BLUE-GREEN DEPLOYMENT MENGGUNAKAN *CONTAINERIZATION* UNTUK MENCAPAI *ZERO DOWNTIME* PADA *WEBSITE* BERORIENTASI SCRUM

SKRIPSI

Restu Bumi Ryan Ramadhan

20210040006

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK, KOMPUTER DAN DESAIN

UNIVERSITAS NUSA PUTRA

SUKABUMI

2025

BLUE-GREEN DEPLOYMENT MENGGUNAKAN *CONTAINERIZATION* UNTUK MENCAPAI *ZERO DOWNTIME* PADA *WEBSITE* BERORIENTASI SCRUM

SKRIPSI

Diajukan Untuk Memenuhi Salah Satu Syarat Dalam Menempuh
Gelar Sarjana Komputer

Restu Bumi Ryan Ramadhan
20210040006

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK, KOMPUTER DAN DESAIN
UNIVERSITAS NUSA PUTRA
SUKABUMI
2025

PERNYATAAN PENULIS

JUDUL	:	*BLUE-GREEN DEPLOYMENT* MENGGUNAKAN *CONTAINERIZATION* UNTUK MENCAPAI *ZERO DOWNTIME* PADA *WEBSITE* BERORIENTASI SCRUM
NAMA	:	RESTU BUMI RYAN RAMADHAN
NIM	:	20210040006

Saya menyatakan dan bertanggungjawab dengan sebenarnya bahwa Skripsi ini adalah hasil karya sendiri kecuali cuplikan dan ringkasan yang masing-masing telah saya jelaskan sumbernya. Jika pada waktu selanjutnya ada pihak lain yang mengklaim bahwa Skripsi ini sebagai karyanya, yang disertai dengan bukti-bukti yang cukup, maka saya bersedia untuk dibatalkan gejar Sarjana Komputer saya beserta segala hak dan kewajiban yang melekat pada gelar tersebut.

Sukabumi, 15 Juli 2025

RESTU BUMI RYAN RAMADHAN

Penulis

PENGESAHAN SKRIPSI

JUDUL	:	*BLUE-GREEN DEPLOYMENT* MENGGUNAKAN *CONTAINERIZATION* UNTUK MENCAPAI *ZERO DOWNTIME* PADA *WEBSITE* BERORIENTASI SCRUM
NAMA	:	RESTU BUMI RYAN RAMADHAN
NIM	:	20210040006

Skripsi ini telah diujikan dan dipertahankan di depan Dewan Penguji pada Sidang Skripsi tanggal 15 Juli 2025 Menurut pandangan kami, Skripsi ini memadai dari segi kualitas untuk tujuan penganugerahan gelar Sarjana Komputer (S.Kom)

Sukabumi, 15 Juli 2025

Pembimbing I	Pembimbing II
Alun Sujjada, S.Kom, M.T	**Anggun Fergina, M.Kom**
NIDN. 0718108001	NIDN. 0407029301
Ketua Penguji	Ketua Program Studi

Ir Somantri, S.T, M.Kom	**Ir Somantri, S.T, M.Kom**
NIDN. 0419128801	NIDN. 0419128801

PLH. Dekan Fakultas Teknik, Komputer dan Desain

Ir. Paikun, S,T., IPM., ASEAN Eng
NIDN. 0402037401

ABSTRACT

The Blue-Green Deployment strategy is an effective approach to achieving zero downtime during system updates. This study aims to implement the strategy using Docker and Traefik-based containerization in the development of a Scrum-oriented website. A mixed method approach was employed, involving literature reviews, observations of practices at PT Sineas Kreatif Indonesia*, and source code analysis of the deployment process.*

The results show that implementing Blue-Green Deployment across two VPS using Docker Swarm and Traefik successfully achieved zero downtime by enabling automatic traffic switching between environments. The CI/CD pipeline built with GitHub Actions effectively prevents failed images from reaching production, maintaining system stability. Monitoring tools such as Uptime Kuma, Grafana, Prometheus, and Loki ensured faster issue detection and resolution. The integration of Cloudflare Zero Trust also enhanced the security of internal access. Overall, this approach offers significant improvements in speed, security, and scalability compared to conventional deployment methods, despite its higher architectural complexity.

