Implementasi Sistem Monitoring dan Logging Menggunakan Ansible Pada Project di PT Itsavirus

I Komang Gde Satya Mahendra[1)], I Gede Suardika[2)], I Made Ari Santosa[3)]
Teknologi Informasi[1)], Sistem Informasi[2)], Sistem Komputer[3)]
Institut Teknologi dan Bisnis STIKOM Bali
Denpasar, Indonesia
e-mail: 210040147@stikom-bali.ac.id[1)], suardika@stikom-bali.ac.id[2)], arisantosa@stikom-bali.ac.id [3)]

Abstrak

Penggunaan Cloud computing dan platform open source menjadi solusi utama dalam meningkatkan efisiensi proses pengembangan sebuah aplikasi. PT itsavirus sebagai software house yang menggunakan AWS (Amazon Web Service) dalam proses pengembangan ataupun penyebaran aplikasi kepada user, menghadapi tantangan dalam melakukan proses monitoring dan logging dalam menggunakan layanan AWS, di mana akses tersebut terbatas bagi tim pengembang, memperlambat proses troubleshooting. Selain itu, otomatisasi dalam melakukan instalasi dan konfigurasi server menjadi kebutuhan penting guna meningkatkan efisiensi kerja. Oleh karena itu, penelitian ini bertujuan mengevaluasi efektivitas platform terpusat dalam menyatukan monitoring sumber daya dan logging aplikasi dengan automation, serta dampaknya terhadap kecepatan identifikasi dan penyelesaian masalah dalam pengelolaan infrastruktur IT di PT Itsavirus. Software Development Life Cycle (SDLC) dengan pendekatan Waterfall Methodology digunakan pada penelitian ini. Dengan melakukan analisa sistem, perancangan sistem dengan merancang cloud architecture diagram, implementasi sistem menggunakan tools Ansible sebagai otomatisasi insitalasi dan konfigurasi server, penggunaan open-source platform seperti Grafana, loki, promtail, Prometheus, node exporter, dan juga cadvisor dalam membagun sistem monitoring dan logging.pengujian akan dilakukan menggunakan stopwatch untuk membandingkan waktu deployment menggunakan Ansible dengan metode manual, serta mengukur waktu yang dibutuhkan developer dalam memperoleh log aplikasi dari cloud menggunakan Grafana dibandingkan dengan metode sebelumnya. Pengujian dilakukan dalam lima iterasi untuk mendapatkan hasil yang lebih akurat. Tercatat dalam penelitan ini rata-rata waktu yang di gunakan Ansible untuk melakukan deployment secara otomatis adalah 87,19 detik dan waktu yang digunakan developer untuk menggunakan Grafana sebagai sistem monitoring dan logging terpusat adalah 24,94 detik. Dengan hasil penelitian ini diharapkan dapat meningkatkan efisiensi operasional, mengurangi kesalahan manusia, serta mempercepat troubleshooting dalam proses pengembangan aplikasi di PT Itsavirus.

***Kata kunci** : Platform, Cloud computing, Amazon web service, Ansible, Grafana*

1. Pendahuluan

Di zaman digitalisasi ini, adanya sebuah aplikasi dan platform sangat membantu para pengguna dengan keberagamannya. Dengan berkembangnya pengguna internet, banyak perusahaan yang berlomba-lomba membuat aplikasi atau platform yang dapat menyelesaikan masalah-masalah yang ada. Dalam prosesnya dibutuhkan tim pengembang yang kompeten dan dibantu oleh teknologi-teknologi pengembangan. *Cloud Computing* merupakan salah satu teknologi yang memungkinkan para *developer* untuk memanfaatkan sumber daya komputasi tanpa perlu memiliki atau mengelola sendiri perangkat keras dan perangkat lunak yang mendasarinya [1]. Selain *cloud computing*, penggunaan platform-platform *open source* juga dapat membantu sebuah proses pengembangan sistem. Dengan berkembangnya teknologi proses kolaborasi antar platform yang digunakan tim pengembang dengan *cloud computing* dapat di dilakukan otomatis, hal ini akan meningkatkan efisiensi pengerjaan dan menghasilkan produk yang maksimal.

