9th Applied Business and Engineering Conference

IMPLEMENTASI CI/CD DALAM PENGEMBANGAN APLIKASI WEB MENGGUNAKAN DOCKER DAN JENKINS

Andrian Alpery[1)], Muhammad Arif Fadhly Ridha[2)]

[1]Teknik Informatika, Politeknik Caltex Riau, Rumbai, Pekanbaru, 28265

E-mail: andrian17ti@mahasiswa.pcr.ac.id

[2]Teknik Informatika, Politeknik Caltex Riau, Rumbai, Pekanbaru, 28265

E-mail: fadhly@pcr.ac.id

Abstract

In the current software development process, the build, testing, and deployment processes are generally carried out conventionally. The software build and testing process is carried out separately by the development and testing teams. After the build and testing process is complete, then the software is handed over to the operational team for deployment. This takes longer because the process is done manually repeatedly. CI/CD is a concept that is implemented as a process that can help application developers to combine the new code with the main, then make it easier to do code testing at each stage, and end with the deployment process. The purpose of this research is to implement and analyze the quality, time required, and process automation in the web application with the CI/CD method using Docker and Jenkins. Based on the tests carried out on web application development using the CodeIgniter framework in 10 deployments, they are as follows: the average time required by docker and jenkins to carry out the jenkins pipeline deployment process is 1 minute 58 seconds, with a success rate of 90%, 78 bugs, and during testing only once a failure occurred in the deployment process.

Keywords: *Continuous Integration, Continuous Delivery, Continuous Deployment, Jenkins, Docker.*

Abstrak

Dalam proses pengembangan perangkat lunak saat ini yaitu proses *build, testing,* dan *deployment* pada umumnya dilakukan secara konvensional. Proses *build* dan *testing* perangkat lunak dilakukan secara terpisah oleh tim pengembang dan pengujian. Setelah proses *build* dan *testing* selesai, kemudian perangkat lunak diserahkan kepada tim operasional untuk melakukan *deployment*. Hal ini memerlukan waktu lebih lama karena proses tersebut dilakukan dengan cara berulang kali secara manual. CI/CD adalah konsep yang diimplementasikan sebagai sebuah proses yang dapat membantu pengembang aplikasi untuk menggabungkan kode yang baru dengan kode utama, kemudian memudahkan untuk melakukan serangkaian *test* pada setiap *stage* yang ada, dan diakhiri dengan proses *deployment*. Tujuan dari penelitian ini adalah mengimplementasikan dan menganalisa kualitas, waktu yang dibutuhkan, dan proses otomatisasi pada proses pengembangan aplikasi *web* dengan metode CI/CD menggunakan *Docker* dan *Jenkins*. Berdasarkan hasil pengujian yang dilakukan pada pengembangan aplikasi *web* yang menggunakan *framework CodeIgniter* dalam 10 kali *deployment* adalah sebagai berikut: rata-rata waktu yang dibutuhkan *Docker* dan *jenkins* dalam melakukan proses *deployment jenkins pipeline*

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

adalah selama 1 menit 58 detik, dengan tingkat keberhasilan sebesar 90%, dan 78 *bugs*, dan selama melakukan pengujian hanya sekali terjadinya kegagalan dalam proses *deployment.*

Kata Kunci: *Continuous Integration, Continuous Delivery, Continuous Deployment, Jenkins, Docker*.

PENDAHULUAN

Dalam pengembangan perangkat lunak memiliki beberapa tahapan proses yang cukup penting dalam melakukan pengembangan. Tahapan tersebut disebut dengan *System Development Life Cycle* (SDLC). *System Development Life Cycle* (SDLC) merupakan siklus dalam pengembangan perangkat lunak dengan tujuan untuk membuat masalah diselesaikan secara efektif sehingga dapat menghasilkan perangkat lunak yang berkualitas dan sesuai dengan tujuan perangkat lunak tersebut. Biasanya perangkat lunak dikembangkan dengan cara yang konvensional. Proses *build* dan *testing* perangkat lunak dilakukan secara terpisah oleh tim pengembang dan pengujian. Setelah proses build dan testing selesai, kemudian perangkat lunak diserahkan kepada tim operasional untuk melakukan *deployment* atau merilis perangkat lunak kepada klien. Seluruh proses pengembangan perangkat lunak tersebut dimulai dari proses *build*, *testing*, dan *deployment* dilakukan secara berulang kali dengan cara manual. Pengembangan perangkat lunak dengan cara seperti itu memerlukan waktu yang lama.

