Jurnal Politeknik Caltex Riau
Terbit Online pada laman https://jurnal.pcr.ac.id/index.php/jkt/
| e- ISSN : 2460-5255 (Online) | p- ISSN : 2443-4159 (Print) |

Implementasi Continuous Delivery dengan Zero – Downtime Rolling Update Menggunakan Ansible

Kiki Harapan Hutapea[1], Muhammad Arif Fadhly Ridha[2]

[1]Politeknik Caltex Riau, Teknik Informatika, email: kiki18ti@mahasiswa.pcr.ac.id
[2]Politeknik Caltex Riau, Teknik Informatika, email: fadhly@pcr.ac.id

Abstrak

Saat ini setiap organisasi membutuhkan aplikasi untuk memberikan layanan pada pelanggan mereka. Faktanya, 62% organisasi mengatakan aplikasi itu penting untuk bisnis mereka, dan 36% lebih lanjut mengatakan aplikasi memberikan keunggulan kompetitif. Hal ini membuat perusahaan dituntut untuk memberikan inovasi secara cepat demi memberikan kepuasan dan kenyamanan bagi pelanggannya. Untuk merespons tuntutan tersebut, maka organisasi perlu melakukan pengiriman pembaruan aplikasi lebih sering. Dalam proses pengiriman tradisional, setiap proses pengiriman dimulai dengan persyaratan yang ditentukan oleh pelanggan dan berakhir pada produksi. Kelemahan pada pengiriman tradisional adalah lambannya proses pengiriman, dimana proses pengiriman dilakukan secara manual dan berbasis langkah berpotensi menyebabkan titik kegagalan serta kesalahan manusia yang berdampak pada penundaan atau penghentian total sistem. Teknik Countinuous Delivery hadir untuk membantu organisasi mempercepat proses pengiriman aplikasi mereka ke pelanggan. Salah satu perangkat lunak yang dapat digunakan untuk membangun Continuous Delivery dengan zero-downtime adalah Ansible. Berdasarkan hasil pengujian, Ansible berhasil menjaga ketersediaan layanan dengan persentase uptime sebesar 100%. Serta mampu mempercepat waktu deployment sebesar 48%. Dari pengujian beban didapatkan bahwa 1 buah server mampu menangani beban sebesar 2000 user per 5 menit dengan persentase keberhasilan sebesar 99%.
Kata Kunci: *Continuous Delivery, zero-downtime, Ansible.*

Abstract

Today every organization relies on application to provide services to their customers. In fact, 62% of organizations say application are essential for their business, and a further 36% say application provide a competitive advantage. This makes companies are required to provide innovation quickly in order to give satisfaction and convenience for their customers. To respond these demands, organizations need to deliver application updates more frequently. In the traditional deployment process, each deployment starts

Dokumen diterima pada 22 Juli, 2022
Dipublikasikan pada 16 Desember, 2022

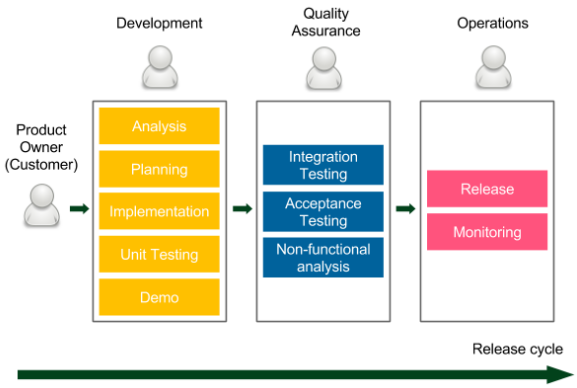
with the requirements spesification and ends at production. The weakness of traditional deployment is the slow delivery process, where its done manually and on a step-by-step basis, which can cause points of failure and human errors that resulting in delays or total system shutdowns. Continuous Delivery help organizations speed up the process of delivering applications to customers. One of the software that can build Continuous Delivery with zero-downtime is Ansible. Based on the test, Ansible managed to maintain the service availability with 100% up time rate and able to speed up deployment time by 48%. From load testing, it was found that 1 server was able to handle a load of 2000 users per 5 minutes with 99% success rate.

