5.14%

SIMILARITY OVERALL

SCANNED ON: 4 FEB 2025, 6:07 PM

Similarity report

Your text is highlighted according to the matched content in the results above.

CHANGED TEXT 5.14%

QUOTES 0.25%

Report #24668451

6 12 18 BAB I PENDAHULUAN 1.1 Latar Belakang Masalah Perkembangan pesat teknologi informasi dan komunikasi telah mengubah berbagai aspek dalam kehidupan manusia, termasuk cara berbelanja. Industri e- commerce, yang menjadi salah satu pilar utama ekonomi digital, telah menunjukkan pertumbuhan signifikan dalam dekade terakhir. Laporan e-Conomy SEA 2022 (Google, Temasek, & Bain & Co., 2022) mengungkapkan bahwa ekonomi digital di Asia Tenggara, dengan e-commerce sebagai kontributor utamanya, telah mengalami pertumbuhan luar biasa. Di Indonesia, e-commerce menyumbang hingga 75% dari seluruh ekonomi digital, dengan proyeksi Gross Merchandise Value (GMV) mencapai 59 miliar USD pada tahun 2022. 37 Hal ini menjadikan Indonesia sebagai salah satu pasar e-commerce dengan pertumbuhan tercepat di Asia Tenggara. Dalam konteks Indonesia, PT. XYZ adalah salah satu pemain dominan di pasar e-commerce. Perusahaan ini telah berhasil memanfaatkan teknologi mutakhir untuk menyediakan layanan yang efisien dan memuaskan bagi konsumennya. Namun, dengan pertumbuhan pesat dan ekspansi yang dilakukan, PT. XYZ menghadapi tantangan besar dalam pengelolaan biaya operasional, khususnya di bidang infrastruktur teknologi. Setelah melakukan Initial Public Offering (IPO), PT. XYZ semakin terdorong untuk mencari strategi yang dapat mengoptimalkan penggunaan sumber daya cloud mereka. Berdasarkan laporan keuangan terbuka perusahaan, salah satu komponen biaya operasional

terbesar berasal dari penggunaan infrastruktur teknologi, terutama cloud computing. Sebagai perusahaan yang kini diawasi oleh investor dan pemegang saham, PT. XYZ harus memastikan bahwa biaya operasional ini dikelola secara efisien untuk menjaga margin keuntungan. Hal inilah yang menjadi tantangan besar untuk PT.XYZ sendiri bagaimana caranya bisa memanfaatkan sistem keuangan secara efisien. Tantangan tersebut membuat PT. XYZ menerapkan strategi pengelolaan sumber daya cloud yang lebih terukur. Fokus utama strategi ini adalah optimasi sumber daya seperti virtual machine (VM), snapshot, disk, dan IP address yang tidak digunakan secara optimal. Beberapa project di Google Cloud Platform (GCP) diketahui hanya mengaktifkan API monitoring tanpa adanya aktivitas layanan lainnya, sehingga menambah biaya tanpa memberikan manfaat yang signifikan. Meskipun cloud computing menawarkan skalabilitas dan fleksibilitas yang tinggi, penggunaan sumber daya yang tidak optimal tetap menjadi tantangan besar bagi efisiensi biaya. Masalah seperti pemilihan tipe mesin (machine type) yang tidak sesuai kebutuhan, keberadaan snapshot yang sudah tidak relevan, serta disk dan IP address yang tidak terpakai dapat berdampak signifikan terhadap biaya operasional. Oleh karena itu, optimasi penggunaan sumber daya cloud menjadi langkah penting yang harus dilakukan PT. XYZ untuk menekan biaya operasional tanpa mengurangi kinerja. 2 Saat ini,

identifikasi sumber daya cloud yang tidak efisien masih dilakukan secara manual, membutuhkan waktu dan tenaga yang besar dari tim operasional. Proses manual ini meningkatkan risiko adanya sumber daya yang terlewat untuk dioptimalkan, sehingga mengakibatkan pemborosan biaya dan ketidakmampuan untuk merespons masalah dengan cepat. Untuk mengatasi keterbatasan ini, PT. XYZ membutuhkan pendekatan yang lebih efisien dan terotomatisasi. Sebagai langkah awal, PT. XYZ telah mengadopsi kebijakan pengelolaan sumber daya cloud. Kebijakan ini mencakup pengurangan VM yang penggunaannya kurang dari 40%, peningkatan kapasitas untuk VM yang penggunaannya melebihi 80%, penghapusan snapshot manual yang berusia lebih dari enam bulan, serta penghapusan disk dan IP address yang tidak digunakan. Kebijakan ini dirancang untuk memastikan efisiensi penggunaan sumber daya cloud, sekaligus mengurangi pemborosan biaya. Penerapan konsep Development and Operations (DevOps) menjadi solusi yang relevan untuk mengatasi tantangan ini. 42 DevOps mendorong kolaborasi antara tim pengembangan dan operasional, dengan fokus pada otomatisasi proses. Dengan integrasi teknologi machine learning, PT. XYZ dapat membangun sistem automasi untuk mendeteksi dan merekomendasikan perubahan tipe VM berdasarkan analisis penggunaan CPU dan memori. Selain itu, dengan memanfaatkan analisis data berbasis BigQuery, sistem ini dapat mengidentifikasi snapshot, disk, dan IP address yang tidak efisien. Penerapan sistem ini akan menggantikan proses manual, meningkatkan akurasi, dan menghemat waktu secara signifikan. Melalui penelitian ini, dikembangkan sistem automasi berbasis machine learning yang mampu mendeteksi dan merekomendasikan optimalisasi penggunaan sumber daya cloud secara efisien. Dengan adanya sistem ini, PT. XYZ diharapkan dapat menekan biaya operasional sekaligus mempertahankan kinerja layanannya. Selain itu, hasil penelitian ini diharapkan dapat menjadi referensi bagi perusahaan e-commerce lain dalam menghadapi tantangan serupa di era digital. 1 14 1.2 Identifikasi Masalah Berdasarkan latar belakang yang telah disampaikan, identifikasi masalah menjadi langkah awal yang krusial

dalam mengembangkan sistem automasi DevOps untuk optimalisasi biaya cloud di lingkungan e-commerce. Berikut adalah beberapa aspek utama yang menjadi dasar dalam identifikasi masalah ini: 1.2.1 Rumusan Masalah Penelitian ini dirancang untuk menjawab pertanyaan-pertanyaan berikut: a. Bagaimana cara mengembangkan sistem otomatisasi yang mampu merekomendasikan perubahan machine type VM berdasarkan analisis penggunaan CPU, memori, serta jumlah permintaan baca dan tulis (read/write request) pada lingkungan Cassandra dengan integrasi machine learning? b. Bagaimana cara mengidentifikasi dan menghapus snapshot yang tidak diperlukan, serta mendeteksi project dan sumber daya cloud lainnya yang tidak digunakan dengan bantuan BigQuery dan analisis data berbasis API GCP? 3 c. Bagaimana cara mengintegrasikan sistem aplikasi dengan praktik DevOps secara efektif untuk mengelola dan mengoptimalkan sumber daya cloud di lingkungan e-commerce? 1.2.2 Batasan Masalah Batasan masalah dalam penelitian ini disesuaikan untuk memfokuskan ruang lingkup dan memastikan kejelasan dan keefektifan dalam pencapaian tujuan penelitian. Berikut adalah batasan yang diterapkan: 1. Penelitian ini terbatas pada penggunaan VM khusus untuk Cassandra di GCP, termasuk pengukuran CPU, memori, dan read/write request untuk analisis optimasi. 2. Penelitian hanya menggunakan teknologi dan alat DevOps yang relevan dengan GCP, termasuk integrasi dengan GitLab CI/CD untuk otomatisasi, serta penggunaan Datadog dan BigQuery untuk monitoring, analisis, dan deteksi anomali sumber daya. 3. Data yang digunakan berasal dari periode Januari 2024 hingga Oktober 2024, dengan asumsi bahwa tren dan teknologi baru setelah periode ini tidak akan menjadi bagian dari analisis. 4. Sistem automasi difokuskan pada rekomendasi terkait: Perubahan machine type VM berdasarkan APU (Average Predicted Usage), Identifikasi dan penghapusan snapshot yang berusia lebih dari enam bulan. Deteksi project dengan hanya Monitoring API yang aktif. Dan deteksi disk dan IP address yang tidak digunakan pada GCP. 12 49 1.3 Tujuan Penelitian Tujuan dari

penelitian ini adalah untuk: 1. Mengembangkan sistem automasi yang mampu mengidentifikasi secara efektif sumber daya cloud yang tidak terpakai atau kurang efisien dalam lingkungan e-commerce, khususnya pada Google Cloud Platform (GCP). 2. Menetapkan ambang batas ideal untuk idle resources di GCP guna mengurangi waktu tidak terpakai (idle time) dan memaksimalkan pemanfaatan sumber daya. Hal ini melibatkan pengembangan model atau algoritma yang dapat menyesuaikan ambang batas berdasarkan analisis data secara real-time. 3. Mengintegrasikan sistem automasi ini dengan praktik DevOps yang sudah ada. Fokus utama adalah pada penerapan strategi untuk meningkatkan efisiensi biaya operasional cloud, mengurangi pemborosan, dan meningkatkan kinerja keseluruhan di lingkungan GCP. 4. Menyediakan solusi yang dapat diadopsi oleh perusahaan e-commerce. Solusi ini mencakup panduan atau kerangka kerja untuk membantu perusahaan mengelola sumber daya cloud dengan lebih hemat biaya, mengoptimalkan alokasi sumber daya, dan memanfaatkan teknologi awan secara lebih efektif. 5. Mengembangkan dan menerapkan model machine learning, seperti Boosted Decision Tree, untuk mengoptimalkan rekomendasi tipe VM berdasarkan analisis penggunaan CPU, memori, dan permintaan baca/tulis (read/write request) pada lingkungan Cassandra.

31 4 1.4 Manfaat Penelitian Berikut adalah beberapa manfaat dari penelitian ini bagi peneliti, ilmu pengetahuan, dan industri e-commerce . 1.4.1 Manfaat bagi peneliti a. Penelitian ini memberikan kesempatan bagi peneliti untuk mengembangkan keahlian dalam pengembangan sistem automasi yang terintegrasi dengan cloud computing dan praktik DevOps b. Penelitian ini dapat memberikan kontribusi baru terhadap pemahaman tentang penerapan dan integrasi teknologi cloud dan DevOps dalam industri e-commerce, terutama dalam mengoptimalkan pengelolaan sumber daya cloud. 1.4.2 Manfaat bagi ilmu pengetahuan Hasil penelitian ini memiliki potensi untuk mendorong inovasi teknologi lebih lanjut dalam bidang cloud computing dan DevOps. Hal ini mendukung kemajuan ilmu pengetahuan dengan memberikan insight dan metodologi baru dalam

pengelolaan sumber daya cloud yang efisien. 1.4.3 Manfaat bagi industri E-commerce Aplikasi dan metodologi yang dikembangkan dalam penelitian ini akan membantu perusahaan e-commerce meningkatkan efisiensi operasional melalui optimalisasi penggunaan sumber daya cloud. Hal ini memungkinkan perusahaan mengurangi pemborosan biaya dan meningkatkan keberlanjutan operasional mereka dalam lingkungan bisnis yang kompetitif. 1.5 Kebaruan Kebaruan dalam penelitian ini adalah pengembangan sistem automasi berbasis machine learning untuk mengoptimalkan penggunaan sumber daya cloud di GCP, dengan fokus pada lingkungan e-commerce. Sistem ini menggunakan Average Predicted Usage (APU) sebagai metrik utama untuk menentukan rekomendasi machine type berdasarkan analisis penggunaan CPU, memori, serta jumlah permintaan baca/tulis (read/write request) pada lingkungan Cassandra. APU dihitung menggunakan integrasi dengan Particle Swarm Optimization (PSO) untuk menghasilkan prediksi yang lebih akurat. Penelitian ini juga mencakup fitur integrasi dengan Google Sheets untuk menyajikan hasil deteksi dan rekomendasi secara langsung, serta memanfaatkan BigQuery untuk analisis data dan identifikasi sumber daya yang tidak digunakan. Implementasi sistem dilakukan dengan mengadopsi menggunakan praktik DevOps, yang mengintegrasikan otomatisasi pipeline CI/CD, monitoring dengan Datadog, dan API langsung ke GCP. Penelitian ini juga memiliki kebaruan dalam penggunaan machine learning sebagai pendekatan untuk otomatisasi optimasi biaya cloud. Sistem yang dikembangkan mampu menyesuaikan sumber daya berdasarkan pola penggunaan aktual dan mengimplementasikan optimasi berbasis pipeline CI/CD. Dengan pendekatan ini, penelitian menawarkan solusi yang lebih efisien dibandingkan metode manual, sekaligus menyediakan framework yang dapat diadopsi oleh 5 perusahaan e-commerce lainnya dalam mengelola sumber daya cloud secara lebih optimal. 1.6 Kerangka Penulisan Penyusunan tugas akhir ini diatur sesuai dengan pedoman akademik yang berlaku dan terbagi dalam beberapa bab utama sebagai berikut: BAB I PENDAHULUAN Bab ini memaparkan latar belakang masalah, identifikasi

masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, kebaruan penelitian, dan kerangka penulisan. Bab ini memberikan gambaran umum dan pengenalan terhadap topik penelitian. 43 BAB II TINJAUAN PUSTAKA Bab ini menguraikan teori-teori dasar dan landasan ilmiah yang mendukung penelitian. Ini termasuk studi literatur yang relevan dan tinjauan terhadap penelitian-penelitian sebelumnya yang berkaitan dengan topik penelitian. 9 33 BAB III TAHAPAN PELAKSANAAN Bab ini menguraikan Langkah – Langkah pelaksanaan , dan mencakup metode pengujian yang akan digunakan. 9 BAB IV PERANCANGAN Bab ini berfokus pada analisis sistem yang telah ada sebelumnya dan menguraikan rincian tentang spesifikasi kebutuhan sistem baru. Bab ini juga menjelaskan perancangan sistem yang dibuat untuk membangun aplikasi atau sistem baru sesuai dengan kebutuhan penelitian. 1 2 4 5 8 39 BAB V HASIL DAN PEMBAHASAN Bab ini menyajikan data hasil pengujian yang dilakukan dan membahas temuan tersebut secara detail. Bab ini dibagi menjadi dua sub bab, yaitu hasil pengujian A/B testing,hasil pengujian white box dan hasil pengujian confusion matrix, dengan analisis dan kesimpulan yang diambil dari hasil pengujian tersebut. 1 4 6 17 BAB VI KESIMPULAN DAN SARAN Bab ini memberikan kesimpulan dari keseluruhan penelitian dan saran untuk penelitian di masa depan serta pengembangan lebih lanjut dari sistem yang telah dikembangkan. 6 BAB II TINJAUAN PUSTAKA Bab ini berisi analisis terhadap penemuan-penemuan ilmiah terdahulu yang berkaitan langsung dengan masalah utama pada penelitian “Pengembangan Sistem Automasi DevOps untuk Optimalisasi Biaya Cloud di E-commerce . Tujuan utama adalah memberikan gambaran tentang capaian sebelumnya dalam menangani masalah serupa serta menjelaskan bagaimana penelitian ini melengkapi atau mengembangkan pemahaman yang telah ada. 2.1 Pencapaian Terdahulu Tinjauan pustaka berfungsi sebagai landasan bagi kerangka teoritis dan metodologi penelitian. Berikut ini adalah pencapaian terdahulu yang relevan: Tabel 2.1 Pencapaian Terdahulu No Nama (Tahun) Judul Hasil Publikasi 1 Srinkanth, K. (2022) DevOps in Cloud computing: An Overview Penelitian ini menyimpulkan bahwa integrasi DevOps dalam cloud

computing sangat penting untuk pertumbuhan bisnis di era digital. DevOps mengubah cara operasional industri dengan otomatisasi pada setiap langkah, menghasilkan perangkat lunak berkualitas tinggi. Kombinasi DevOps dengan cloud computing mengarah pada aplikasi yang lebih skalabel, biaya operasional yang lebih rendah, dan jangkauan global lebih luas. Studi ini juga menyoroti pentingnya integrasi keamanan dalam pengembangan cloud. IJEAST: International Journal of Engineering Applied Sciences and Technology, 2022 Vol. 6, Issue 10, ISSN No. 2455-2143, Pages 195- 201 2 Krushang Patel, Tirth Chauhan (2023) DevOps and Cloud computing: Best Practices and Integration Challenges Penelitian ini membahas bagaimana DevOps dan cloud computing saling melengkapi dalam revolusi pengembangan perangkat lunak. Praktik DevOps mempercepat pengiriman perangkat lunak yang andal, sementara cloud computing memfasilitasi penyebaran dan skalabilitas aplikasi. 40 Artikel ini mengidentifikasi tantangan JETIR: Journal Of Emerging Technologies and Innovative Research. Volume 10, Issue 6. ISSN: 2349-5162 7 integrasi keduanya dan pentingnya proses berkelanjutan untuk keberhasilan implementasi. 3 Osypanka, P & Nawrocki, P. (2020) Resource Usage Cost Optimization in Cloud computing Using Machine learning Studi ini mengusulkan penggunaan deteksi anomali, machine learning, dan optimasi berbasis Particle Swarm Optimization (PSO) untuk konfigurasi sumber daya cloud yang lebih efisien. Dengan data dari Microsoft Azure, penelitian ini menunjukkan pengurangan biaya hingga 85% dalam 10 bulan. IEEE Transactions on Cloud Computing, 10(3), 2079– 2089. https: //doi.org/1 0.1109/TCC.202 0.3015769 4 Balalaie, A., Heydarnoo ri, A. , Jamshidi, P. (2016) Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture Paper ini membahas pengalaman penulis dalam migrasi arsitektur monolitik ke arsitektur berbasis microservices dalam konteks DevOps. Migrasi ini membawa banyak manfaat seperti peningkatan fleksibilitas dalam menyesuaikan dengan perubahan teknologi dan manajemen sumber daya yang

independen untuk komponen sistem yang berbeda. Mereka mengusulkan pola migrasi berbasis microservices dan menjelaskan bagaimana praktik DevOps memfasilitasi proses migrasi yang mulus. Selain itu, mereka membahas tentang pembuatan pola migrasi microservices berdasarkan pengalaman di berbagai project untuk dapat digunakan kembali dalam praktik migrasi. IEEE Software, 33(3), 42–52. https://doi.org/ 10.1109/MS.20 16.64
5 Fang, D., Liu, X., Liu, L., Yang, H. (2020) OCSO: Off-the-cloud service optimization for green efficient service resource utilization Penelitian ini membahas pendekatan optimasi konsumsi sumber daya layanan dari sisi pengguna melalui OCSO (Off- the-Cloud Service Optimization), yang memungkinkan optimasi proaktif dan berkelanjutan dari sumber daya layanan IaaS dan PaaS melalui API layanan cloud. Eksperimen menunjukkan peningkatan signifikan dalam penggunaan Journal of Green Computing (DOI: 10.1016/j.jogc. 2020.100052) 8 sumber daya dibandingkan dengan opsi auto-scaling yang disediakan oleh penyedia layanan cloud, terutama dalam manajemen beban kerja dinamis. Penelitian ini menyebutkan bahwa batas “green CPU utilization range yang optimal adalah antara 40% hingga 80%, dan VM yang melampaui atau di bawah batas tersebut dianggap tidak efisien dalam penggunaan sumber daya.
2.2 Tinjauan Teoritis Landasan teori dalam penelitian ini mengacu pada kumpulan konsep, prinsip, dan teori yang relevan untuk menjelaskan fenomena yang berkaitan dengan pengembangan sistem automasi DevOps untuk optimalisasi biaya cloud di e- commerce. Landasan ini digunakan sebagai dasar dalam mengembangkan aplikasi, merancang kebijakan, dan melaksanakan kegiatan penelitian. 2.2 20 1 E-commerce E-commerce, atau perdagangan elektronik, adalah proses pembelian dan penjualan barang atau jasa melalui internet, termasuk transfer uang dan data untuk mendukung transaksi ini. 27 Aktivitas e-commerce mencakup berbagai model bisnis, seperti Business-to-Consumer (B2C), Business-to-Business (B2B), Consumer-to- Consumer (C2C), dan Consumer-to-Business (C2B). Pertumbuhan e-commerce yang pesat didorong oleh evolusi teknologi informasi dan

komunikasi, yang memperluas cakupan pasar dan mengubah paradigma perdagangan tradisional. Dalam konteks Indonesia, e-commerce telah menjadi katalis utama dalam pertumbuhan ekonomi, terutama dengan memberikan akses bagi pelaku Usaha Mikro, Kecil, dan Menengah (UMKM) untuk bersaing di pasar global. Menurut Ayu dan Lahmi (2020), e-commerce tidak hanya meningkatkan daya saing UMKM tetapi juga menciptakan lapangan kerja baru dan menarik investasi. Peningkatan akses internet dan penggunaan media sosial di kalangan generasi muda yang melek digital telah mengubah perilaku konsumen, menjadikan transaksi online semakin nyaman dan efisien. Pradana (2021) mencatat bahwa kemudahan metode pembayaran non-tunai dan Cash On Delivery (COD) telah meningkatkan preferensi konsumen terhadap e- commerce. Selain itu, integrasi e-commerce dalam ekosistem digital mendorong inovasi dan peningkatan kualitas layanan, memperkuat posisi Indonesia dalam ekonomi global. Kunthi et al. (2023) menunjukkan bahwa investasi yang didukung kebijakan pemerintah dapat mempercepat pertumbuhan e-commerce di Indonesia, sekaligus memastikan adanya regulasi yang adil untuk melindungi konsumen. Regulasi ini penting untuk menciptakan ekosistem e-commerce yang berkelanjutan dan inklusif. 9 2.2.2 Cloud computing Cloud computing telah menjadi paradigma komputasi fundamental yang mendukung operasi e-commerce secara efektif (Mell & Grance, 2011). Paradigma ini menyediakan akses mudah dan on-demand melalui jaringan ke sekumpulan sumber daya komputasi yang dapat dikonfigurasi, seperti server, penyimpanan, jaringan, dan layanan. Pendekatan ini telah mengubah cara bisnis e-commerce mengelola infrastruktur TI mereka, memberikan fleksibilitas yang signifikan dalam menyesuaikan sumber daya sesuai permintaan pasar yang fluktuatif. Keunggulan utama cloud computing dalam e-commerce adalah pada skalabilitas dan fleksibilitasnya. Bisnis dapat dengan cepat meningkatkan atau mengurangi kapasitas sumber daya mereka sesuai kebutuhan operasional yang dinamis. Hal ini memungkinkan optimalisasi efisiensi operasional dan mengurangi pemborosan

sumber daya. Model pembayaran cloud computing yang berbasis penggunaan (pay-as-you-go) menawarkan efisiensi biaya yang signifikan, menghilangkan kebutuhan investasi awal yang besar pada perangkat keras. Selain efisiensi, cloud computing juga mendukung inovasi melalui kemampuan integrasi teknologi terbaru seperti DevOps, machine learning, dan analitik data real-time. Dalam konteks penelitian ini, cloud computing menyediakan infrastruktur yang mendukung otomatisasi pengelolaan sumber daya dengan memanfaatkan data penggunaan untuk menghasilkan rekomendasi berbasis model prediktif. Hal ini relevan untuk mengidentifikasi sumber daya yang tidak efisien, seperti VM dengan kapasitas berlebih, snapshot yang tidak lagi diperlukan, atau disk yang tidak digunakan, sehingga dapat membantu mengurangi pemborosan biaya. Namun, adopsi cloud computing juga menghadapi tantangan, seperti pengelolaan biaya yang tidak efisien akibat penggunaan sumber daya yang tidak optimal. Dalam penelitian ini, tantangan tersebut menjadi fokus utama, di mana pendekatan berbasis machine learning akan diterapkan untuk memberikan rekomendasi optimal berdasarkan analisis metrik seperti APU. Pendekatan ini memastikan manfaat cloud computing dapat dimaksimalkan tanpa meningkatkan biaya operasional secara signifikan. 2.2.3 Google Cloud Platfrom (GCP) Google Cloud Platform (GCP) adalah rangkaian layanan cloud computing yang disediakan oleh Google, mendukung berbagai kebutuhan komputasi organisasi bisnis, termasuk e-commerce. GCP menawarkan solusi infrastruktur yang skalabel, aman, dan berkinerja tinggi, sehingga perusahaan dapat fokus pada pengembangan produk tanpa perlu memikirkan pengelolaan infrastruktur fisik (Google Cloud, 2024). Dalam penelitian ini, GCP menjadi platform utama untuk mengelola sumber daya cloud, seperti VM, snapshot, disk, dan layanan analitik seperti BigQuery. Gambar 2.1 GCP Logo 10 a. Virtual Machine (Compute Engine) Virtual Machine (VM) di GCP, dikenal sebagai Compute Engine, adalah layanan Infrastructure as a Service (IaaS) yang memungkinkan pengguna menjalankan mesin virtual di infrastruktur Google (Google Cloud, 2024).

