Jurnal Publikasi Sistem Informasi dan Manajemen Bisnis
Volume.4, Number. 3 September 2025
e-ISSN: 2808-8980; p-ISSN: 2808-9383, Pages 360-371
DOI: https://doi.org/10.55606/jupsim.v4i3.5398
Tersedia : https://journalcenter.org/index.php/jupsim

Evaluasi AWS CodeDeploy dalam Meningkatkan Efisiensi *Deployment* Website Personal Image Studio (PIS) *Japan Career* Berbasis *Cloud*

Maria Amsilia Koza[1*]
[1] Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Flores, Indonesia
[*]*Penulis Koresspondensi: kozamaria78@gmail.com*[1]

Abstract. *This study evaluates the effectiveness of AWS CodeDeploy in enhancing the efficiency and reliability of website deployment for Personal Image Studio (PIS) Japan Career by utilizing* [cloud]*-based infrastructure. Manual deployment processes often involve multiple steps, are time-consuming, and highly susceptible to human error, which can lead to service disruption and operational inefficiencies. To address this issue, a quantitative experimental method was conducted by comparing two deployment approaches: traditional manual deployment and automated deployment using AWS CodeDeploy. The evaluation focused on two primary indicators: deployment time and system reliability. The results revealed a significant improvement in time efficiency, where automated deployment reduced the average deployment duration from 240 minutes to just 33 minutes — an efficiency gain of 86.25%. In terms of reliability, automated deployment achieved an 80% success rate, doubling the 40% success rate observed in manual deployment processes. Moreover, AWS CodeDeploy offers several advanced features such as automatic rollback on failure, seamless integration with monitoring services like AWS CloudWatch, and support for continuous deployment pipelines. These capabilities not only minimize downtime and configuration errors but also ensure consistent and scalable application delivery. Overall, the findings demonstrate that AWS CodeDeploy is a powerful tool that significantly improves the performance and dependability of cloud-based deployment workflows for modern web applications.*

Keywords: *AWS CodeDeploy; Automated Deployment; Cloud Computing; Efficiency; Reliability*

Abstrak. Penelitian ini mengevaluasi efektivitas AWS CodeDeploy dalam meningkatkan efisiensi dan keandalan proses deployment situs web untuk Personal Image Studio (PIS) Japan Career dengan memanfaatkan infrastruktur berbasis cloud. Proses deployment secara manual sering kali melibatkan banyak tahapan, memakan waktu lama, dan sangat rentan terhadap kesalahan manusia, yang dapat menyebabkan gangguan layanan dan inefisiensi operasional. Untuk mengatasi permasalahan ini, digunakan metode eksperimen kuantitatif dengan membandingkan dua pendekatan deployment: secara manual dan otomatis menggunakan AWS CodeDeploy. Evaluasi difokuskan pada dua indikator utama: waktu deployment dan keandalan sistem. Hasil penelitian menunjukkan peningkatan efisiensi waktu yang signifikan, di mana deployment otomatis berhasil memangkas rata-rata waktu dari 240 menit menjadi hanya 33 menit — peningkatan efisiensi sebesar 86,25%. Dari sisi keandalan, deployment otomatis mencapai tingkat keberhasilan 80%, dua kali lipat dari metode manual yang hanya mencapai 40%. Selain itu, AWS CodeDeploy menawarkan berbagai fitur unggulan seperti rollback otomatis saat terjadi kegagalan, integrasi dengan layanan monitoring seperti AWS CloudWatch, serta dukungan pipeline deployment berkelanjutan. Kemampuan ini secara signifikan mengurangi downtime dan kesalahan konfigurasi, serta memastikan distribusi aplikasi yang konsisten dan skalabel. Temuan ini menunjukkan bahwa AWS CodeDeploy merupakan solusi yang efektif dan andal untuk deployment aplikasi berbasis cloud.

Kata kunci: AWS CodeDeploy; Cloud Computing; Deployment Otomatis; Efisiensi; Keandalan

1. LATAR BELAKANG

Perkembangan teknologi informasi mendorong organisasi untuk menyediakan layanan berbasis web yang cepat, andal, serta responsif terhadap kebutuhan pengguna. Salah satu tantangan utama dalam praktiknya adalah tahap *deployment* aplikasi, yaitu proses pemindahan sistem dari lingkungan pengembangan ke lingkungan produksi. Menurut (Shahin et al., 2017), *deployment pipeline* memegang peran penting karena memungkinkan organisasi merilis perangkat lunak secara berkesinambungan dengan tetap menjaga kualitas.

