CONTINUOUS INTEGRATION PIPELINE WITH JENKINS

Alexander[1]; Wella[2*]

Information System Study Program[1, 2]
Universitas Multimedia Nusantara, Tangerang, Indonesia[1, 2]
www.umn.ac.id[1, 2]
alexander2@student.umn.ac.id[1], wella@umn.ac.id[2*]
(*) Corresponding Author

Abstract— *The advancement of technology has led to continuous improvement in application development. As a result, there is a growing demand for application software. Tech companies are constantly working on building and updating their existing applications. The development process is prolonged because to the intricate nature of the build and deployment procedures, ensuring that the application software can be accessed and utilized by individuals across the internet. To address these issues, this research aims to construct and enhance a system capable of automating the entire build and deployment process. By eliminating human intervention, potential errors and downtime that may prevent access to the deployed application can be avoided. The system was developed using the RAD or Rapid Application Development method, with the goal of simplifying and expediting the development process. A DevOps Engineer facilitates the implementation of Continuous Integration in order to reduce the duration of the entire Software Development Life Cycle (SDLC) by utilizing an open-source tool named Jenkins. This ensures that the application development process is efficient and adheres to the designated schedule, allowing all users to benefit from timely delivery.*

Keywords: *continuous integration, devOps engineer, jenkins.*

Abstrak— *Kemajuan teknologi telah menyebabkan perbaikan terus-menerus dalam pengembangan aplikasi. Oleh karena itu, permintaan akan perangkat lunak aplikasi semakin meningkat. Perusahaan teknologi terus berupaya membangun dan memperbarui aplikasi mereka yang sudah ada. Proses pengembangan memakan waktu lama karena rumitnya prosedur pembuatan dan penerapan, sehingga memastikan bahwa perangkat lunak aplikasi dapat diakses dan digunakan oleh individu melalui internet. Untuk mengatasi masalah ini, penelitian ini bertujuan untuk membangun dan meningkatkan sistem yang mampu mengotomatisasi seluruh proses pembangunan dan penerapan. Dengan menghilangkan campur tangan manusia, potensi kesalahan dan downtime yang mungkin menghalangi akses ke aplikasi yang diterapkan dapat dihindari. Sistem dikembangkan dengan menggunakan metode RAD atau Rapid Application Development, dengan tujuan untuk mempermudah dan mempercepat proses pengembangan. Seorang Insinyur DevOps memfasilitasi penerapan Integrasi Berkelanjutan untuk mengurangi durasi seluruh Siklus Hidup Pengembangan Perangkat Lunak (SDLC) dengan memanfaatkan alat sumber terbuka bernama Jenkins. Hal ini memastikan bahwa proses pengembangan aplikasi berjalan efisien dan mematuhi jadwal yang ditentukan, sehingga semua pengguna dapat memperoleh manfaat dari pengiriman yang tepat waktu.*

Kata Kunci*: continuous integration, devOps engineer, jenkins.*

INTRODUCTION

In the world of information technology, the role of the developer is an essential role in the successful implementation of an information system (Akdur et al., 2024). However, it is not uncommon for developers in an organization to face difficulties, especially in making updates or changes (Bai et al., 2023). This is because the application has not been containerized so changes must be made manually in each file that needs to be changed, so changes take a long time (Kumar et al., 2024). Apart from that, errors often occur when the update stage has reached the staging server (Prigent et al., 2024). Errors in this section could be caused by missing dependencies so the quality assurance (QA) party needs to set aside their time to install the applications one by one that are needed to run well, such as the database, bash version and node version if the application runs on Node.JS (Jiang et al., 2021).

PILAR Nusa Mandiri: Journal of Computing and Information System
Vol. 21, No. 1 March 2025 | DOI: 10.33480/pilar.v21i1.6062

One organization experiencing similar problems is PT Emporia Digital Raya.

PT. Emporia Digital Raya is a company founded under the auspices of PT. Anabatic Digital Raya which operates in the field of Finance and Technology, PT. Emporia Digital provides several platforms related to the financial system such as payments for electricity tokens, credit and PDAM called IKIMitra, apart from that, there is also a peer-to-peer lending platform where this platform connects lenders and borrowers. Peer-to-peer lending is a method of borrowing money that is being widely discussed, where this method makes it easier for an individual to borrow from another individual who is ready to borrow (Permatasari et al., 2024).