Software house merupakan sebuah perusahaan yang berfokus pada perancangan dan pengembangan aplikasi atau platform dari ide klien. PT Itsavirus yang merupakan *software house*, menggunakan teknologi-teknologi terkenal dan *cloud* server dalam mengembangkan sebuah aplikasi. Peranan *cloud* server pada sebuah *software house*, membantu para tim pengembang dalam melakukan pengujian dan demonstrasi kepada klien. Banyaknya proyek yang di kerjakan oleh PT Itsavirus dengan tenggat waktu yang terbatas, penting adanya pengoptimalan dan efisiensi yang dilakukan dalam proses

pengembangannya. Di PT Itsavirus, *monitoring* dan *logging* pada *cloud* masih menjadi kekurangan yang ada, karena *developer* dan DevOps harus bekerja sama dalam memeriksa log dan sumber daya server pada halaman amazon web service (AWS). Hal ini memerlukan akses langsung ke server, di mana hanya DevOps yang mempunyai akses terhadap AWS. Dengan alur seperti itu pastinya akan memperlambat proses *troubleshooting* dari perangkat lunak yang dikembangkan ataupun dari perangkat keras *cloud.* Selain masalah tersebut, otomatisasi dalam melakukan konfigurasi sebuah server merupakan sebuah hal yang patut di implementasi pada *software house*. Penambahan kompleksitas dalam pemeliharaan aplikasi akan dirasakan, dengan ini dibutuhkan sebuah sistem dan alur yang mudah agar dapat membantu tim pengembang dalam mencari informasi tanpa harus mempengaruhi waktu pengerjaan.

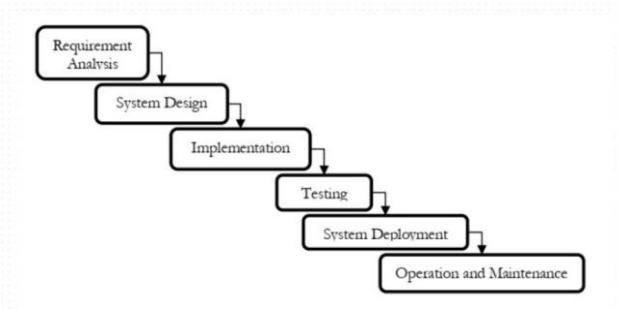
Dengan mengimplementasi sistem *logging* dan *monitoring* sistem ke dalam sebuah platform terpusat akan membantu, baik DevOps atau *developer*. Mengurangi ketergantungan antar tim, dapat meningkatkan efisiensi waktu dalam pencarian masalah, proses pengambilan keputusan, hingga proses eksekusi kode. penggunaan platform terpusat memungkinkan deteksi dini terhadap masalah, yang pada akhirnya meningkatkan ketersediaan dan performa layanan. Integrasi sistem monitoring dan logging dengan teknologi terbaru, serta otomatisasi tugas-tugas yang repetitif, semakin meningkatkan efisiensi dalam pengerjaan proyek di PT Itsavirus. Dengan menggabungkan alat-alat seperti Grafana, Prometheus, dan Loki, bersama dengan otomatisasi konfigurasi server menggunakan Ansible, organisasi dapat mengoptimalkan kinerja dan mengurangi kesalahan manusia dalam proses pengelolaan sistem. Pendekatan ini memungkinkan tim DevOps dan *developer* di PT Itsavirus untuk bekerja lebih efektif dan dapat berinovasi.

Sistem *monitoring* untuk server terbukti dalam kemampuannya meningkatkan efisiensi operasional, meningkatkan waktu respons, dan memberikan wawasan berharga mengenai kinerja sistem [2]. Dengan Grafana yang bersifat *open source*, dapat terintegrasi dengan Prometheus dan Loki, penggunaan berbagai eksportir, kemampuan *query* yang kuat, fitur peringatan, dan kemampuan menganalisis data yang dikumpulkan [3]. Dari banyak manfaat penggunaan *monitoring* sistem dan grafana sebagai platform, menunjukkan bahwa penting adanya sebuah pengumpulan informasi *monitoring* dan *logging*. Selain itu, sebuah penelitian yang dilakukan oleh Sang Putu Nanda Suhendra dengan menggunakan ansible sebagai alat bantu *deployment*, memberikan dampak yang baik buat sebuah perusahaan. Dengan berkurangnya waktu *deployment* secara signifikan hingga 90% dan meningkatnya frekuensi *deployment* yang dilakukan, membuat otomatisasi ini sangat diperlukan pada sebuah *software house* [4]. Otomatisasi yang dilakukan Ansible ini berperan dalam menyederhanakan penerapan konfigurasi server sistem secara otomatis, mengurangi kesalahan manusia dalam proses konfigurasi, dan meningkatkan efisiensi[5].