Compute Engine memberikan akses on-demand ke sumber daya komputasi yang dapat dikonfigurasi, sehingga pengguna dapat menyesuaikan ukuran dan jenis VM sesuai kebutuhan beban kerja. Fleksibilitas ini memungkinkan perusahaan e-commerce untuk menskalakan sumber daya secara dinamis dan sesuai permintaan, yang sangat penting untuk mendukung kebutuhan operasional yang berfluktuasi. Dalam konteks penelitian ini, Compute Engine menyediakan infrastruktur inti untuk menjalankan sistem otomatisasi berbasis DevOps dan machine learning. Melalui analisis data penggunaan CPU, memori, dan APU, sistem ini dapat mendeteksi VM yang tidak efisien dan merekomendasikan tindakan seperti peningkatan (upgrade) atau pengurangan kapasitas (downgrade). Gambar 2.2 Compute Engine icon b. BigQuery BigQuery adalah solusi data warehouse terkelola sepenuhnya di GCP, dirancang untuk analisis data berskala besar dan real-time (Google Cloud, 2024). Dengan memanfaatkan arsitektur serverless, BigQuery memudahkan perusahaan menjalankan SQL query untuk menganalisis data tanpa perlu mengelola infrastruktur yang kompleks. Fitur ini mendukung perusahaan e- commerce dalam mengoptimalkan strategi bisnis dan memahami pola konsumsi pelanggan. Dalam penelitian ini, BigQuery digunakan untuk menganalisis data terkait penggunaan sumber daya cloud, seperti memantau VM yang tidak aktif, mendeteksi snapshot yang tidak lagi relevan, dan memvalidasi efisiensi disk serta IP address. Kemampuan BigQuery dalam memproses data besar secara cepat memungkinkan identifikasi sumber daya yang tidak efisien, yang menjadi dasar rekomendasi optimasi biaya. Gambar 2.3 BigQuery Icon 11 2.2 11 4 DevOps DevOps adalah konsep yang mengintegrasikan pengembangan (Development) dan operasi (Operations) dalam teknologi informasi, dengan tujuan mempercepat siklus hidup pengembangan perangkat lunak serta meningkatkan kualitas produk secara keseluruhan (Bass, Weber, & Zhu, 2015). Dengan pendekatan ini, kolaborasi antara tim pengembangan dan operasi diperkuat, memungkinkan otomatisasi proses dari tahap pengembangan hingga operasional. Gambar 2.4 DevOps Dalam konteks e-commerce yang dinamis,

DevOps memungkinkan perusahaan untuk merespons perubahan kebutuhan pasar dengan cepat, mengimplementasikan fitur baru, serta memperbarui sistem tanpa mengorbankan stabilitas atau keandalan layanan. Praktik DevOps mendukung pengembangan, pengujian, dan penerapan aplikasi secara efisien melalui pendekatan Continuous Integration dan Continuous Deployment (CI/CD). Melalui CI/CD, perubahan kode dapat diintegrasikan secara berkelanjutan ke dalam repositori bersama, diikuti oleh pengujian otomatis untuk memastikan stabilitas, performa, dan keamanan aplikasi. Dalam penelitian ini, DevOps menjadi pendekatan utama untuk mendukung otomatisasi pengelolaan sumber daya cloud. Praktik CI/CD memungkinkan pengujian dan penerapan algoritma berbasis machine learning secara terstruktur, termasuk model prediksi berbasis APU yang digunakan untuk memberikan rekomendasi optimasi sumber daya. Dengan otomatisasi ini, proses pengelolaan VM, snapshot, dan sumber daya lain dapat dilakukan secara lebih cepat, akurat, dan efisien. 2.2.5 Gitlab / Gitlab C I Gambar 2.5 Gitlab logo GitLab adalah platform DevOps yang menyediakan alat kolaborasi dan kontrol versi berbasis Git, memungkinkan tim pengembangan mengelola kode dan project secara efisien dalam satu platform terpadu (GitLab, 2024). 45 GitLab mendukung berbagai bahasa pemrograman, seperti Python, Java, Ruby, dan Node.js. 23 Salah satu fitur utamanya adalah GitLab CI/CD (Continuous Integration/Continuous Deployment), sebuah pipeline otomatis untuk proses integrasi dan pengiriman berkelanjutan. Dengan menggunakan file konfigurasi 12 .gitlab-ci.yml yang ditulis dalam format YAML, pengembang dapat mengatur tahapan CI/CD, termasuk build, test, dan deploy. Proses ini memastikan kualitas kode tetap terjaga sambil mempercepat siklus pengembangan (GitLab, 2024). Dalam konteks pengembangan sistem automasi DevOps untuk optimasi biaya cloud, GitLab CI memungkinkan pelaksanaan tugas-tugas otomatis yang dapat dijadwalkan secara fleksibel. Dengan fitur GitLab Scheduler, tugas-tugas tertentu dapat dijalankan secara otomatis pada interval waktu yang ditentukan. Hal ini relevan untuk

pengelolaan sumber daya cloud, di mana sistem dapat secara dinamis menyesuaikan alokasi sumber daya berdasarkan analisis kebutuhan aktual. Kemampuan ini membantu mengurangi biaya operasional cloud dan meningkatkan efisiensi pengelolaan sumber daya, seperti mengatur VM, snapshot, dan penggunaan disk yang optimal. Dalam penelitian ini, GitLab CI menjadi komponen kunci untuk mendukung otomatisasi pipeline pengembangan. Pipeline ini digunakan untuk mengelola proses pengumpulan data, pelatihan model machine learning, hingga penerapan sistem berbasis APU. Dengan otomatisasi ini, sistem dapat dijalankan secara efisien dan memastikan bahwa setiap perubahan kode atau model dapat diuji dan diterapkan secara cepat tanpa mengorbankan kualitas. 2.2 29 6 Python Gambar 2.6 Python Logo Python adalah bahasa pemrograman tingkat tinggi yang dikembangkan oleh Guido Van Rossum dan pertama kali dirilis pada tahun 1991. 32 Python telah menjadi salah satu bahasa pemrograman paling populer, khususnya dalam pengembangan cloud dan e-commerce (Van Rossum & Drake, 2009). Bahasa ini berorientasi objek dan memiliki semantik dinamis, sehingga cocok untuk berbagai aplikasi, mulai dari analisis data hingga pengembangan sistem berbasis kecerdasan buatan. Keunggulan utama Python terletak pada sintaksnya yang sederhana dan mudah dipelajari, sehingga meningkatkan produktivitas dan efisiensi programmer. Python juga memiliki ekosistem pustaka yang luas, seperti pandas, NumPy, scikit-learn, dan TensorFlow, yang mendukung pengolahan data, pembelajaran mesin, serta pengembangan sistem berbasis DevOps. Fleksibilitas ini menjadikan Python pilihan utama dalam penelitian berbasis data besar dan sistem otomatisasi. Dalam konteks penelitian ini, Python digunakan untuk membangun pipeline sistem, mulai dari pengumpulan data, preprocessing, hingga pelatihan model machine learning. Bahasa ini juga mendukung integrasi dengan API GCP, BigQuery, dan GitLab CI untuk memungkinkan sistem bekerja secara otomatis dan terkoordinasi. Dengan pustaka seperti scikit-learn dan pandas, Python mempermudah analisis metrik APU serta pengembangan algoritma prediktif untuk mendukung

optimasi sumber daya cloud. 13 2.2 7 7 Visual Code Studio Gambar 2.7 Visual Code Studio Logo Visual Studio Code (VS Code), yang dikembangkan oleh Microsoft, telah menjadi salah satu lingkungan pengembangan terintegrasi (Integrated Development Environment atau IDE) paling populer di kalangan pengembang, termasuk dalam pengembangan cloud dan e-commerce. Menurut dokumentasi Microsoft (2021), VS Code menawarkan fitur canggih seperti dukungan untuk berbagai bahasa pemrograman, pengeditan kode yang efisien, debugging terintegrasi, dan kemampuan ekstensi yang luas. Fitur-fitur ini menjadikan VS Code sebagai alat yang sangat bermanfaat dalam pengembangan sistem yang kompleks. Kemampuannya untuk mendukung integrasi dengan Git dan GitLab, serta kompatibilitas dengan berbagai pustaka dan framework Python, memungkinkan pengembang untuk bekerja secara kolaboratif dalam project DevOps dan cloud computing. Dalam penelitian ini, VS Code digunakan sebagai alat utama untuk menulis dan mengelola kode, termasuk pipeline CI/CD, dan skrip Python 2.2.8 Application Programming Interface (API) Application Programming Interface (API) memegang peran kunci dalam ekosistem digital modern, khususnya dalam pengembangan cloud dan e-commerce. API adalah sekumpulan protokol dan definisi yang memungkinkan aplikasi perangkat lunak untuk berkomunikasi dan berbagi data atau fungsi (Fielding & Taylor, 2002). Dalam konteks e-commerce, API mendukung integrasi dengan berbagai layanan seperti sistem pembayaran, pengelolaan inventaris, dan penyedia logistik. Hal ini menciptakan ekosistem yang terhubung dan responsif, di mana data dapat ditransfer dengan lancar antara aplikasi, meningkatkan efisiensi operasional dan pengalaman pengguna. Dalam cloud computing, API memungkinkan aplikasi berinteraksi langsung dengan infrastruktur cloud, mengakses sumber daya komputasi, penyimpanan, dan layanan analitik secara dinamis. Platform seperti Google Cloud Platform (GCP) menyediakan API untuk mempermudah pengelolaan VM, snapshot, disk, dan layanan lainnya. Dalam penelitian ini, API menjadi komponen penting untuk memungkinkan integrasi otomatis antara pipeline

sistem dengan layanan GCP dan BigQuery, mendukung sistem berbasis DevOps yang efisien. 2.2.9 Machine learning Machine learning adalah teknologi yang memungkinkan sistem untuk belajar dari data tanpa instruksi eksplisit dari pengguna (Takdirillah, 2020). Dalam konteks cloud computing, machine learning memainkan peran penting dalam mengidentifikasi sumber daya yang kurang efisien dan memberikan rekomendasi untuk optimasi. Penelitian ini menggunakan machine learning untuk menganalisis 14 pola penggunaan CPU dan memori pada VM Cassandra guna menghasilkan rekomendasi terkait jenis machine type yang lebih efisien. Proses machine learning yang digunakan dalam penelitian ini diilustrasikan pada Gambar 2.8. Setelah data dikumpulkan melalui proses data collection, data diproses dalam tahap data preprocessing untuk memastikan format yang optimal, termasuk normalisasi dan perhitungan nilai puncak (peak values). Data yang telah diproses kemudian dianalisis menggunakan algoritma Isolation Forest untuk mendeteksi anomali, seperti pola penggunaan sumber daya yang tidak efisien. Selanjutnya, model Boosted Decision Tree digunakan untuk melatih data yang telah melalui proses deteksi anomali, menghasilkan prediksi tipe VM yang optimal. Evaluasi model dilakukan dengan Mean Squared Error (MSE) untuk mengukur akurasi prediksi, diikuti oleh proses optimasi menggunakan algoritma Particle Swarm Optimization (PSO) guna menyempurnakan rekomendasi berdasarkan kebutuhan aktual. Proses ini diakhiri dengan menghasilkan rekomendasi tipe VM yang sesuai untuk meningkatkan efisiensi dan menekan biaya operasional. 36 2.2 36 10 Data Collection Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari platform Datadog. Dalam konteks penelitian ini, data yang dikumpulkan mencakup metrik penggunaan CPU, memori, serta jumlah permintaan baca dan tulis (read/write requests) pada VM Cassandra yang dioperasikan di GCP. Gambar 2.8 Flow Machine Learning 15 Pengumpulan data dilakukan selama periode 10 bulan, dari Januari 2024 hingga Oktober 2024. Data ini memberikan representasi lengkap

dari pola penggunaan sumber daya pada lingkungan cloud, yang menjadi dasar untuk analisis dan pengembangan model prediktif dalam penelitian ini. 2.2.11 Data Preprocessing Data preprocessing adalah tahap persiapan data mentah untuk memastikan bahwa data yang digunakan dalam analisis dan pelatihan model memiliki kualitas yang optimal. Proses ini melibatkan beberapa langkah utama yang dirancang untuk meningkatkan keakuratan dan efisiensi model (Salam et al., 2020, p. 532). Langkah pertama adalah data cleansing, yaitu proses menghilangkan data yang tidak lengkap, tidak sesuai, atau mengandung anomali. Dengan menghilangkan data yang tidak valid, dataset yang dihasilkan menjadi lebih bersih dan konsisten, sehingga mendukung analisis yang lebih akurat. Selanjutnya, data yang telah dibersihkan diolah menggunakan teknik normalisasi untuk memastikan bahwa nilai dari berbagai metrik memiliki skala yang seragam. Dalam penelitian ini, normalisasi dilakukan menggunakan teknik min-max scaling. Teknik ini mentransformasi nilai setiap metrik ke dalam rentang 0 hingga 1, dengan menggunakan rumus berikut: $x' = x - x_{!"\#} \times !\$\% - x_{!"\#}$ di mana x' adalah n ilai hasil normalisasi, x adalah nilai aktual, $x_{!"\#}$ adalah nilai maksimum, dan $x_{!\$\%}$ adalah nilai minimum dari metrik tersebut. Normalisasi in i penting untuk memastikan bahwa data dari berbagai metrik tidak mendominasi analisis atau model karena perbedaan skala, yang dapat meningkatkan performa model (Suryanegara et al., 2021, p. 117). Selain itu, data preprocessing mencakup penghitungan nilai puncak (peak values) untuk setiap metrik. Nilai puncak ini memberikan informasi tentang penggunaan maksimum sumber daya, yang relevan untuk memahami kebutuhan kapasitas pada skenario penggunaan tertinggi. Indikator kinerja utama yang digunakan dalam penelitian ini adalah Average Predicted Usage (APU), yang mencerminkan rata-rata penggunaan CPU, memori, serta permintaan baca dan tulis pada lingkungan Cassandra. Sebagai bagian dari proses normalisasi, data harian digunakan untuk memastikan analisis yang lebih detail dan akurat. Dengan fokus pada daily

average, analisis ini mengurangi risiko bias yang dapat muncul dari variasi ekstrem pada tingkat harian, sekaligus memberikan pemahaman yang lebih stabil terkait pola penggunaan sumber daya. 2.2.12 Pembentukan Model Pada tahap ini, dua algoritma utama diterapkan dalam pembuatan model, yaitu Isolation Forest untuk deteksi anomali dan Boosted Decision Tree untuk menghitung nilai APU yang digunakan oleh PSO dalam menentukan konfigurasi machine type yang optimal. 16 1. Isolation Forest Isolation Forest adalah algoritma yang digunakan untuk mendeteksi anomali dalam pola penggunaan CPU pada VM. Anomali ini mencakup penggunaan CPU yang sangat rendah atau sangat tinggi dibandingkan dengan kapasitas yang disediakan. Algoritma ini bekerja dengan membangun pohon keputusan yang secara efektif “mengisolasi” titi k data anomali lebih cepat daripada data normal, menjadikannya ideal untuk mendeteksi outliers pada dataset yang besar (Liu, Ting, & Zhou, 2008). Dalam penelitian ini, Isolation Forest digunakan untuk menghapus 0.01% data yang terdeteksi sebagai outliers, memastikan dataset bersih untuk pelatihan model selanjutnya. Rumus utama yang digunakan dalam Isolation Forest untuk menghitung skor isolasi adalah: Score = − $E(h(x))$ $c(n)$ di mana: $E(h(x))$ adalah rata-rata ked alaman pohon untuk titik data x $c(n)$ adalah nilai normalisas i untuk sampel n 2. Boosted Decision Tree Boosted Decision Tree adalah algoritma yang digunakan untuk menghitung nilai APU berdasarkan pola penggunaan CPU, memori, serta permintaan baca dan tulis (read /write requests). Nilai APU yang dihasilkan dari algoritma ini digunakan oleh PSO untuk menentukan konfigurasi machine type 1 ang optimal. Algoritma ini bekerja dengan membangun sejumlah pohon keputusan secara bertahap, di mana setiap pohon baru dilatih untuk mengoreksi kesalahan yang dibuat oleh pohon sebelumnya (Fr edman, 2001). Prediksi akhir dari model Boosted Decision Tree dapat dinyatakan sebagai: $F(x)$ = - ! h ! (x) & !'(di mana: F (x) adalah predi ksi akhir dari model, M adalah jumlah pohon, ! adalah bobo