Keywords: Blue-Green Deployment, Containerization, Docker, *Traefik, Zero Downtime,* CI/CD*, Virtual Private Server* (VPS)

ABSTRAK

Strategi *Blue-Green Deployment* merupakan pendekatan efektif untuk mencapai *zero downtime* dalam proses pembaruan sistem. Penelitian ini bertujuan untuk mengimplementasikan strategi tersebut dengan menggunakan *containerization* berbasis Docker dan Traefik dalam pengembangan *website* yang berorientasi Scrum. Metode yang digunakan adalah pendekatan *mixed method*, dengan pengumpulan data melalui studi pustaka, observasi terhadap praktik di PT Sineas Kreatif Indonesia, serta analisis *source code deployment*.

Hasil penelitian menunjukkan bahwa penerapan *Blue-Green Deployment* pada dua VPS menggunakan Docker Swarm dan Traefik berhasil mencapai *zero downtime* dengan memanfaatkan pengalihan trafik antar lingkungan secara otomatis. CI/CD *pipeline* yang dibangun dengan GitHub Actions terbukti mencegah *image* gagal masuk ke *production* dan menjaga stabilitas sistem. *Monitoring* dengan Uptime Kuma, Grafana, Prometheus, dan Loki mempercepat deteksi serta respons terhadap gangguan. Integrasi Cloudflare Zero Trust juga menambah keamanan akses internal. Secara keseluruhan, pendekatan ini memberikan peningkatan signifikan dalam keamanan, kecepatan, dan skalabilitas dibanding metode konvensional, meski dengan kompleksitas yang lebih tinggi.

Kata Kunci: *Blue-Green Deployment*, *Containerization*, Docker, Traefik, *Zero Downtime*, CI/CD *Pipeline*, *Virtual Private Server* (VPS)

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul:
"*Blue-Green Deployment* Menggunakan *Containerization* untuk Mencapai *Zero Downtime* pada *Website* Berorientasi Scrum"
Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Informatika, Fakultas Teknik, Komputer, dan Desain, Universitas Nusa Putra.
Penelitian ini bertujuan untuk mengimplementasikan strategi *Blue-Green Deployment* dengan pendekatan *containerization* dalam rangka mengurangi *downtime* saat pembaruan aplikasi, khususnya pada pengembangan perangkat lunak berbasis Scrum. Harapannya, skripsi ini dapat menjadi referensi teknis maupun akademis dalam praktik DevOps modern, serta memberikan kontribusi nyata dalam peningkatan kualitas *deployment* aplikasi *web*.
Dalam proses penyusunan skripsi ini, penulis mendapatkan dukungan dari banyak pihak. Oleh karena itu, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua penulis, Eni Nuraeni dan Maryata, beserta seluruh keluarga yang selalu memberikan doa, dukungan, kasih sayang, dan semangat yang tiada henti dalam setiap langkah perjuangan penulis.
2. Bapak Dr. H. Kurniawan, ST., M.Si., MM., selaku Rektor Universitas Nusa Putra yang telah memberikan dukungan dalam pelaksanaan kegiatan akademik.
3. Bapak Anggy Pradiftha Junfitharana, S.Pd., MT., selaku Wakil Rektor I Bidang Akademik Universitas Nusa Putra, atas arahan dan kebijakan yang memudahkan proses studi.
4. Bapak Ir. Somantri, ST., M.Kom., selaku Ketua Program Studi Teknik Informatika, atas segala bimbingan dan motivasinya selama masa perkuliahan.

5. Bapak Alun Sujjada, S.Kom., MT., selaku Dosen Pembimbing I yang telah memberikan bimbingan teknis, dorongan, serta arahan sejak awal hingga akhir penulisan skripsi ini.
6. Ibu Anggun Fergina, M.Kom., selaku Dosen Pembimbing II yang telah memberikan masukan yang sangat bermanfaat dalam menyempurnakan isi dan struktur penulisan.
7. PT. Sineas Kreatif Indonesia yang telah memberikan izin dan kepercayaan kepada penulis untuk menjadikan salah satu proyeknya sebagai objek penelitian, sehingga penulis dapat mengaplikasikan konsep *Blue-Green Deployment* secara nyata dalam lingkungan industri.
8. Rekan-rekan mahasiswa Program Studi Teknik Informatika Universitas Nusa Putra yang senantiasa memberikan semangat dan kebersamaan dalam menyelesaikan studi.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan untuk perbaikan di masa yang akan datang. Semoga skripsi ini dapat memberikan manfaat bagi pembaca, khususnya di bidang Teknologi Informasi dan pengembangan perangkat lunak modern.

