# Evaluasi Performa CI/CD Menggunakan Cloud Build Pada Aplikasi Web

**Vincentius Agung Prabandaru[1], Andi Wahju Rahardjo Emanuel[2], Findra Kartika Sari Dewi[3]**
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta
Jl. Babarsari No. 43, Sleman 55281, Daerah Istimewa Yogyyakarta, Indonesia Email:
[1]200710683@students.uajy.ac.id, [2]andi.emanuel@uajy.ac.id, [3]findra.dewi@uajy.ac.id

***Abstract.*** *In the application development process, the deployment phase is a crucial step to ensure the application reaches its users. Frequent application development and repeated processes, even for minor changes, can be cumbersome. Automating this process can positively impact time management, allowing resources to be allocated to other tasks. Automating the deployment phase of application development leverages the implementation of continuous integration/continuous deployment (CI/CD). This study uses a web application built with React, JavaScript, and MySQL. The services utilized include GCP Cloud Build for integration, building, testing, and deployment, and Compute Engine as the virtual machine (VM) running Ubuntu as the operating system. The implementation of CI/CD in this study demonstrates significant differences in efficiency compared to manual deployment in terms of time and resources. Regarding costs, analysis shows that saving time and resources through CI/CD results in lower operational expenses. Parallel use of CI/CD facilitates team collaboration in resolving code conflicts but imposes higher demands on resources and time.*
***Keywords:*** *CI/CD, cloud build, compute engine, GCP*

***Abstrak.*** *Dalam proses pengembangan aplikasi, tahapan deployment adalah tahapan yang penting dilakukan agar aplikasi sampai pada pengguna. Pengembangan aplikasi yang sering dan harus dilakukan berulang kali, meskipun perubahannya minor akan merepotkan. Proses ini jika diubah menjadi otomatis akan memberikan dampak positif terhadap waktu, sehingga bisa dialokasikan untuk hal lainnya. Pengembangan aplikasi tahapan deployment menjadi otomatis memanfaatkan implementasi continuous integration/continuous deployment. Penelitian menggunakan aplikasi web menerapkan React, Javascript dan MySQL. Layanan yang digunakan adalah GCP Cloud Build sebagai integration, build, testing dan deployment. Compute Engine sebagai VM dengan ubuntu sebagai operating system. Hasil penelitian implementasi CI/CD bahwa efisiensi CI/CD dibandingkan manual menunjukkan perbedaan signifikan pada waktu dan sumber daya. Mengenai biaya, analisis dilakukan membandingkan waktu dan sumber daya yang digunakan dan melalui efisiensi waktu dan sumber daya, maka menghemat biaya operasional. Penggunaan CI/CD paralel membantu kerja tim mengatasi konflik kode, tetapi memberatkan sumber daya dan waktu yang dibutuhkan.*
***Kata Kunci:*** *CI/CD, cloud build, compute engine, GCP*

## 1. Pendahuluan

### 1.1. Latar Belakang

Dalam proses pengembangan sebuah aplikasi, pengembang harus melalui tahapan framework activities. Tahapan berupa *communication, planning, modelling, construction*, dan tahapan terakhir yang harus dilalui sebelum aplikasi sampai kepada *user* adalah *deployment [1]*. Deployment adalah proses yang penting dalam pengembangan aplikasi karena memastikan bahwa aplikasi dapat digunakan oleh *end user*.

Dalam pengembangan perangkat lunak, proses perilisan aplikasi atau deployment sering kali menjadi hambatan. Pengembangan perangkat lunak secara *minor* dan

frekuensinya sering akan sangat merepotkan pengembang dalam waktu perilisannya jika dilakukan secara manual. Pengembang harus melakukan langkah-langkah manual dan berulang, proses *build*, *test* dan *deploy* secara terus menerus, meskipun perubahannya adalah *minor*. Hal ini tidak hanya memakan waktu dan sumber daya, tetapi juga rentan terhadap kesalahan manusia. Maka dari itu, perlu mengubah proses dari yang awalnya manual menjadi otomatis, sehingga proses yang terjadi lebih efektif dan efisien [2]. Melalui penerapan CI/CD juga bisa meningkatkan produktivitas dari pembuatan aplikasi dengan metode *agile* [3].

*Continuous integration* (CI) adalah praktik yang mengharuskan pengembang untuk menggabungkan kode secara berkala ke dalam *repository*. Melalui proses *continuous integration* (CI), setiap kali penggabungan kode dilakukan, pengecekan juga akan dilaksanakan untuk memastikan kualitas dan keamanan kode. Proses yang terjadi dalam CI adalah proses *integration*, *build* dan *testing* secara otomatis. Sedangkan, *continuous deployment* (CD) adalah praktik yang mengharuskan aplikasi yang telah dibangun di *continuous integration* (CI) untuk didistribusikan baik ke lingkungan *staging* atau *production* secara otomatis atau disebut otomatisasi [4].