Oleh karena itu, penelitian ini juga bertujuan untuk mengevaluasi efektivitas penggunaan platform terpusat untuk menyatukan informasi *monitoring* sumber daya dan *logging* aplikasi dengan *automation* pada proyek di PT Itsavirus, serta dampaknya terhadap kecepatan identifikasi dan pemecahan masalah dalam pengelolaan infrastruktur IT. Selain itu, penelitian ini juga akan mengkaji bagaimana integrasi platform tersebut dapat meningkatkan efisiensi kerja dan menghilangkan tugas-tugas repetitif yang sering terjadi dalam proses *monitoring* dan konfigurasi sistem.

2. Metode Penelitian

Software Development Life Cycle (SDLC) merupakan sebuah proses terstruktur yang menguraikan tahap-tahap yang terlibat dalam pengembangan perangkat lunak. Tahapan SDLC mencakup perancangan, pengembangan, pengujian, dan menyebarkan perangkat lunak dengan kualitas yang tinggi [6]. SDLC menyediakan pendekatan sistematis dengan membagi proses pengembangan menjadi beberapa fase berbeda dengan tujuan yang spesifik. Tujuan adanya SDLC ini untuk memastikan efisiensi, konsistensi, dan kualitas di seluruh proses pengembangan dengan prinsip kolaborasi yang transparan. Metodologi *Waterfall* adalah serangkaian proses pengembangan perangkat lunak yang terstruktur dengan pendekatan linier dan berurutan. Penggunaan *waterfall* ini membantu proses pengembangan perangkat lunak dengan menyediakan struktur yang jelas dan mengharuskan penyelesaian setiap fase sebelum beralih ke fase selanjutnya.



Gambar 1. Alur Kerja *Waterfall Methodology [6]*

2.1. Pengumpulan Data

Pengumpulan data merupakan tahapan awal yang akan dilakukan saat mencari solusi dari sebuah masalah. Adanya pengumpulan data dan informasi akan membantu proses pengembangan ataupun pencarian solusi yang tepat. Pada tahap ini dilakukan pencarian jurnal pendukung terhadap teknologi-teknologi yang digunakan dengan masalah yang dihadapi PT Itsavirus, serta melakukan observasi secara langsung terhadap kendala *developer* di PT Itsavirus dalam mencari log dari sebuah aplikasi yang sedang dikembangkan.

2.1.1. Studi Literatur

Pada penelitian ini, data-data serta informasi yang dikumpulkan dengan membaca dan memahami berbagai macam jenis literatur yang tersedia, seperti jurnal, buku, karya tulis, dan penelitian yang memiliki ruang lingkup yang sama dengan penelitian ini, baik dalam bentuk media elektronik ataupun media cetak. Tahap dapat memberikan kemudahan dan sebagai acuan terhadap perancangan dan implementasi sistem ini.

2.1.2. Observasi

Observasi merupakan sebuah metode pengumpulan informasi dan juga data yang dilakukan dengan pengamatan langsung ke lapangan. Pada penelitian ini, observasi akan melihat alur kerja pegawai di PT. Itsavirus yang telah digunakan dan mencari sebuah kelemahan pada alur tersebut.

2.2. Analisis Kebutuhan (*Requirement Analysis*)

Pada tahap perencanaan dilakukannya pengumpulan terhadap informasi dan kebutuhan untuk memperoleh alur proses yang lebih terarah serta mempersiapkan kemungkinan masalah yang akan terjadi dalam proses pengembangan agar dapat ditanggulangi dengan baik. Penyelesaian masalah juga akan di disesuaikan terhadap kebutuhan dan pengambilan informasi yang telah dilakukan.

2.3. Desain Sistem (*System Design*)

Pada tahap desain sistem ini, akan membuat perancangan terhadap sistem yang akan dibuat serta sistem pengukuran yang akan dilakukan. Proses ini akan menetapkan penggunaan teknologi yang akan membantu proses pengembangan, dengan membuat *Cloud Architecture Diagram* sebagai acuan dan referensi dalam membangun sistem ini dalam lingkungan *cloud [7]*.

2.4. Implementasi *(Implementation)*

Tahapan ini merupakan sebuah proses pengembangan yang mencakup implementasi hasil dari perencanaan dan desain sistem yang telah dibangun. Adapun perangkat yang digunakan yaitu *Visual Studio Code* sebagai aplikasi kode *editor*. Ansible sebagai otomasi deployment dengan menggunakan YAML yang akan melakukan instalasi dan konfigurasi terhadap *deployment* sistem *monitoring* dan *logging* yang akan berjalan pada sebuah Docker *container*.