Naskah Masuk: Juli 16, 2025; Revisi: Agustus 01, 2025; Diterima: September 25, 2025; Tersedia : September 29, 2025

Namun, penerapan *continuous deployment* tidak lepas dari sejumlah kendala, di antaranya masalah reliabilitas pada rilis dengan frekuensi tinggi, keterbatasan arsitektur aplikasi, serta kebutuhan koordinasi dan transparansi tim. Jika tantangan ini tidak dikelola dengan baik, kontinuitas layanan kepada pengguna akhir dapat terganggu. Untuk itu, beberapa pendekatan seperti otomasi pengujian, perbaikan desain arsitektur menuju microservices, serta strategi rilis bertahap seperti *canary deployment* diusulkan guna meminimalkan risiko kegagalan layanan (Proulx et al., 2018).

Metode deployment manual memerlukan intervensi manusia di setiap tahap, mulai dari mengunduh kode, memperbarui dependensi, hingga me-restart layanan. Proses ini sangat memakan waktu, melelahkan, dan rawan kesalahan, misalnya kegagalan instalasi dependensi, masalah koneksi server, atau konfigurasi yang salah saat me-restart aplikasi. Akibatnya, waktu implementasi menjadi tidak menentu, produktivitas pengembang menurun, dan sistem sulit merespons cepat ketika terjadi keadaan darurat. Kondisi ini menunjukkan bahwa deployment manual berpotensi mengganggu kontinuitas layanan dan menurunkan keandalan sistem (Hyun et al., 2024). (Pagels et al., 2015) menemukan bahwa banyak perusahaan yang masih mengandalkan tahapan manual dalam proses deployment menghadapi hambatan berupa keterlambatan rilis dan kualitas layanan yang tidak konsisten. Untuk mengatasi hal tersebut, otomatisasi dalam pipeline deployment dinilai penting karena dapat mengurangi pekerjaan manual, mempercepat waktu rilis, serta meningkatkan efisiensi dan stabilitas sistem.

Kemunculan teknologi cloud computing telah membawa perubahan mendasar dalam strategi pengelolaan aplikasi. Melalui layanan yang ditawarkan penyedia seperti Amazon Web Services (AWS), organisasi dapat memanfaatkan berbagai fasilitas untuk mendukung otomatisasi, termasuk AWS CodeDeploy. Layanan ini memungkinkan proses deployment dilakukan secara otomatis, cepat, dan konsisten, sehingga mengurangi risiko kesalahan konfigurasi yang sering muncul pada metode manual (Andriani, 2013). Hasil penelitian oleh (Gadani & Devan, 2024) menunjukkan bahwa penerapan praktik Continuous Integration/Continuous Deployment (CI/CD) dengan dukungan AWS CodePipeline, CodeBuild, dan CodeDeploy mampu meningkatkan efisiensi pengiriman, mempercepat waktu rilis, serta menjaga stabilitas dan kualitas sistem perangkat lunak.

Namun, penelitian yang secara khusus mengevaluasi kinerja AWS CodeDeploy dalam konteks peningkatan efisiensi dan keandalan masih relatif terbatas. Kebaruan penelitian ini terletak pada analisis empiris perbandingan metode *deployment* manual dan otomatis dengan menggunakan AWS CodeDeploy pada studi kasus website Personal Image Studio (PIS) Japan Career.

Penelitian ini memberikan kontribusi praktis dalam menunjukkan efektivitas layanan *cloud* AWS untuk mendukung organisasi kecil hingga menengah yang membutuhkan solusi *deployment* efisien dan andal.

Tujuan dari penelitian ini adalah untuk mengevaluasi peran AWS CodeDeploy dalam meningkatkan efisiensi waktu dan keandalan proses *deployment* aplikasi berbasis *cloud*. Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi teoretis pada pengembangan kajian CI/CD sekaligus kontribusi praktis bagi organisasi yang mengadopsi layanan *cloud* dalam pengelolaan sistem informasinya.