t dari pohon ke- m , h ! (x) adalah prediksi dari pohon ke- m . Proses optimasi dalam Boosted Decision Tree bertujuan untuk meminimalkan Mean Squared Error (MSE), yang digunakan sebagai fungsi loss selama pelatihan. Penggunaan MSE sebagai metrik evaluasi memastikan model memberikan prediksi APU yang akurat, mendukung proses optimasi konfigurasi sumber daya cloud oleh PSO. Dengan iterasi berkelanjutan, algoritma ini mampu menangkap pola kompleks dalam data, menjadikannya ideal untuk aplikasi dalam penelitian ini (Chen & Guestrin, 2016). 17 2.2.13 Evaluasi Model Evaluasi model dilakukan dengan menggunakan metrik Mean 2 quared Error (MSE). MSE adalah metrik evaluasi yang paling umum digunakan dalam model regresi, karena mengukur rata-rata dari kuadrat perbedaan antara nilai predik i dan nilai aktual. Dalam penelitian ini, MSE digunakan untuk menilai akurasi prediksi dari algoritma Boosted Decision Tree dalam menghitung Average Predicted Usage (APU), yang menjadi dasar rekomendasi tipe VM. Rumus MSE yang digunakan adalah sebagai ber kut: MSE = 1 n -(y " −y ^ ") * # " '(di mana: y $: nilai aktual y ^ $: nilai prediksi n : jumlah data MSE yang lebih renda h menunjukkan bahwa model memiliki prediksi yang lebih akurat dan sesuai dengan nilai sebenarnya. Dalam konteks penelitian ini, optimasi MSE sangat penting untuk memastikan bahwa prediksi APU mendekati nilai aktual. Hal ini memungkinkan sistem memberikan rekomendasi tipe VM yang lebih efisien, menghindari alokasi sumber daya yang berlebihan (overprovisioning) atau kurang optimal (underprovisioning). Keunggulan utama MSE adalah kemampuannya untuk memperbesar pengaruh kesalahan prediksi besar, karena kesalahan tersebut dikuadratkan dalam perhitungan. Dengan demikian, model yang dioptimalkan untuk MSE akan lebih fokus dalam meminimalkan kesalahan besar, sehingga meningkatkan kualitas rekomendasi tipe VM yang dihasilkan. Dalam penelitian ini, optimasi MSE dilakukan selama tahap pelatihan algoritma Boosted Decision Tree, sehingga model dapat memberikan rekomendasi tipe VM yang sesuai berdasarkan pola

penggunaan CPU, memori, serta jumlah p rmintaan baca dan tulis. 2.2.14 Particle Swarm Optimization (PSO) Particle Swarm Optimization (PSO) adalah algoritma optimasi berbasis populasi yang meniru perilaku sosial kawanan, seperti burung yang mencari sumber makanan atau ikan yang bergerak secara kolektif. PSO digunakan untuk menemukan solusi optimal di ruang pencarian besar dengan cara memperbarui posisi dan kecepatan partikel berdasarkan informasi individu dan kolektif kawanan (Kennedy & Eberhart, 1995). Dalam penelitian ini, PSO digunakan untuk mengoptimalkan hasil prediksi APU dari Boosted Decision Tree guna menentukan konfigurasi machine 13 type VM yang paling efisien. PSO bekerja dengan konsep partikel yang mewakili calon solu i dalam ruang pencarian. Setiap partikel memiliki posisi dan kecepatan yang diperbarui secara iteratif berdasarkan posisi terbaik yang ditemukan oleh partikel itu sendiri (personal best) dan posisi terbaik global yang ditemukan oleh seluruh kawanan (global best). Rumus untuk memperbarui kecepatan dan posisi partikel adalah sebagai berikut: 18 v " (t+1) = w ·v " (t)+c (·r (·(p " −
 ")+c * ·r * ·(g−x ") x " (t+1) = x " (t)+v " (t+1) di mana: v $ (t + 1) : Kecepa tan partikel i pada iterasi 3 t + 1. w: Bobot inersia, mengon trol kontribusi kecepatan sebelumnya. c ' dan c (: Koefisie n untuk faktor kognitif dan sosial. r ' dan r (: Nilai aca k antara 0 dan 1. p $: Posisi terbaik yang ditemukan oleh partikel i . g : Posisi terbaik global yang ditemukan ol eh seluruh kawanan. Dalam penelitian ini, PSO digunakan untuk menemukan konfigurasi tipe VM yang paling efisien dalam hal biaya dan performa. Misalnya, jika VM menggunakan CPU jauh di bawah kapasitas yang disediakan selama periode tertentu, Isolation Forest akan menandainya sebagai sumber daya yang tidak efisien. Boosted Decision Tree, yang didukung oleh PSO, kemudian memberikan rekomendasi untuk menurunkan machine type VM tersebut agar lebih sesuai dengan

kebutuhan aktual. Proses ini memungkinkan pengurangan biaya operasional tanpa mengorbankan performa sistem. 2.2.15 Generate Recommendations Tahap Generate Recommendations adalah langkah akhir dalam proses analisis dan optimasi penggunaan sumber daya VM. Pada tahap ini, sistem menghasilkan rekomendasi berbasis data yang konkret berdasarkan hasil deteksi anomali, pembentukan model, evaluasi, dan optimasi. Rekomendasi ini bertujuan untuk meningkatkan efisiensi penggunaan sumber daya sekaligus mengurangi biaya operasional. Rekomendasi yang diberikan sistem mencakup tiga tindakan utama: a. Upgrade: Jika hasil analisis menunjukkan bahwa penggunaan sumber daya melebihi 80% dari kapasitas VM saat ini, sistem akan merekomendasikan upgrade untuk meningkatkan kapasitas CPU, memori, atau jenis VM agar memenuhi kebutuhan aktual. b. Downgrade: Jika penggunaan sumber daya berada di bawah 40% dari kapasitas VM, sistem akan merekomendasikan downgrade untuk menyesuaikan kapasitas VM dengan kebutuhan aktual, sehingga mengurangi pemborosan biaya. c. Maintain: Jika penggunaan sumber daya berada dalam batas optimal (40%- 80%), sistem akan merekomendasikan untuk mempertahankan (maintain) konfigurasi VM saat ini tanpa perubahan. Proses ini menggabungkan prediksi APU yang dihasilkan oleh model Boosted Decision Tree dengan optimasi berbasis Particle Swarm Optimization (PSO). Setelah rekomendasi awal (upgrade, downgrade, atau maintain) ditentukan, PSO digunakan untuk membandingkan konfigurasi machine type yang direkomendasikan dengan konfigurasi saat ini. Sistem kemudian mencari jenis machine type terdekat yang tersedia dalam katalog GCP untuk memenuhi konfigurasi yang dihasilkan. Jika tidak ada machine type yang sesuai dengan hasil 19 optimasi, maka sistem akan mempertahankan konfigurasi as-is guna memastikan kesesuaian kapasitas sumber daya tanpa mengorbankan performa aplikasi. 2.2.16 Cassandra Gambar 2.9 Cassandra logo Apache Cassandra adalah basis data terdistribusi yang dirancang untuk menangani data dalam skala besar dengan tingkat ketersediaan tinggi. Dikembangkan oleh Facebook dan sekarang dikelola

oleh komunitas open-source di bawah naungan Apache Software Foundation (Apache Cassandra, 2023), Cassandra memungkinkan penyimpanan data terstruktur dengan arsitektur masterless, di mana setiap node dalam cluster memiliki peran yang sama. Hal ini memungkinkan skalabilitas yang mudah di berbagai data center tanpa mengorbankan performa atau ketersediaan. Dalam konteks penelitian ini, data yang digunakan berasal dari sumber daya VM yang difungsikan sebagai node Cassandra. VM ini digunakan untuk menjalankan layanan basis data Cassandra yang dirancang khusus untuk menangani kebutuhan penyimpanan data dalam skala besar pada aplikasi e-commerce. Cassandra di sini digunakan sebagai database utama untuk menyimpan data penting dalam operasional e-commerce, seperti data chat pengguna, log aktivitas sistem, riwayat transaksi, serta katalog produk yang sering diakses. Penelitian ini fokus pada analisis performa VM yang digunakan untuk menjalankan Cassandra, terutama dalam hal penggunaan CPU dan memori, yang merupakan komponen utama dalam menentukan konfigurasi optimal VM. Dua metrik yang sangat diperhatikan dalam konteks ini adalah: • Write Requests (CPU Usage) Permintaan tuli s pada Cassandra membutuhkan daya komputasi yang signifikan dari CPU. Setiap operasi tulis, seperti pencatatan transaksi atau penyimpanan data chat, membutuhkan CPU yang cukup untuk memastikan data dapat ditulis dengan cepat dan konsisten ke dalam cluster Cassandra. • Read Request s (Memory Usage) Permintaan baca pada Cassandra sangat terkait dengan kapasitas memori. Data yang sering diakses, seperti informasi katalog produk atau data log, biasanya di-cache di memori agar dapat diakses lebih cepat. Dengan memori yang memadai, Cassandra dapat memberikan waktu respons yang cepat, terutama ketika volume permintaan baca meningkat. 20 2.2.17 Datadog Gambar 2.10 Datadog logo Datadog adalah platform pemantauan dan analitik yang dirancang untuk memberikan visibilitas penuh terhadap kinerja aplikasi dan infrastruktur di cloud. Platform ini mendukung pengumpulan metrik secara real-time, memungkinkan perusahaan untuk memantau dan menganalisis penggunaan sumber daya

seperti CPU, memori, disk, serta performa aplikasi seperti latensi permintaan baca dan tulis (Datadog, 2024). Dalam penelitian ini, Datadog berperan sebagai alat utama untuk mengumpulkan data pemantauan dari VM Cassandra. Metrik yang dikumpulkan dari Datadog digunakan dalam tahap Data Collection dan diproses lebih lanjut dalam tahapan Machine learning untuk membangun model prediksi. Beberapa fitur penting yang disediakan oleh Datadog dalam konteks penelitian ini meliputi: a. Real-time Data Datadog menyediakan data real-time yang mencakup penggunaan CPU, memori, serta jumlah permintaan baca dan tulis pada setiap VM. Data ini digunakan untuk menganalisis pola penggunaan dan menentukan tindakan optimasi sumber daya yang diperlukan. b. Integration with Cloud Infrastructure Datadog mendukung integrasi dengan berbagai layanan infrastruktur cloud, termasuk Cassandra, yang memungkinkan pengumpulan data dari berbagai sumber dalam satu platform, memberikan gambaran holistik tentang kinerja sistem. 21 BAB III TAHAPAN PELAKSANAAN 3.1 Langkah-langkah Pelaksanaan Dalam kerangka pengembangan sistem automasi DevOps untuk optimalisasi biaya cloud di e-commerce, terdapat serangkaian langkah pelaksanaan yang terstruktur dan sistematis. Berikut adalah detail dari masing-masing tahap pelaksanaan yang dijelaskan pada gambar 3.1 Gambar 3.1 Tahapan Pelaksanaan Tahap-tahap pelaksanaan ini dirancang untuk mendukung pengembangan sistem yang efektif dan efisien, dimulai dari dentifikasi kebutuhan hingga penyusunan laporan akhir. Berikut adalah penjelasan mendalam mengenai mas ng-masing tahapan yang terdapat dalam gambar tersebut: 1. Analisis kebutuhan Tahap pertama dalam pelaksanaan sistem adalah analisis kebutuhan. Di sini, identifikasi kebutuhan bisnis, teknis, dan operasional dilakukan untuk memahami permasalahan yang dihadapi dalam pengelolaan sumber daya cloud di lingkungan e-commerce. Fokus utama adalah optimalisasi biaya cloud, di mana tantangan pengelolaan infrastruktur dan sumber daya yang efisien menjadi prioritas utama dalam perancangan sistem. 2. Studi literatur Pada tahap ini, dilakukan pengumpulan pengetahuan terkait