The problems raised in this research have been resolved by several previous studies. Similar research has been carried out using Docker tools (Giallorenzo et al., 2021). Docker is used as a virtualization platform to containerize virtual machines so that they run more efficiently in terms of virtual machine maintenance costs. Other research has also been carried out based on the problem of the large number of application requests from an organization so that a CI (Continuous Integration) system is implemented (Elazhary et al., 2021). The CI used is the open-source tool Jenkins (Patil et al., 2022). In addition, the CI developed uses Agile methods to increase flexibility during implementation (Noor et al., 2020). However, this Agile method requires professional staff to carry out system development because documentation in this method is not really prioritized (Rahmanti et al., 2022). Another main problem experienced in other research was regarding the length of time required in the build and deployment process, so the idea emerged to automate all processes from build to deployment, using GitlabCI which is provided by Gitlab (Wahyu & Guna Noviantama, 2021). The application design process with a cloud system is developed in such a way that it uses a web server and uses a cloud database, namely Google Firebase (Rahmanti et al., 2022). The design in this research also implements a Cloud Computing system with a server that is no longer maintained personally but uses the cloud provider Google Cloud (Rahmanti et al., 2022).

Based on the previous explanation, the solution offered to every problem faced by developers is to build a CI or Continuous Integration system using Jenkins. Jenkins is used because Jenkins is one of the oldest CI/CD tools released since 2011, it is open source and designed in Java, apart from that, Jenkins has many advantages, including supporting many plugins such as Sonarqube or Sonarscanner which are used to detect vulnerabilities in an environment. Project code that is being built, this process is often referred to as DAST (Dynamic Analysis Security Testing) (Koskela, 2021).

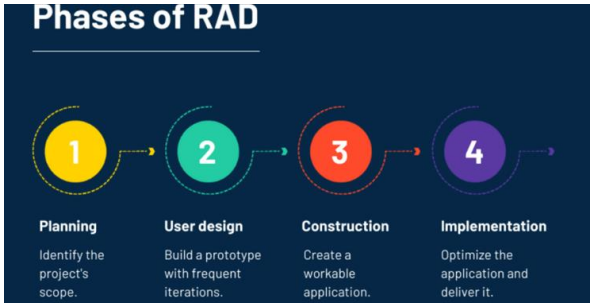
Apart from Jenkins, the tool used for containerization is Docker, chosen because Docker supports many types of repositories such as Gitlab, Bitbucket, Github, et al. With Docker, one server can have several containers containing several applications, so that one server can no longer only handle one application, but rather many applications in each container built with Docker (Poulton, 2023).

This research will focus on the CI (Continuous Integration) system implementation process which will be implemented using an open-source tool called Jenkins, apart from that there will be 2 workers who will act as builders with instance specifications that have been determined using Google Cloud Compute Engine. This research is expected to help developer performance to be more efficient and effective using the environment that has been built with the aim of automating all build and deployment processes without the high risk of manually deploying to the server, besides that it is also hoped that the rollback process on deployments will be If something doesn't go according to plan, it can be done quickly to minimize down time that occurs during servicing.

MATERIALS AND METHODS

Rapid Application Development (RAD)

RAD is a method and framework for creating or developing an application that takes less time, Rapid itself is interpreted as fast or takes less time. RAD or Rapid Application Development has 4 phases, namely Requirements Planning, User Design, Construction and Cutover Phase (UAT) (Breyter, 2022). Figure 1 is a visualization of each phase of RAD (Rapid Application Development).



Source: (Breyter, 2022)

Figure 1. RAD Phases

Docker

Continuous Integration is not far from Docker, these 2 tools are interconnected to perform automation of the build and deployment process,

Docker is one of the tools that functions as a containerization tool which was published in 2013 (Cuadra et al., 2024). Every developer can isolate an application from the host or virtual existing machines so that one server can accommodate many containerized applications without disturbing the host system itself (Yang et al., 2024). The relevance of Docker to VMs is that Docker will use the resources of the VM itself such as CPU, RAM and Memory (Baresi et al., 2024).