Pada *report* yang dibuat oleh *DevOps Research and Assesment* (DORA) dan Google Cloud pada 2022, *software delivery performance* (SDP) akan menilai efektivitas proses *deployment*. Berdasarkan *report* tersebut, *continuous integration* (CI) dan *continuous deployment* (CD) adalah indikator yang sangat berpengaruh dalam *software delivery performance* (SDP) [5]. *Report* tersebut merupakan program terbesar dan terpanjang yang berupaya untuk memahami kemampuan yang mendorong nilai dari *software delivery performance* yang dijalankan setiap tahunnya. Salah satu parameter yang ada dari *report* tersebut adalah *lead time for changes* yang mengukur waktu yang dibutuhkan untuk menerapkan perubahan kode dari saat perubahan itu dibuat atau *code commit*, hingga perubahan tersebut berhasil berjalan. Salah satu faktor yang sangat mempengaruhi *lead time for changes* adalah dari penerapan CI/CD.

Melalui permasalahan yang dihadapi dalam proses perilisan yang dilakukan secara manual atau tradisional dan penelitian yang telah dilakukan bahwa CI/CD mempengaruhi performa di dalam SDP. Maka akan dilakukan pengujian dari implementasi CI/CD mengenai waktu atau *lead time for changes*, biaya sumber daya dan pemanfaatan paralel yang dibutuhkan dalam penerapannya yang kemudian akan dibandingkan dengan yang bukan CI/CD. Obyek yang akan diteliti adalah aplikasi web menggunakan React dan Express dalam pengembangannya. Hal ini dilakukan untuk membuktikan CI/CD memiliki efisiensi dan performa yang lebih baik daripada deployment secara manual atau tradisional.

Berdasarkan latar belakang masalah di atas dapat dirumuskan masalah sebagai berikut: (1) Bagaimana menilai performa efisiensi waktu dari *pipeline* CI/CD dibandingkan dengan proses manual dalam *deployment* aplikasi web? (2) Bagaimana menilai penggunaan sumber daya dari *pipeline* CI/CD dibandingkan dengan proses manual dalam *deployment* aplikasi web? (3) Bagaimana menilai biaya yang dibutuhkan dari *pipeline* CI/CD dibandingkan dengan proses manual dalam *deployment* aplikasi web? (4) Bagaimana pengaruh kemampuan paralel dari CI/CD dengan kerja sama tim dalam *deployment* aplikasi web?

Penelitian memerlukan batasan agar tidak terlalu luas dalam arah penelitiannya. Terdapat batasan penelitian dalam melakukan penelitian, yaitu: (1) Penelitian ini hanya

membahas tentang pengembangan dan implementasi CI/CD, bukan tentang proses pengembangan perangkat lunak secara keseluruhan. (2) Penjelasan aplikasi yang digunakan hanya mengenai kompleksitas dari aplikasi tersebut. (3) Penelitian menggunakan Google Cloud Platform sebagai penyedia komputasi awan, dengan memanfaatkan dana dari Bangkit Academy dan *free trial* yang tersedia. (4) Penelitian ini terbatas pada *framework* Express dan *library* React, serta *database* MySQL. Penggunaan *services* pihak ketiga tertentu, seperti: Cloud Source Repositories, Cloud Build, Compute Engine, Artifact Registry dan Cloud Monitoring.

Berdasarkan rumusan masalah yang ada, maka penelitian ini bertujuan untuk: (1) Meneliti performa waktu *pipeline* CI/CD aplikasi web dibandingkan dengan proses manual. (2) Meneliti sumber daya yang dibutuhkan untuk proses CI/CD aplikasi web dibandingkan proses manual. (3) Meneliti biaya yang dibutuhkan dalam proses CI/CD aplikasi web dibandingkan dengan proses manual. (4) Meneliti pengaruh kemampuan paralel dari CI/CD dengan kerja sama tim dalam deployment aplikasi.

## 2. Tinjauan Pustaka

Penerapan CI/CD yang memberikan banyak dampak positif sudah dilaksanakan oleh banyak orang. Tinjauan Pustaka ini akan membandingkan penelitian yang telah dilaksanakan sebelumnya. Perbandingan ini diharapkan memberikan acuan dan menjelaskan keunikan penelitian yang ditulis.

Penelitian pertama bertujuan untuk mengimplementasikan CI/CD menggunakan Jenkins, Docker Gitlab, dan Portainer pada aplikasi React dan Node JS. Parameter yang dicari dari penelitian ini adalah waktu yang dibutuhkan untuk proses manual dan dibandingkan dengan proses CI/CD. Aplikasi yang ada akan di-containerized yang kemudian akan dirilis pada virtual machine dengan *operation system* ubuntu. Proses yang dilalui pada pengujian secara manual dimulai dari *code commit*, *pull code*, *containerized* lalu *running* dari *container*. Sedangkan, pengujian pada CI dan CD dibagi menjadi kedua hal tersebut, yaitu CI untuk proses build dan testing, lalu CD untuk proses *deployment*. Prosesnya sama, yaitu *code commit*, lalu CI dan dilanjutkan pada CD. Waktu yang didapatkan pada pengujian tersebut membuktikan bahwa CI/CD lebih baik untuk performa dari segi waktu. Waktu yang dibutuhkan untuk proses manual adalah delapan menit 48 detik, sedangkan setelah implementasi CI/CD didapatkan waktu dua menit 56 detik [6].