teori dan konsep yang relevan dengan penelitian, termasuk DevOps, cloud computing, dan teknik optimisasi biaya di lingkungan e-commerce. Penelitian sebelumnya, jurnal ilmiah, dan literatur lainnya digunakan untuk memberikan dasar yang kuat bagi perancangan dan implementasi sistem yang lebih baik. 3. Pengumpulan data 22 Tahap pengumpulan data melibatkan pengumpulan data terkait penggunaan sumber daya cloud yang relevan dalam pengelolaan infrastruktur e-commerce. Data yang dikumpulkan meliputi informasi mengenai konfigurasi VM, penggunaan sumber daya, serta biaya operasional yang dikeluarkan selama periode tertentu. Data ini digunakan untuk analisis lebih lanjut dalam proses pelatihan model dan optimasi biaya. 4. Implementasi system Tahap ini melibatkan penerapan praktik DevOps dalam sistem yang telah dikembangkan. Fokus utamanya adalah mengintegrasikan proses otomatisasi, seperti continuous integration dan continuous deployment (CI/CD), dengan alur kerja DevOps di lingkungan e-commerce. Ini termasuk pengujian berkelanjutan dan penerapan perubahan otomatis ke dalam produksi, sehingga meningkatkan efisiensi dan mengurangi intervensi manual dalam pengelolaan infrastruktur. 5. Implementasi DevOps Tahap ini melibatkan penerapan praktik DevOps dalam sistem yang telah dikembangkan. Fokus utamanya adalah mengintegrasikan proses otomatisasi, seperti continuous integration dan continuous deployment (CI/CD), dengan alur kerja DevOps di lingkungan e-commerce. Ini termasuk pengujian berkelanjutan dan penerapan perubahan otomatis ke dalam produksi, sehingga meningkatkan efisiensi dan mengurangi intervensi manual dalam pengelolaan infrastruktur. 6. Analisis hasil Setelah implementasi DevOps, tahap selanjutnya adalah melakukan analisis hasil untuk mengevaluasi efektivitas sistem yang telah diterapkan. Analisis ini bertujuan untuk menilai seberapa baik sistem dapat mengidentifikasi dan mengurangi pemborosan sumber daya cloud serta dampaknya terhadap pengurangan biaya operasional dalam lingkungan e-commerce. Hasil analisis 5 ni juga digunakan untuk menyempurnakan sistem lebih lanjut. 7. Kesimpulan Berdasarkan hasil

analisis, ditarik kesimpulan mengenai efisiensi dan efektivitas sistem automasi DevOps yang telah dikembangkan. Kesimpulan ini mencakup sejauh mana sistem berhasil mengoptimalkan penggunaan sumber daya cloud dan mengurangi biaya operasional di sektor e-commerce. Temuan ini memberikan dasar yang jelas bagi perusahaan dalam menerapkan sistem serupa untuk kebutuhan mereka. 8. Laporan/Dokumentasi Tahap akhir melibatkan penyusunan laporan atau dokumentasi yang mencakup seluruh proses pengembangan sistem, mulai dari analisis kebutuhan hingga hasil implementasi dan evaluasi. Laporan ini juga mencakup deskripsi rinci mengenai tahapan pelaksanaan, tantangan yang dihadapi, solusi yang diterapkan, serta hasil dan temuan penting selama penelitian yang dapat digunakan oleh pihak terkait dalam pengelolaan sumber daya cloud secara lebih efisien. 3.2 Metode Pengujian Pada bagian ini menguraikan tentang metodologi pengujian yang diterapkan dalam penelitian ini, yang mencakup dua pendekatan utama: pengujian pengujian 23 white box dan metode confusion matrix. Penjelasan terperinci tentang masi 22 g- masing metode akan disajikan dalam sub-bab berikut. 3.2.1 Metode A/B Testing A/ B Testing adalah metode pengujian yang membandingkan dua versi sistem atau parameter untuk menentukan mana yang me berikan hasil lebih baik berdasarkan metrik tertentu. Metode ini sering digunakan untuk mengevaluasi perubahan sistem, model, atau konfigurasi, dan pengaruhnya terhadap metrik performa seperti biaya, efisiensi, atau akurasi (Kohavi et al., 2020). Dalam konteks penelitian ini, A/B Testing diterapkan untuk membandingkan biaya penggunaan sumber daya cloud sebelum dan sesudah diterapkannya metode optimisasi berbasis PSO. Pengujian ini melibatkan langkah-langkah sebagai berikut: 1. Pengumpulan Data Data biaya sebelum dan sesudah optimisasi dihitung menggunakan data penggunaan sumber daya cloud dan tipe mesin virtual yang direkomendasikan oleh PSO. 2. Perhitungan Biaya Menghitung total biaya untuk kondisi awal (Cost Before) dan setelah optimisasi (Cost After PSO). 3. Analisis Penghematan Menghitung penghematan biaya baik dalam

nilai absolut (USD) maupun persentase, dengan rumus sebagai berikut: S
$avings= (USD) = =Cost=Bef$
$re -Cost=After=PSO\ Savin$
$gs= (\%) == Savings= (USD) C$
$ost=Before =x$=100 4. Perhitungan Optimizat
ion Recommendation Selain penghematan biaya, tahap ini juga mencakup perhitungan optimization recommendation. Sistem akan mengevaluasi rekomendasi yang dihasilkan oleh PSO, kemudian mengklasifikasikan rekomendasi tersebut sebagai valid atau tidak valid berdasarkan kriteria yang telah ditentukan. Hasil akhirnya akan menunjukkan persentase rekomendasi yang valid dibandingkan dengan total rekomendasi yang dihasilkan oleh sistem. a. Valid Recommendation: Rekomendasi yang menghasilkan penghematan biaya dan memenuhi kriteria performa yang ditetapkan. b. Invalid Recommendation: Rekomendasi yang tidak memberikan penghematan biaya atau tidak sesuai dengan kebutuhan operasional yang sebenarnya. Persentase rekomendasi yang valid dihitung dengan rumus berikut: $valid=recommendatio$
$s= (\%) == Number=of=valid=$
$ecommendations\ Total=r$
$commendations =x$=100 Metode in
i mendukung pengambilan keputusan berbasis data untuk mengevaluasi efektivitas metode optimisasi yang diterapkan. A/B Testing memberikan bukti empiris apakah perubahan konfigurasi sistem melalui PSO 24 benar-benar memberikan dampak positif terhadap biaya operasional. Menurut Kohavi et al. (2020), A/B Testing adalah metode yang sangat direkomendasikan untuk mengukur dampak perubahan pada sistem berbasis data karena kemampuannya untuk memberikan hasil yang objektif, valid, dan dapat diandalkan dalam pengujian hipotesis. 3.2.2 Metode White box Pengujian white box, sebagaimana dijelaskan oleh Khan, Mohd.E. (2010), berfokus pada analisis struktur internal dan detail teknis perangkat lunak. Metode ini melibatkan pemberian input ke dalam

sistem dan mengamati bagaimana sistem tersebut mengolah input untuk menghasilkan keluaran yang diharapkan. Pengujian white box memerlukan pemahaman mendalam mengenai kode sumber sistem dan umumnya diterapkan pada tingkat integrasi, unit, serta sistem secara keseluruhan selama proses pengujian. Dalam konteks penelitian ini, white box testing diterapkan untuk unit testing pada bagian-bagian tertentu dari sistem automasi DevOps yang telah dikembangkan. Pengujian ini difokuskan untuk memastikan bahwa setiap unit atau modul dalam sistem berfungsi dengan benar sesuai dengan tujuan yang telah ditentukan, terutama dalam pengolahan input yang terkait dengan algoritma dan perhitungan penghematan biaya. Unit testing dilakukan untuk memverifikasi bahwa setiap bagian dari kode, termasuk komponen sistem yang mengimplementasikan algoritma seperti PSO dan Boosted Decision Tree, berfungsi sesuai dengan ekspek 2 asi. Ini dilakukan dengan menguji unit-unit kode secara terpisah untuk memverifikasi bahwa mereka memberikan keluaran yang benar ketika diberikan input yang sudah diketahui. 3.2.3 Metode Confusion Matrix Confusion matrix adalah metode evaluasi yang digunakan untuk mengukur kinerja model klasifikasi berdasarkan kombinasi nilai sebenarnya (actual) dan nilai prediksi (prediction) yang dihasilkan oleh model. Confusion matrix sangat berguna dalam masalah klasifikasi dengan dua atau lebih kelas, karena memberikan inf 3 rmasi mengenai hasil prediksi yang benar maupun salah (Putra, Suprapto, & Bukhori, 2022). Matriks ini terdiri dari empat tribut utama yang menjadi dasar perhitungan, yaitu: 1. True Positive (TP): Jumlah data yang diklasifikasikan sebagai positif bai pada kategori prediksi maupun kategori sebenarnya. 2. False Positive (FP): Jumlah data yang diklasifikasikan sebagai positif pa 3 34 a kategori prediksi tetapi sebenarnya bernilai negatif. 3. True Negative (TN): Jumlah data yang diklasifikasikan sebagai negatif baik pada ategori prediksi maupun kategori sebenarnya. 4. False Negative (FN): Jumlah data yang diklasifikasikan sebagai negatif pada kategori prediksi tetapi sebenarnya bernilai

positif. Confusion matrix memungkinkan penghitungan berbagai metrik evaluasi, seperti akurasi, presisi, recall, dan F1 35 score, yang memberikan pemahaman lebih mendalam terkait kelebihan dan kelemahan model dalam memprediksi data. Perhit ngan dari keempat atribut ini dijelaskan sebagai berikut: 1. Akurasi 25 Akurasi adalah pendekatan evaluasi yang mengukur seberapa baik prediksi model sesuai dengan nilai aktual. Dengan menghitung jumlah data yang diklasifikasikan dengan benar, tingkat keakuratan dari prediksi tersebut dapat ditentukan. $accu$ $racy = ' TP + TN\ TP + TN + FP +$ FN 2. Presisi Presisi merupakan metode evaluasi yang membandingka n jumlah informasi relevan yan 16 ditemukan oleh sistem dengan total informasi yang diperoleh, termasuk yang relevan maupun yang tidak relevan. Formula untuk menghitung presisi adalah sebagai berikut: pr $ecision == TP\ TP + FP$ 3. Recal Recall adalah metode evaluasi yang membandingkan jumlah informasi relevan 30 ang berhasil ditemukan oleh sistem dengan total informasi relevan dalam dataset, baik yang ditemukan maupun yang tidak ditemukan. Formula untuk menghitung re all adalah sebagai berikut: $recall == TP\ TP + FN$ 4. F 1 Score F1 Score adalah metrik evaluasi yang menghitung rata-rata harmonis antara presisi dan recall, yang bertujuan untuk menemukan keseimbangan antara keduanya. F1 Score sangat bermanfaat terutama dalam kasus di mana terdapat ketidakseimbangan kelas, karena metrik ini memberikan bobot yang sama pada presisi dan recall. Semakin tinggi nilai F1 Score, semakin baik kinerja model dalam mengidentifikasi observasi positif tanpa terlalu banyak menghasilkan kesalahan. $F1 = s$ $ore = = 2 = x = = precisio$ 41 $= x = recall\ pre$ $cision + recall$ 26 BAB IV PERANCANGAN BAB ini merinci rancangan s stem automasi DevOps yang dirancang untuk mengidentifikasi dan mengoptimalkan sumber daya cloud yang tidak digunakan atau tidak efisien. Ini akan dilakukan melalui analisis sistem terdahulu, spesifikasi kebutuhan sistem baru, dan perancangan sistem tersebut. Sistem baru yang akan

dikembangkan bertujuan untuk mengotomatisasi deteksi dan optimasi sumber daya cloud yang tidak digunakan atau tidak efisien di PT. XYZ, dengan integrasi DevOps yang melibatkan GitLab. 4.1 Analisis Sistem Terdahulu Sistem yang ada sebelumnya di PT. XYZ melibatkan pemantauan dan pengelolaan sumber daya cloud secara manual. Analisis ini mengungkap bahwa proses manual tersebut tidak hanya memakan waktu dan sumber daya manusia yang signifikan tetapi juga sering kali menghasilkan identifikasi yang lambat terhadap sumber daya yang tidak optimal, menyebabkan penggunaan biaya cloud yang lebih tinggi dari yang seharusnya. Kelemahan utama dari sistem ini adalah kurangnya visibilitas atas penggunaan sumber daya secara real-time dan keterbatasan dalam respons otomatis terhadap perubahan kebutuhan sumber daya. Gambar 4.1 Flowchart Sistem Terdahulu 27 Pada gambar 4.1, dapat dilihat alur sistem yang ada sebelumnya di PT. XYZ. Di tahap pertama, dilakukan identifikasi penggunaan sumber daya, di mana data dikumpulkan untuk menilai penggunaan cloud. Kemudian dilakukan analisis kebutuhan sumber daya untuk mengevaluasi apakah kapasitas yang tersedia sudah sesuai dengan permintaan operasional. Jika hasil analisis menunjukkan bahwa sumber daya tidak optimal, maka langkah selanjutnya adalah melakukan penyesuaian untuk menyesuaikan penggunaan sumber daya dengan kebutuhan yang ada. Proses ini dilakukan secara manual dan memerlukan banyak waktu untuk memonitor dan menyesuaikan, yang mengarah pada pemborosan biaya dan ketidakmampuan untuk merespons secara cepat terhadap perubahan beban kerja. 4.2 Spesifikasi Kebutuhan Sistem Baru Setelah melakukan analisis terhadap sistem yang ada, didapatkan hasil yang mencakup spesifikasi yang diperlukan untuk sistem baru, yaitu sistem automasi DevOps yang berfokus pada optimalisasi biaya cloud di e-commerce. Untuk membangun sistem ini, diperlukan spesifikasi yang meliputi perangkat keras, perangkat lunak, serta kebutuhan input dan output yang akan diproses oleh sistem. Selain itu, perancangan model untuk machine learning dan perancangan DevOps