The reason for using Docker containers is very simple, the concept of containers is flexibility and several other reasons, namely (Devopedia, 2024):

a. Creating environments for developers such as production, staging and isolated development.
b. Creating building blocks for service oriented or microservices, with Docker making it easier to isolate applications and use Docker to create several microservices with containers.
c. Create CI/CD systems easily.
d. Creating a stand-alone container for testing and research by developers.

Then several components are also explained so that Docker can run on a server, including (Rajyashree et al., 2024):

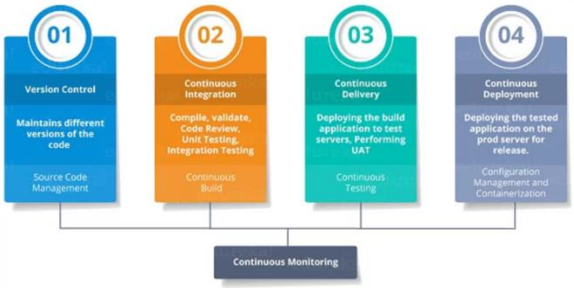
a. Docker engine, which is useful as a client of Docker or can be called a Docker daemon which has the task of executing containers.
b. Images, before Docker can run, images are needed which act as installers which contain various dependencies. Docker images can be built yourself or accessed via Dockerhub.
c. Registry, acts as storage which is divided into 2, namely public which can be accessed via Dockerhub and private via images designed and built personally.
d. Container, after the image has been successfully executed a container will be formed. It is easy to explain that a container is a running image.

Continuous Integration

Continuous Integration (CI) is here to implement the function of the DevOps role where CI is a method that plays a role in continuously integrating the code created, when a developer pushes/commits to one of the repositories in a particular branch, the project will automatically builds and ensures that the code runs well (Robinson et al., 2021). Figure 2 is an illustration of the CI/CD lifecycle.

There are several requirements that will be implemented to ensure the implementation of CI, including (Pratama & Sulistiyo Kusumo, 2021):

a. Using a version repository for source code pushed by developers such as Github, Gitlab, Bitbucket, et al.
b. Automatic testing of code changes,
c. Build process automation,
d. Deploy or release code into pre-production areas such as staging and development.



Source: (Pratama & Sulistiyo Kusumo, 2021)
Figure 2. CI/CD Lifecycle

Jenkins

Jenkins is one of the many tools available on the internet for automating the build and deployment process, but many large companies operating in the world of IT (Information Technology) choose Jenkins for their CI/CD processes because this tool is classified as safe and open-source (Zhao et al., 2024). so not a penny comes out of the company's pocket to use this tool, then Jenkins is one of the tools that is classified as complete in integration and can run on any cloud provider (Patil et al., 2022), even though it is classified as free, aka open source, Jenkins ensures that this tool is very complete. Figure 3 is a comparison of CI/CD tools.

CI/CD Tools Throwdown

Jenkins vs. TeamCity vs. Bamboo

	Jenkins	TeamCity	Bamboo
Open Source?	Yes	No	No
Ease of Use/Setup	3/5	5/5	4/5
Built-In Features	2/5	5/5	4/5
Integrations	1447	338	221
Support	4/5	5/5	5/5
Run on Cloud?	Yes	Yes	Yes
Pricing	Free	From $299	From $888

Powered by Stackify

Source: (Light et al., 2021)
Figure 3. Comparison of Several CI/CD Tools

It is literally written down what is the meaning of Jenkins itself and what is the use of this tool (Light et al., 2021), Jenkins is one of the tools used for the process of implementing Continuous Integration with pipelines or workflows, Jenkins

PILAR Nusa Mandiri: Journal of Computing and Information System
Vol. 21, No. 1 March 2025 | DOI: 10.33480/pilar.v21i1.6062

works with a Jenkinsfile which will then be detected by the machine to run build and deployment processes. The advantages of using Jenkins pipelines include (Moradvandi et al., 2024):

a. Code: pipelines run using code that has been written and checked in source control, giving each team in the organization the freedom to review, edit and iterate the pipeline.
b. Durable: where pipelines can go through unplanned stages such as planned or unplanned restarts from the Jenkins controller.
c. Pausable: the pipeline can be stopped temporarily to wait for input from the developer and requires approval to continue working on the pipeline itself.
d. Versatile: the Jenkins pipeline also supports quite complex workflows such as looping, fork/join and IF/ELSE conditions.
e. Extensible: Jenkins has a lot of features that

RESULTS AND DISCUSSION

The data collection process will be carried out using qualitative methods, which are unstructured research that is flexible and aims to understand, explain, examine, and find certain phenomena based on deductive logic. This research was conducted by conducting focus group discussions (FGD).