Penelitian kedua bertujuan untuk melakukan penelitian tentang pengembangan CI/CD untuk pengujian performa dari DevOps tersebut melalui aplikasi yang dibuat di pendidikan tinggi sebagai tugas akhir. Penelitian ini mengimplementasikan CI/CD menggunakan jenkins, docker, kubernetes dan argocd. Penelitian ini dilakukan karena proses pengembangan secara tradisional atau manual terjadi secara repetitif dalam *developing*, *testing* dan *deploying*. Dalam pengembangan aplikasi dengan tim juga ditemukan masalah mengenai tim operasional harus menunggu developer untuk menyelesaikan testing sebelum rilis aplikasi. Setelah proses testing selesai, konflik sering ditemukan karena lingkungan pengembangan yang berbeda, yang memperlambat proses rilis. Masalah-masalah ini dapat menyebabkan penundaan dalam memberikan rilis aplikasi kepada klien. Implementasi CI/CD menjadi jawaban untuk permasalahan tersebut dikarenakan proses pengembangan menjadi otomatis sehingga mempercepat proses dari perilisan aplikasi [7].

Penelitian ketiga bertujuan untuk mengimplementasikan CI/CD menggunakan AWS CodePipeline, AWS CodeDeploy dan Amazon EC2. Objek penelitiannya berupa *academic administrative services*. Penelitian ini mendiskusikan mengenai implementasi dari CI/CD dalam pengembangan *academic information system application* di

Paramadina University yang masih menggunakan proses manual dalam pengembangannya. Implementasi dari CI/CD diharapkan dapat lebih efektif dan kualitas produk menjadi lebih baik. Penelitiannya sendiri menggunakan metode kualitatif dengan mengobservasi data yang terkumpul. Hasil dari penelitian ini membuktikan bahwa Implementasi CI/CD mempercepat proses *deployment* aplikasi, meminimalisir kemungkinan adanya *bug*, dan tim pengembang menjadi lebih mudah dalam melakukan pembaruan dan perubahan kapan saja ketika dibutuhkan [8].

Penelitian keempat bertujuan untuk mengimplementasikan DevOps pada aplikasi skrining. Penelitian ini membahas penerapan DevOps menggunakan *tools* yang tidak terlalu banyak, yaitu Gitlab untuk mengimplementasikan CI/CD. Penelitian ini memberikan solusi untuk masalah dari pengembangan agile scrum yang memakan waktu banyak dan menyebabkan rilisan tidak sesuai dengan jadwal. Solusi untuk mengotomatisasi *build, test,* dan *deploy* bagi *project* yang dijalankan di lingkungan SDLC Melalui penelitian ini menghasilkan kesimpulan bahwa implementasi CI/CD memudahkan penggabungan kode, kelancaran build, dan pemeriksaan kelayakan kode setiap kali terjadi *trigger* dari pengembangan aplikasi. Penelitian ini mengamati penggunaan dari sumber daya pada proses CI/CD [9].

Penelitian kelima bertujuan untuk melakukan penelitian tentang penerapan CI/CD dengan mengembangkan aplikasi web melalui Docker, Jenkins, Sonarqube dan Kubernetes. Penelitian ini mengimplementasikan CI/CD menggunakan keempat *tools* tersebut pada sebuah Aplikasi myITS *Single Sign On*, yaitu aplikasi dari kampus ITS untuk memudahkan mahasiswanya. Proses pengembangan dari aplikasi tersebut masih manual, sehingga penelitian dilakukan untuk membuat proses pengembangan menjadi otomatis. *Tools* yang digunakan seperti Docker untuk mengemas aplikasi menjadi image hingga deploy pada Kubernetes. Jenkins yang digunakan untuk mengintegrasikan lingkungan *development* dan *production* serta tempat *pipeline* berada. Sonarqube digunakan untuk analisis *source code*, serta Kubernetes sebagai tempat aplikasi di-*deploy* dan dikelola [10].

Penelitian keenam bertujuan untuk melakukan penelitian tentang penerapan CI/CD dengan mengembangkan aplikasi web melalui CodePipeline, GitHub, AWS Elastic Beanstalk, Slack dan EC2. Penelitian ini mengimplementasikan CI/CD menggunakan tools tersebut pada sebuah aplikasi web WeShop, yaitu aplikasi toko *online* yang berfungsi untuk jual beli barang. Aplikasi menggunakan bahasa pemrograman PHP, PHP Framework, HTML, Python dan Node JS. Github digunakan untuk *source repository* yang kemudian diintegrasikan dengan Slack untuk *monitoring*. CodePipeline digunakan untuk integrasi antara Github dengan Elastic Beanstalk dan EC2 untuk kemudian dilakukan perilisan secara otomatis. Parameter yang dihitung adalah waktu setiap bahasa pemrograman yang ada dari aplikasi web WeShop. Kelima web dengan bahasa pemrograman yang berbeda, tetapi dengan tampilan dan fungsi yang sama kemudian dibandingkan secara waktu [11].