Beberapa perusahaan, baik besar maupun kecil membutuhkan produksi yang efektif dan efisien. Produksi perangkat lunak yang efektif dan efisien merupakan salah satu kunci sukses pada pengembangan perangkat lunak untuk menjangkau pelanggan. Metode CI/CD digunakan untuk mengatasi hal tersebut. CI/CD atau *Continuous Integration/Continuous Deployment* merupakan jembatan antara tim operasional dan tim pengembang dalam melakukan automasi *build*, *testing*, dan *deployment* aplikasi. Dengan adanya CI/CD, pengembang dapat melakukan proses automasi dan *monitoring* secara berkelanjutan pada pengembangan perangkat lunak. CI/CD dapat mempermudah pengembang dalam merilis maupun memperbarui perangkat lunak secara aman dan cepat secara terorganisir.

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

Pada penelitian proyek akhir ini akan mengimplementasikan proses CI/CD dalam pengembangan aplikasi *web* menggunakan *Docker* dan *Jenkins*. *Docker* merupakan perangkat lunak yang memungkinkan dalam membuat, menguji dan menerapkan aplikasi dengan cepat. Dengan adanya *Docker*, perangkat lunak disimpan dan dikemas ke dalam unit standar yang disebut kontainer yang memiliki seluruh hal yang diperlukan perangkat lunak agar dapat berfungsi. *Jenkins* adalah sebuah automasi *server* yang berbasis *open source* yang ditulis menggunakan bahasa pemrograman *Java*. Dengan dukungan komunitas yang cukup besar dan jumlah plugin yang cukup banyak, *Jenkins* menjadi salah satu *tools* yang cukup populer untuk mengimplementasikan *Continuous Integration* dan *Continuous Deployment* atau biasa yang disebut CI/CD.

Berdasarkan uraian diatas maka akan dibuat suatu sistem CI/CD yang digunakan untuk pengembangan aplikasi web menggunakan *Docker* dan *Jenkins.* Sistem ini dibuat dengan tujuan untuk menganalisa kualitas, waktu yang dibutuhkan, dan proses otomatisasi pada proses pengembangan aplikasi web mulai dari *build, testing,* dan *deploy* dengan metode CI/CD menggunakan *Docker* dan *Jenkins.*

METODE PENELITIAN

Metode penelitian yang digunakan dalam pembuatan proyek akhir ini adalah:

1) Studi Literatur

Kumpulan referensi dalam penelitian ini berasal dari membaca, mempelajari literatur penelitian sejenis yang memiliki permasalahan terkait, menganalisa buku, artikel, jurnal yang berkaitan dengan penelitian, dan literatur mengenai *Ubuntu*, *Version Control System* (VCS), *Git*, *Jenkins*, *Continuous Integration/Continuous Delivery*, Virtualisasi. *Container*, dan *Docker*.

2) Perancangan dan Desain

Perancangan dan desain terdiri dari rancangan infrastruktur jaringan seperti *development* dan *production server*, alur kerja CI/CD menggunakan *Docker* dan *Jenkins*, alur penelitian, dan metode pengujian. Aplikasi *web* yang digunakan pada

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

penelitian ini adalah sebuah *web* yang dikembangkan dengan menggunakan *framework codeigniter*

3) Implementasi

Pada tahap implementasi akan mengimplementasikan proses otomatisasi pada CI/CD dalam pengembangan aplikasi *web* menggunakan *Docker* dan *Jenkins* berdasarkan perancangan dan desain yang telah dibuat.

4) Pengujian

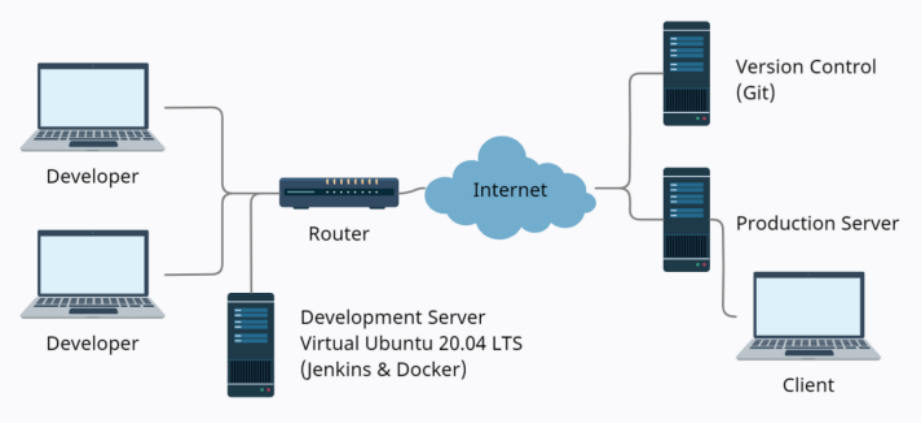
Pada tahap pengujian dilakukan pengujian dengan cara menguji kinerja jenkins berdasarkan kualitas, waktu yang dibutuhkan, dan proses otomatisasi dalam pengembang aplikasi *web* dengan mengimplementasikan CI/CD menggunakan *Docker* dan *Jenkins*.