2. KAJIAN TEORITIS

Efisiensi dan keandalan merupakan dua aspek utama dalam menilai kualitas deployment aplikasi. Efisiensi tercermin dari kemampuan sistem melakukan implementasi secara cepat dengan sumber daya minimal, seperti ditunjukkan oleh (Saintikom et al., 2024) yang menemukan bahwa penerapan CI/CD berbasis Kubernetes dan Jenkins lebih efektif dibanding metode manual karena mampu mengotomatisasi proses serta mengurangi kesalahan konfigurasi. Di sisi lain, keandalan berkaitan dengan probabilitas sistem berfungsi dengan benar dalam periode tertentu, sebagaimana ditunjukkan oleh (Wira Pradipta et al., 2024) yang menyatakan bahwa kualitas sistem dan informasi berpengaruh signifikan terhadap kepuasan pengguna, di mana faktor keamanan dan konsistensi informasi menjadi indikator penting dari keandalan aplikasi.

Kemunculan cloud computing telah mendorong perubahan signifikan dalam strategi pengelolaan sistem dan aplikasi modern. Layanan AWS, seperti CodePipeline, CodeBuild, CodeDeploy, CloudFormation, dan Elastic Beanstalk, menyediakan infrastruktur yang elastis, skalabel, serta hemat biaya, memungkinkan organisasi melakukan deployment aplikasi secara otomatis, cepat, dan konsisten. Integrasi antar layanan tersebut mendukung penerapan CI/CD dan *Infrastructure as Code*, sehingga pipeline deployment menjadi lebih efisien sekaligus mengurangi potensi kesalahan konfigurasi (Gadani & Devan, 2024). Menurut (Cahya Kurniawan et al., 2020) cloud computing memiliki lima karakteristik utama, yaitu *on-demand self-service, broad network access, resource pooling, rapid elasticity,* dan *measured service* yang menjadikannya fleksibel serta dapat diukur sesuai kebutuhan pengguna.

AWS menyediakan beragam layanan deployment yang dirancang untuk mengotomatisasi proses pengelolaan aplikasi di lingkungan cloud. Salah satu layanan utamanya adalah AWS CodeDeploy, yaitu layanan yang mengoordinasikan proses deployment aplikasi ke Amazon EC2, AWS Lambda, maupun server on-premises secara otomatis. CodeDeploy mendukung strategi seperti *in-place deployment*, *blue-green*

deployment, hingga *rolling update*, sehingga meminimalkan downtime dan mengurangi risiko kegagalan konfigurasi (AWS, 2025). Amazon EC2 digunakan sebagai infrastruktur komputasi inti dalam sistem, dengan konfigurasi *auto scaling* di beberapa zona ketersediaan untuk menjaga *high availability*. Database MySQL ditempatkan di dalam instance EC2 sehingga sistem tetap dapat diakses dengan performa yang stabil meski jumlah pengguna meningkat (Lukita & Setyo Utomo, 2024).

Dalam pengembangan perangkat lunak modern, AWS CodeDeploy berperan penting dalam mendukung praktik CI/CD melalui deployment otomatis, rollback saat terjadi kegagalan, serta strategi *rolling updates*, *canary*, dan *blue/green deployments* yang memungkinkan *zero-downtime deployment* (AWS, 2025). Untuk infrastruktur komputasi, Amazon EC2 menyediakan server virtual dengan penskalaan otomatis sehingga sistem tetap andal meskipun terjadi peningkatan jumlah pengguna (Lukita & Setyo Utomo, 2024). Sementara itu, Amazon S3 digunakan sebagai penyimpanan objek yang andal dan berketersediaan tinggi untuk menyimpan artefak deployment maupun data aplikasi (Syed, 2025). Guna mendukung monitoring, AWS CloudWatch menyediakan metrik, log, dan alarm yang terintegrasi dengan proses deployment, sehingga membantu mendeteksi kegagalan lebih cepat dan menjaga stabilitas sistem (Teja Thallam, 2025).