Penelitian ketujuh bertujuan untuk mengimplementasikan CI/CD menggunakan Jenkins, Slack dan Docker. Objek penelitiannya berupa aplikasi web. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap sumber daya, seperti kesehatan mesin, CPU dan RAM. Implementasi CI/CD dimulai dengan mengatur Jenkins sebagai *server* CI/CD, yang secara otomatis akan membangun, menguji, dan mendistribusikan aplikasi web tersebut setiap kali ada perubahan pada kode sumber. Jenkins akan diintegrasikan dengan Docker untuk memastikan setiap dilakukan dalam lingkungan yang konsisten, serta dengan Slack untuk memberikan notifikasi real time kepada tim pengembang mengenai status *build* dan *deploy*. Penelitian ini juga melakukan pengawasan dan pencatatan kinerja mesin untuk membandingkan dua skenario: penggunaan Docker dan tidak menggunakan Docker. Dalam skenario pertama, aplikasi dan Jenkins berjalan di dalam kontainer Docker, sementara pada skenario kedua, mereka berjalan langsung pada *host machine* tanpa *containerized* [12].

Penelitian kedelapan bertujuan untuk membandingkan *deployment* secara manual dan CI/CD secara paralel dan berurutan. Objek penelitiannya berupa aplikasi mobile. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap efisiensi waktu. Pengujian dilakukan pada VM dan layanan *cloud* yang digunakan adalah Azure. Sedangkan, layanan CI/CD yang digunakan adalah Azure DevOps. Penelitian mengamati proses terjadinya deployment berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD secara paralel menghemat lebih banyak waktu [13].

Penelitian kesembilan bertujuan untuk membandingkan CI/CD yang menggunakan GitHub Actions dan Jenkins. Objek penelitiannya berupa aplikasi web. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap waktu lalu membandingkan kedua metode antara GitHub Actions dan Jenkins. Penelitian mengamati proses terjadinya *deployment* berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD dengan GitHub Actions menghemat lebih banyak waktu daripada Jenkins [14].

Penelitian kesepuluh bertujuan untuk membandingkan CI/CD yang menggunakan Gitlab dan Jenkins. Objek penelitiannya berupa Aplikasi *monolithic* Basu *Daily Farm Admin Website*. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap waktu lalu membandingkan kedua metode antara Gitlab dan Jenkins. Penelitian mengamati proses terjadinya *deployment* berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD dengan Jenkins menghemat lebih banyak waktu daripada Gitlab [15].

Berdasarkan berbagai penelitian yang sudah dilakukan, maka akan dilakukan implementasi CI/CD. Hal ini dilakukan untuk pembuktian dari performa CI/CD yang dikatakan lebih baik daripada dilakukan secara manual atau tradisional. Berikutnya akan dilakukan analisis dampak CI/CD yang diperoleh berdasarkan waktu, biaya, sumber daya dan penggunaan paralel. Analisis yang dilakukan ada empat dikarenakan kebanyakan penelitian hanya menganalisis dari segi waktu. Sedangkan, CI/CD juga bisa berdampak pada segi sumber daya yang digunakan dan biaya dalam pengembangannya. Penggunaan GCP, terutama Cloud Build juga masih jarang disinggung, bahkan berdasarkan 10 jurnal yang ada belum ada yang membahas Cloud Build. Sehingga, penelitian yang akan dilakukan menggunakan layanan GCP, terutama Cloud Build serta menganalisis dampak dari segi waktu, biaya, sumber daya dan penggunaan paralel untuk pembuktian dari efisiensi CI/CD.

## 3. Metodologi Penelitian

Pertama adalah studi literatur, pada tahap ini, peneliti akan mencari dan mengevaluasi berbagai sumber informasi, termasuk artikel, jurnal, penelitian sebelumnya, dan buku yang terkait dengan implementasi CI/CD. Tujuan studi literatur adalah untuk memperoleh pemahaman mendalam tentang proses-proses terkait dan kemajuan terkini dalam lingkup penelitian ini.

Kedua adalah analisis kebutuhan, pada tahap ini, peneliti akan mencari kebutuhan yang perlu dipersiapkan sebelum penelitian dimulai, seperti tools yang digunakan dan kebutuhan lainnya. Setelah menganalisis kebutuhan ini, penelitian akan mendapatkan kerangka utama yang akan digunakan sebagai basis dari penelitian.

Ketiga adalah implementasi penelitian, setelah merinci kebutuhan penelitian, tahapan penerapan CI/CD akan dilaksanakan. Proses ini akan mengikuti kerangka yang telah disusun sebelumnya. Penerapan ini akan berfokus pada CI/CD dan bukan pada pengembangan aplikasi keseluruhan.

Keempat adalah pengujian penelitian, tahapan ini akan memastikan bahwa implementasi yang dilakukan di tahap implementasi berjalan dengan baik. Tahapan ini akan dilakukan untuk menjawab persoalan dari rumusan masalah penelitian ini. Pada tahap ini akan dilakukan pengujian keberhasilan implementasi, performa dan efisiensi CI/CD.

Kelima adalah evaluasi kinerja, pada tahap ini, *feedback* dan hasil dari tahap sebelumnya, yaitu tahap pengujian dibahas dan dianalisis. Selain itu, evaluasi terkait hasil pengujian juga dilakukan. Tahapan ini bertujuan untuk mengidentifikasi potensi perbaikan dan peningkatan yang dapat diterapkan pada *output* yang sedang dievaluasi.