5) Analisis dan Evaluasi

Setelah tahap pengujian, data terkumpul akan dianalisis kualitas, waktu yang dibutuhkan, dan proses otomatisasi pada proses pengembangan aplikasi *web* mulai dari *build*, *testing*, dan *deploy* dengan metode CI/CD menggunakan *Docker* dan *Jenkins*.

HASIL DAN PEMBAHASAN

Perancangan infrastruktur jaringan yang akan digunakan dalam penelitian ini adalah seperti Gambar 1.



Gambar 1. Infrastruktur jaringan

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

Berdasarkan Gambar 1., *developer* mengakses *development server* yang menggunakan virtual *ubuntu* yang berisi kontainer *jenkins* yang dijalankan melalui *docker*. Rancangan infrastruktur memiliki 3 *server* yaitu *server version control*, *development server,* dan *production server*. perancangan infrastruktur jaringan menggunakan *Git* sebagai *version control*, *Jenkins* sebagai *server development* dan *automation server* untuk melakukan seluruh tugas dalam tahap pengembangan seperti proses *build* dan *testing* secara otomatis sebelum aplikasi dimasukkan ke dalam kontainer menggunakan *docker* dan di-*deploy* ke *production server*.

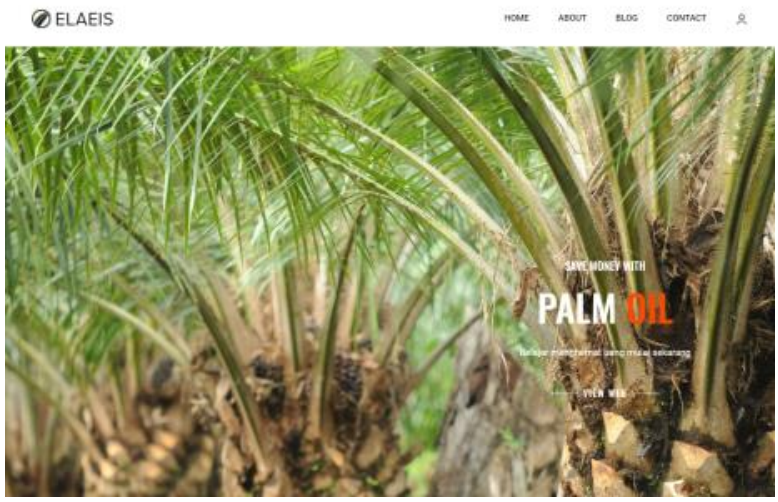
Pengerjaan penelitian ini diawali dengan merancang desain dan infrastruktur yang akan dibangun. Adapun yang dilakukan pada tahap ini adalah menginstal sistem operasi *Ubuntu* 20.04 LTS, dan *docker*.

Setelah membangun infrastruktur, maka dari rancangan tersebut akan diimplementasikan yaitu melakukan instalasi *Git, Jenkins,* dan *Docker*. Setelah instalasi perangkat lunak, akan membuat *job* di *Jenkins* untuk melakukan proses *fetch* dari *git repository* menuju *Jenkins.* Kemudian, melakukan proses *build* dan *testing* aplikasi web sebelum dimasukkan ke dalam kontainer menggunakan *docker* dan di-*deploy* ke *production server.*

Pada penelitian ini menggunakan aplikasi *web* bernama *Elaeis* untuk pengujian CI/CD dengan *docker* dan *jenkins*. *Elaeis* merupakan *website* yang dikembangkan dengan menggunakan *framework codeigniter*. *Web* yang digunakan pada penelitian ini disimpan pada *private repository* dengan menggunakan *Github*. Gambar 2 merupakan halaman utama dari website Elaeis.

