OTOMATISASI PENERAPAN APLIKASI UNTUK PROYEK INMEET MENGGUNAKAN CI/CD DI PT. INDO TRANS TEKNOLOGI BANDUNG

1st Muhamad Hafizh Nuryasin
Fakultas Ilmu Terapan
Telkom University
Bandung, Indonesia
hafizhyasin@student.telkomuniversity.ac.id

2nd Sungondo Hadiyoso
Fakultas Ilmu Terapan
Telkom University
Bandung, Indonesia
sugondo@telkomuniversity.ac.id

***Abstrak* — Penelitian ini secara komprehensif membahas dan menganalisis implementasi sistem otomatisasi *Continuous Integration/Continuous Delivery* (CI/CD) pada proyek pengembangan aplikasi Sistem Informasi Ruang Rapat (Meeting Room Information System/MRIS) yang dikerjakan di lingkungan PT. Indo Trans Teknologi Bandung. Tujuan utama dari inisiatif ini untuk mempersingkat siklus *development*, meningkatkan efisiensi operasional tim pengembang, mempercepat responsifitas perubahan kebutuhan bisnis yang dinamis, serta mengintensifkan kolaborasi tim. Metodologi penelitian melibatkan konfigurasi dan optimasi *runner* khusus yang didedikasikan untuk tahapan *pipeline* seperti *build*, *publish*, dan *deployment* ke lingkungan *staging*, seluruhnya beroperasi di Google Cloud Platform (GCP). Selain itu, penelitian ini menginvestigasi penggunaan variabel CI/CD untuk manajemen kredensial dan konfigurasi yang aman serta dinamis. Hasil yang diperoleh menunjukkan kinerja *pipeline* yang sangat efisien: pekerjaan *build* berhasil diselesaikan dalam rata-rata 28 detik, pekerjaan *publish* dalam 18 detik, dan *deployment* ke *staging* dalam 16 detik. Secara kualitatif, keberhasilan *pipeline* ini divalidasi dengan suksesnya akses ke aplikasi MRIS di lingkungan *staging*, yang menampilkan halaman *login* yang berfungsi penuh. Kesimpulan utama dari penelitian ini adalah bahwa implementasi CI/CD tidak hanya berhasil mempercepat proses *delivery* fitur secara substansial, tetapi juga mengurangi beban kerja manual, meminimalisir potensi kesalahan, dan meningkatkan kualitas serta keandalan aplikasi secara keseluruhan.**

***Kata kunci* — CI/CD, otomatisasi, deployment, pipeline, GitLab, runner**

I. PENDAHULUAN

Dalam pengembangan perangkat lunak modern yang didominasi oleh persaingan ketat dan ekspektasi pasar yang terus meningkat, kemampuan untuk merilis fitur baru dengan cepat dan merespons perubahan kebutuhan pengguna secara efisien telah menjadi imperatif bisnis. Proyek Sistem Informasi Ruang Rapat (MRIS) di PT. Indo Trans Teknologi Bandung, sebuah aplikasi kritikal untuk manajemen sumber daya internal, menghadapi tantangan yang sering ditemui dalam siklus pengembangan tradisional, termasuk durasi development yang cenderung panjang, keterlibatan intervensi manual yang ekstensif dalam proses penyebaran aplikasi, dan kapasitas yang terbatas untuk beradaptasi secara lincah terhadap perubahan kebutuhan bisnis yang muncul secara tiba-tiba. Permasalahan ini secara langsung berdampak pada produktivitas tim dan waktu time-to-market produk.