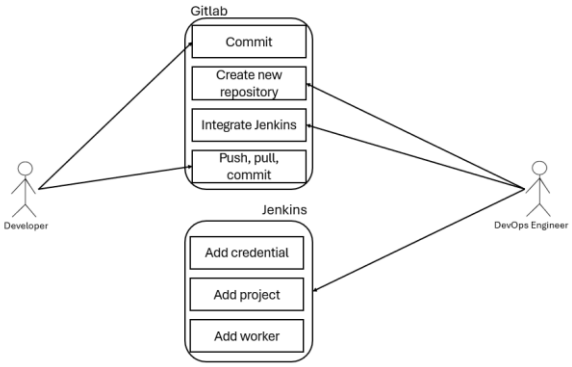
Based on the FGD with 5 participants, each of which came from a different background, namely 2 (two) people as mobile and web app developers, 2 (two) people as DevOps Engineer (infrastructure), and 1 (one) person as QA Tester, the results were found that developers find it difficult and feel that the build and deployment process of an application is quite time-consuming because of the lengthy process because the application that you want to deploy must be communicated to the infrastructure team in charge of doing manual updates, apart from that from the infrastructure side, they also experience the same thing. Which is a long time to deploy, especially if a rollback of applications that have been deployed is needed. While the QA Tester always wants the application to run smoothly in a short time.

Planning

The object studied in this study is a Continuous Integration system using open-source tools, namely Jenkins. To ensure that the output is running correctly, a web or mobile application developer is needed to run the application via Gitlab. The requirements for the system designed are a PC of at least 8GB RAM, 2 (two) Google Cloud Instances, and a Termius SSH agent.

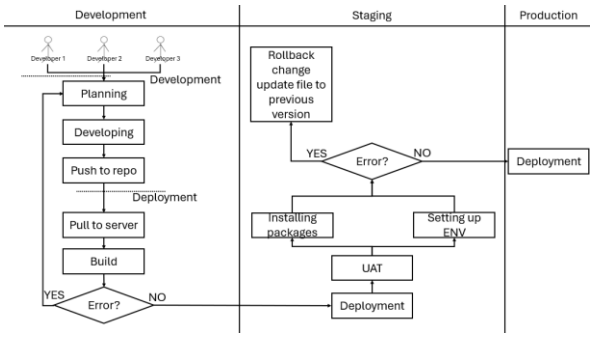
User Design

The designed system can be used by web and mobile application developers by committing to each repo, while the DevOps Engineer has full access to the resources in the Jenkins CI system.



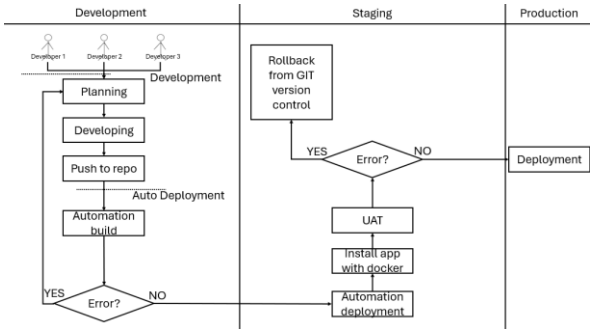
Source: (Research Result, 2024)
Figure 4. Use Case Diagram

Figure 4 is an activity diagram for the manual deployment process, where the process is quite time-consuming because the manual deployment process involves the infrastructure team and software engineers to carry out the deployment, then for the staging stage the QA team must install dependencies one by one for the application to run. In addition, the rollback process must be done manually by replacing files that have been backed up before deployment. The activity diagam for manual deployment can be seen on Figure 5.