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021



Gambar 2. Halaman utama *Elaeis*

Pengujian dilakukan dengan menjalankan proses *deployment* pada saat pengembangan aplikasi *web* dengan menggunakan *jenkins* yang secara otomatis menjalankan proses pengembangan seperti *test, build,* dan *deploy. Deployment* yang dilakukan sampai aplikasi *web* selesai dikembangkan berjumlah 10 kali *deployment.* Beberapa hal yang diuji pada proyek akhir ini berdasarkan waktu yang dibutuhkan dan kualitas dari proses CI/CD yang dimana adalah sebagai berikut: *deployment time, test pass rate,*dan *number of bugs.* Berikut adalah hasil pengujian yang telah dilakukan pada penelitian ini adalah mengimplementasikan metode CI/CD pada pengembangan aplikasi web menggunakan *docker* dan *jenkins* seperti pada Tabel 1.

Tabel 1
Hasil Pengujian CI/CD Menggunakan *Docker* dan *Jenkins*

Job Build Number	Job Build Time	Bugs	Status
1	2 min 21 sec	41	Fail
2	3 min 42 sec	0	Pass
3	3 min 20 sec	1	Pass
4	1 min 29 sec	0	Pass
5	1 min 28 sec	3	Pass
6	1 min 30 sec	0	Pass

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

7	1 min 25 sec	7	Pass
8	1 min 34 sec	10	Pass
9	1 min 33 sec	9	Pass
10	1 min 22 sec	7	Pass

A. Time-based Metrics

Metrik ini mengukur waktu yang dibutuhkan *jenkins* melakukan seluruh proses *pipeline* seperti *testing, build,* dan *deployment* pada pengembangan aplikasi *web*.

- *Deployment Time*

Berdasarkan Tabel 1, rata-rata waktu yang dibutuhkan *jenkins* dalam menyelesaikan proses CI/CD pada aplikasi web adalah selama 118 detik atau 1 menit 58 detik. Tercatat waktu terlama pada proses pengujian adalah pada pengujian kedua dengan waktu 222 detik atau 3 menit 43 detik. Hal yang menyebabkan pengujian kedua menghabiskan waktu yang lebih lama daripada pengujian pertama adalah karena terjadinya kegagalan pada *deployment* pertama. Waktu terkecil yang tercatat pada pengujian adalah pada pengujian ke-10 yaitu selama 82 detik atau 1 menit 22 detik.

Pada saat melakukan pengujian *deployment time*, ditemukan beberapa hal yang dapat mempengaruhi waktu yang dibutuhkan *jenkins* dalam menjalankan proses CI/CD pada aplikasi web adalah sebagai berikut: kecepatan internet yang digunakan *jenkins* karena CI/CD dengan menggunakan *docker* dan *jenkins* terdapat proses *pulling* dan *pushing docker image* dari *docker registry;* jumlah dan ukuran kapasitas *docker image;* dan kode baru yang di-*commit* dan *push* ke *repository*.

B. Quality Metrics

Mengukur metrik kualitas merupakan salah satu aspek yang penting dalam implementasi CI/CD. Adapun pengukuran untuk metrik kualitas yang digunakan adalah sebagai berikut.

- *Test Pass Rate*

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

Berdasarkan Tabel 1, tingkat keberhasilan *jenkins* dalam melakukan proses CI/CD dalam 10 kali deployment adalah sebesar 90%. Hasil ini didapatkan dari jumlah *deployment* yang berhasil yaitu sebanyak 9 kali dan dibagi dengan total *deployment* yang dijalankan yaitu 10 kali. Hal ini membuktikan bahwa *docker* dan *jenkins* dapat menjalankan proses CI/CD dengan baik.

- *Number of Bugs*

Berdasarkan Tabel 1, *jenkins* mencatat jumlah *bugs* yang terdeteksi pada tahap pengembangan aplikasi web dengan total 78 *bugs* dari seluruh pengujian CI/CD. Total *bugs* terbanyak terdapat pada pengujian pertama yaitu dengan total 41 *bugs*. Hal ini yang menyebabkan terjadinya kegagalan *deployment* pada pengujian pertama. Kegagalan tersebut terjadi karena *deployment* aplikasi tidak dapat melewati salah satu tahap dari CI/CD yaitu tes dan analisis kode program. Pada pengujian kedua, keempat, dan keenam tidak ditemukan adanya *bugs.*