Penelitian sebelumnya memperkuat pentingnya otomatisasi dalam proses deployment. (Rudrabhatla, 2020) menunjukkan bahwa strategi *Blue-Green Deployment* mampu secara efektif mengurangi downtime pada infrastruktur cloud publik. Sejalan dengan itu, (Cepuc et al., 2020) berhasil mengimplementasikan pipeline CI/CD berbasis Jenkins, Ansible, dan Kubernetes di AWS, yang meningkatkan kecepatan, konsistensi, serta menjamin zero downtime melalui penerapan *rolling update*. Selanjutnya, (Hyun et al., 2024) menegaskan bahwa penggunaan sistem otomatis berbasis Jenkins untuk deployment kontainer dapat secara signifikan mempercepat proses sekaligus menurunkan risiko kesalahan manusia. Walaupun menggunakan pendekatan yang berbeda, ketiga penelitian tersebut sepakat bahwa otomatisasi merupakan faktor utama dalam peningkatan kinerja deployment.

Berdasarkan landasan teori dan penelitian terdahulu tersebut, dapat ditegaskan bahwa efisiensi waktu dan keandalan sistem merupakan indikator utama yang harus diperhatikan dalam evaluasi deployment. Cloud computing memberikan fondasi yang memungkinkan penerapan otomatisasi melalui pipeline CI/CD, sedangkan AWS CodeDeploy dipandang sebagai salah satu solusi yang dapat meningkatkan kedua aspek tersebut.

3. METODE PENELITIAN

Desain Penelitian

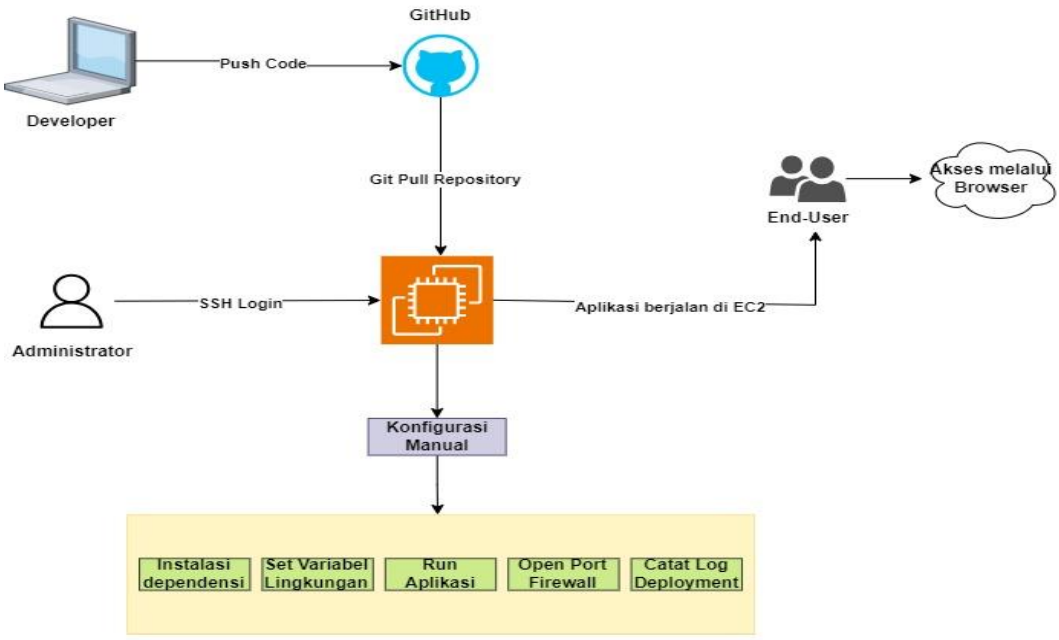
Penelitian ini menggunakan desain eksperimen kuantitatif dengan pendekatan perbandingan dua metode *deployment* aplikasi, yaitu manual dan otomatis menggunakan AWS CodeDeploy. Objek penelitian adalah website Personal Image Studio (PIS) Japan Career, yang dioperasikan pada infrastruktur *cloud* AWS. Data dikumpulkan pada rentang waktu April–Mei 2025 melalui lima kali percobaan *deployment* untuk masing-masing metode.

Instrumen penelitian berupa pencatatan waktu dengan stopwatch dan log AWS CloudWatch untuk menghitung durasi, serta pencatatan status *deployment* (berhasil atau gagal) untuk mengukur tingkat keandalan. Variabel yang diamati adalah:

1) Durasi deployment (menit),
2) Tingkat keberhasilan deployment (%).