Keywords: *Continuous Delivery, zero-downtime, Ansible*

1. Pendahuluan

Saat ini setiap organisasi membutuhkan aplikasi untuk memberikan layanan pada pelanggan mereka. Faktanya, 62% organisasi mengatakan aplikasi itu penting untuk bisnis mereka, dan 36% lebih lanjut mengatakan aplikasi memberikan keunggulan kompetitif [1]. Dengan berkembangnya teknologi dimana memberikan para pelanggan akses ke internet dengan mudah, membuat mereka menuntut lebih banyak perubahan dengan cepat serta akses layanan yang dapat diandalkan setiap saat, hal ini membuat perusahaan harus memberikan inovasi secara cepat demi memberikan kepuasan dan kenyamanan bagi pelanggannya. Oleh karena itu pengembangan aplikasi yang cepat dan dapat diandalkan sangat penting untuk sukses di dunia digital.

Untuk merespons tuntutan pelanggan terhadap inovasi dan kebutuhan layanan yang dapat diandalkan setiap saat, maka organisasi akan melakukan pengiriman pembaruan aplikasi lebih sering tanpa mengurangi ketersediaan dari layanan tersebut. Dalam proses pengiriman tradisional, setiap proses pengiriman dimulai dengan persyaratan yang ditentukan oleh pelanggan dan berakhir pada produksi. Proses tradisional tersebut dijelaskan pada diagram siklus rilis berikut:



Gambar 1 Diagram Siklus Rilis

Gambar I.1 menjelaskan alur siklus rilis dimulai dengan persyaratan yang gmaildiberikan oleh pemilik produk (pelanggan / pemangku kepentingan). Kemudian ada tiga fase yang harus dilewati produk selama proses pengembangan, setiap fase tersebut dikerjakan oleh tim yang berbeda :

- *Development*: Di sini, tim Pengembang mengerjakan produk. Biasanya mereka menggunakan teknik *Agile* untuk meningkatkan kecepatan pengembangan dan komunikasi dengan klien. Sesi demo dilakukan untuk mendapatkan umpan balik cepat dari pelanggan. Setelah implementasi selesai, kode diteruskan ke tim *QA*.
- *Quality Assurance*: Fase ini biasanya disebut pengujian penerimaan pengguna dan memerlukan pembekuan kode pada basis kode *trunk*, sehingga tidak ada pengembangan baru yang akan merusak tes. Tim *QA* akan melakukan serangkaian pengujian integrasi, pengujian penerimaan, dan pengujian non-fungsional. Setiap bug yang terdeteksi akan dikembalikan ke tim Pengembang. Setelah proses pengujian penerimaan pelanggan selesai, tim *QA* menyetujui fitur yang direncanakan untuk rilis berikutnya.
- *Operations*: Fase terakhir adalah menyerahkan kode ke tim Operasi, sehingga mereka dapat melakukan rilis dan memantau produksi. Apabila ada kesalahan, mereka akan menghubungi tim Pengembang untuk membantu sistem produksi.

Kelemahan yang paling signifikan dari proses pengiriman tradisional tersebut adalah lambannya proses pengiriman, hal ini dapat mengakibatkan penundaan antara penemuan dan penyelesaian *bug* perangkat lunak, serta menjadi penghalang pada respon terhadap permintaan pelanggan dan pasar. Kurangnya otomatisasi juga menjadi masalah, dimana proses pengiriman secara manual dan berbasis langkah beresiko mengarah ke proses yang lebih memakan waktu dan kompleks. Hal ini berpotensi menyebabkan titik kegagalan dan kesalahan manusia yang berdampak pada penundaan atau bahkan penghentian total sistem sehingga menyebabkan hambatan operasional dan biaya.