Sukabumi, 5 Juli 2025

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik UNIVERSITAS NUSA PUTRA, saya yang bertanda tangan di bawah ini:

Nama : Restu Bumi Ryan Ramadhan

NIM 20210040006

Program Studi : Teknik Informatika

Jenis karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Nusa Putra **Hak Bebas Royalti Non Eksklusif (*Non-exclusive Royalty- Free Right*)** atas karya ilmiah saya yang berjudul :

"BLUE-GREEN DEPLOYMENT MENGGUNAKAN *CONTAINERIZATION* UNTUK MENCAPAI *ZERO DOWNTIME* PADA *WEBSITE* BERORIENTASI SCRUM"

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Nusa Putra berhak menyimpan, mengalih media/format- kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : SUKABUMI

Pada tanggal : 15 Juli 2025

Yang menyatakan

(Restu Bumi Ryan Ramadhan)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

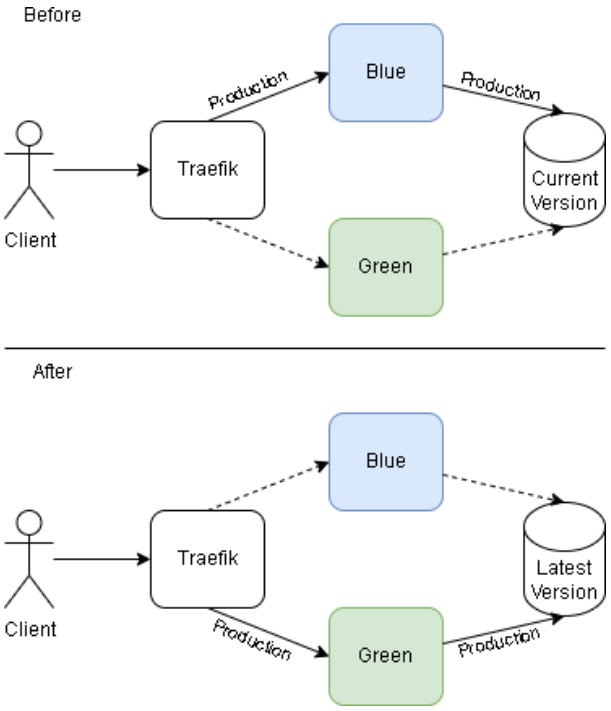
PT. Sineas Kreatif Indonesia adalah perusahaan yang bergerak di bidang teknologi serta Film Media *Production* dan *Creative Agency*, dengan produk unggulannya berupa *platform* OTT (*Over the Top*) untuk penayangan produksi film lokal karya mahasiswa di Indonesia. Sebagai layanan berbasis digital, *platform* ini harus memastikan ketersediaan layanan (*availability*) yang tinggi agar pengguna dapat mengakses konten tanpa gangguan. *Availability* merupakan salah satu faktor krusial dalam pengalaman pengguna dan keberhasilan bisnis, terutama dalam industri yang mengandalkan akses daring secara kontinu. Dalam standar ISO/IEC 25010, *availability* dikategorikan sebagai bagian dari *reliability*, yang didefinisikan sebagai tingkat di mana suatu sistem tetap operasional dan dapat diakses saat dibutuhkan. Selain itu, *reliability* mencakup elemen seperti *faultlessness* (kemampuan sistem untuk beroperasi tanpa kesalahan), *fault tolerance* (kemampuan sistem untuk tetap berfungsi meskipun terdapat kesalahan perangkat keras atau perangkat lunak), serta *recoverability* (kemampuan sistem untuk memulihkan data dan keadaan yang diinginkan setelah gangguan atau kegagalan terjadi) [1].

Idealnya, setiap proses pembaruan atau *deployment* aplikasi dilakukan tanpa mengganggu layanan yang sedang berjalan. Hal ini sangat relevan bagi aplikasi berbasis *web*, yang dirancang untuk mendukung aktivitas pengguna secara terus-menerus dengan tuntutan tinggi terhadap stabilitas, aksesibilitas, dan ketersediaan layanan (*availability*). Namun, dalam praktiknya, banyak perusahaan *startup* menghadapi tantangan berupa gangguan atau *downtime* selama proses pembaruan, yang dapat berdampak buruk pada produktivitas pengguna dan menimbulkan kerugian finansial maupun reputasi [2].