Infrastruktur Penelitian

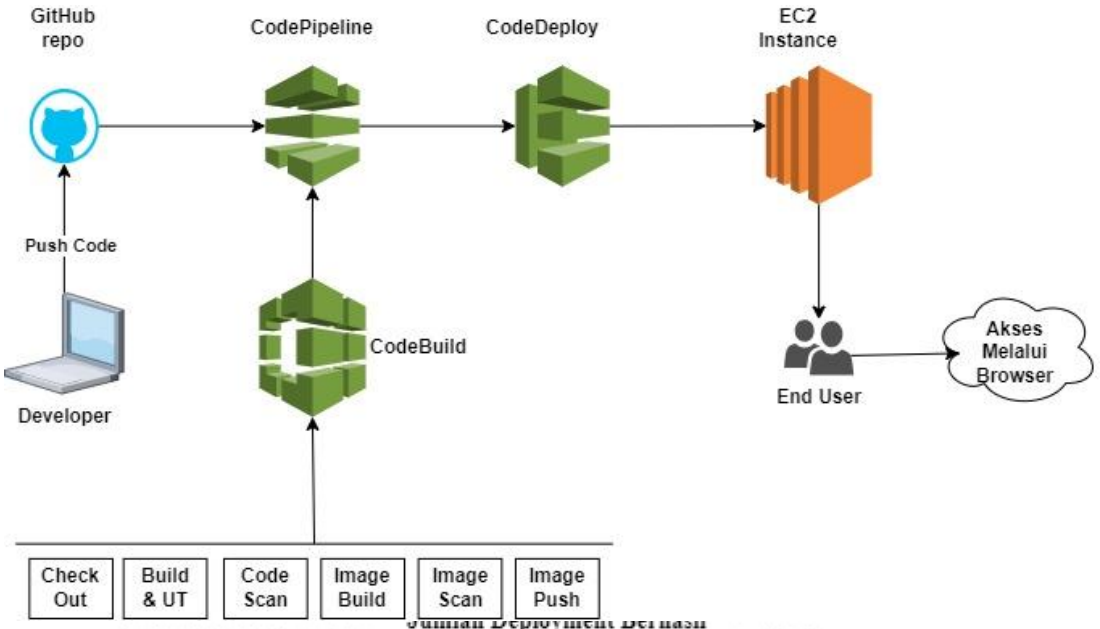
A. Deployment Manual



Gambar 1. Infrastruktur deployment manual

Gambar 1 menunjukkan arsitektur deployment manual, di mana proses implementasi aplikasi sepenuhnya dijalankan oleh admin melalui konfigurasi langsung di server. Model ini cenderung lebih lambat karena semua tahapan, mulai dari upload file hingga restart service, harus dilakukan secara manual.

B. Deployment Otomatis



$$\text{Tingkat Keberhasilan} = \frac{\text{Jumlah Deployment Berhasil}}{\text{Total Deployment}} \times 100\%$$

Gambar 2. Infrastruktur deployment otomatis

Gambar 2 memperlihatkan arsitektur deployment otomatis menggunakan AWS CodeDeploy. Proses deployment dijalankan melalui pipeline otomatis yang terintegrasi dengan EC2, S3, dan CloudWatch, sehingga lebih cepat dan minim kesalahan konfigurasi.

Teknik Analisis Data

Analisis data dilakukan secara deskriptif kuantitatif, dengan menghitung rata-rata waktu *deployment* dan persentase tingkat keberhasilan. Rumus yang digunakan adalah:

a. Rata-rata waktu deployment

$$\text{Efisiensi} = \frac{\text{Rata-rata Waktu Manual} - \text{Rata-rata Waktu Otomatis}}{\text{Rata-rata Waktu Manual}} \times 100\%$$