Dalam penelitian yang dilakukan oleh A. Alpery and M. A. F. Ridha dengan judul "Implementasi CI/CD Dalam Pengembangan Aplikasi Web Menggunakan Docker dan Jenkins," didapatkan waktu rata – rata yang dibutuhkan Jenkins untuk menjalakan proses *deployment* dalam impelmentasi CI/CD adalah 1menit 58 detik, dengan keberhasilan sebesar 90%, dan selamat testing hanya tercatat satu kali kegagalan dalam proses pengiriman aplikasi ke produksi[2]. Penelitian dengan judul "Penerapan CI/CD dalam Pengembangan Aplikasi Web Menggunakan Docker dan Gitlab" yang dilakukan oleh R. J. Dwiputra, menunjukan bahwa CI/CD mempercepat proses pengirimaan palikasi ke produksi dengan tingkat keberhasilan 100%[3]. Dari kedua penelitiain tersebut dapat diambil kesimpulan bahwa implementasi CI/CD ekefetif dalam mengatasi masalah lambanya proses pengiriman aplikasi ke produksi serta mampu meningkatkan tingkat keberhasilan proses pengiriman aplikasi ke produksi.

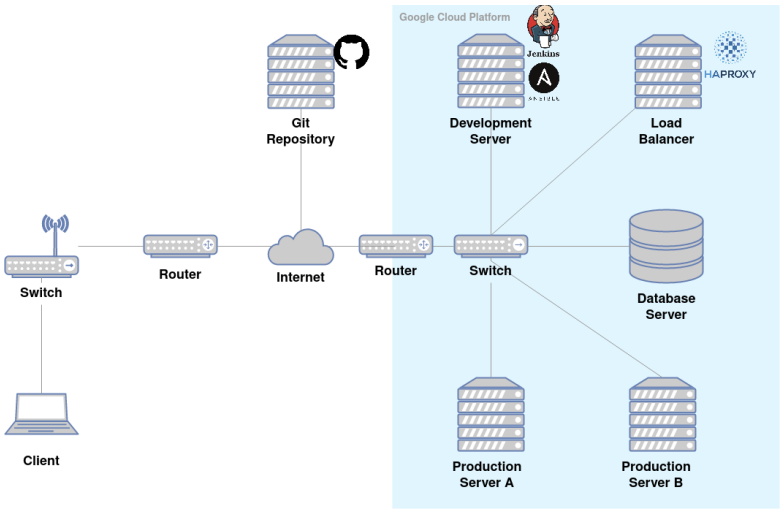
Dalam penelitian I. P. Hariyadi, dengan judul "*Implementation Of Configuration Management Virtual Private Server Using Ansible*," yang berfokus pada otomatisasi manajemen konfigurasi *Virtual Private Server* (*VPS*) menggunakan Asible untuk mendukung kegiatan praktikum mata kuliah manajemen jaringan. Hasil dari penelitian menunjukan penggunaan ansible meningkatkan waktu pembuatan VPS lebih cepat dua kali lipat. Selain itu penggunaan playbook berhasil mengotomatiskan menjalankan dan menghentikan container secara dinamis sehingga menjaga ketersediaan layanan *PVE Cluster[4]*. Penelitina dengan judul "*Implementation Of DevOps Method For Automation Of Server Management Using Ansible*" yang dilakukan oleh A. Khumaidi, menunjukkan

kemampuan ansible dalam mengkonfigurasi dan mengelola pengaturan akun pengguna pada seluruh server secara otomatis[5]. Dari kedua penelitian tersebut didapatkan bahwa Ansibel dapat mengelola konfigurasi pada lebih dari 1 server serta menjanga ketersediaan layanan secara otomatis. Hal ini menjawab rentannya kesalahan manusia dalam konfigurasi layanan dalam proses manual yang berbasis langkah dimana dapat mengakibatkan kegagalan atau penghentian total layanan.