2.5. Pengujian (*Testing*)

Pengujian sistem ini akan dilakukan dengan menggunakan *cloud* server pribadi yang akan menjalankan sistem ini. Dalam pengujiannya akan dilakukan pencatatan waktu menggunakan *stopwatch* untuk mendapatkan waktu yang dibutuhkan ansible sebagai alat otomasi *deployment* dan melakukan *deployment* secara manual. Dalam tahapan ini juga akan mencatat beberapa masalah yang terjadi saat melakukan *deployment*.

2.6. Penyebaran Sistem *(System Deployment)*

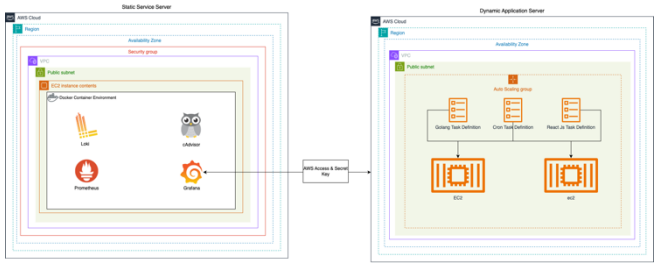
Tahapan ini mencakup *deployment* sistem menggunakan *cloud* server dari PT. Itsavirus, dan menggunakan hasil yang telah dikerjakan pada tahap implementasi. Hal ini mencakup penggunaan sistem ini dalam sebuah *project* dan dapat digunakan langsung oleh pegawai di PT. Itsavirus dalam serta memberikan *feedback.*

E-ISSN : 3031-9692 Vol. 2 No. 1 2025
Institut Teknologi dan Bisnis STIKOM Bali, 8 Maret 2025

3. Hasil dan pembahasan

Implemetasi Sistem *Monitoring* dan *Logging* menggunakan Ansible pada Project di PT Itsavirus, memberikan sebuah kemudahan kepada para developer dan juga *devops*. Penyediaan platform *monitoring* dan *logging* ini membantu para developer dalam mencari akar masalah dan melakukan akses pada platform terpusat. Selain itu juga Ansible sebagai alat otomatisasi membantu *devops* dalam melakukan instalasi dan konfigurasi yang berulang dan mengurangi *human error*. Dengan menggunakan perhitungan yang dibantu dengan stopwatch, serta data akan di ambil dan di proses menghasilkan sebuah tabel [8].

3.1. *Perancangan*



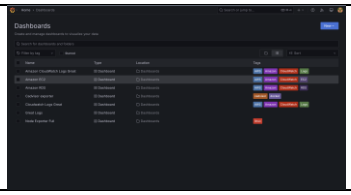
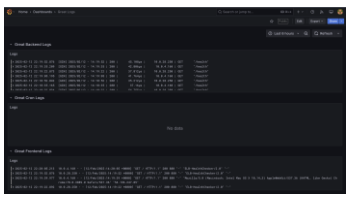
Gambar 2. *Cloud Architecture Diagram*

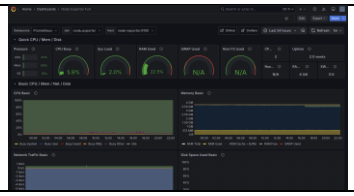
Pada *Cloud Architecture Diagram* diatas, terdapat dua AWS *architecture* berbeda. *Dynamic Application Server* yang berada di sebelah kanan merupakan sebuah *Cloud Achitecture* yang berisi aplikasi utama yang digunakan oleh pengguna dalam mengakses *website* yang dibangun. Pada *architecture* tersebut terdapat tiga aplikasi yaitu React js yang merupakan *frontend*, golang sebagai *backend*, dan *cron* sebagai *scheduler application*. Sistem ini menggunakan AWS ECS yang mendukung sebuah server dapat secara otomatis menangani penggunaan sumber daya server yang dibutuhkan dalam menjalankan tiga aplikasi tersebut [9]. *Static Service Server* merupakan sebuah server yang menampung beberapa aplikasi yang mendukung berjalannya aplikasi utama yang berada pada *Dynamic Application Server*. Dalam hal ini grafana sebagai alat visualisasi membutuhkan data dari *Dynamic Application* Server berupa *metrics* server dan juga *logging* dari setiap aplikasi yang berjalan. Maka dari hal ini dibutuhkan sebuah *access key* dan *secret key* yang dapat dikonfigurasi dengan AWS IAM [10].