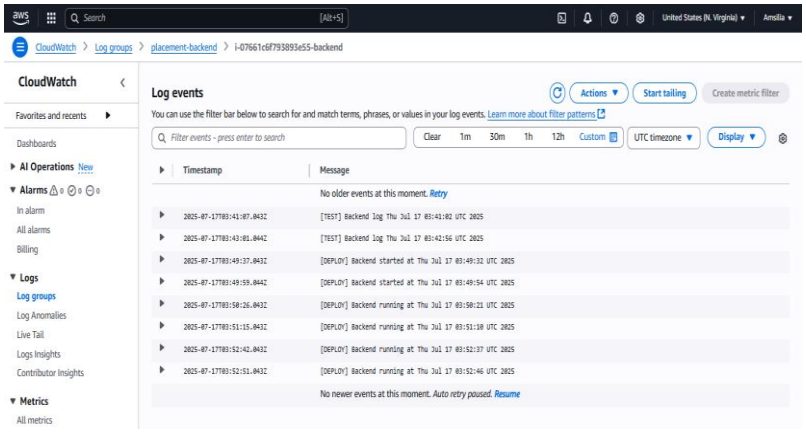
b. Tingkat keandalan deployment

4. HASIL DAN PEMBAHASAN

Waktu Deployment

$$\text{Tingkat Keberhasilan} = \frac{\text{Jumlah Deployment Berhasil}}{\text{Total Deployment}} \times 100\%$$

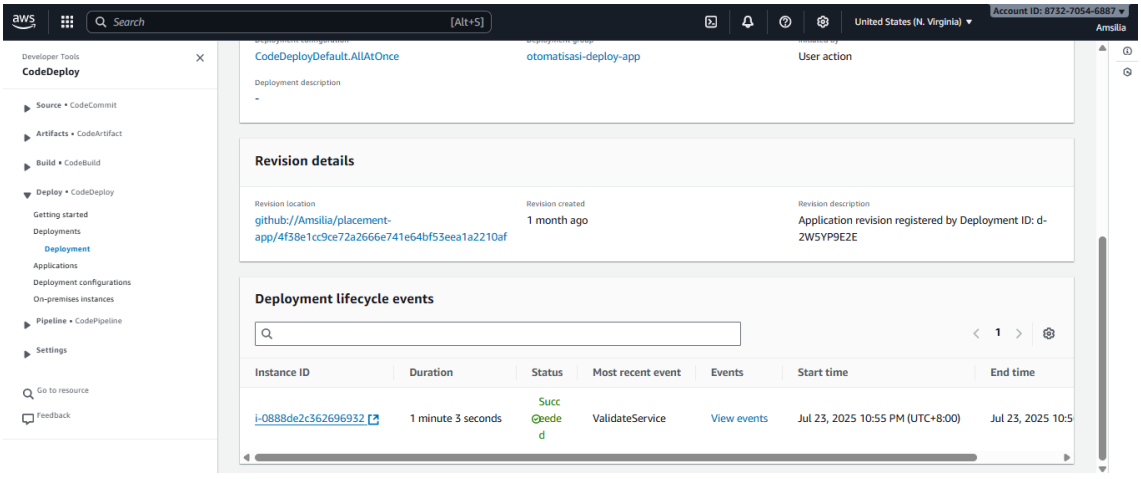
A. Hasil pengujian manual



Gambar 3. Hasil pengujian manual.

Hasil pemantauan melalui AWS CloudWatch menunjukkan bahwa durasi *deployment* dengan metode manual berada di kisaran 230 hingga 250 menit. Variasi waktu yang relatif lebar ini terjadi karena setiap tahapan proses dilakukan secara manual oleh admin, mulai dari pemindahan file, pengaturan konfigurasi, hingga restart layanan. Kondisi tersebut menjadikan metode manual tidak hanya lebih lambat, tetapi juga kurang konsisten dalam hal efisiensi waktu.

B. Hasil pengujian otomatis



Gambar 4. Hasil pengujian otomatis.

Pada metode otomatis menggunakan AWS CodeDeploy, hasil monitoring memperlihatkan durasi yang jauh lebih singkat, yaitu stabil di kisaran 30 hingga 35 menit pada setiap percobaan. Pola waktu yang konsisten ini menunjukkan bahwa otomatisasi mampu mengurangi ketergantungan pada intervensi manusia sekaligus meminimalkan potensi keterlambatan. Dengan demikian, penerapan AWS CodeDeploy tidak hanya meningkatkan efisiensi, tetapi juga memberikan kestabilan dalam proses *deployment*.