Berdasarkan uraian di atas, kemampuan Ansible dalam mengotomasi pengelolaan beberapa server di waktu yang bersamaan dapat meminimalisir kesalahan manusia serta menjaga ketersediaan layanan, apablia dikombinasikan dengan CI/CD maka Ansbile dapat memberikan kecepatan pengiriman sekaligus menjaga ketersediaan layanan di sisi pengguna selama proses pengiriman aplikasi ke produksi berlangsung. Penelitian ini akan membahas impementasi Ansibel untuk menjalankan proses pengiriman aplikasi ke produksi secara otomatis menggunakan konsep CI/CD, serta menjaga ketersediaan layanan selama proses pengiriman berlangsung dengan menggunakan pendekantan atau metode penerapan *Rolling Update*. *Rolling Update sendiri* merupakan sebuah strategi penerapan yang melibatkan peluncuran versi baru dengan mengganti *instance* lama secara bertahap[6].

2. Metode Penelitian

Penelitian ini akan menggunakan 5 buah server, dengan rincian 2 server bertindak sebagai *production server* dimana applikasi *production* akan di-*deploy*, 1 server sebagai *development server* disini akan dipasangkan Jenkins dan Ansible untuk membangung CI/CD, 1 server sebagai *load balancer* yang berperan sebagai penyeimbang beban dan menjaga ketersediaan layanan aplikasi ke pengguna, terakhir 1 server sebagai database yang berperan menyimpan data applikasi. Adapun perancangan topologi yang dilakukan pada penelitian ini adalah sebagai berikut:



Gambar 2 Rancangan Topologi Jaringan

Penelitian ini akan menggunakan metode penerapan *Rolling Update*, dimana pada metode ini penerapan pembaruan akan dilakuka secara bertahap satu demi satu hingga setiap server production menjalankan versi terbaru. Pendekatan ini akan mengalihkan lalulitas ke versi terbaru hanya ketika instances production server dengan versi terbaru telah siap menangani permintaan sehingga memunkinan penerapan dilakukan tanpa adanya *downtime* atau mengganggu kinerja layanan yang sedang berjalan.

Adapun metode pengujian yang dilakukan dalam penelitian ini adalah sebagai berikut:

a) Pengujian Waktu Aktif (*Uptime Test*)
   Pengujian waktu aktif dilakukan untuk mendeteksi kegagalan aplikasi yang menyebabkan penundaan dan waktu henti layanan selama proses *deployment* berlangsung. Pengujian ini dilakukan sebanyak 10 kali dan dilakukan dengan memantau metrik waktu aktif (*uptime*) layanan saat proses pengiriman kode program ke *production server* sedang berlangsung dengan menggunakan alat atau layanan Google Cloud Monitoring dari Google Cloud Platform.
b) Pengujian Waktu Deployment (*Time Based Metric*)
c) Pengujian ini dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan selama proses deployment. Pengujian ini akan dilakukan sebanyak 10 kali dan dilakukan dengan menghitung waktu proses *deploy* dari awal sampai berhasil. Waktu akan dicatat dan dibandingkan antara menggunakan *CD* Ansible dan tanpa *CD* Ansible atau tradisional.
d) Pengujian Beban (*Load Testing*)
   Pengujian beban bertujuan untuk mengetahui seberapa besar jumlah permintaan yang dapat ditangani oleh 1 *production server* ketika proses penerapan aplikasi berlangsung. Pengujian dilakukan dengan menggunakan perangkat lunak JMeter untuk melakukan *HTTP request* ke *production server*. Jmeter akan memberikan beban ke *production server* saat proses penerapan aplikasi berlangsung. Data yang dicatat adalah rata – rata *latency* saat mengakses *production server.*

3. Hasil dan Pembahasan

3.1 Pengujian dan Analisa

Untuk mengetahui kelayanan implementasi *continuous delivery* dengan *zero - downtime rolling update* menggunakan ansible dilakukan pengujian waktu aktif (*Uptime Test*), pengunjian waktu deployment (*Time Based Metric*), dan pengujian beban (*Load Testing*).