Sebagai respons terhadap tantangan tersebut, implementasi praktik Continuous Integration/Continuous Delivery (CI/CD) muncul sebagai strategi transformatif yang menjanjikan solusi komprehensif melalui otomatisasi penuh pada siklus hidup pengembangan perangkat lunak. CI/CD mengintegrasikan proses pengembangan, pengujian, dan deployment ke dalam alur kerja yang mulus dan berulang. Konsep Continuous Integration (CI) mendorong pengembang untuk secara teratur mengintegrasikan perubahan kode mereka ke repositori bersama, diikuti dengan build otomatis dan pengujian untuk mendeteksi bug atau konflik integrasi sejak dini. Sementara itu, Continuous Delivery (CD) memastikan bahwa kode yang telah melewati tahap CI selalu dalam keadaan siap untuk dirilis ke lingkungan produksi kapan saja, dengan Continuous Deployment sebagai evolusi selanjutnya yang mengotomatisasi rilis hingga ke produksi tanpa intervensi manual.

Tinjauan literatur dan praktik industri (state of art) menunjukkan bahwa adopsi sistem CI/CD secara luas telah terbukti secara empiris mampu meningkatkan kecepatan delivery, mengurangi risiko deployment, memperbaiki kualitas produk, dan secara signifikan meningkatkan kolaborasi tim. Berangkat dari latar belakang ini, penelitian ini secara spesifik bertujuan untuk merancang, mengimplementasikan, dan mengevaluasi efektivitas sistem CI/CD untuk proyek MRIS. Tujuan utamanya adalah untuk secara substansial meningkatkan kecepatan delivery fitur

baru, mengoptimalkan efisiensi kerja tim pengembang dengan mengurangi beban manual, mempercepat respons terhadap perubahan kebutuhan bisnis yang dinamis, dan memperkuat sinergi serta kolaborasi antar anggota tim yang terlibat dalam pengembangan aplikasi MRIS.

II. KAJIAN TEORI

Bagian ini menyajikan dan menjelaskan secara rinci teori-teori dasar, konsep-konsep kunci, serta teknologi yang menjadi landasan filosofis dan teknis dalam perancangan dan implementasi sistem CI/CD dalam penelitian ini.

A. Continuous Integration/Continuous Delivery (CI/CD)

CI/CD adalah sebuah paradigma pengembangan perangkat lunak yang mengintegrasikan serangkaian praktik secara otomatis untuk mempercepat dan meningkatkan keandalan proses delivery perangkat lunak. Continuous Integration (CI) adalah praktik di mana anggota tim mengintegrasikan hasil pekerjaan mereka sesering mungkin, biasanya beberapa kali sehari. Setiap integrasi diverifikasi oleh build otomatis (termasuk kompilasi kode) dan pengujian otomatis untuk mendeteksi kesalahan integrasi secepat mungkin. Tujuannya adalah untuk menghindari "integration hell" yang sering terjadi pada proyek besar. Continuous Delivery (CD) adalah perpanjangan dari CI yang memastikan bahwa perangkat lunak dapat dirilis ke produksi secara andal kapan saja. Setiap perubahan kode, setelah melewati semua tahapan CI dan pengujian otomatis, siap untuk deployment manual ke lingkungan produksi dengan satu tombol. Continuous Deployment adalah langkah selanjutnya di mana setiap perubahan yang lolos semua pengujian otomatis secara otomatis disebarkan ke produksi tanpa intervensi manusia, menghasilkan rilis yang sangat cepat.

B. Gitlab CI/CD

GitLab CI/CD merupakan sebuah sistem built-in dan terintegrasi penuh dalam platform pengembangan DevOps GitLab. Sistem ini menyediakan fungsionalitas komprehensif untuk mengotomatisasi seluruh siklus hidup pengembangan perangkat lunak. Konfigurasi GitLab CI/CD didefinisikan melalui berkas YAML bernama .gitlab-ci.yml yang ditempatkan di root repositori proyek. Berkas ini secara deklaratif menjelaskan pipeline, yaitu alur kerja otomatis yang terdiri dari tahapan (stages) dan pekerjaan (jobs). Tahapan dijalankan secara sekuensial, sedangkan pekerjaan dalam satu tahapan dapat berjalan secara paralel .