## 4. Hasil dan Diskusi

### 4.1 Perancangan Sistem

Perancangan sistem CI/CD pada penelitian ini memerlukan *tools* dan *virtual machine* dari pihak ketiga. Pihak ketiga yang digunakan adalah Google Cloud Platform. Dalam pelaksanaannya agar bisa mengukur waktu dan *monitoring* sumber daya, maka perlu dibuat dua *virtual machine* dengan spesifikasi sama untuk mendukung penelitian. *Virtual machine* yang digunakan adalah Compute Engine dari GCP. Spesifikasi dari VM yang perlu dibuat seperti pada Tabel 1.

**Tabel 1 Tabel Spesifikasi Compute Engine**

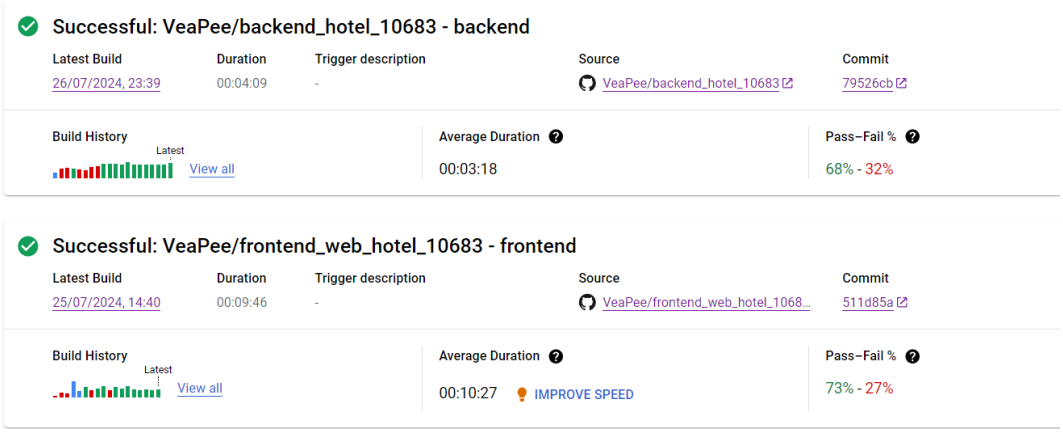
| | |
|---|---|
| *Provider* | Google Cloud Platform |
| *Machine Type* | e2-*medium* |
| vCPU | 1-2 vCPU (1 *Shared Core*) |
| RAM | 4 GB |
| *Operating System* | Ubuntu 20.04 LTS |
| *Disk Type* | *Balanced persistent disk* |
| *Disk Size* | 40 GB |
| *Location* | asia-southeast1-a (Singapore) |

Pada Tabel 1 terlihat spesifikasi lengkap dari *virtual machine* yang akan digunakan. *Virtual machine* akan digunakan sebagai tempat tujuan akhir dari alur *pipeline* dan akan merilis aplikasi web dalam bentuk *container*. Perlu dibuat dua VM yang identik dengan tujuan uji coba secara manual dan secara CI/CD untuk keperluan pencatatan waktu dan *monitoring* sumber daya selama pelaksanaan *build, testing* dan *deployment*. Obyek yang akan digunakan untuk pengujian adalah aplikasi web React dan Express dengan database MySQL.

### 4.2 Hasil Pengujian

#### 4.2.1 Hasil Pengujian Waktu

Pengujian waktu dilakukan masing-masing lima kali untuk pengujian *frontend* dan *backend*, secara manual dan melalui penerapan CI/CD. Waktu ini dipengaruhi oleh proses yang terjadi di dalam *file configuration* dan kompleksitas dari aplikasi yang ada. Kecepatan yang diperoleh didapatkan dari proses *commit, build, testing,* dan *deploy*. Pada Gambar 1 terlihat *dashboard* dari pengujian CI/CD menggunakan Cloud Build dalam melakukan proses CI/CD. Terlihat persentase keberhasilan dan rata-rata waktu dari proses CI/CD. Namun, pencatatan tidak melihat *dashboard*, tetapi sudah ada pencatatan dengan penggunaan *timestamp* pada setiap tahapan pengujian. Pencatatan akan melihat proses dari lima uji coba *frontend* dan *backend* CI/CD dibandingkan dengan pengujian manual. Pencatatan akan menggunakan tabel untuk melihat *timestamp* dan waktu yang dibutuhkan dalam prosesnya.




**Gambar 1. Tampilan Dashboard CI/CD**

Pengujian waktu secara CI/CD dan manual dilaksanakan bersamaan dan dimulai saat dilakukan *code commit* pada *repository*. Setiap tahapan dimulai dan berakhir akan ditandai menggunakan *timestamp*. Telah diperoleh hasil pengujian yang dilakukan pada tanggal 25 Juli 2024 dimulai dari jam 12:00 hingga jam 16:00 dengan pengubahan fitur pada tampilan untuk *frontend* dan pengubahan pesan respons pada *backend*. Dampak yang diberikan pada waktu yang diperlukan dalam proses *deployment* antara CI/CD dan manual terjadi pemangkasan waktu yang diperlukan untuk waktu dari *commit* hingga *deploy* selesai atau disebut *lead time*.