Downtime dalam proses *deployment* dapat berdampak negatif pada pengalaman pengguna, terutama ketika sebuah aplikasi atau layanan daring tidak dapat diakses saat pembaruan sedang dilakukan. Sebuah *deployment website* yang ideal seharusnya tidak menyebabkan gangguan bagi pengguna akhir, di mana versi lama

tetap berjalan hingga versi baru siap sepenuhnya, sehingga lalu lintas dapat dialihkan secara lancar ke versi terbaru tanpa interupsi atau gangguan layanan [3]. Ketika *deployment* tidak dirancang untuk menghindari *downtime*, pengguna dapat mengalami keterlambatan akses, gangguan dalam penggunaan aplikasi, atau bahkan ketidakmampuan untuk mengakses layanan sama sekali, yang dapat berdampak pada kepuasan pelanggan dan keberhasilan bisnis. *Downtime* ini umumnya disebabkan oleh proses *deployment* yang dilakukan langsung pada lingkungan *production*, tanpa isolasi atau replikasi versi aplikasi, sehingga layanan harus dihentikan sementara saat *update* berlangsung. Untuk menjawab tantangan ini, *Continuous Integration/Continuous Deployment* (CI/CD) telah menjadi salah satu pendekatan utama dalam pengembangan perangkat lunak. CI/CD memungkinkan proses pengembangan, pengujian, dan *deployment* aplikasi berjalan secara otomatis dan efisien [4]. Namun, penerapan CI/CD yang benar-benar menjamin *zero downtime* seringkali menghadapi berbagai kendala, mulai dari kompleksitas implementasi hingga keterbatasan infrastruktur. Salah satu solusi yang dikenal efektif adalah pendekatan *blue-green deployment* [3]. Dengan metode ini, dua versi aplikasi berupa versi lama dan baru yang dapat berjalan bersamaan, sehingga versi baru dapat diuji secara langsung tanpa mengganggu versi lama yang masih melayani pengguna [3].

Pendekatan *blue-green deployment* umumnya diterapkan pada infrastruktur dengan *horizontal scaling*, di mana beban kerja didistribusikan ke beberapa *instance* layanan untuk meningkatkan kapasitas dan keandalan. Pada awalnya, metode ini dapat diadaptasi oleh perusahaan *startup* dengan sumber daya terbatas melalui penggunaan infrastruktur sederhana seperti satu *Virtual Private Server* (VPS). Namun, seiring dengan meningkatnya kebutuhan sistem dan kompleksitas layanan, penerapan *blue-green deployment* juga dapat dikembangkan lebih lanjut dengan menambahkan VPS kedua untuk memisahkan beban kerja secara fungsional seperti memisahkan *server* aplikasi dari *server* layanan eksternal seperti metrik, ataupun *database*. Dengan demikian, perusahaan *startup* tetap dapat mempertahankan prinsip *zero downtime deployment* secara efisien, tanpa harus langsung beralih ke arsitektur skala besar berbasis *container orchestration* seperti Kubernetes [5].



Gambar 1. 1 Konsep *Blue Green Deployment*

Untuk mengatasi tantangan dalam pengelolaan *pipeline* yang melibatkan integrasi Docker dan Traefik, penelitian ini menawarkan solusi dengan pendekatan yang lebih adaptif terhadap keterbatasan sumber daya, sambil tetap mendukung praktik *blue-green deployment*. Penelitian sebelumnya oleh Taufik Irfan (2024) berjudul "*Performa Quality of Service* (QoS) melalui *Implementasi Blue-Green Deployment* pada Infrastruktur *Multi-Server*" menunjukkan efektivitas strategi ini pada lingkungan *multi-server* dengan HAProxy sebagai *load balancer*. Dalam penelitian tersebut, pengujian dilakukan terhadap parameter QoS seperti *throughput, response time, dan packet loss* di Google Cloud Platform (GCP). HAProxy berfungsi untuk membagi lalu lintas antara server aktif dan *server* cadangan selama proses *deployment*, memastikan layanan tetap tersedia tanpa *downtime*. Namun, pendekatan tersebut menuntut arsitektur yang kompleks serta investasi infrastruktur yang cukup tinggi [6], [7].