juga merupakan bagian integral dalam membangun sistem ini untuk memastikan optimasi yang lebih efisien dan otomatis. 4.2.1 Spesifikasi Perangkat Keras Dalam pengembangan system aplikasi, perangkat keras yang digunakan untuk menjalankan sistem ini harus sesuai dengan kebutuhan pemrosesan data dan kapasitas untuk mendukung alur kerja sistem yang efisien. Berikut adalah detail spesifikasi perangkat keras yang diperlukan yang tercantum pada Tabel 4.1 Tabel 4.1 Spesifikasi Perangkat Keras Perangkat Konfigurasi Prosesor Amd64 2 vCPU Memori 8 GB Sistem Operasi Linux x86-64 Tipe Sistem 64-bit 4.2.2 Spesifikasi Perangkat Lunak Untuk mendukung pengembangan sistem ini, sejumlah perangkat lunak dan alat pendukung diperlukan. Berikut adalah spesifikasi perangkat lunak yang digunakan dalam pengembangan aplikasi pada Tabel 4.2 Tabel 4.2 Spesifikasi Perangkat Lunak Perangkat Deskripsi Bahasa Pemograman Python Browser Mozilla Firefox Library Google APIs, pandas, python-dotenv, oauth2client, db-types dan Jinja2 Tools Visual Code Studio CI/CD Gitlab Alat Monitoring Datadog 28 4.2.3 Spesifikasi Kebutuhan Input Data yang digunakan dalam penelitian ini diambil dari beberapa sumber yang menyediakan metrik terkait penggunaan sumber daya cloud, baik untuk VM di GCP maupun untuk anali 46 is optimasi sumber daya. Sumber data yang digunakan adalah sebagai berikut: 1. Datadog Da a utama mengenai penggunaan CPU, memori, serta metrik terkait permintaan baca/tulis pada VM yang berfungsi sebagai node Cassandra dikumpulkan melalui platform pemantauan Datadog. Pemantauan ini berlangsung selama periode 10 bulan, mulai dari Januari hingga Oktober 2024. Fokus utama pada metrik CPU dan memori memungkinkan penelitian ini untuk mendeteksi penggunaan sumber daya yang tidak efisien, sehingga dapat memberikan rekomendasi perubahan machine type atau alokasi sumber daya pada VM untuk mengoptimalkan biaya operasional dan performa sistem. 2. BigQuery BigQuery digunakan untuk mendeteksi project di GCP yang hanya mengaktifkan API monitoring tanpa ada layanan lain yang aktif. Selain itu, BigQuery membantu mengidentifikasi snapshot yang

sudah lama tidak digunakan dan IP address yang tidak terpakai. Data dari BigQuery ini meliputi informasi terkait pemakaian penyimpanan, biaya operasional, dan penggunaan komponen cloud lainnya yang tidak efisien, sehingga memudahkan proses identifikasi sumber daya yang perlu dioptimalkan. 4.2.4 Spesifikasi Kebutuhan Output Output yang dihasilkan oleh sistem ini berupa laporan yang diintegrasikan dengan Go 47 gle Sheets. Laporan ini akan berisi informasi tentang sumber daya cloud yang diidentifikasi sebagai tidak terpakai atau tidak efisien, serta rekomendasi optimasi yang diperlukan untuk menyesuaikan alokasi sumber daya dan mengurangi biaya operasional. Laporan ini menjadi dasar bagi pengambilan keputusan dalam mengoptimalkan sumber daya di PT. XYZ. 4.3 Perancangan Sistem
Untuk mengembangkan sistem automasi DevOps yang mampu mendeteksi dan mengoptimalkan penggunaan sumber daya cloud yang tidak efisien atau tidak terpakai, peneliti merancang sebuah sistem yang komprehensif. Rancangan ini menggambarkan cara kerja sistem secara keseluruhan sehingga proses pengembangan dapat dilakukan dengan lebih rinci dan efisien. Sistem ini didesain untuk melakukan beberapa fungsi utama, seperti deteksi VM dengan jenis machine type yang tidak sesuai, deteksi snapshot lama yang tidak diperlukan, serta deteksi project GCP yang hanya mengaktifkan API monitoring tanpa layanan lain. Selain itu, sistem juga mendeteksi IP address dan disk yang tidak terpakai.
Untuk memberikan rekomendasi optimasi, sistem akan memanfaatkan algoritma seperti PSO untuk mengoptimalkan machine type berdasarkan pola penggunaan CPU dan memori yang terdeteksi. 29 Rancangan sistem ini menggunakan Flow Diagram untuk memvisualisasikan setiap alur kerja dalam sistem. Diagram ini membantu peneliti dan pengembang untuk memetakan keseluruhan proses, mulai dari pengumpulan data melalui GCP Monitoring dan BigQuery, analisis, hingga penyajian hasil di Google Sheets.
Dengan menggunakan Flow Diagram, alur kerja sistem dapat tergambar lebih sederhana dan jelas, tanpa memerlukan detail teknis mendalam seperti pada diagram Use Case, Activity, atau Sequence. Berikut adalah

proses Flow Diagram yang dibagi menjadi beberapa bagian Gambar 4.3 Process Flow Diagram Rekomendasi VM Gambar 4.3 adalah Flow Diagram untuk proses rekomendasi VM berdasarkan penggunaan CPU dan memori. Proses dimulai ketika pengguna menjalankan pipeline DevOps dengan memasukkan variabel yang diperlukan. Sistem kemudian mengumpulkan data metrik CPU dan memori dari Datadog, melakukan preprocessing data dengan normalisasi dan agregasi mingguan, dan melatih model deteksi anomali menggunakan Isolation Forest. Hasil deteksi tersebut dioptimalkan menggunakan PSO untuk menghasilkan rekomendasi tipe VM yang lebih efisien. Terakhir, hasil rekomendasi disimpan ke Google Sheets agar dapat diakses oleh tim DevOps. 30 Gambar 4.4 Process Flow Diagram mendeteksi project GCP Gambar 4.4 adalah Flow Diagram untuk mendeteksi project GCP yang hanya mengaktifkan API monitoring tanpa layanan lain. Proses dimulai ketika pengguna menjalankan pipeline DevOps. Sistem terhubung ke BigQuery untuk mengumpulkan data proyek, kemudian melakukan query untuk menemukan proyek-project yang hanya memiliki API monitoring aktif tanpa layanan lain. Hasil deteksi disimpan ke Google Sheets untuk memudahkan pemantauan dan tindak lanjut oleh tim. Gambar 4.5 Process flowchart identifikasi snapshot Gambar 4.5 adalah Flow Diagram untuk proses identifikasi snapshot lama yang tidak diperlukan untuk mengoptimalkan penggunaan penyimpanan. Proses ini dimulai dengan pengguna yang menjalankan pipeline di DevOps, kemudian sistem terhubung ke BigQuery untuk mengambil data snapshot. Selanjutnya, sistem memfilter snapshot yang sudah tidak aktif atau lama, dan menyimpan daftar snapshot ini ke Google Sheets untuk memudahkan penghapusan. 31 Gambar 4.6 Process flowdiagram deteksi IP address and disk Gambar 4.6 adalah Flow Diagram untuk mendeteksi IP address dan disk yang tidak terpakai, sehingga dapat mengurangi biaya operasional cloud. Pengguna menjalankan pipeline melalui DevOps, kemudian sistem terhubung ke BigQuery untuk mengumpulkan data IP address dan disk yang tidak digunakan. Setelah data dikumpulkan dan dianalisis untuk aktivitas,

sistem menyimpan daftar resource yang tidak terpakai ke Google Sheets sebagai rekomendasi penghapusan atau pengoptimalan. 4.3.1 Perancangan Model Machine Learning Gambar 4.7 Flow Chart Pembuatan Model Gambar 4.7 menunjukkan flowchart perancangan model yang diusulkan untuk sistem optimasi sumber daya cloud menggunakan algoritma machine learning dan DevOps, sebagaimana yang telah dijelaskan pada Bab 2. Proses ini dimulai dengan pengumpulan data, dilanjutkan dengan tahapan data preprocessing, deteksi anomali, pembentukan model, evaluasi model, hingga optimasi dan pembuatan rekomendasi. 32 Data yang digunakan dalam penelitian ini dikumpulkan dari platform pemantauan berbasis cloud seperti Datadog, yang memuat metrik penggunaan CPU, memori, dan permintaan baca/tulis pada VM Cassandra. Data yang terkumpul kemudian diproses melalui tahapan preprocessing untuk memastikan konsistensi dan kesiapan data, termasuk normalisasi dan perhitungan Average Predicted Usage (APU) sebagai indikator kinerja. Setelah proses data cleansing dan normalisasi selesai, data yang sudah diproses akan dianalisis menggunakan algoritma Isolation Forest untuk mendeteksi anomali dalam pola penggunaan sumber daya. Langkah ini bertujuan untuk mengidentifikasi pola penggunaan yang tidak efisien, seperti penggunaan sumber daya yang berlebihan atau kurang optimal 24 Setelah anomali berhasil dideteksi, proses dilanjutkan ke tahap pembentukan model menggunakan algoritma Boosted Decision Tree. Algoritma ini dilatih untuk mengenali pola data guna merek mendasikan tipe VM yang optimal berdasarkan data penggunaan CPU dan memori. Model yang terbentuk kemudian dievaluasi dengan metrik Mean Squared Error (MSE), yang membantu mengukur akurasi model terhadap data aktual dan mengidentifikasi potensi kesalahan dalam prediksi. Tahap terakhir adalah optimasi model menggunakan PSO (Particle Swarm Optimization), yang bertujuan untuk menyempurnakan rekomendasi berdasarkan data penggunaan sumber daya yang dinamis dan kondisi lingkungan yang berubah. Melalui proses optimasi ini, sistem dapat memberikan rekomendasi tipe VM yang paling efisien dan sesuai dengan

kebutuhan operasional VM. Gambar 4.8 Flowchart pengumpulan data Gambar 4.8 Flowchart Pengumpulan Data menunjukkan tahapan pengumpulan data dalam penelitian ini. Proses dimulai dengan mengimpor library yang diperlukan, seperti datetime, timedelta, re, logging, csv, os, dan datadog, yang mendukung pengambilan, pemrosesan, dan penyimpanan data. Setelah itu, API Datadog diinisialisasi dengan API key yang telah dikonfigurasi, yang memungkinkan akses ke Datadog untuk memperoleh data metrik yang dibutuhkan. 33 Rentang waktu untuk pengambilan data kemudian ditentukan menggunakan datetime dan timedelta, memastikan data yang diambil berada dalam periode yang jelas. Setelah rentang waktu ditentukan, permintaan data dikirim ke API Datadog untuk mengambil metrik penggunaan CPU dan memori. Proses ini dilakukan berulang kali hingga seluruh data yang dibutuhkan berhasil diambil. Kemudian diberi timestamp untuk mempermudah analisis dan pelacakan. Pada tahap akhir, data yang telah bersih disimpan dalam file CSV di direktori yang telah ditentukan, sehingga siap digunakan untuk tahap analisis atau diproses oleh komponen lainnya dalam alur penelitian. Gambar 4.9 Flowchart data preprocessing Gambar 4.9 menunjukkan alur proses data preprocessing dalam penelitian ini. Tahapan ini dimulai dengan mengimpor library penting seperti pandas, numpy, dan glob yang berfungsi untuk mempermudah pengolahan dan manipulasi data. Selanjutnya, data file CSV yang diambil dari proses pengambilan data yang berisi metrik-metrik utama, yaitu cpu_usage, memory_usage, cassandra_read_request, dan cassandra_write_request, digabungkan ke dalam satu file harian untuk mempermudah langkah-langkah analisis selanjutnya. Pada tahap berikutnya, dilakukan normalisasi untuk menyesuaikan skala data. Normalisasi CPU dan memori menggunakan formula yang lebih spesifik. CPU Usage dinormalisasi dengan rumus norm_cpu_usage = 100 − cpu_usage yang memberikan inform asi kapasitas CPU yang terpakai dalam persentase Memory Usage dinormalisasi dengan rumus (1 − memory_usage) × 100 yang menampil kan kapasitas memory terpakai. Sementara itu, untuk permintaan read

dan write dari Cassandra, dilakukan transformasi menggunakan log-normalisasi untuk mengatasi variasi nilai yang tinggi pada data. Langkah ini bertujuan agar skala penggunaan sumber daya lebih seragam untuk analisis lebih lanjut. Selanjutnya tahap untuk menghitung nilai peak usage atau puncak penggunaan harian untuk setiap metrik, dengan mengambil nilai tertinggi pada data harian. Selanjutnya menghitung nilai APU yang dihasilkan dari metrik rata-rata harian. PT. XYZ telah menetapkan bobot khusus untuk setiap metrik dalam penilaian APU 34 Tabel 4 3 Tabel bobot APU Metriks Bobot CPU Usage 48% Memory Usage 48% Wrire Request 2 % Read Request 2 % Rumus perhitungan APU adalah: APU = 0.48 × norm_average_cpu_usage + 0.48 × norm_ average_memory_usage +0.02 × norm_average_write_request + 0.02 × norm_average_read_request Nilai APU ini kemudian disimpan untu k digunakan dalam proses analisis dan optimasi lebih lanjut. Gambar 4.10 Flowchart Pembentukan Model 35 Gambar 4.10 merupakan flowchart yang menggambarkan alur pembentukan model dalam penelitian ini. Proses dimulai dengan mengimpor library yang diperlukan, seperti pandas, json, logging, dan sklearn. Library ini mendukung berbagai fungsi, seperti manipulasi data, penyimpanan hasil dalam format JSON, logging untuk pencatatan proses, dan implementasi model Machine Learning. Library tambahan seperti tqdm juga digunakan untuk menampilkan progress bar selama proses iterasi hyperparameter. Setelah tahap import library selesai, proses dilanjutkan dengan inisialisasi direktori yang akan digunakan untuk menyimpan model, hyperparameter, serta hasil performa model. Langkah inisialisasi direktori ini bertujuan untuk menyiapkan ruang penyimpanan terstruktur, sehingga setiap artefak yang dihasilkan dari pembentukan model dapat dikelola dan diakses dengan mudah selama proses pengembangan maupun analisis lebih lanjut. Langkah berikutnya adalah mendefinisikan model utama, yaitu Boosted Decision Tree, dan menentukan grid hyperparameter yang akan dievaluasi. Model ini dipilih untuk memprediksi APU, yang kemudian akan menjadi dasar dalam