3.1.1 Pengujian Waktu Aktif (*Uptime Test*)

Pengujian ini bertujuan untuk menguji kemampuan Ansible menjaga ketersediaan layanan selama proses deployment berlangsung. Pengujian dilakukan dengan melakukan skenario deployment sebanyak 10 kali.

Tabel 1 Data olahan pengujian Waktu Aktif

Jumlah Hambatan Minimal	0
Jumlah Hambatan Maskimal	0
Rata – Rata Jumlah Hambatan	0

Tabel 2 Data olahan pengujian Waktu Aktif

Durasi Hambatan Minimal	0 detik
DurasiHambatan Maskimal	0 detik
Rata – Rata Durasi Hambatan	0 detik

Berdasarkan tabel 3.1 dan table 3.2, rata – rata hambatan yang terjadi selama proses *deployment* adalah sebanyak 0 kali dengan rata – rata durasi hambatan sebesar 0 detik, hal ini terjadi karena proses *deployment* dilakukan menggunakan Ansible untuk melakukan *deployment* dengan metode *zero – downtime rolling update*, dimana server akan diupdate satu – persatu secara bergantian sehingga selama proses *deployment* berlangsung terdapat minimal 1 buah server yang aktif unutk menangani permintaan klien. Dengan demikian ketersediaan layanan aplikasi web dapat aktif dengan persentase 100% selama proses *deployment* berlangsung.

3.1.2 Pengujian Waktu (*Time Based Metric*)

Pada pengujian ini bertujuan mengukur waktu yang dibutuhkan untuk menerapkan update terbaru aplikasi ke setiap *production server*. Pengujian ini dilakukan dengan cara melakukan *deployment* 10 fitur app dan membandingkan waktu *deployment* antara proses *deployment* tradisional dan proses *deployment Continuous Delivery* menggunakan Ansible.

Tabel 3: Perbandingan data olahan metode *deployment* tradisional dan *Continuous Delivery* menggunakan Ansible

Parameter	Tradisional	*CD* Ansible
Waktu Minimal	100 detik	83 detik
Waktu Maksimal	456 detik	200 detik
Rata – Rata Waktu	266.4 detik	136.3 detik
Tingkat Kesuksesan	100%	100%

Berdasarkan tabel 3.3 didapatkan waktu yang dibutuhkan untuk melakukan *deployment* pada metode tradisional memakan waktu yang lebih lama dimana metode tradisional memakan waktu rata – rata 266.4 detik dengan waktu maksimal 456 detik dan waktu minimal 100 detik. Sedangkan waktu rata – rata yang dibutuhkan oleh *CD* Ansible adalah 136.3 detik dengan waktu maksimal 200 detik dan minimal 83 detik.

3.1.3 Pengujian Beban (*Load Testing*)

Pengujian beban bertujuan untuk mengetahui beban yang dapat ditangani oleh sebuah server selama proses *deployment* berlangsung. Pengujian ini dilakukan dengan memberi beban ketika server pada 1 buah *production server*. Pengujian ini akan melakukan *HTTP Request* ke *production server* yang aktif menggunakan perangkat lunak Jmeter Pembebanan dilakukan dengan skenario 500 *user*, 1000 *user*, dan 2000 *user* selama 5 menit sebanyak 10 kali pengulangan.