Hasil Pengujian Waktu Deployment

A. Deployment Manual

Tabel 1. Hasil pengujian waktu.

Percobaan ke-	Durasi (jam)	Durasi (menit)
1	5	300
2	5	300
3	4	240
4	3	180
5	3	180
Total		**1200 menit**
Rata-rata		**240 menit**

Rata-rata waktu *deployment* pada metode manual mencapai 240 menit. Durasi yang panjang ini menegaskan bahwa pendekatan manual kurang efisien dan sulit memenuhi kebutuhan sistem berbasis *cloud*.

B. Deployment Otomatis

Tabel 2. Hasil pengujian waktu deployment.

Percobaan ke-	Durasi (Menit)
1	45
2	40
3	30
4	30
5	20
Total	**165 menit**
Rata-rata	**33 enit**

Metode otomatis hanya membutuhkan rata-rata 33 menit. Perbedaan waktu yang signifikan dengan metode manual membuktikan efektivitas otomatisasi melalui AWS CodeDeploy dalam mempercepat proses *deployment*.

Hasil Pengujian Keandalan Deployment

A. Deployment Manual

Tabel 3. Hasil pengujian keandalan deployment.

Percobaan	Status	Catatan Kesalahan
1	Gagal	Install dependensi, salah memasukan perintah
2	Gagal	Port berubah-ubah
3	Gagal	*Error* saat migrasi database
4	Berhasil	-
5	Berhasil	-

Pengujian dengan metode manual menghasilkan keberhasilan dua kali dari lima percobaan, dengan tingkat keandalan 40%. Kegagalan terutama disebabkan oleh kesalahan konfigurasi serta migrasi basis data.

B. Deployment Otomatis

Tabel 4. Hasil pengujian keandalan deployment.

Percobaan	Status	Catatan Kesalahan
1	Gagal	Kesalahan membuat file *script* yml
2	Berhasil	-
3	Berhasil	-
4	Berhasil	-
5	Berhasil	-

Pengujian metode otomatis menghasilkan empat keberhasilan dari lima percobaan, dengan tingkat keandalan 80%. Satu kegagalan yang terjadi disebabkan oleh kesalahan pada file skrip, namun masih dapat diperbaiki melalui validasi konfigurasi.

C. Hasil Perbandingan Keandalan Deployment

Tabel 5. Hasil keandalan deployment manual dan otomatis

No	Parameter	Manual	Otomatis
1.	Jumlah Percobaan	5	5
2.	Jumlah Berhasil	2	4
3.	Jumlah Gagal	3	1
4.	Tingkat Keberhasilan (%)	40%	80%
5.	Sumber Kesalahan	Sistem/ konfigurasi (port, dependensi, migrasi DB)	Human error (file script
6.	Kemampuan *Rollback*	Manual	Belum aktifkan *rollback* otomatis

Perbandingan langsung antara kedua metode memperlihatkan bahwa otomatisasi jauh lebih andal, didukung fitur *rollback* otomatis yang menjaga stabilitas sistem. Sebaliknya, metode manual masih sangat bergantung pada intervensi manusia sehingga lebih rawan kesalahan.

5. KESIMPULAN DAN SARAN

KESIMPULAN

Penelitian ini menunjukkan bahwa penggunaan AWS CodeDeploy mampu meningkatkan efisiensi dan keandalan proses *deployment* aplikasi dibandingkan metode manual. Waktu *deployment* berkurang signifikan dari rata-rata 240 menit menjadi 33 menit, sedangkan tingkat keberhasilan meningkat dari 40% menjadi 80%. Dengan demikian, penerapan otomatisasi *deployment* berbasis AWS terbukti lebih efektif dalam mendukung kebutuhan sistem berbasis *cloud*.

SARAN

Penelitian selanjutnya dapat memperluas kajian dengan membandingkan AWS CodeDeploy terhadap layanan otomatisasi lain, serta menguji integrasi penuh dalam pipeline *Continuous Integration/Continuous Deployment* (CI/CD). Selain itu, uji coba pada skala aplikasi yang lebih besar dan beragam dapat memberikan gambaran yang lebih komprehensif mengenai efektivitas layanan ini.