C. Runner

Runner adalah komponen inti dalam ekosistem GitLab CI/CD yang berfungsi sebagai agen eksekusi. Tugas utamanya adalah mengambil (pick up) dan menjalankan pekerjaan (job) yang telah didefinisikan dalam pipeline GitLab CI/CD . Runner dapat diinstal pada berbagai infrastruktur dan sistem operasi, termasuk server on-premise yang menjalankan Linux Ubuntu atau di lingkungan cloud computing seperti Google Cloud Platform (GCP). Untuk memastikan alokasi pekerjaan yang efisien dan spesifik, runner dapat dikonfigurasi dengan satu atau lebih tag. Tag ini memungkinkan job tertentu dalam pipeline untuk dieksekusi hanya oleh runner yang memiliki tag yang sesuai, sehingga memfasilitasi isolasi lingkungan dan pemanfaatan sumber daya yang optimal untuk jenis pekerjaan yang berbeda .

D. Variable CI/CD

Variabel CI/CD adalah pasangan kunci-nilai yang berfungsi sebagai sarana untuk menyimpan dan menyediakan informasi konfigurasi, kredensial sensitif, atau data dinamis yang dibutuhkan selama eksekusi pipeline. Variabel ini dapat didefinisikan pada tingkat proyek, grup, atau bahkan pipeline individu. Untuk meningkatkan keamanan, variabel dapat dikonfigurasi sebagai Protected, yang berarti variabel tersebut hanya akan tersedia untuk pipeline yang berjalan pada branch atau tag yang dilindungi. Selain itu, variabel dapat diatur sebagai Masked, yang secara otomatis menyembunyikan nilai variabel dalam log eksekusi pipeline untuk mencegah kebocoran informasi sensitif . Penggunaan variabel sangat krusial untuk menjaga konfigurasi pipeline tetap generik dan terpisah dari data sensitif.

E. Docker dan Docker Compose

Docker adalah sebuah platform open-source yang merevolusi cara aplikasi dikembangkan, dikirim, dan dijalankan dengan memanfaatkan teknologi kontainerisasi. Kontainer adalah unit executable perangkat lunak yang mandiri, ringan, dan portabel yang mengemas kode aplikasi beserta semua dependensinya (termasuk runtime, pustaka sistem, dan pengaturan konfigurasi). Dengan demikian, aplikasi dapat berjalan secara konsisten di lingkungan komputasi mana pun, mulai dari laptop pengembang hingga server produksi. Docker Compose adalah alat pendamping Docker yang mempermudah pendefinisian dan manajemen aplikasi multi-container. Dengan Docker Compose, seluruh arsitektur aplikasi yang terdiri dari beberapa service (misalnya, web server, database, backend API) dapat didefinisikan dalam satu berkas YAML. Hal ini memungkinkan pengembang untuk mengelola siklus hidup seluruh aplikasi (mulai dari build, run, stop, hingga remove) hanya dengan satu perintah.

F. Lingkungan Kerja Deployment

Dalam konteks pengembangan perangkat lunak dan alur kerja CI/CD, terdapat beberapa jenis lingkungan deployment yang memiliki peran spesifik:

- Staging: Lingkungan ini dirancang untuk mereplikasi lingkungan produksi sesedekat

mungkin dalam hal konfigurasi perangkat keras, perangkat lunak, dan data. Tujuannya adalah untuk melakukan pengujian akhir yang komprehensif, uji integrasi sistem, dan user acceptance testing (UAT) dalam kondisi yang realistis sebelum aplikasi dilepaskan ke publik. Deployment ke staging merupakan tahapan validasi krusial sebelum masuk ke produksi .

- Produksi: Ini adalah lingkungan live di mana aplikasi diakses dan digunakan oleh pengguna akhir. Lingkungan produksi harus memiliki tingkat stabilitas, keamanan, dan kinerja yang sangat tinggi.