Source: (Research Result, 2024)
Figure 5. Manual Deployment Activity Diagram

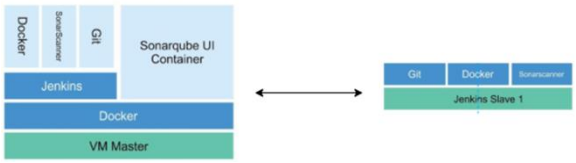
Figure 6 is the activity diagram for automated deployment. When the CI process has been implemented, it can be seen that the differences cut a lot of time because some processes are eliminated due to automation, the build and deploy processes are no longer done manually but Jenkins workers will automatically detect every commit made by the company's developers, in addition to the process rollbacks can also be done automatically with Gitlab, which is a version control platform that provides a history of every commit made by developers.



Source: (Research Result, 2024)
Figure 6. Automation Deployment Activity Diagram

Construction

The first thing that must be done for the CI implementation process using Jenkins is to create several servers or instances as needed, at least to run Jenkins it takes 2 instances where the second instance is a slave or worker so that the master server does not have such a heavy burden. Illustration can be seen on Figure 7 below.



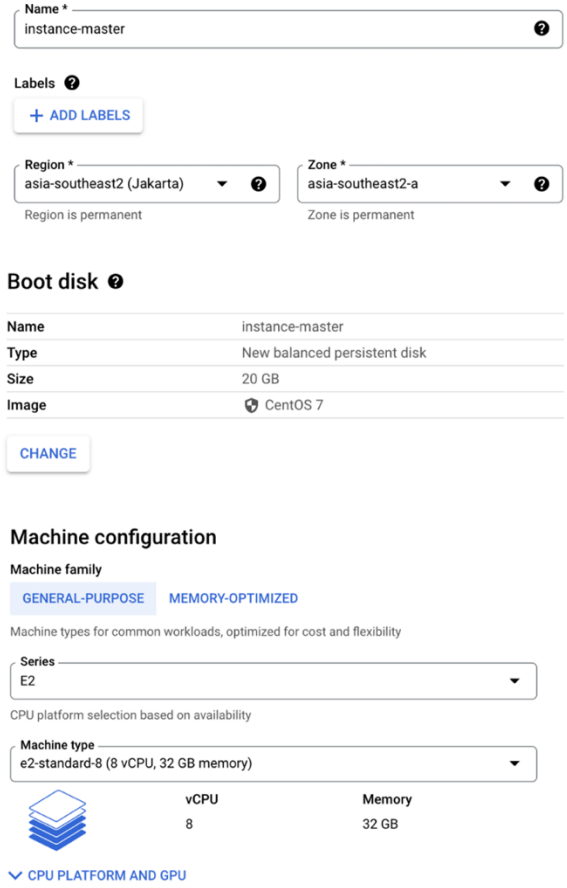
Source: (Research Result, 2024)
Figure 7. Jenkins Structure

The specifications used are instances with the instance-master tag, located in Jakarta zone A, using the CentOS 8vCPU 32GB memory operating system.

Figure 8 is instance details configuration. After provisioning the instance, Docker must be installed on each server because Jenkins is run through Docker to make it easier and doesn't take a long time. Docker will act as an executor or containerization tool on a server, the image of an application running with Docker can be retrieved via the Docker repository site or Docker hub.

Jenkins installation can be done after the Docker engine installation process is complete, the Jenkins version used is the latest release version which can be found on the Docker hub. Jenkins installation will be exposed to port 8686 and forwarded to 8080, so it will be accessible in a web browser with the format Error! Hyperlink reference not valid.>. To ensure that Jenkins is running, Docker provides logs from running containers in the command 'Docker logs <container-name> format.

After that, configuration of Jenkins, Jenkins Slave, Gitlab, integration of Jenkins into Gitlab repository, integration of Gitlab repository into Jenkins, installation of Jenkins plugins, creation of Dockerfile, creation of Jenkinsfile, and commit to staging branch were carried out.