UCAPAN TERIMA KASIH

Ucapan syukur senantiasa penulis panjatkan kehadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penelitian ini dapat terselesaikan dengan baik. Penulis menyampaikan terima kasih kepada kedua orangtua tercinta, Bapak Silavanus Padhang dan Mama Odilia Dhagheng, atas doa dan dukungan yang tiada henti. Terima kasih juga kepada bapak Kristianus Jago Tute, S.Kom., M.Pd, serta Ibu Elvira Esperanza Sala, S.T., M.Kom selaku dosen pembimbing yang telah memberikan bimbingan dan arahan dalam penyusunan penelitian ini.

DAFTAR REFERENSI

Andriani, A. (2013). Pemanfaatan cloud computing dalam pengembangan bisnis. *Seminar Nasional Teknologi Informasi Dan Multimedia*, 3-18.

AWS. (2025). AWS CloudFormation - Overview of deployment options on AWS. *March*. https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/aws-cloudformation.html

Cahya Kurniawan, A., Tibyani, T., & Amalia, F. (2020). Implementasi teknologi cloud computing untuk e-learning berbasis website dengan framework Laravel (Studi Kasus: MAN 9 Jombang). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 4*(11), 3833-3844. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/8141

Cepuc, A., Botez, R., Craciun, O., Ivanciu, I. A., & Dobrota, V. (2020). Implementation of a continuous integration and deployment pipeline for containerized applications in Amazon Web Services using Jenkins, Ansible and Kubernetes. *Proceedings - RoEduNet IEEE International Conference, 2020-Decem*. https://doi.org/10.1109/RoEduNet51892.2020.9324857

Gadani, N. N., & Devan, K. (2024). Leveraging the AWS cloud platform for CI/CD and infrastructure automation in software development. *International Journal of Intelligent Systems and Applications in Engineering, 2024*(23s), 350-356. www.ijisae.org

Hyun, G., Oak, J., Kim, D., & Kim, K. (2024). The impact of an automation system built with Jenkins on the efficiency of container-based system deployment. *Sensors, 24*(18), 1-12. https://doi.org/10.3390/s24186002

Lukita, D., & Setyo Utomo, F. (2024). Sistem informasi pengolahan data nilai siswa menggunakan AWS berbasis web. *Journal of Informatics and Interactive Technology, 1*(3), 206-214. https://doi.org/10.63547/jiite.v1i3.24

Pagels, M., Eloranta, V., & Itkonen, J. (2015). Focus: Release engineering.

Proulx, A., Raymond, F., Roy, B., & Petrillo, F. (2018). Problems and solutions of continuous deployment: A systematic review. http://arxiv.org/abs/1812.08939

Rudrabhatla, C. K. (2020). Comparison of zero downtime based deployment techniques in public cloud infrastructure. *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, 1082-1086. https://doi.org/10.1109/I-SMAC49090.2020.9243605

Saintikom, J., Sains, J., Informatika, M., Maulana, I., Umar, R., & Yudhana, A. (2024). Implementasi deployment layanan website menggunakan Kubernetes dengan CI/CD Jenkins. *23*, 290-296. https://doi.org/10.53513/jis.v23i2.9992

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access, 5*(Ci), 3909-3943. https://doi.org/10.1109/ACCESS.2017.2685629

Syed, F. A. (2025). Performance and features of Amazon S3. *International Journal of Scientific Research and Engineering Trends, 11*(1), 252-258. https://doi.org/10.61137/ijsret.vol.11.issue1.119

Teja Thallam, N. S. (2025). AI-powered monitoring and predictive maintenance for cloud infrastructure: Leveraging AWS CloudWatch and ML. *International Journal of Artificial Intelligence, Data Science, and Machine Learning, 6*(1), 55-61. https://doi.org/10.63282/3050-9262.IJAIDSML-V6I1P107

Wira Pradipta, K., Presiana Desi, Y., Khuntari, D., Tinggi Multi Media, S., Alamat, Y., & Magelang Km, J. (2024). Pengaruh kualitas sistem dan kualitas informasi aplikasi Jogja Smart Services (JSS) terhadap kepuasan pengguna pada masyarakat Yogyakarta. *Jurnal Ilmiah Manajemen Informasi Dan Komunikasi, 8*(1), 45-57. https://doi.org/10.56873/jimik.v8i1.333