III. METODE

Alur Kerja Continuous Integration dan Continuous Deployment (CI/CD) dalam pengembangan perangkat lunak. Diagram ini memvisualisasikan tahapan-tahapan kunci yang terlibat dalam proses otomatisasi build, pengujian, dan penyebaran aplikasi, yang relevan dengan pembahasan implementasi CI/CD untuk proyek MRIS dalam jurnal kita.

Secara sederhana gambar ini menunjukkan alur yang dimulai dari seorang pengembang yang membuat branch baru dari repositori kode. Setelah melakukan perubahan kode (push code changes), sistem CI secara otomatis melakukan build dan pengujian otomatis. Jika ada perbaikan kode (push code fixes), proses build dan pengujian akan diulang dalam lingkungan Continuous Integration. Setelah dianggap stabil, perubahan kode dapat melalui tahap review dan persetujuan. Setelah disetujui, branch akan di-merge ke branch utama. Dalam skenario Continuous Deployment, setelah merge, sistem akan secara otomatis melakukan build, pengujian, dan deployment aplikasi ke lingkungan produksi. Diagram ini menekankan otomatisasi dalam setiap langkah, mulai dari integrasi kode hingga penyebaran aplikasi, yang merupakan prinsip inti dari CI/CD.

A. Konfigurasi Lingkungan GitLab CI/CD

Tahap awal penelitian ini melibatkan analisis mendalam terhadap kebutuhan spesifik proyek MRIS terkait otomatisasi proses pengembangan perangkat lunak. Ini mencakup identifikasi bottleneck dalam alur kerja pengembangan aplikasi yang sudah ada, khususnya pada aspek build, test, dan deployment. Selain itu, tahap ini juga meliputi penentuan lingkungan target yang diperlukan untuk deployment aplikasi, seperti lingkungan pengembangan (dev), staging, dan produksi, serta spesifikasi persyaratan fungsional dan non-fungsional untuk sistem CI/CD yang akan dibangun.

B. Konfigurasi Lingkungan GitLab CI/CD

Pengaturan lingkungan inti GitLab CI/CD dilakukan sebagai platform utama untuk otomatisasi. Langkah ini dimulai dengan otentikasi akun GitLab dengan repositori kode proyek MRIS. Selanjutnya, sebuah berkas konfigurasi bernama .gitlab-ci.yml dibuat dan disesuaikan di root repositori. Berkas ini secara deklaratif mendefinisikan seluruh alur pipeline CI/CD, termasuk urutan stages (tahapan) dan jobs (pekerjaan) yang akan dieksekusi secara otomatis. Konfigurasi ini mencakup definisi untuk jobs build, test, publish, staging, dan release.

C. Pendaftaran dan Konfigurasi Runner

Untuk memastikan eksekusi jobs dalam pipeline, runner didaftarkan dan dikonfigurasi secara spesifik untuk proyek MRIS. Dalam penelitian ini, empat (4) runner proyek khusus telah dialokasikan dan diatur: MRIS Pre-Production GCP, MRIS Staging GCP, MRIS Testing Tags GCP, dan MRIS Build GCP . Setiap runner ini diberi tag unik dan spesifik (misalnya, preprod-gcp, staging-gcp, testing-gcp, build-gcp) untuk memungkinkan pipeline mengarahkan job ke runner yang paling sesuai dengan jenis tugasnya, sehingga memastikan isolasi lingkungan dan pemanfaatan sumber daya yang optimal. Seluruh runner dioperasikan pada sistem operasi Linux (Ubuntu) dan memanfaatkan Docker sebagai executor untuk menjalankan pekerjaan dalam lingkungan kontainer yang terisolasi .