menentukan rekomendasi optimal untuk tipe VM. Pengaturan grid hyperparameter meliputi parameter seperti jumlah n_estimators, learning_ rate, max_leaf_nodes, dan min_samples_leaf. Setiap kombinasi dari hyperparameter ini akan dievaluasi selama pelatihan untuk menemukan konfigurasi terbaik yang me 2 13 28 ghasilkan nilai MSE terendah. Selanjutnya, data yang telah diproses pada tahap data preprocessing akan dimuat untuk digunakan dalam proses pelatihan model. Jika pemuatan data berhasil, data tersebut dibagi menjadi dua bagian: set pelatihan dan set pengujian. Set pelatihan digunakan untuk melatih model, sedangkan set pengujian digunakan untuk mengukur performa model pada data yang belum pernah dilihat sebelumnya. Pada tahap ini, Isolation Forest diterapkan untuk mendeteksi dan menghilangkan outliers, guna memastikan bahwa data yang digunakan adalah representatif dan bebas dari anomali yang dapat mempengaruhi hasil pembentukan model. Setelah persiapan data selesai, proses pelatihan model dilakukan menggunakan Boosted Decision Tree pada set pelatihan, dengan mencoba setiap kombinasi hyperparameter yang tersedia dalam grid. Untuk setiap kombinasi, dilakukan evaluasi menggunakan cross-validation untuk menghitung nilai MSE. Proses cross-validation ini dilakukan berulang kali hingga seluruh kombinasi hyperparameter dievaluasi, dan model dengan MSE terendah dipilih sebagai model terbaik. 36 Gambar 4.11 Flowchart Optimization PSO Gambar 4.11 menunjukkan flowchart yang mendeskripsikan alur optimisasi menggunakan Particle Swarm Optimization (PSO) dalam penelitian ini. Proses dimulai dengan mengimpor library yang diperlukan, seperti pso, pandas, joblib, dan logging, yang mendukung pelaksanaan PSO serta pemuatan model yang telah dilatih. Setelah library di import, parameter PSO diinisialisasi. Parameter ini mencakup jumlah partikel, kecepatan awal, dan batas iterasi, yang sangat penting untuk mengatur perilaku partikel selama proses optimisasi. Tahap selanjutnya adalah mendefinisikan fungsi objektif, yang dibangun berdasarkan output dari model yang telah dilatih. Fungsi objektif ini dirancang untuk menentukan konfigurasi VM

yang optimal dengan mempertimbangkan efisiensi 48 penggunaan sumber daya. Setiap partikel dalam swarm kemudian diperbarui dalam hal kece atan dan posisi berdasarkan solusi optimal terbaik yang ditemukan, baik pada tingkat individu (personal best) maupun pada tingkat swarm secara keseluruhan (global best). Pada setiap iterasi, fitness dari setiap partikel dihitung menggunakan fungsi objektif. Jika solusi yang ditemukan lebih optimal dari solusi sebelumnya, maka solusi terbaik personal dan global akan diperbarui. Setelah evaluasi, sistem melakukan pencarian tipe machine yang paling mendekati solusi optimal yang ditemukan. Proses ini bertujuan untuk memastikan bahwa rekomendasi yang diberikan bisa diimplementasikan sesuai dengan ketersediaan tipe machine yang ada. Jika solusi terbaik ditemukan, proses akan diakhiri. Namun, jika kriteria terminasi, seperti jumlah iterasi atau ambang perbedaan solusi, belum tercapai, proses akan berlanjut hingga kriteria tersebut terpenuhi. 37 4.3.2 Perancangan DevOps Perancangan DevOps dalam penelitian ini dibagi menjadi dua bagian utama, yaitu pipeline untuk Machine Learning dan pipeline untuk analisis sumber daya. Pipeline ini dirancang dengan struktur yang mencakup semua langkah utama mulai dari persiapan lingkungan, pengumpulan data, hingga pengoptimalan sumber daya. Penjelasan detail terkait setiap alur pipeline dijelaskan dalam gambar di bawah ini. Gambar 4.12 diaram alir CI Machine learning Gambar 4.12 menunjukkan alur pipeline untuk Machine Learning dalam penelitian ini. Pipeline ini mencakup beberapa tahap utama untuk mengotomatisasi proses analisis dan pengoptimalan sumber daya. Berikut adalah penjelasan per stage dalam pipeline ini: Gambar 4.13 diagram alir stage setup 38 Gambar 4.13 menjelaskan pada stage setup, terdapat job setup environment yang dirancang untuk menyiapkan environment Python yang akan digunakan oleh stage lainnya dalam pipeline Machine learning. Job ini terdiri dari beberapa langkah, yaitu pembuatan virtual environment, aktivasi virtual environment, dan instalasi dependensi yang terdapat pada file requirements.txt. Hasil dari job

ini disimpan sebagai artifact dalam direktori venv, yang kemudian digunakan oleh job lain pada pipeline. Gambar 4.14 diagram alir stage collect data Gambar 4.14 menjelaskan pada stage collect_data, terdapat job collect data yang dirancang untuk mengumpulkan data terkait penggunaan sumber daya dari platform monitoring, seperti CPU, memori, disk, dan koneksi. Job ini menjalankan script collect_data.py untuk menghasilkan data mentah yang disimpan dalam direktori output ($OUTPUT_DIR). Data yang dihasilkan pada job ini akan digunakan sebagai input pada stage berikutnya dalam pipeline Machine learning. Gambar 4.15 flowchart stage preprocess data Gambar 4.15 menjelaskan pada stage preprocess_data, terdapat job preprocess data yang dirancang untuk menjalankan proses preprocessing data. Job ini memiliki dependencies dari job sebelumnya, yaitu setup-job dan collect_data. Pada job ini, virtual environment diaktifkan, dan script data_ preprocessing.py dijalankan dengan parameter input dan output. Hasil dari job ini disimpan sebagai artifact dalam direktori $OUTPUT_DIR, yang akan digunakan oleh stage berikutnya. 39 Gambar 4.16 flowchart stage calculate apu Gambar 4.16 menjelaskan pada stage calculate_apu, terdapat job calculate APU yang dirancang untuk menghitung metrik APU dari data yang telah diproses pada stage sebelumnya. Job ini memiliki dependencies dari setup-job dan preprocess_data. Pada job ini, virtual environment diaktifkan, dan script calculate_apu.py dijalankan. Hasil dari perhitungan disimpan sebagai artifact dalam direktori $VISUALIZATION_DIR, yang kemudian digunakan pada stage berikutnya. Gambar 4.17 flowchart stage train model Gambar 4.17 menjelaskan pada stage train_model, terdapat job train model yang dirancang untuk melatih model Machine Learning berdasarkan data APU yang telah dihitung pada stage sebelumnya. Job ini memiliki dependencies dari setup-job dan calculate_apu. Virtual environment diaktifkan, dan script train_model.py dijalankan dengan data input dan parameter yang telah ditentukan. Hasil dari pelatihan model disimpan

sebagai artifact dalam direktori $OUTPUT_DIR, yang akan digunakan oleh stage selanjutnya untuk optimasi sumber daya. 40 Gambar 4.18 flowchart stage optimize resources Gambar 4.18 menjelaskan pada stage optimize_resources, terdapat job resource optimization yang dirancang untuk menjalankan optimasi sumber daya berdasarkan model yang telah dilatih. Job ini memiliki dependencies dari setup-job dan train_model. Setelah virtual environment diaktifkan, script resources_optimize.py akan dijalankan. Hasil dari optimasi memberikan rekomendasi tindakan seperti upgrade, downgrade, atau maintain tipe mesin berdasarkan analisis sumber daya. Gambar 4.19 flowchart pipeline resource analysis Gambar 4.19 menjelaskan perancangan pipeline untuk resources analysis, yang terdiri dari beberapa tahapan utama untuk menganalisis dan mengoptimalkan penggunaan sumber daya cloud. Pipeline ini dimulai dari stage setup, yang bertugas mempersiapkan environment yang akan digunakan oleh seluruh job pada stage berikutnya. Setelah tahap setup selesai, pipeline berlanjut ke stage resource analysis, yang terdiri dari beberapa job untuk menjalankan analisis secara paralel. Berikut adalah penjelasan per stage dalam pipeline ini: Pada stage setup, sama seperti yang telah dijelaskan pada Gambar 4.8, terdapat job setup environment, yang dirancang untuk menyiapkan environment Python yang akan digunakan oleh semua job di stage berikutnya. Job ini terdiri dari beberapa langkah utama, yaitu pembuatan virtual environment, aktivasi virtual environment, dan instalasi dependensi yang terdapat pada file requirements.txt. 41 Hasil dari job ini disimpan sebagai artifact di direktori venv, yang kemudian digunakan pada semua job di stage resource analysis. Gambar 4.20 flowchart job rekomendasi vm Gambar 4.20 menjelaskan pada stage resource analysis, terdapat job recommendation-vm, yang bertugas menjalankan program Machine learning untuk memberikan rekomendasi tipe mesin virtual berdasarkan data penggunaan sumber daya. Job ini memiliki dependencies dari setup-job dan dijalankan hanya jika commit branch adalah main, sesuai dengan validasi yang didefinisikan dalam

pipeline CI/CD. Pada job ini, script resources_optimize.py dijalankan menggunakan environment Python yang telah disiapkan sebelumnya, dan hasil rekomendasi dihasilkan berdasarkan model yang telah dilatih. Gambar 4.21 flowchart job get cost project Gambar 4.21 menjelaskan job cost project job yang terdapat pada stage resource analysis, yang bertugas untuk mengevaluasi efisiensi biaya project berdasarkan data penggunaan sumber daya terkini. Job ini juga memiliki dependencies dari setup-job dan hanya dijalankan jika commit branch adalah main. Pada job ini, script main.py dengan perintah cost-project dijalankan untuk menghasilkan laporan biaya. 42 Gambar 4.22 flowchart deattach ip disk Gambar 4.22 menjelaskan job deattach IP disk job yang terdapat pada stage resource analysis, yang bertugas mendeteksi disk dan IP address yang tidak terpakai di lingkungan cloud. Dependencies dari job ini adalah setup-job, dan validasi commit branch memastikan job hanya dijalankan pada branch main. Analisis ini dilakukan menggunakan script main.py dengan perintah deattach-ip- disk. Gambar 4.23 flowchart job get old snapshot Gambar 4.23 menjelaskan job old snapshots job, yang bertugas untuk menganalisis snapshot lama guna mengidentifikasi snapshot yang dapat dihapus. Job ini memiliki dependencies dari setup-job dan hanya dijalankan jika commit branch adalah main, sesuai validasi yang didefinisikan. Pada job ini, script main.py dijalankan dengan perintah old-snapshots, yang menggunakan parameter seperti JOBS, YEARS, MONTHS, DAYS, dan DISK_SIZE untuk menentukan kriteria analisis snapshot. 43 Gambar 4.24 CI pipeline merge request Gambar 4.24 menjelaskan perancangan pipeline untuk tahapan unit testing, yang dijalankan secara otomatis saat terjadi merge request. Merge request adalah permintaan yang diajukan oleh pengembang untuk menggabungkan perubahan kode pada sebuah branch ke dalam main branch, setelah melalui proses review dan pengujian untuk memastikan bahwa kode yang diubah tidak menimbulkan error atau konflik. Pipeline untuk unit testing ini terdiri dari dua stages, yaitu setup dan

unit test. Berikut adalah penjelasan per stage dalam pipeline ini: Pada stage setup, sama seperti yang telah dijelaskan pada bagian gambar 4.8 Hasil dari stage ini disimpan sebagai artifact di direktori venv, yang kemudian digunakan pada semua job di stage unit testing. Gambar 4.25 flowchart unit testing ml job Gambar 4.25 menjelaskan job unit testing Machine Learning, yang bertugas menjalankan pengujian fungsi-fungsi pada modul Machine Learning, seperti 44 perhitungan APU dan pelatihan model. Job ini memiliki dependencies dari setup- job dan hanya dijalankan jika pipeline dijalankan dalam konteks merge request, sesuai validasi yang didefinisikan. Pada job ini, pengujian dilakukan dengan menggunakan pytest untuk memastikan bahwa fungsi dalam modul recommendation_vm bekerja dengan baik. Hasil dari pengujian ini disimpan dalam laporan dengan format file machine_ learning.xml, yang dapat digunakan untuk memeriksa status pengujian dan menganalisis hasilnya. Gambar 4.26 flowchart unit testing gcp report Gambar 4.26 menjelaskan job unit testing GCP resources, yang bertugas menjalankan pengujian fungsi-fungsi terkait integrasi GCP, seperti pengumpulan laporan dan analisis resource. Job ini juga memiliki dependencies dari setup-job dan hanya dijalankan jika pipeline dijalankan dalam konteks merge request, sesuai validasi yang telah ditentukan. Pada job ini, pengujian dilakukan dengan menggunakan pytest untuk memastikan bahwa fungsi dalam modul gcp_reports bekerja dengan baik. Hasil dari pengujian ini disimpan dalam laporan dengan format file gcp_reports.xml, yang dapat digunakan untuk memeriksa status pengujian dan menganalisis hasilnya. 4.3.3 Perancangan Pengujian Pengujian sistem dilakukan dengan menggunakan metode A/B Testing, white box, dan confusion matrix. Tujuan dari perancangan pengujian ini adalah untuk memberikan gambaran dan memastikan sistem bekerja sesuai fungsinya tanpa ada kesalahan, baik dari sisi antarmuka maupun proses internal. Berikut adalah rincian rancangan pengujian sistem yang akan dilakukan: 45 4.3.3.1 Perancangan Pengujian A/B Testing Tabel 4.4