Berbeda dari penelitian tersebut, penelitian ini menggunakan pendekatan berbasis Docker dan Traefik, yang lebih cocok untuk perusahaan *startup* dengan infrastruktur yang awalnya sederhana namun berkembang menjadi dua *Virtual Private Server* (VPS) karena meningkatnya kebutuhan layanan. Docker memungkinkan pengelolaan aplikasi yang modular melalui kontainerisasi,

sedangkan Traefik, sebagai *application proxy* dan *load balancer*, menyederhanakan proses distribusi lalu lintas antar-*container* secara otomatis [8]. Dengan mengadopsi pendekatan ini, perusahaan *startup* dapat merasakan manfaat dari *blue-green deployment* tanpa memerlukan investasi besar dalam infrastruktur *multi-server*. Selain itu, integrasi Docker dan Traefik memungkinkan proses *deployment* yang lebih efisien dengan mendeteksi otomatis layanan baru, meminimalkan risiko konfigurasi manual yang rumit. Penelitian ini juga menitikberatkan pada keamanan *pipeline*, termasuk pengelolaan *environment variables*, serta memastikan validitas pengujian untuk menjaga stabilitas sistem secara keseluruhan selama proses *deployment*. Dengan menggabungkan pendekatan *blue-green deployment* dan Traefik sebagai *application proxy* modern pada infrastruktur dua VPS, penelitian ini bertujuan untuk mengembangkan *pipeline* CI/CD yang andal, efisien, dan mampu menjawab kebutuhan tersebut.

Berdasarkan latar belakang yang telah disampaikan, penelitian ini bertujuan untuk mengeksplorasi penerapan *pipeline* CI/CD dengan pendekatan *blue-green deployment*, dengan memanfaatkan integrasi Docker dan Traefik dalam *environment* infrastruktur yang telah berkembang dari satu menjadi dua *Virtual Private Server* (VPS). Penelitian ini bertujuan untuk mengembangkan solusi yang efektif dalam mengurangi *downtime* serta memastikan stabilitas dan keamanan data pada proses deployment aplikasi *web*. Adapun judul penelitian ini adalah: *"Blue-Green Deployment* Menggunakan *Containerization* untuk Mencapai *Zero Downtime* pada *Website* Berorientasi Scrum.*"*

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana perancangan *blue-green deployment* menggunakan Docker dan Traefik dapat mendukung proses *deployment* aplikasi dengan *zero downtime*?
2. Bagaimana penerapan dan integrasi Traefik dapat menyederhanakan proses *deployment* serta mendukung pengujian efisiensi dan keamanan?

1.3 Batasan Masalah

Agar penelitian ini lebih terarah, beberapa batasan masalah yang diterapkan adalah sebagai berikut:

1. Penelitian ini berfokus pada penerapan *blue-green deployment* menggunakan Docker dan Traefik untuk mendukung *pipeline* CI/CD dalam infrastruktur yang disesuaikan dengan kebutuhan perusahaan *startup*.
2. Infrastruktur yang digunakan mencakup beberapa *instance* dan layanan yang saling terhubung dalam satu sistem *deployment*, termasuk penggunaan *Docker Swarm* dan pengaturan *cluster*, namun tidak mencakup skenario *multi-server* berskala *enterprise* seperti *high availability* dengan *auto-scaling* penuh.
3. Penelitian ini menitikberatkan pada pengembangan *pipeline* CI/CD yang mampu menjamin *zero downtime* dan efisiensi *deployment* aplikasi, serta pengelolaan keamanan *pipeline* seperti manajemen *environment variables*, integrasi *Cloudflare Zero Trust*, dan proses *debugging*.
4. Validasi pengujian dilakukan pada *website* berorientasi Scrum yang dikembangkan sebagai studi kasus, tanpa mencakup analisis performa secara mendalam di lingkungan *production* berskala besar atau pengukuran parameter *Quality of Service* (QoS) secara formal.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Menganalisis penerapan *blue-green deployment* menggunakan Docker dan Traefik dalam mendukung proses *deployment* aplikasi dengan *zero downtime*.
2. Mengembangkan *pipeline* CI/CD yang efisien, andal, dan sesuai kebutuhan perusahaan melalui integrasi Docker dan Traefik sebagai solusi *deployment* yang terukur dan adaptif terhadap perkembangan infrastruktur.

1.5 Manfaat Penelitian

Hasil penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Bagi Mahasiswa

Penelitian ini dapat menjadi referensi dan ilmu dalam memahami konsep *blue-green deployment*, serta mengimplementasikan *pipeline* CI/CD yang sederhana namun efektif menggunakan Docker dan Traefik. Hal ini membantu mahasiswa mengembangkan keterampilan *hard skill* yang relevan dengan kebutuhan industri.

2. Bagi Perguruan Tinggi

Penelitian ini dapat menjadi tambahan literatur ilmiah untuk mendukung pengembangan kurikulum di bidang DevOps (*Development Operations*), CI/CD (*Continuous Integration* and *Continuous Delivery*/*Continuous Deployment*), dan manajemen infrastruktur. Fokus penelitian pada solusi berbasis sumber daya terbatas juga relevan dalam membantu perguruan tinggi mempersiapkan mahasiswa menghadapi tantangan industri.