D. Manajemen Variabel CI/CD

Variabel CI/CD yang krusial untuk proses build dan deployment, termasuk konfigurasi lingkungan (misalnya, DOCKER_COMPOSE_INGRIX, ENV_PREPROD, ENV_STAGING) serta kredensial sensitif (seperti $CI_REGISTRY_USER, $CI_REGISTRY_PASSWORD), dikelola secara aman dalam pengaturan CI/CD GitLab. Variabel-variabel ini dikonfigurasi sebagai Masked untuk menyembunyikan nilainya dari log eksekusi, dan Protected untuk membatasi akses hanya pada branch yang dilindungi, guna menjaga kerahasiaan dan keamanan informasi sensitif . Hal ini esensial untuk memisahkan konfigurasi pipeline dari data sensitif dan memfasilitasi penggunaan kembali pipeline tanpa perlu mengubah kredensial secara manual.

E. Pengembangan dan Pengujian Pipeline

Pipeline CI/CD dikembangkan secara iteratif, dimulai dengan definisi dasar dan kemudian disempurnakan. Pipeline MRIS dirancang untuk mencakup beberapa stages, termasuk build (membuat citra Docker), test (menjalankan pengujian unit/integrasi), publish (mengunggah citra ke registry), staging (menyebarkan ke lingkungan staging), dan release (membuat rilis resmi), serta stage database untuk pengelolaan basis data . Setiap kali ada commit baru atau merge request yang diajukan ke repositori kode, pipeline ini akan secara otomatis dipicu. Proses ini mencakup langkah-langkah seperti validasi sintaksis kode, pembangunan citra Docker aplikasi, publikasi citra tersebut ke registry Docker, dan penyebaran otomatis ke lingkungan staging yang telah ditentukan.

F. Verifikasi Hasil Deployment

Setelah pekerjaan deployment ke lingkungan staging berhasil diselesaikan oleh pipeline, tahapan verifikasi fungsionalitas dilakukan secara manual. Verifikasi ini melibatkan akses langsung ke URL aplikasi yang telah di-deploy (misalnya, mris-staging.transtrack.id) melalui peramban web . Keberhasilan akses dan tampilan halaman login aplikasi yang berfungsi penuh menjadi indikator

kualitatif utama bahwa aplikasi telah berhasil di-deploy dan beroperasi sebagaimana mestinya di lingkungan staging.

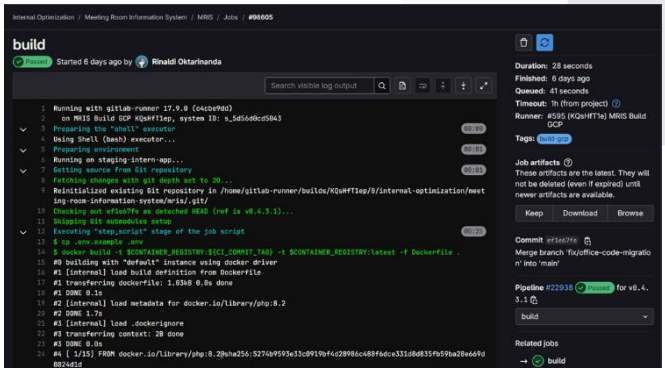
Penomoran untuk persamaan disusun berurutan. Nomor persamaan, di dalam tanda kurung, harus memposisikan rata ke kanan dengan menggunakan penghentian tab kanan.

IV. HASIL DAN PEMBAHASAN

Bagian ini memaparkan secara objektif hasil-hasil yang diperoleh dari implementasi sistem CI/CD untuk proyek MRIS, diikuti dengan analisis mendalam dan interpretasi terhadap temuan-temuan tersebut dari perspektif kualitatif dan kuantitatif. Pembahasan ini mencakup hasil spesifik dari otomatisasi proses build, publish, dan deployment ke lingkungan staging.