Source: (Research Result, 2024)
Figure 8. Instance Details Configuration

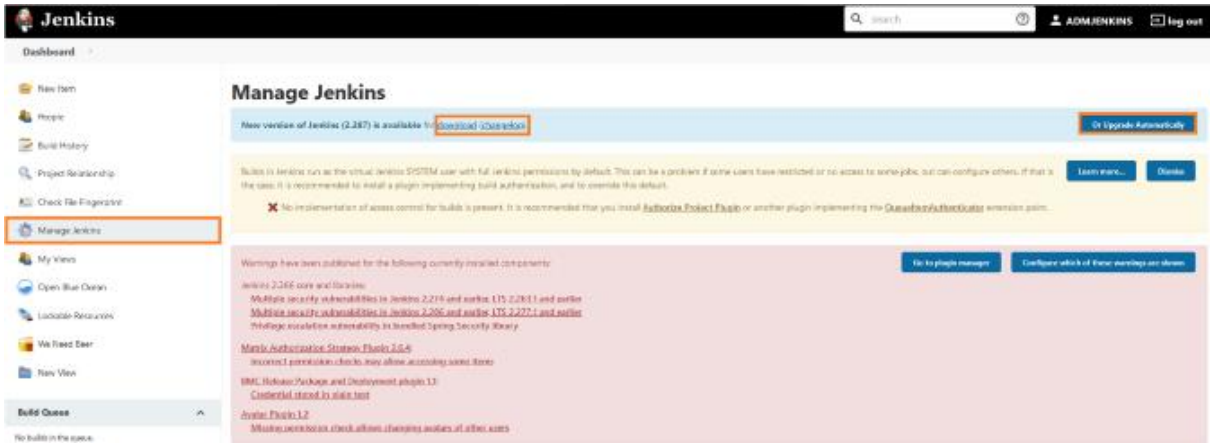
User Design

After Jenkins is integrated by DevOps Engineer and it is feasible to experiment on the system, then a test will be carried out on the system by integrating one of the active repositories, then Jenkins will detect every branch in the repository.

Referring to an interview conducted with a DevOps Engineer from PT. Emporia Digital Raya named Alvian Rizaldi said that the CI system was proven to speed up the deployment and build process, before using CI the deployment process took approximately 30 minutes, Alvian said that the reason for the long deployment and build time was the result of manually changing the update file and one by one, while when using CI, every build and deploy process can be done automatically with the system just by creating a Jenkins script.

After conducting an interview with one of the mobile app and website developers from PT. Emporia Digital Raya, namely Jontathan Atmadja, said that the system from Jenkins CI has proven to

PILAR Nusa Mandiri: Journal of Computing and Information System
Vol. 21, No. 1 March 2025 | DOI: 10.33480/pilar.v21i1.6062



Source: (Research Result, 2024)

Figure 9. Jenkins UI

simplify the design process of an application, both web -based and mobile, one of the frameworks that are often used is Node.JS and also React Native, " Jenkins CI simplifies the development process of an application because The build will be done automatically by Jenkins workers at every commit made by the developers, so bugs will be automatically detected before the application is deployed to an environment, whether it's the Development stage or staging for the UAT stage ".

Jenkins CI is proven to increase efficiency in the build and deployment process, according to Alvian Rizaldi as DevOps Engineer at PT. Emporia Digital Raya explained that the system from Jenkins CI automates every build and deployment process so that it doesn't take a lot of personnel to upgrade an application because the application can go up to the internet automatically without any human intervention in the middle of the process, a DevOps Engineer only needs to create a script which is important for Jenkins workers to run according to the script that was created earlier.

Discussion

Table 1 shows the results before the implementation of the Jenkins System and after the implementation of the CI (Continuous Integration) system from Jenkins. Before the CI system, the application build process was carried out manually, while after the CI system, the build process was carried out automatically by Jenkins Workers. Then it can be concluded that the deployment process before the CI system was carried out manually by the infrastructure team and after the Jenkins CI system, the process was carried out automatically and faster.

Figure 9 is a display of the Jenkins UI for performing maintenance in the form of updates to the Jenkins system, the maintenance update process can be done manually or automatically, the automatic method can be done by going directly to the Manage Jenkins options from the UI but the manual update process can be done by executing into the container and downloading the latest application from Jenkins from Docker Hub.

Table 1. Before and After Jenkins CI Implementation

No	Scenario	Before	After
1	Application build process	Done manually only when the developer wants to test the application that has been developed	Performed every time the code is pushed or uploaded to the repository
2	Deployment process	Done manually by the infrastructure team when receiving a request for an application ready to be deployed to the server, the process takes up to 30 minutes	Occurring automatically, the deployment process is carried out by the Jenkins worker and takes only half the time of the manual deployment process, which is 15 minutes.

