea з приватними сховищами коду. Розроблено рекомендації щодо організації навчального процесу шляхом впровадження практичних і лабораторних робіт, орієнтованих на реальні сценарії розгортання програмного забезпечення та автоматизації тестування.

Результати дослідження підтверджують ефективне використання засобів CI/CD в освітніх цілях, забезпечуючи формування компетенцій, необхідних для роботи в сучасних IT-командах. Використання CI/CD підвищує рівень обізнаності в питаннях кібербезпеки та оптимізації процесів DevOps.

Key words: CI/CD, DevOps, Jenkins, Git, Gitea, автоматизоване тестування, pipeline.