A. Otomatisasi Pekerjaan Build

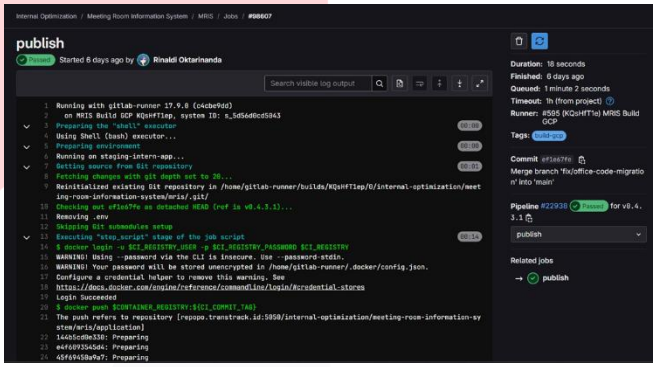
Pekerjaan build merupakan tahapan fundamental dalam pipeline CI/CD, yang bertanggung jawab untuk mengkompilasi kode sumber proyek dan membentuknya menjadi citra Docker aplikasi. Berdasarkan log eksekusi yang terekam, pekerjaan build secara konsisten berhasil diselesaikan dengan status "Passed" . Secara kualitatif, keberhasilan ini menegaskan bahwa kode aplikasi bebas dari kesalahan kompilasi, konflik dependensi, atau error sintaksis yang dapat menghambat proses pembangunan. Hal ini juga mengindikasikan bahwa semua tool dan library yang diperlukan untuk build tersedia dan terkonfigurasi dengan benar di runner. Dari aspek kuantitatif, pekerjaan build menunjukkan efisiensi tinggi dengan durasi eksekusi rata-rata hanya 28 detik. Selain itu, job ini tercatat memiliki waktu antrian 41 detik sebelum dieksekusi oleh Runner #5985, yang diidentifikasi sebagai MRIS Build GCP. Angka-angka ini secara signifikan menunjukkan pengurangan waktu yang dibutuhkan untuk verifikasi awal setiap perubahan kode, yang berdampak langsung pada percepatan siklus umpan balik bagi pengembang.



OTOMATISASI PEKERJAAN BUILD
(A)

B. Otomatisasi Pekerjaan Publish

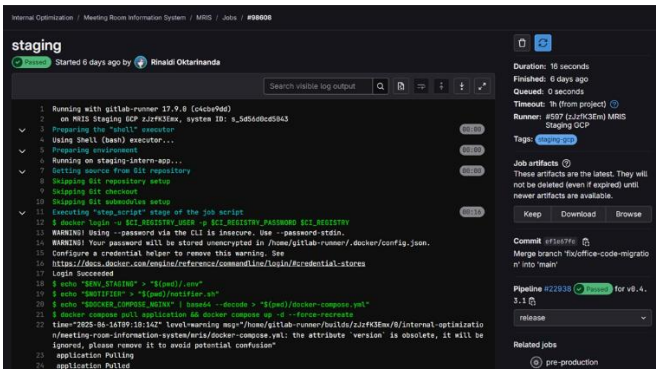
Setelah citra Docker aplikasi berhasil di-build, tahap selanjutnya adalah publikasi citra tersebut ke registry Docker melalui pekerjaan publish. Log eksekusi mengkonfirmasi bahwa pekerjaan publish juga secara konsisten berhasil dengan status "Passed" . Secara kualitatif, ini membuktikan bahwa citra Docker aplikasi yang telah dibangun berhasil diunggah dengan aman ke repositori pusat (registry Docker repo.transtrack.id/internal-optimization/meeting-room-information-system/mris/application), sehingga citra tersebut siap diakses dan ditarik untuk proses deployment selanjutnya. Proses ini melibatkan otentikasi yang sukses ke registry (Login Succeeded) menggunakan kredensial yang dienkripsi melalui variabel CI/CD, serta eksekusi perintah docker push. Dari sisi kuantitatif, pekerjaan publish menunjukkan kinerja yang sangat efisien dengan rata-rata durasi eksekusi 18 detik. Meskipun terdapat waktu antrian 1 menit 2 detik, kecepatan eksekusi aktual pekerjaan ini memastikan ketersediaan citra aplikasi terbaru secara cepat.