Pada Tabel 2 dapat dilihat bahwa pada *frontend* terjadi pemangkasan dari yang 17 menit 13 detik menjadi 10 menit 33 melalui penerapan CI/CD. Waktu yang dipangkas sebesar enam menit 40 detik. Pada *backend* terjadi pemangkasan dari yang lima menit 27 detik menjadi tiga menit 40 detik. Waktu yang dipangkas sebesar satu menit 47 detik. Selain itu, *lead time* yang diperoleh juga lebih konsisten untuk penerapan CI/CD. Pemangkasan waktu akan membuat proses berjalannya pengembangan aplikasi dapat lebih cepat. Selain itu, proses CI/CD juga bisa berjalan secara paralel, sehingga proses perubahan dari *frontend* dan *backend* dapat berjalan bersamaan. Proses ini juga berjalan secara otomatis, sehingga waktu selama proses berjalannya *pipeline* dapat digunakan untuk pekerjaan lain.

**Tabel 2 Tabel Hasil Uji Coba Waktu**

| Pengujian | Bagian | Waktu |
|---|---|---|
| CI/CD | *Frontend* | 10 menit 33 detik |
| | *Backend* | 3 menit 40 detik |
| Manual | *Frontend* | 17 menit 13 detik |
| | *Backend* | 5 menit 27 detik |

### 4.2.2 Hasil Pengujian Sumber Daya

Berdasarkan Tabel 3, perbandingan sumber daya yang digunakan pada VM dengan penerapan CI/CD dan pengujian manual menunjukkan beberapa perbedaan utama. VM dengan CI/CD cenderung lebih ringan dalam pemakaian CPU dan lebih stabil dalam penggunaan RAM dibandingkan dengan pengujian manual. Namun, penggunaan disk pada VM CI/CD relatif lebih boros dibandingkan VM manual, yang lebih hemat.

Penerapan Cloud Build sebagai alat CI/CD memungkinkan pengelolaan proses pengujian secara otomatis sehingga meminimalkan beban kerja pada CPU dan RAM. Namun, untuk mendukung proses yang optimal, penggunaan disk menjadi lebih besar guna menyimpan log pipeline, cache, dan artefak hasil pengujian. Tanpa kapasitas disk yang memadai, potensi terjadinya bottleneck dapat meningkat, yang akan memengaruhi performa pipeline CI/CD.

Penerapan CI/CD mengurangi beban CPU karena proses pengujian dilakukan secara otomatis dan terjadwal. Hal ini memastikan tugas-tugas yang dijalankan pada VM tersebar secara efisien, mengurangi lonjakan penggunaan CPU yang sering terjadi pada pengujian

manual, di mana pengujian dilakukan tanpa pengaturan otomatis. Pada CI/CD, penggunaan RAM cenderung stabil karena pipeline dirancang untuk memproses data secara terstruktur dan terotomatisasi. Sebaliknya, pengujian manual sering kali menghasilkan penggunaan RAM yang fluktuatif, terutama ketika menjalankan berbagai proses secara bersamaan tanpa manajemen sumber daya yang optimal. Penerapan CI/CD memerlukan penggunaan disk yang lebih besar karena harus menyimpan berbagai data tambahan seperti log pengujian, cache build, dan artefak hasil build. Selain itu, riwayat pipeline yang disimpan juga meningkatkan kebutuhan kapasitas disk. Sementara itu, pengujian manual lebih hemat disk karena log dan data hasil pengujian tidak dihasilkan secara berulang atau dalam jumlah besar. Pada pengujian manual, penggunaan disk lebih sedikit karena prosesnya tidak membutuhkan penyimpanan log dan artefak yang terstruktur. Meskipun lebih hemat dalam hal ini, pengujian manual dapat meningkatkan beban kerja CPU dan RAM secara signifikan, terutama jika pengujian dilakukan tanpa jeda atau kontrol sumber daya.

Penerapan CI/CD memberikan manfaat yang signifikan dalam hal efisiensi penggunaan CPU dan stabilitas RAM, menjadikannya solusi yang ideal untuk pengujian otomatis. Namun, kebutuhan disk yang lebih besar menjadi faktor yang perlu dipertimbangkan, terutama untuk organisasi yang memiliki keterbatasan kapasitas penyimpanan. Oleh karena itu, pengaturan kapasitas disk yang memadai sangat penting untuk memastikan implementasi CI/CD berjalan optimal tanpa hambatan.