3.2. Implementasi

Implementasi di lakukan sesuai dengan hasil dari perancangan yang telah di lakukan. Dengan menggunakan vscode sebagai *code editor* dan Ansible sebagai alat otomatisasi instalasi dan konfigurasi, akan menghasilkan sebuah sistem *monitoring* dan *logging* seperti tabel dibawah ini:

Tabel 1. Implementasi Sistem *Monitoring* dan *Logging*

No	Nama Rancangan Antarmuka	Gambar Rancangan	Penjelasan
1.	Halaman Utama		Halaman ini berisi informasi terkait *dashboard* yang telah dikonfigurasi dari berbagai data *source*, yang memungkinkan pengguna untuk melihat *montioring* dan *logging* yang tersedia.
2.	Halaman *Great Logging*		Halaman great logging berisi informasi terkait log dari aplikasi seperti *frontend*, *backend*, dan juga *cron application*. Halaman ini menggunakan konsep *real-time* yang memungkinkan pengguna melihat *logs* dengan mudah

3.	Halaman Cadvisor		Halaman ini berisi informasi terkait penggunaan sumber daya pada docker *environment* di mana, setiap aplikasi yang berjalan pada docker akan terlihat secara terpisah dan pengguna dapat melihat informasi yang ada.
4.	Halaman Node *Exporter*		Halaman ini berisi informasi terkait penggunaan sumber daya secara sistem dimana, setiap penggunaan cpu, ram, dan juga jaringan.

3.3. Pengujian

Dalam tahap pengujian dilakukan pengukuran waktu, dalam penelitian ini terdapat dua hal yang dapat di ukur untuk mengevaluasi efisiensi pekerjaan di PT Itsavirus dengan menggunakan *stopwatch*. Pertama penggunaan Ansible sebagai alat otomatisasi *deployment* dan konfigurasi dapat melakukan efisiensi waktu dibandingkan dengan melakukan secara instalasi dan konfigurasi secara manual melalui terminal server. Kedua mengukur waktu yang dibutuhkan developer dalam mencari *logs* dan *metrics* dari sistem yang terpusat dibandingkan dengan menggunakan layanan AWS.

Tabel 2. Perbandingan Waktu Dalam Melakukan *Deployment* dan Konfigurasi

Iterasi ke	Ansible (detik)	Manual (detik)
1	90,95 detik	787,76 detik
2	89,19 detik	788,31 detik
3	85,69 detik	791,57 detik
4	84,02 detik	785,07 detik
5	86,10 detik	789,98 detik
Rata-rata	87.19 detik	788.54 detik

Berdasarkan Tabel 2. Perbandingan Waktu Dalam Melakukan *Deployment* dan Konfigurasi, menujukan bahwa dengan menggunakan otomatisasi *deployment* dan konfigurasi dapat mengurangi waktu pengerjaan hingga 88.93%. Efisiensi tersebut terjadi dikarenakan adanya pengurangan *human error* dan juga beberapa proses yang tidak dibutuhkan seperti kesalahan mengetik, masalah koneksi jaringan, serta proses terminal yang dibutuhkan untuk melakukan *deployment* dan konfigurasi.

Tabel 3. Perbandingan Waktu Akses Sistem *Monitoring* dan *Logging*

Iterasi ke	Grafana	AWS Logs dan Server Metrics
1	25,63 detik	72,78 detik
2	24,60 detik	53,96 detik
3	23,72 detik	62,19 detik
4	23,60 detik	59,34 detik
5	27,16 detik	65,22 detik
Rata-rata	24.94 detik	62,70 detik

Berdasarkan Tabel 3. Perbandingan Waktu Akses Sistem *Monitoring* dan *Logging*, menunjukkan bahwa dengan menggunakan grafana sebagai *platform monitoring* dan *logging* terpusat akan mempersingkat waktu developer dalam mengakses *metrics* dan *logs* hingga 60.22%. Dalam kasus ini developer tidak perlu masuk ke halaman AWS yang bergantung jaringan internet bergantung pada region yang digunakan, mempersingkat proses pencarian setiap servis yang ingin di tinjau, serta memberikan kemudahan untuk developer dalam mengakses sebuah *metrics* dan *logging*.