Source: (Research Result, 2024)

CONCLUSION

The conclusion of this study is that the build and deployment process is no longer done manually with human intervention, but all processes are automated with the Jenkins CI system that has been built using the Rapid Application Development approach and Jenkins workers, indirectly the risk of errors in the build and deployment process can be reduced because all processes can be tracked through Gitversion control on Gitlab. Jenkins CI has been proven to reduce human resources and time in the build and deployment process because the process has been automated, the infrastructure team no longer bothers to deploy manually and can-do research on other things rather than doing monotonous and time-consuming activities.

ACKNOWLEDGMENT

This research will not be successful without the help of various internal and external parties. Acknowledgments are addressed to Universitas Multimedia Nusantara and to all information system lecturers who have participated in this research.

REFERENCE

Akdur, G., Aydin, M. N., & Akdur, G. (2024). Understanding Virtual Onboarding Dynamics and Developer Turnover Intention in the Era of Pandemic. *Journal of Systems and Software*, 216, 112136. https://doi.org/10.1016/j.jss.2024.112136

Bai, S., Liu, L., Meng, C., & Liu, H. (2023). Automating discussion structure re-organization for GitHub issues. *Expert Systems with Applications*, 225, 120024. https://doi.org/10.1016/j.eswa.2023.120024

Baresi, L., Quattrocchi, G., & Rasi, N. (2024). A qualitative and quantitative analysis of container engines. *Journal of Systems and Software*, 210, 111965. https://doi.org/10.1016/j.jss.2024.111965

Breyter, M. (2022). Agile Estimation and Planning. *Agile Product and Project Management*, 161–185. https://doi.org/10.1007/978-1-4842-8200-7_7

Cuadra, J., Hurtado, E., Sarachaga, I., Estévez, E., Casquero, O., & Armentia, A. (2024). Enabling DevOps for Fog Applications in the Smart Manufacturing domain: A Model-Driven based Platform Engineering approach. *Future Generation Computer Systems*, *157*, 360–375. https://doi.org/10.1016/j.future.2024.03.053

Devopedia. (2024). *Continuous Integration*. Devopedia. 2022. "Continuous Integration." Version 6, February 15. Accessed 2024-06-25. Https://Devopedia.Org/Continuous-Integration. https://devopedia.org/continuous-integration

Elazhary, O., Werner, C., Li, S., Lowlind, D., Ernst, N. A., & Storey, M.-A. (2021). Uncovering the benefits and challenges of continuous integration practices. *IEEE Transactions on Software Engineering*, 48(7), 2570–2583. https://doi.org/10.1109/tse.2021.3064953

Giallorenzo, S., Mauro, J., Poulsen, M. G., & Siroky, F. (2021). Virtualization Costs: Benchmarking Containers and Virtual Machines Against Bare-Metal. *SN Computer Science*, *2*(5). https://doi.org/10.1007/S42979-021-00781-8

Jiang, Z., Zhong, H., & Meng, N. (2021). Investigating and recommending co-changed entities for JavaScript programs. *Journal of Systems and Software*, 180, 111027. https://doi.org/10.1016/j.jss.2021.111027

Koskela, P. (2021). *Automated Security Testing Utilizing Continuous Integration and Continuous Delivery Technologies*. https://www.theseus.fi/bitstream/handle/10024/502952/Opinnaytetyto_Koskela_Pyry.pdf

Kumar, B., Verma, A., & Verma, P. (2024). Optimizing resource allocation using proactive scaling with predictive models and custom resources. *Computers and Electrical Engineering*, *118*, *109419*. https://doi.org/10.1016/j.compeleceng.2024.109419

Light, J., Pfeiffer, P., & Bennett, B. (2021). An Evaluation of Continuous Integration and Delivery Frameworks for Classroom Use. *Proceedings of the 2021 ACM Southeast Conference*, 204–208. https://doi.org/10.1145/3409334.3452085

Moradvandi, A., Abraham, E., Goudjil, A., De Schutter, B., & Lindeboom, R. E. F. (2024). An identification algorithm of switched Box-Jenkins systems in the presence of bounded disturbances: An approach for approximating complex biological wastewater treatment models. *Journal of Water Process Engineering*, *60*. https://doi.org/10.1016/j.jwpe.2024.105202