Tabel 4: Data olahan pengujian beban

Skenario Pengujian	Rata – Rata Error %	Rata – Rata Throughput	Rata – Rata Latency
500	0.00%	1.64	419.81
1000	0.00%	3.31	426.79
2000	0.01%	6.48	407.48

Berdasarkan hasil pengujian pada table 3.4 didapatkan bahwa sebuah server dengan spesifikasi 1 VCPU dan 3.75GB RAM dapat menangani 2000 *user* dalam kurun waktu 5 menit dengan *throughput* sebesar 6.48 transaksi perdetik dan rata – rata *latency* sebesar 407.48 ms, meski dalam pengujian server sempat mengalami overload namun pesentase *error* yang dihasilkan masih terbilang kecil sebesar 0.01%. Perbandingan rata – rata *latency* dari ketiga skenario tersebut tidak terlalu jauh dengan rata – rata *latency* terendah berada di 407.48 ms dan rata – rata *latency* tertinggi berada di 426.79 ms.

4. Kesimpulan

Dari hasil penerapan *Continuous Delivery* dengan *Zero – Downtime Rolling Update* menggunakan Ansible, maka didapatkan kesimpulan sebagai berikut:

1. Penerapan Ansible sebagai *Continuous Delivery* berhasil menjaga ketersediaan layanan aplikasi selama proses *deployment* terjadi dengan tingkat kesuksesan 100% yang berarti tidak terjadi masalah yang mengakibatkan layanan tidak dapat diakses. Hal ini dimungkinkan karena Ansible memastikan selama deployment berlangsung minimal terdapat 1 buah server siap menangani request.
2. Penerapan Ansible sebagai *Continuous Deliver* mampu mempercepat proses *deployment* sebuah aplikasi. Hal ini terbukti dari hasil pengujian waktu *deployment* yang dilakukan dimana *deployment* menggunakan *CD* Ansible lebih cepat 48.8% dimana Ansible membutuhkan waktu rata - rata 136.3 detik sedangkat pada deployment tradisional membutuhkan waktu rata – rata 266.4 detik. Hal ini disebabkan karena Ansible menjalan proses *deployment* yang kompleks secara otomatis sehingga meminimalisir kesalahan manusia yang dapat terjadi dalam proses *deployment* tradisional.
3. Dari hasil pengujian beban pada 1 buah server dengan spesifikasi 1 VCPU dan 3.75 GB RAM didapatkan hasil bahwa server tersebut mampu menangani 2000 *user* dalam kurun waktu 5 menit dengan cukup baik, dimana didapatkan persentase keberhasilan sebesar 99.99%. Dengan kata lain selama proses *deployment* dalam penerapan *Continuous Delivery* dengan *Zero – Dowtime Rolling Update* menggunakan Ansible, server masih mampu menagani 6.48 transaksi perdetik.

Daftar Pustaka

[1] Red Hat Inc., “Streamline CI/CD pipelines with Red Hat Ansible Automation Platform,” 2020. https://www.redhat.com/en/resources/ansible-continuous-integration-delivery-whitepaper (accessed Oct. 23, 2021).

[2] A. Alpery and M. A. F. Ridha, “Implementasi CI/CD Dalam Pengembangan Aplikasi Web Menggunakan Docker dan Jenkins,” *ABEC Indonesia*, vol. 9, pp. 287–296, Aug. 2021.

[3] R. J. Dwiputra;, *Penerapan CI/CD dalam Pengembangan Aplikasi Web Menggunakan Docker dan Gitlab*. Pustaka Politeknik Caltex Riau, 2021. Accessed: Dec. 02, 2021. [Online]. Available: //opac.lib.pcr.ac.id/index.php?p=show_detail&id=13767&keywords=GitLab

[4] I. P. Hariyadi, “Implementation Of Configuration Management Virtual Private Server Using Ansible,” *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 19, pp. 347–357, May 2020, doi: 10.30812/matrik.v19i2.724.

[5] A. Khumaidi, “Implementation of DevOps Method for Automation of Server Management Using Ansible,” *Jurnal Transformatika*, vol. 18, no. 2, Art. no. 2, Jan. 2021, doi: 10.26623/transformatika.v18i2.2447.

[6] N. Sabharwal and P. Pandey, *Pro Google Cloud Automation: With Google Cloud Deployment Manager, Spinnaker, Tekton, and Jenkins*. Apress, 2020.