3. Bagi Pengembangan Perangkat Lunak

Penelitian ini memberikan wawasan tentang cara membangun pipeline CI/CD yang mendukung *zero downtime* dengan memanfaatkan teknologi modern seperti Docker, Traefik, dan *blue-green deployment*. Hasil penelitian ini diharapkan membantu pengembangan perangkat lunak dalam mengatasi tantangan seperti efisiensi *deployment*, pengelolaan keamanan *pipeline*, serta adaptasi terhadap kebutuhan infrastruktur yang berkembang, baik dalam skala kecil maupun menengah.

1.6 Sistematika Penulisan

Penelitian ini disusun dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas teori-teori yang relevan, seperti konsep *blue-green deployment*, CI/CD *pipeline*, teknologi Docker dan Traefik.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan metode penelitian yang digunakan, termasuk pendekatan penelitian, parameter keberhasilan *pipeline* CI/CD, pengujian sistem, serta alat dan bahan yang digunakan.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini membahas hasil implementasi *blue-green deployment* menggunakan Docker dan Traefik, serta pengujian ketersediaan layanan, efisiensi pipeline, dan efektivitas pengurangan *downtime* selama proses *deployment*.

BAB V PENUTUP

Bab ini berisi kesimpulan dari hasil penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut, khususnya dalam penerapan strategi *zero-downtime deployment* pada infrastruktur berbasis *containerization*.

DAFTAR PUSTAKA

Berisi daftar referensi yang digunakan dalam penelitian ini dengan sumber yang relevan.

LAMPIRAN

Berisi dokumen pendukung penelitian, seperti konfigurasi *pipeline* CI/CD, konfigurasi *containerization*, hasil pengujian, serta diagram arsitektur sistem.

RIWAYAT HIDUP

Berisi informasi pribadi penulis, pengalaman organisasi, pelatihan, seminar atau konferensi yang pernah diikuti, serta publikasi ilmiah yang mendukung kompetensi dan rekam jejak akademik penulis dalam bidang penelitian.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian terhadap penerapan strategi *Blue-Green Deployment* dengan pendekatan *containerization* pada pengembangan *website* Scrum di PT Sineas Kreatif Indonesia, diperoleh beberapa kesimpulan sebagai berikut:

1. Strategi *Blue-Green Deployment* terbukti efektif dalam meminimalisir *downtime*, bahkan mampu mencapai *zero downtime* selama proses *deployment* berlangsung. Dengan memanfaatkan dua *environment* (*blue* dan *green*) yang dijalankan pada dua *host* terpisah (dua VPS), sistem dapat melakukan *switching* versi aplikasi secara otomatis dan tanpa gangguan layanan bagi pengguna.
2. Teknologi *containerization* melalui Docker Swarm dan Traefik berhasil membentuk arsitektur yang modular dan skalabel. Docker Swarm mempermudah orkestrasi layanan dalam *cluster*, sementara Traefik memungkinkan pengelolaan trafik secara dinamis antar lingkungan *blue* dan *green*.
3. *Monitoring* sistem dengan Uptime Kuma dan *observability* seperti Grafana, Prometheus, dan Loki berperan penting dalam menjaga reliabilitas sistem. Pemantauan *real-time* dan notifikasi otomatis terbukti mempercepat respons terhadap gangguan, serta memastikan ketersediaan layanan pasca-*deployment* tetap terjaga.
4. Pengujian terhadap *workflow* GitHub Actions menunjukkan bahwa CI/CD *pipeline* dengan kontrol ketat pada tahap *build* mampu mencegah *image* yang gagal di*deploy* ke *production*, sehingga menjaga kestabilan sistem. Selain itu, metrik performa *pipeline* menunjukkan bahwa meskipun terdapat *error*, tidak satu pun yang berdampak pada *downtime* secara langsung.
5. Penggunaan Cloudflare Zero Trust memberikan lapisan keamanan tambahan pada sistem, khususnya untuk akses internal seperti *dashboard* manajemen Traefik. Dengan autentikasi berbasis organisasi GitHub dan

whitelist email, hanya pengguna terverifikasi yang dapat mengakses area sensitif aplikasi.