3.4. Penyebaran

Sistem *monitoring* dan *logging* terpusat disebarkan menggunakan layanan AWS serta digunakan pada *developer* di PT Itsavirus. Penyebaran di lakukan menggunakan Ansible secara otomatis ke dalam *server* yang terisolasi ke dalam sebuah Docker *environment*. Dengan itu semua *developer* di PT Itsavirus dapat mengakses sistem ini dengan mudah.

4. Kesimpulan

Penelitian ini telah menghasilkan sebuah sistem monitoring dan logging terpusat yang digunakan pada project di PT Itsavirus. Pengujian dalam penelitian ini menunjukkan bahwa penggunaan Ansible, sistem *monitoring* dan *logging* secara terpusat memiliki dampak secara signifikan terhadap waktu yang digunakan. Sistem ini telah digunakan oleh *developer* dalam mencari sebuah sumber masalah dalam proses pengembangan ataupun menentukan penggunaan *resource cloud* yang lebih efisien. Penggunaan Ansible membantu proses *deployment* dan juga konfigurasi akan lebih optimal dan stabil secara otomatis. Dengan hal tersebut PT Itsavirus dapat menyimpan lebih banyak waktu untuk mengembangkan sebuah aplikasi dengan lebih produktif dan optimal. Secara keseluruhan, sistem yang dikembangkan memberikan dampak positif terhadap efisiensi operasional PT Itsavirus, memungkinkan tim untuk lebih fokus pada pengembangan aplikasi tanpa terbebani proses konfigurasi dan *troubleshooting* yang memakan waktu. Dengan solusi ini, PT Itsavirus dapat meningkatkan produktivitas serta kualitas layanan dalam pengelolaan infrastruktur IT.

Daftar Pustaka

[1] A. Y. Arisandy, S. Della Permatasari, S. Izaroh, R. Hidayat, and M. Ikaningtyas, "Adopsi Cloud Computing Dalam Perencanaan Dan Pengembangan Bisnis Usaha Kecil Menengah (UKM)," *Economics And Business Management Journal (EBMJ) Februari*, vol. 3, no. 1, 2024.

[2] A. Rahma, F. Indriyani, T. Alfian, and A. Sandi, "Perancangan Dan Implementasi Monitoring Perangkat Server Menggunakan Zabbix Pada PT. Rizki Tujuh Belas Kelola," *Jurnal INSAN (Journal of Information Systems Management Innovation*, vol. 3, no. 2, 2023, [Online]. Available: http://jurnal.bsi.ac.id/index.php/jinsan

[3] Matti Holopainen, "Monitoring Container Environment with Prometheus and Grafana," May 2021.

[4] Sang Putu Nanda Suhendra, "Sistem Otomatisasi Deployment Menggunakan Ansible (Studi Kasus : Lab Networking Stikom Bali)," 2019.

[5] A. F. Aditya, M. U. H. Al Rasyid, A. Basofi, and W. M. Rahmawati, "Otomatisasi Deployment Web OpenSID Menggunakan Ansible," in *Prosiding Seminar Nasional Terapan Riset Inovatif (SENTRINOV)*, 2024, pp. 346–353.

[6] H. J. Christanto and Y. A. Singgalen, "Analysis and Design of Student Guidance Information System through Software Development Life Cycle (SDLC) dan Waterfall Model," *Journal of Information Systems and Informatics*, vol. 5, no. 1, pp. 259–270, Mar. 2023, doi: 10.51519/journalisi.v5i1.443.

[7] S. B. Vermaa, B. Pandeyb, and B. K. Gupta, "Containerization and its Architectures: A Study," *Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 11, no. 4, pp. 395–409, 2022, doi: 10.14201/adcaij.28351.

[8] S. Wangchuk and A. K. Madan, "Stopwatch Method For Assembly Line Production," *International Journal of Advances in Engineering and Management (IJAEM)*, vol. 5, p. 39, 2023, doi: 10.35629/5252-05013945.

[9] K. R. Ahmed and M. Islam, "A Comparative Analysis of AWS Cloud-Native Application Deployment Model," in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2022, pp. 429–441. doi: 10.1007/978-981-19-2445-3_29.

[10] N. Kamble, R. Gandhi, and V. Shahji, "Access Control Model Based on AWS IAM Article in International Journal of Innovative Research in Computer and Communication Engineering," *International Journal of Innovative Research in Computer and Communication Engineering*, 2021, doi: 10.15680/IJIRCCE.2021.0911024.