**Tabel 3 Tabel Perbandingan Sumber Daya**

| | CI/CD | Manual |
|---|---|---|
| Sumber Daya CPU | Lebih Ringan | Lebih Berat |
| Sumber Daya RAM | Lebih Stabil | Fluktuatif |
| Sumber Daya *Disk* | Lebih Boros | Lebih Hemat |

Selain dari perbandingan sumber daya pada infrastruktur, penggunaan sumber daya manusia untuk proses manual akan lebih besar daripada proses CI/CD. Analisis dilakukan berdasarkan dampak dari implementasi CI/CD terhadap waktu. Proses yang terjadi pada CI/CD berjalan secara otomatis untuk semua tahapan, yaitu *integration, build, testing* dan *deployment*. Terutama pada proses *testing*, pengujian kode secara otomatis menggunakan unit testing yang dilakukan pada penelitian ini akan membuat tim QA tidak perlu melakukan pengujian fungsional dasar. Jika pengujian kode dilakukan pada tahap berikutnya, yaitu *integration testing* atau bahkan *end to end testing,* maka sumber daya manusia yang diperlukan untuk proses *testing* bisa berkurang. Sehingga, sumber daya manusia yang dibutuhkan untuk proses tertentu, terutama proses *testing* dapat dialihkan untuk hal lain.

### 4.2.3 Hasil Pembahasan Biaya

Perbandingan biaya untuk pengembangan aplikasi secara manual dengan implementasi CI/CD *pipeline* yang berjalan otomatis akan tergantung dari setiap perusahaan. Penggunaan layanan Cloud Build pada GCP yang biayanya berdasarkan pemakaian, maka biaya dapat berkurang tergantung seberapa sering dilakukan perubahan kode, bahkan jika pemakaiannya tinggi, maka perusahaan dapat menghemat waktu lebih banyak. Kemudian, berdasarkan analisis waktu yang telah dihemat selama proses CI/CD dapat diperoleh informasi bahwa biaya yang digunakan untuk pegawai melakukan tugas berulang dapat dipindahkan pada hal lainnya.

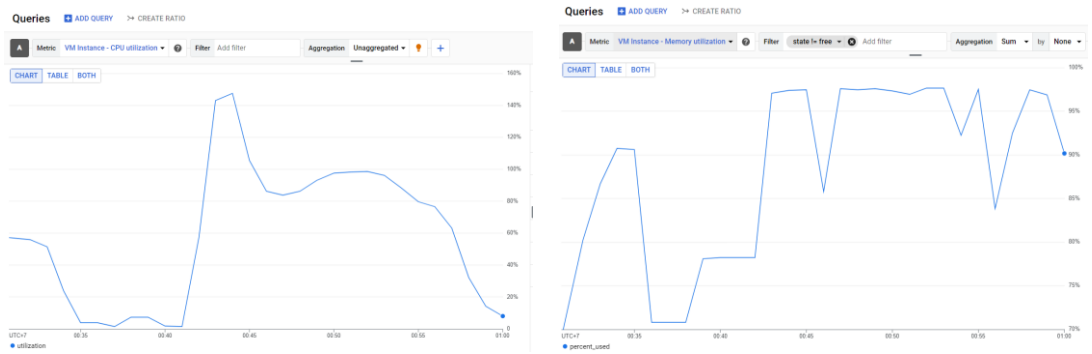
Perusahaan memiliki kebijakan yang berbeda, tetapi untuk perusahaan yang membayar pegawai dengan hitungan per jam, maka implementasi CI/CD *pipeline* akan sangat berpengaruh dalam mengurangi beban waktu pegawai yang kemudian bisa dialokasikan untuk pekerjaan lainnya. Penghematan waktu, maka akan membuat pengeluaran biaya per jam akan lebih efektif dan efisien. Biaya untuk memulai CI/CD membutuhkan investasi awal, tetapi akan berdampak ke depannya.

### 4.2.4 Hasil Pengujian Paralel

Kemampuan paralel CI/CD dalam proses deployment aplikasi dapat membantu untuk perubahan atau penambahan fungsionalitas dari banyak commit yang terjadi secara bersamaan, sehingga memungkinkan untuk kerja sama tim menjadi lebih baik dan terhindar dari konflik kode. Namun, beban kerja yang lebih tinggi membuat waktu dari deployment menjadi lebih lama dibandingkan dengan sekuensial, terutama karena keterbatasan dari sumber daya yang dimiliki, sehingga banyak pekerjaan yang dijalankan secara paralel dapat menyebabkan kelebihan beban yang membuat *deployment* menjadi lebih lama.

Terlihat pada Gambar 2 bahwa pemakaian CPU lebih banyak pada paralel dibandingkan dengan penggunaan CPU saat CI/CD berjalan sekuensial. Selain itu, terlihat juga pemakaian RAM juga jauh lebih banyak bahkan hampir menyentuh batas dari sumber daya yang digunakan. Sedangkan, pada pemakaian RAM CI/CD sekuensial hanya sekitar 50 persen.

Manfaat dari paralelisme dapat meningkatkan kolaborasi tim, memungkinkan banyak pengembang untuk mengerjakan fitur yang berbeda secara bersamaan. Ini juga mengurangi konflik kode karena perubahan digabungkan dan diuji lebih sering. Namun, memiliki tantangan berupa waktu *deployment* yang lebih lama, keseluruhan waktu deployment bisa menjadi lebih lama karena peningkatan beban kerja pada sistem. Keterbatasan sumber daya, jika sumber daya komputasi terbatas, menjalankan terlalu banyak tugas secara paralel dapat menyebabkan kelebihan beban, yang memperlambat semua tahapan yang ada.