6. Dibandingkan dengan pendekatan *two-branch deployment* konvensional, implementasi *Blue-Green Deployment* menunjukkan peningkatan signifikan dalam hal keamanan, kecepatan, skalabilitas, dan kontrol *environment*, meskipun dengan kompleksitas arsitektur yang lebih tinggi.
7. Selama proses pengembangan dan *deployment*, manajemen variabel *secret* telah dilakukan menggunakan fitur GitHub Actions *Repository Secret* untuk menjaga keamanan *pipeline* dan mencegah kebocoran data sensitif.

5.2 Saran

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut antara lain:

1. Perluasan penerapan *monitoring* ke seluruh layanan mikro yang berjalan di dalam *cluster*, termasuk metrik sistem, *log*, dan *health check container*, untuk mendapatkan visibilitas yang lebih komprehensif terhadap performa sistem secara menyeluruh.
2. Meskipun Traefik telah memberikan solusi *routing* dan *observability* yang efisien untuk sistem saat ini, penggunaan *service mesh* seperti Istio atau Linkerd dapat dipertimbangkan pada tahap pengembangan lanjutan, khususnya jika arsitektur berkembang menjadi sistem *microservices* berskala besar. Pendekatan ini dapat membantu meningkatkan pengawasan menyeluruh (*end-to-end observability*), menambah lapisan keamanan untuk komunikasi antar layanan, serta memberikan kontrol yang lebih rinci terhadap alur trafik dalam sistem.
3. Peningkatan dokumentasi dan standar DevOps di internal tim perlu dilakukan, termasuk pembakuan struktur Dockerfile, *pipeline* CI/CD, dan penggunaan *branch*. Hal ini akan mendukung proses kolaborasi antar *developer* serta mempercepat proses *deployment* yang berulang.
4. Eksperimen lebih lanjut terhadap strategi *deployment* lain seperti *Canary Deployment* atau A/B *Testing* dapat dilakukan untuk mengevaluasi keunggulan dan kelemahan relatif terhadap *Blue-Green Deployment*,

khususnya pada skala sistem yang lebih besar atau dengan kebutuhan segmentasi pengguna.

5. Penerapan redundansi infrastruktur (*multi-host deployment*) dapat menjadi langkah berikutnya dalam meningkatkan reliabilitas sistem, terutama untuk menghadapi skenario kegagalan total pada salah satu *node* VPS.

DAFTAR PUSTAKA

[1] M. D. Mulyawan, I. N. S. Kumara, I. B. A. Swamardika, and K. O. Saputra, "Kualitas Sistem Informasi Berdasarkan ISO/IEC 25010: Literature Review," *Maj. Ilm. Teknol. Elektro*, vol. 20, no. 1, p. 15, 2021, doi: 10.24843/mite.2021.v20i01.p02.

[2] J. Bogojeska, I. Giurgiu, G. Stark, and D. Wiesmann, "IBM predictive analytics reduces server downtime," *INFORMS J. Appl. Anal.*, vol. 51, no. 1, pp. 63–75, 2021, doi: 10.1287/INTE.2020.1064.

[3] Chaitanya K. Rudrabhatla, "Comparison of zero downtime based deployment techniques in public cloud infrastructure," no. June, 2020, doi: 10.13140/RG.2.2.14933.84969.

[4] F. Zampetti, S. Geremia, G. Bavota, and M. Di Penta, "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, Sep. 2021, pp. 471–482. doi: 10.1109/ICSME52107.2021.00048.

[5] V. Subrahmanyam *et al.*, "Optimizing horizontal scalability in cloud computing using simulated annealing for Internet of Things," *Meas. Sensors*, vol. 28, no. May, p. 100829, 2023, doi: 10.1016/j.measen.2023.100829.

[6] T. Irfan, F. M. Wijaya, and S. Slameta, "Performa quality of service (QoS) melalui implementasi blue-green deployment pada infrastruktur multiple server," *JITEL (Jurnal Ilm. Telekomun. Elektron. dan List. Tenaga)*, vol. 4, no. 2, pp. 167–176, Aug. 2024, doi: 10.35313/jitel.v4.i2.2024.167-176.

[7] Traefik, "Welcome." Accessed: Jan. 20, 2025. [Online]. Available: https://doc.traefik.io/traefik/

[8] Docker, "What is Docker?" Accessed: Jan. 20, 2025. [Online]. Available: https://docs.docker.com/get-started/docker-overview/

[9] E. R. F. N. U. Antara, I. R. J-, J. Pocket, and S. Vihar, "Advanced Cloud Automation Workflows for CI / CD Pipelines : Tools and Techniques," vol. 2, no. 5, pp. 296–317, 2024.