OTOMATISASI PEKERJAAN PUBLISH
(B)

C. Otomatisasi Pekerjaan Staging (Deployment)

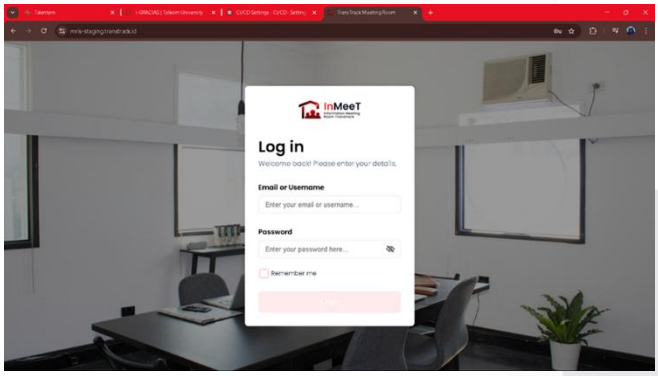
Log eksekusi dengan jelas menunjukkan bahwa pekerjaan ini berhasil diselesaikan dengan status "Passed" . Secara kualitatif, keberhasilan ini merupakan validasi fundamental bahwa aplikasi telah berhasil disebarkan dan dioperasikan di lingkungan staging, yang disiapkan untuk menyerupai lingkungan produksi. Docker login yang sukses, penarikan citra terbaru (docker compose pull), dan startup aplikasi (docker compose up) di server staging yang dikelola oleh Runner #5977 (MRIS Staging GCP). Penggunaan variabel lingkungan spesifik staging (misalnya, $ENV_STAGING) dan injeksi konfigurasi dinamis (seperti konfigurasi Nginx dari $DOCKER_COMPOSE_INGRIX yang didekode) juga berfungsi dengan benar, memastikan aplikasi terkonfigurasi secara tepat dan aman untuk lingkungan staging. Dari aspek kuantitatif, deployment ke staging ini diselesaikan dengan sangat cepat, hanya memerlukan rata-rata 16 detik, dengan catatan efisiensi tinggi pada waktu antrian 0 detik, menunjukkan ketersediaan runner yang optimal.



OTOMATISASI PEKERJAAN STAGING (DEPLOYMENT) (C)

D. Verifikasi Fungsionalitas Aplikasi di Staging

Sebagai puncak dari keseluruhan pipeline CI/CD yang berhasil, aplikasi "InMeET Information Meeting Room System" dapat diakses dan berfungsi penuh, menampilkan halaman login yang responsif melalui URL mris-staging.transtrack.id . Secara kualitatif, ini adalah bukti empiris yang paling jelas dan tangible dari keberhasilan seluruh proses CI/CD. Hal ini menegaskan bahwa setiap tahapan dalam pipeline, mulai dari commit kode (terkait dengan commit ef1e077e dari pipeline #22938) hingga deployment aplikasi, telah berfungsi dengan sempurna, menghasilkan aplikasi yang berjalan di lingkungan yang dapat diakses pengguna. Keberhasilan ini menandakan bahwa aplikasi sudah siap untuk menjalani pengujian fungsional yang lebih ekstensif dan User Acceptance Testing (UAT) oleh tim Quality Assurance (QA) atau pemangku kepentingan bisnis di lingkungan yang representatif sebelum dipertimbangkan untuk rilis ke lingkungan produksi.