**Gambar 2. Grafik CPU dan RAM CI/CD Paralel**

## 5. Kesimpulan dan Saran

Penelitian ini menunjukkan bahwa penerapan CI/CD secara signifikan meningkatkan efisiensi proses deployment dengan memangkas waktu dan mengurangi keterlibatan manual, memungkinkan pengembang fokus pada tugas lain. Penggunaan sumber daya lebih hemat pada CPU dan lebih stabil pada RAM, meskipun membutuhkan kapasitas disk yang lebih besar. Biaya operasional dapat ditekan berkat waktu yang dihemat dan pengurangan kebutuhan sumber daya manusia. Namun, frekuensi deployment memengaruhi biaya layanan CI/CD, sehingga perlu manajemen yang bijak. Paralelisme dalam CI/CD mempercepat pengembangan, tetapi harus disesuaikan agar tidak membebani sumber daya. Ke depan, spesifikasi dan kebutuhan layanan harus direncanakan dengan baik untuk menghindari biaya tak terduga. Otomatisasi infrastruktur dengan alat seperti Terraform atau Ansible dapat meningkatkan efisiensi dan mengurangi risiko kesalahan.

## Referensi

[1] R. S. Pressman dan B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th Edition, New York, NY: McGraw-Hill Professional, 2020.

[2] H. Toba, T. K. Gautama, J. Narabel, A. Widjaja dan S. F. Sujadi, "Evaluasi Metodologi CI/CD untuk Pengembangan Perangkat Lunak dalam Perkuliahan," *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, vol. 8, no. 2, pp. 227-234, 2022.

[3] S. Arachchi dan P. Indika, "Continuous Integration and Continuous Delivery Pipeline

Automation for Agile Software Project Management," *2018 Moratuwa Engineering Research Conference (MERCon),* pp. 156-161, 2018.

[4] Gitlab, "What is CI/CD?," [Online]. Available: https://about.gitlab.com/topics/ci-cd/. [Diakses 3 Desember 2023].

[5] C. Peters, D. Farley, D. Villalba, D. Stanke, D. DeBellis, E. Maxwell, J. S. Meyers, K. ". Xu, N. Harvey dan T. Kulesza, "2022 Accelerate State of DevOps Report," 2022. [Online]. Available: https://cloud.google.com/devops/state-of-devops/. [Diakses 26 October 2023].

[6] Nurhayati, "Implementation of Continuous Integration and Continuous Deployment (CI/CD) to Speed up the Automation Process of Software Delivery In the Production Process Using Node.Js, Docker, and React.Js," *Jurnal Info Sains : Informatika dan Sains,* vol. 14, no. 2, pp. 15-28, 2024.

[7] J. Jaeni, N. A. S. dan L. D. Arif, "Implementasi Continuous Integration/Continuous Delivery (CI/CD) Pada Performance Testing Devops," *Journal of Information System Management (JOISM),* vol. 4, no. 1, pp. 62-66, 2022.

[8] R. Indriyanto dan D. G. Purnama, "CI/CD Implementation Application Deployment Process Academic Information System (Case Study Of Paramadina University)," *Jurnal Indonesia Sosial Teknologi,* vol. 4, no. 9, pp. 1503-1516, 2023.

[9] Tohirin, S. F. Utami, S. R. Widianto dan W. A. Mauludyansah, "Implementasi DevOps pada Pengembangan Aplikasi e-Skrining Covid-19," *MULTINETICS,* vol. 6, no. 1, p. 15–20, 2020.

[10] R. A. Parama, H. S. dan R. J. Akbar, "Implementasi Continuous Integration dan Continuous Delivery Pada Aplikasi myITS Single Sign On," *JURNAL TEKNIK ITS,* vol. 11, no. 3, pp. 264-269, 2022.

[11] A. Alanda, H. Mooduto dan R. Hadelina, "Continuous Integration and Continuous Deployment (CI/CD) for Web Applications on Cloud Infrastructures," *JITCE (Journal of Information Technology and Computer Engineering),* vol. 6, no. 2, pp. 50-55, 2022.

[12] R. T. Kusumadewi dan R. Adrian, "Performance Analysis of Devops Practice Implementation Of CI/CD Using Jenkins," *MATICS : Jurnal Ilmu Komputer dan Teknologi Informasi,* vol. 15, no. 2, pp. 90-95, 2023.

[13] A. D. Setyoko dan A. Zahra, "Perbandingan Efisiensi Proses CI/CD Multi-Lingkungan melalui Implementasi Paralel dan Berurutan," *MALCOM: Indonesian Journal of Machine Learning and Computer Science,* vol. 4, no. 3, pp. 911-925, 2024.

[14] Z. Zulhakim dan A. Kurniawan, "Implementasi Continuous Integration Dan Continuous Deployment Pada Pengembangan Aplikasi Website Menggunakan Docker Dan Github Actions," *Jurnal Manajemen Informasi,* vol. 16, no. 1, pp. 1-11, 2023.

[15] A. B. Kuncara, D. S. Kusumo dan M. Adrian, "COMPARISON OF JENKINS AND GITLABCI/CD TO IMPROVE DELIVERY TIME OF BASU DAIRY FARM ADMIN WEBSITE," *Jurnal Teknik Informatika (JUTIF),* vol. 5, no. 3, pp. 747-756, 2024.