[10] S. Amgothu, "Innovative CI/CD Pipeline Optimization through Canary and

Blue-Green Deployment," *Int. J. Comput. Appl.*, vol. 186, no. 50, pp. 975–8887, 2024, doi: 10.5120/ijca2024924141.

[11] A. Malhotra, A. Elsayed, R. Torres, and S. Venkatraman, "Evaluate Canary Deployment Techniques Using Kubernetes, Istio, and Liquibase for Cloud Native Enterprise Applications to Achieve Zero Downtime for Continuous Deployments," *IEEE Access*, vol. 12, pp. 87883–87899, 2024, doi: 10.1109/ACCESS.2024.3416087.

[12] F. Sinlae, F. S. Rosyad, F. Nurhidayat, and W. Jannah, "Evolusi Teknologi Web dan Dampaknya Terhadap Masyarakat Digital," *J. Ilmu Multidisplin*, vol. 3, no. 2, pp. 146–154, 2024, doi: https://doi.org/10.38035/jim.v3i2.608.

[13] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, doi: 10.1109/MS.68.

[14] C. Gea, K. J. D. Lase, and M. Syamsudin, "Implementasi Virtual Private Server untuk Mini Hosting," *J. Sains Dan Komput.*, vol. 7, no. 01, pp. 5–9, 2023, doi: 10.61179/jurnalinfact.v7i01.402.

[15] T. Ylonen, "SSH key management challenges and requirements," *10th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2019 - Proc. Work.*, pp. 1–5, doi: 10.1109/NTMS.8763773.

[16] M. R. Yaswinski, M. M. Chowdhury, and M. Jochen, "Linux security: A survey," *IEEE Int. Conf. Electro Inf. Technol.*, vol. May, pp. 357–362, doi: 10.1109/EIT.8834112.

[17] S. B. Vermaa, B. Pandeyb, and B. K. Gupta, "Containerization and its Architectures: A Study," *Adv. Distrib. Comput. Artif. Intell. J.*, vol. 11, no. 4, pp. 395–409, 2022, doi: 10.14201/adcaij.28351.

[18] Docker, "Docker Compose." Accessed: Jan. 20, 2025. [Online]. Available: https://docs.docker.com/compose/

[19] Docker, "Docker Swarm." Accessed: Jan. 20, 2025. [Online]. Available: https://docs.docker.com/engine/swarm/

[20] Docker, "Docker Stack." Accessed: Jan. 20, 2025. [Online]. Available: https://docs.docker.com/reference/cli/docker/stack/

[21] Docker, "Docker Contexts." Accessed: Jan. 20, 2025. [Online]. Available: https://docs.docker.com/engine/manage-resources/contexts/

[22] Docker, “Docker Hub.” Accessed: Jan. 20, 2025. [Online]. Available: https://hub.docker.com/

[23] Traefik, “Quick Start.” Accessed: Jan. 25, 2025. [Online]. Available: https://doc.traefik.io/traefik/getting-started/quick-start/

[24] Traefik, “Overview.” Accessed: Jan. 25, 2025. [Online]. Available: https://doc.traefik.io/traefik/routing/overview/

[25] Git, “Documentation.” Accessed: Jan. 29, 2025. [Online]. Available: https://git-scm.com/about/branching-and-merging

[26] GitHub, “About GitHub and Git.” Accessed: Jan. 29, 2025. [Online]. Available: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

[27] O. Bashiru and O. G. Olufemi, “An Enhanced CICD Pipeline: A DevSecOps Approach,” *Int. J. Comput. Appl.*, vol. 184, no. 48, pp. 8–13, 2023, doi: 10.5120/ijca2023922594.

[28] J. S. Schwaber, Ken, “Scrum Definition.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#scrum-definition

[29] J. S. Schwaber, Ken, “Product Backlog.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#product-backlog

[30] J. S. Schwaber, Ken, “Sprint Planning.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#sprint-planning

[31] J. S. Schwaber, Ken, “Daily Scrum.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#daily-scrum

[32] J. S. Schwaber, Ken, “Sprint Backlog.” [Online]. Available: https://scrumguides.org/scrum-guide.html#sprint-backlog

[33] J. S. Schwaber, Ken, “Sprint Review.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#sprint-review

[34] J. S. Schwaber, Ken, “Sprint Retrospective.” Accessed: Jan. 29, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html#sprint-retrospective