Noor, S., Koehler, B., Steenson, A., Caballero, J., Ellenberger, D., & Heilman, L. (2020). IoTDoc: A Docker-Container Based Architecture of IoT-Enabled Cloud System. *Big Data, Cloud Computing, and Data Science Engineering*, 51–68. https://doi.org/10.1007/978-3-030-24405-7_4

Patil, K., Kapadnis, S., Waghmare, R., & Thakare, H., & Raut, R. (2022). Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins. *Interantional Journal of Scientific Research in Engineering and Management*, 06(11). https://doi.org/10.55041/ijsrem16948

Permatasari, E., Fatimah, S., Safitri, N., & Wijaya, R. (2024). Problems of Peer-to-Peer Lending (P2PL) in Indonesia from an Islamic Law Perspective. *Jurnal Ilmiah Mizani: Wacana Hukum, Ekonomi Dan Keagamaan*, 11(1), 115. https://doi.org/10.29300/mzn.v11i1.3440

Poulton, N. (2023). *Docker Deep Dive: Zero to Docker in a single book*. https://books.google.com/books?hl=en&lr=&id=tJnMEAAAQBAJ&oi=fnd&pg=PP1&dq=%5B3%5D%09Poulton,+N.,+Docker+deep+dive:+Zero+to+docker+in+a+single+book+.+London:+Leanpub+publishing,+2018.&ots=nzPrXI0mPu&sig=XLJ7l_J6W-z2BTl2yh_rroel_BY

Pratama, M. R., & Sulistiyo Kusumo, D. (2021). Implementation of Continuous Integration and Continuous Delivery (CI/CD) on Automatic Performance Testing. 2021 9th International Conference on Information and Communication Technology (ICoICT), 230–235. https://doi.org/10.1109/icoict52021.2021.9527496

Prigent, C., Costan, A., Antoniu, G., & Cudennec, L. (2024). Enabling federated learning across the computing continuum: Systems, challenges and future directions. *Future Generation Computer Systems*, *160*, 767–783. https://doi.org/10.1016/j.future.2024.06.043

Rahmanti, F. Z., Permata, O. A., Amiroh, K., Daely, P. T., Ittaqullah, A., & Saputro, D. B. (2022). An Improvement Using Global Positioning System (GPS) and Cloud Firestore for Integration of Information System in Surabaya Public Transportation. *EDUTEC : Journal of Education and Technology*, 5(4), 894–909. https://doi.org/10.29062/edu.v5i4.294

Rajyashree, R., Mathi, S., Saravanan, G., & Sakthivel, M. (2024). An Empirical Investigation of Docker Sockets for Privilege Escalation and Defensive Strategies. *Procedia Computer Science*, *233*, 660–669. https://doi.org/10.1016/j.procs.2024.03.255

Robinson, A. C., Drake, R. R., Swan, M. S., Bennett, N. L., Smith, T. M., Hooper, R., & Laity, G. R. (2021). A software environment for effective reliability management for pulsed power design. *Reliability Engineering and System Safety*, 211, 107580. https://doi.org/10.1016/j.ress.2021.107580

Wahyu, A. P., & Guna Noviantama, I. (2021). IMPLEMENTASI CONTIONOUS INTEGRATION DAN CONTINOUS DEPLOYMENT PADA APLIKASI LEARNING MANAGEMENT SYSTEM DI PT. MILLENNIA SOLUSI INFORMATIKA. *Jurnal Ilmiah Teknologi Infomasi Terapan*, 8(1), 183–186. https://doi.org/10.33197/jitter.vol8.iss1.2021.744

Yang, H., Shao, R., Cheng, Y., Chen, Y., Zhou, R., Liu, G., Xie, G., & Zhou, Q. (2024). REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems. *Journal of Systems Architecture*, 151, 103135. https://doi.org/10.1016/j.sysarc.2024.103135

Zhao, X., Clear, T., & Lal, R. (2024). Identifying the primary dimensions of DevSecOps: A multi-vocal literature review. *Journal of Systems and Software*, 214, 112063. https://doi.org/10.1016/j.jss.2024.112063