SIMPULAN DAN SARAN

Adapun kesimpulan yang diperoleh dari hasil analisis yang didapatkan terhadap data pengujian adalah sebagai berikut: *Jenkins* dan *Docker* dapat digunakan dengan baik pada pengembangan aplikasi *web*. Berdasarkan hasil pengujian yang dilakukan pada pengembangan aplikasi *web* yang menggunakan *framework CodeIgniter* dengan metode CI/CD dalam 10 kali *deployment* adalah sebagai berikut: rata-rata waktu yang dibutuhkan *jenkins* dalam melakukan proses *deployment jenkins pipeline* adalah selama 1 menit 58 detik, tingkat keberhasilan jenkins dalam melakukan proses *deployment* adalah sebesar 90%, dan *jenkins* mendeteksi adanya *bugs* dengan total 78 *bugs* dalam proses *deployment* pada proses CI/CD dengan menggunakan *docker* dan *jenkins.*

Adapun saran yang dapat digunakan untuk pengembangan pada penelitian selanjutnya adalah sebagai berikut: Memanfaatkan layanan berbasis cloud seperti AWS, *Google Cloud* atau *Microsoft Azure* sebagai *development server* yang dapat menjalankan *jenkins* beserta proses CI/CD dengan *bandwidth* yang lebih besar dan

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

kecepatan baca *file* yang lebih cepat dengan tujuan untuk memaksimalkan efisiensi dari metode CI/CD dan membandingkan kinerja *jenkins* dengan tools CI/CD lainnya.

DAFTAR PUSTAKA

Adiputra, F. (2015). Container dan Docker: Teknik Virtualisasi Dalam Pengelolaan Banyak Aplikasi Web. *Jurnal Simantec,* 4(3), pp. 167-176, doi: https://doi.org/10.21107/simantec.v4i3.1384.

Afriandi, A. (2012). Perancangan, Implementasi, dan Analisis Kinerja Virtualisasi Server Menggunakan Proxmox, VMWare ESX, dan Openstack. *Jurnal Teknologi, 5*(2), pp. 182-191.

Arachchi, S A I B & Perera, Indika. (2018). Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. *2018 Moratuwa Engineering Research Conference (MERCon),* Moratuwa, 2018, pp. 156-161, doi: 10.1109/MERCon.2018.8421965.

Docker Inc. (2013). *Docker Overview*. Retrieved February 15, 2021, from Docker: https://docs.docker.com/get-started/overview/index.html

Dwiyatno, S., Rakhmat, E., & Gustiawan, O. (2020). Implementasi Virtualisasi Server Berbasis Docker Container. *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer,* 7(2), pp. 165-175.

Farizy, S. (2019). Implementasi Teknologi Virtualisasi Private Server Menggunakan Hyper-V Pada STIMIK Pranata Indonesia. *Jurnal Teknologi Informasi ESIT*, 14(1), pp. 31-40.

Git community. (n.d.). *Git*. Retrieved February 17, 2021, from Git: https://git-scm.com/

Jenkins. (2011). *Jenkins User Documentation*. Retrieved February 15, 2021, from Jenkins: https://www.jenkins.io/doc/index.html

Karamitsos, Ioannis & Albarhami, Saeed & Apostolopoulos, Charalampos. (2020). Applying DevOps Practices of Continuous Automation for Machine Learning. *Journal of Information,* 11(7), pp. 363-378, doi: 10.3390/info11070363.

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, 5, pp. 3909-3943, doi: 10.1109/ACCESS.2017.2685629.

Trinh, H., & Doan, H.A. (2016). *Implementation of Continuous Integration and Continuous Delivery in Scrum: Case Study: Food 'N Stuff and WebRTC*

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021

Applications [Bachelor Thesis, Lahti University of Applied Sciences]. Theseus. https://www.theseus.fi/bitstream/handle/10024/117725/Trinh_Huy.pdf

Ubuntu. (2021). *The Story of Ubuntu*. Retrieved February 15, 2021, from Ubuntu: https://ubuntu.com/about.html

Wahanani, H. E., Saputra, W. S., & Wahono, B. H. (2019). Perancangan Infrastruktur Server VCS (Version Control System) Dengan Gitlab Berbasis Git. *SCAN-Jurnal Teknologi Informasi dan Komunikasi*, 14(2), pp. 68-73, doi: 10.33005/scan.v14i2.1490.

ISSN: 2339 – 2053

Pekanbaru, 25 Agustus 2021