VERIFIKASI FUNGSIONALITAS APLIKASI DI STAGING (D)

V. KESIMPULAN

Implementasi sistem Continuous Integration/Continuous Delivery (CI/CD) pada proyek Meeting Room Information System (MRIS) telah berhasil secara komprehensif mencapai seluruh tujuan yang telah ditetapkan dalam penelitian ini. Otomatisasi penuh pada proses build, publish, dan deployment aplikasi ke lingkungan staging secara fundamental meningkatkan efisiensi operasional dan kualitas produk. Hal ini secara signifikan memangkas siklus pengembangan aplikasi, secara dramatis mengurangi waktu delivery dari potensi hitungan jam atau hari menjadi hanya hitungan detik untuk setiap commit kode. Keberhasilan pipeline ini secara empiris divalidasi oleh status "Passed" yang konsisten pada job build (rata-rata 28 detik), publish (rata-rata 18 detik), dan deployment ke staging (rata-rata 16 detik). Puncak dari keberhasilan ini adalah kemampuan untuk mengakses dan memverifikasi aplikasi yang berfungsi penuh pada URL lingkungan staging, menampilkan halaman login yang responsif. Pencapaian ini secara efektif meningkatkan efisiensi tim pengembang dengan membebaskan mereka dari tugas-tugas repetitif yang rentan kesalahan manual, mempercepat respons terhadap perubahan kebutuhan bisnis melalui deployment yang gesit dan andal, serta memperkuat kolaborasi tim dengan menyediakan alur kerja yang transparan, otomatis, dan terpadu.

REFERENSI

Direkomendasikan menggunakan *reference management tools* (*mendeley*), format style menggunakan IEEE. Contoh penulisan referensi IEEE Style:

[1] H. P. D. Gohae, F. Anglhoma, J. Khu and E. Indra, "Rancang Bangun Aplikasi E-Booking Ruangan dan Zoom Meeting Premium Berbasis Web Dengan Pendekatan Design Thinking," Jurnal Ekonomi Manajemen Sistem Informasi (JEMSI), vol. 6, no. 1, pp. 494-504, 2024.

[2] A. Raheem, A. M. Osilaja, I. Kolawole and V. E. Essien, "Exploring Continuous Integration and deployment strategies for streamlined DevOps processes in software engineering practices," World Journal of Advanced Research and Review, vol. 24, no. 3, pp. 2813-2830, 2024.

[3] A. Chava, "CI/CD and Automation in DevOps Engineering," Asian Journal of Research in Computer Science, vol. 17, no. 11, pp. 73-80, 2024.

[4] H. Ho-Dac and L. V. Van, "An Approach to Enhance CI/CD Pipeline with Open-Source Security Tools," European Modern Studies Journal, vol. 8, no. 3, pp. 408-413, 2024.

[5] 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.

[6] Simplilearn, "Apa itu Kontainerisasi?," Simplilearn, 9 Juni 2025. [Online]. Available: https:// [Accessed 2025 Juni 22].

[7] R. Patria, "Visual Studio Code Pengertian, Fitur, Kelebihannya Lengkap!," DomaiNesia, 28 Juni 2023. [Online]. Available: https://www.domainesia.com/berita/visual-studio-code/#:~:text=mulai%20membahas%20bersama!-,Visual%20Studio%20Code%20adalah%20Code%20Editor,%2C%20Windows%2C%20dan%20juga%20Linux.. [Accessed 22 Juni 2025].

[8] R. Setiawan, "Apa Itu CI/CD? IT Developer Harus Tau," dicoding, 27 Oktober 2021. [Online]. Available: https://www.dicoding.com/blog/apa-itu-ci-cd/. [Accessed 22 Juni 2025].

[9] Meiliaeka, "Simak Pengertian, Sejarah, Kelebihan dan Kekurangan Linux Ubuntu," Telkom University, 4 Mei

2023. [Online]. Available: https://it.telkomuniversity.ac.id/simak-pengertian-sejarah-kelebihan-dan-kekurangan- linux-ubuntu/. [Accessed 22 Juni 2025].

[10] Zenarmor, "Introduction to OpenVPN," Zenarmor, 12 Oktober 2023. [Online]. Available: https://www.zenarmor.com/docs/network-security-tutorials/what-is-openvpn. [Accessed 22 Juni 2025].