Rancangan pengujian A/B Testing Skenario Pengujian Parameter yang dievaluasi Hasil yang diharapkan Evaluasi biaya sebelum dan sesudah optimisasi Cost Before, Cost After PSO, Cost Saving (USD), Cost Saving (%) PSO memberikan penghematan biaya yang signifikan dalam penggunaan sumber daya cloud dibandingkan kondisi awal. Evaluasi rekomendasi optimasi berdasarkan PSO Total recommendation, valid recommendation, percentage recommendation optimization PSO memberikan rekomendasi yang tepat dan efisien dalam penyesuaian konfigurasi VM sesuai dengan kebutuhan sumber daya. Pada Tabel 4.4, rancangan pengujian A/B Testing bertujuan untuk mengevaluasi efektivitas metode PSO dalam mengoptimalkan biaya penggunaan sumber daya cloud. Pengujian dilakukan dengan membandingkan biaya awal (Cost Before) dan biaya setelah optimisasi (Cost After PSO) untuk setiap bulan, dengan hasil penghematan dihitung dalam satuan dolar (Cost Saving (USD)) dan persentase (Cost Saving (%)). Hasil pengujian ini diharapkan dapat menunjukkan bahwa optimisasi berbasis PSO mampu mengurangi biaya penggunaan sumber daya cloud secara signifikan. Selain itu, pengujian ini juga akan mengevaluasi rekomendasi optimasi yang dihasilkan oleh PSO, dengan parameter yang dievaluasi berupa Total recommendation, valid recommendation, dan percentage recommendation optimization. Hasil yang diharapkan adalah PSO memberikan rekomendasi yang tepat dan efisien dalam penyesuaian konfigurasi VM sesuai dengan kebutuhan sumber daya. 4.3.3.2 Perancangan Pengujian Whitebox Tabel 4.5 Rancangan Pengujian White Box Skenario Pengujian Hasil yang Diharapkan Unit testing untuk fungsi pengumpulan data dari Datadog API Fungsi mengambil data metrik dari Datadog API dan mengembalikannya dalam format yang sesuai untuk analisis, tanpa ada error koneksi atau format data yang salah. Unit testing untuk fungsi preprocessing data Fungsi melakukan normalisasi dan agregasi data harian menjadi data mingguan dengan benar, serta memeriksa data yang hilang atau tidak valid. Unit testing untuk model Machine learning dalam tahap pelatihan

Model berhasil dilatih dengan data yang telah dipreproses, tanpa ada error selama proses training, dan menghasilkan model yang sesuai untuk deteksi anomali dan rekomendasi VM. Unit testing untuk algoritma PSO dalam optimasi rekomendasi Algoritma PSO berjalan dengan benar, mencari solusi optimal berdasarkan input model Machine learning, dan mengeluarkan hasil optimasi tipe VM tanpa kesalahan komputasi. Pada Tabel 4.5, rancangan pengujian white box bertujuan untuk menguji bagian dalam sistem dengan menggunakan pendekatan unit testing. Pengujian ini fokus 46 pada pengujian logika setiap fungsi utama dan integrasi antar-modul yang terkait dalam pipeline DevOps dan Machine learning. Proses pengujian dimulai dari tahap pengumpulan data, preprocessing, pelatihan model, hingga tahap optimasi menggunakan algoritma PSO dan penyimpanan hasil. Unit testing memastikan bahwa setiap fungsi bekerja sesuai dengan spesifikasinya, baik dalam hal logika komputasi maupun format data, sehingga setiap komponen berkontribusi pada keseluruhan fungsi sistem dengan optimal. 4.3.3.3 Perancangan Pengujian Confusion Matrix Tabel 4.6 Perancangan Confusion Matrix Pengukuran Performance Perhitungan Hasil yang diharapkan Akurasi $accuracy$ = ' TP + TN TP + TN + FP + FN > 0 -1 Presisi $precisio$ == TP TP +FP > 0 -1 Recall $recall$ == TP TP +FN > 0 -1 F1 Score $F1$=$scor$ e = =2=x =$precision$=x=$recal$ l $precision$ +$recall$ > 0 -1 P ada tabel 4.6, dirancang skenario pengujian untuk confusion matrix yang meliputi perhitungan akurasi, presisi, recall, dan F1 score, yang menjadi indikator utama performa model. Pengukuran ini memberikan pandangan menyeluruh terkait efektivitas model dalam membuat rekomendasi yang sesuai dengan kondi 1 2 4 5 8 21 i aktual. Pengujian ini bertujuan untuk mengevaluasi performa model dalam memberikan rekomendasi optimal untuk tipe mesin virtual berdasarkan data penggunaan sumber daya di

lingkungan cloud. Semakin t nggi nilai akurasi mendekati 1, semakin baik performa model dalam mendeteksi kebutuhan upgrade atau downgrade sumber daya pada GCP. BAB V HASIL DAN PEMBAHASAN 47 Bab ini membahas hasil dari penelitian yang dilakukan serta pembahasan mengenai sistem yang telah dikembangkan untuk optimasi penggunaan sumber daya cloud pada GCP. Bagian hasil akan membahas setiap komponen utama dari sistem yang dirancang, termasuk proses data preprocessing, pembentukan model, serta implementasi sistem secara keseluruhan. Bagian pembahasan akan menjelaskan analisis terkait hasil yang diperoleh, efektivitas metode yang digunakan, serta potensi pengembangan lebih lanjut untuk meningkatkan kinerja sistem. 5.1 Hasil Hasil dari penelitian ini adalah sebuah sistem automasi DevOps berbasis Machine learning yang dirancang untuk mendeteksi dan mengoptimalkan penggunaan sumber daya pada cloud GCP. Sistem ini mencakup berbagai proses utama, termasuk preprocessing data, pembentukan model dengan algoritma Machine learning, dan optimasi melalui Particle Swarm Optimization (PSO). Pada bagian ini, akan dijelaskan hasil dari setiap tahap pengembangan sistem, mulai dari data preprocessing hingga hasil dari rekomendasi optimasi yang diberikan oleh sistem. 5.1.1 Hasil data preprocessing Data yang diperoleh melalui proses data collection memerlukan pembersihan awal agar siap digunakan dalam pelatihan model. Preprocessing data adalah tahapan untuk menyiapkan data mentah menjadi format yang dapat dipahami, diprediksi, dan dianalisis oleh mesin melalui algoritma Machine learning (Tabassum & Patil, 2020). Preprocessing ini mencakup beberapa langkah, antara lain penggabungan data harian, normalisasi, dan perhitungan APU 1) Penggabungan data harian Gambar 5.1 Penggalan kode untuk penggabungan data harian Gambar 5.1 menunjukkan kode sumber yang digunakan untuk menggabungkan data harian dari beberapa metrik utama, seperti cpu_usage, memory_ usage, cassandra_read_latency, dan cassandra_write_latency. Pada fungsi ini, setiap file yang berisi data harian dimuat, dan kolom

timestamp diubah menjadi date. Fungsi ini juga mengganti nama kolom value menjadi nama metrik yang sesuai untuk mempermudah proses analisis selanjutnya. 48 Setelah data dimuat, semua metrik digabungkan ke dalam satu DataFrame menggunakan metode concat, sehingga analisis berbagai metrik dapat dilakukan dalam satu struktur data. Hasil akhir dari penggabungan data ini disimpan sebagai file CSV untuk referensi di tahap berikutnya. Output dari kode di atas dirangkum pada tabel berikut, yang memperlihatkan hasil penggabungan data dari file-file terpisah menjadi satu DataFrame. Tabel 5.1 Hasil penggabungan data harian Sebelum Sesudah cpu_usage_20240101.csv memory_usage_20241001.csv cassandra_read_latency_20241001.csv cassandra_write_latency_20241001.csv merged_daily_data.csv Tabel 5.1 menunjukkan hasil dari proses penggabungan data harian menggunakan fungsi merge_daily_data. Sebelum proses penggabungan, data untuk setiap metrik utama, seperti cpu_ usage, memory_usage, cassandra_read_latency, dan cassandra_write_ latency, disimpan dalam file terpisah berdasarkan tanggal, seperti cpu_ usage_20240101.csv atau memory_usage_20241001.csv. File-file ini hanya mencakup satu metrik pada satu waktu, sehingga analisis data lintas metrik menjadi sulit. Setelah fungsi merge_daily_data dijalankan, semua data dari file-file terpisah ini digabungkan ke dalam satu file bernama merged_daily_data.csv. Proses ini melibatkan beberapa langkah, yaitu memuat setiap file, mengubah kolom timestamp menjadi format date, mengganti nama kolom value dengan nama metrik yang sesuai, dan akhirnya menyusun semua data ke dalam satu DataFrame. Hasil penggabungan ini memuat seluruh informasi metrik dalam satu struktur data yang terorganisasi, sehingga mempermudah analisis lebih lanjut terhadap data. 2) Normalisasi Gambar 5.2 Penggalan kode sumber normalisasi Gambar 5.2 adalah kode sumber yang digunakan untuk melakukan normalisasi pada data harian. Pada fungsi ini, data hasil data collection awal menunjukkan sisa penggunaan sumber daya, bukan penggunaan aktual. Oleh karena itu, dilakukan kalkulasi untuk

mendapatkan persentase penggunaan yang benar. Output dari proses normalisasi ini dapat dilihat pada tabel berikut, yang menunjukkan perubahan nilai sebelum dan sesudah normalisasi untuk metrik utama: Tabel 5.2 Hasil Normalisasi data Jenis Metrik Sebelum Sesudah 49 CPU Usage 98.50 1.5 Memory Usage 0.85 85 CassandraWrite Request 123 4.82 Cassandra Read request 1200 7.09 Tabel 5.2 merupakan hasil dari perbandingan nilai metrik sebelum dan sesudah proses normalisasi menggunakan berbagai metode yang sesuai dengan karakteristik data. Sebelum normalisasi, nilai metrik seperti CPU Usage, Memory Usage, Cassandra Write Request, dan Cassandra Read Request ditampilkan dalam skala asli. Misalnya, CPU Usage bernilai 98.50% yang menunjukkan kapasitas CPU yang tersisa. Setelah dilakukan normalisasi, nilai metrik diubah untuk memberikan representasi yang lebih bermakna dan terstandarisasi, seperti CPU Usage menjadi 1.5%, yang merepresentasikan kapasitas pemakaian dari CPU. Demikian pula, Memory Usage yang awalnya 0.85 (85% digunakan) diubah menjadi 85%, yang pemakaian kapasitas memori. Untuk Cassandra Write Request dan Cassandra Read Request, nilai awal masing-masing 123 dan 1200 dinormalisasi menggunakan fungsi logaritma natural, menghasilkan nilai 4.82 dan 7.09. Proses ini membantu mengurangi skala data besar dan mempermudah analisis dengan tetap menjaga informasi yang relevan. Gambar 5.3 Penggalan kode sumber pengisian nilai NaN Selain proses agregasi harian, dilakukan juga pengisian nilai kosong pada data. Nilai not a number (NaN) pada setiap metrik utama diisi dengan nilai 0 agar data tetap konsisten dan dapat digunakan dalam analisis lebih lanjut tanpa mengganggu hasil. Pengisian nilai kosong ini penting agar setiap metrik memiliki data yang lengkap, sehingga perhitungan lebih stabil dan tidak terpengaruh oleh adanya data yang hilang. Gambar 5.3 adalah kode sumber yang menunjukkan proses pengisian nilai kosong di setiap kolom metrik. Tabel 5.3 Hasil pengisian nilai NaN Sebelum Sesudah ,,,, 0,0,0,0,0 Pada Tabel 5.3, hasil dari pengisian nilai NaN diubah

menjadi nilai 0 (kosong). Proses ini memastikan bahwa semua kolom dalam data memiliki nilai, sehingga dapat mendukung langkah-langkah analisis dan pelatihan model tanpa gangguan. 3) Perhitungan APU 50 Gambar 5.4 penggalan kode sumber perhitungan APU Gambar 5.4 menunjukkan kode sumber untuk menghitung APU menggunakan bobot yang ditentukan untuk setiap metrik. Pada kode ini, APU dihitung dengan menggunakan bobot 0.48 untuk cpu_usage dan memory_usage, serta 0.02 untuk read_request dan write_request. Perhitungan ini menghasilkan nilai yang menggambarkan penggunaan sumber daya secara keseluruhan. Tabel 5.4 Hasil Calculate APU Nilai Metrics Sebelum Sesudah Cpu_ usage: 45 Memory_usage: 60 Peak_cpu: 85 Peak_memory: 90 Write_ req: 100 Write_peak: 250 Read_req: 80 Pead_peak: 200 APU_CPU: 0 APU_Memory: 0 APU_CPU: 70.75 APU_Memory: 88.50 Tabel 5.4 memperlihatkan hasil dari perhitungan APU menggunakan data input yang telah dinormalisasi, baik untuk cpu_usage maupun memory_usage. Nilai APU ini memberikan representasi tingkat pemakaian sumber daya secara keseluruhan berdasarkan bobot kontribusi masing-masing metrik. 5.1.2 Hasil Pembuatan model Proses pembuatan model dilakukan melalui beberapa tahapan utama, dimulai dari inisialisasi direktori untuk input, output, model, dan hasil performa. Model yang digunakan adalah Boosted Decision Tree dengan pengaturan hyperparameter yang mencakup learning_ rate, n_estimators, max_leaf_nodes, dan min_samples_leaf. 51 Gambar 5.5 kode sumber inisialisasi model Gambar 5.5 menunjukkan bahwa model Boosted Decision Tree diinisialisasi dengan learning_rate berkisar antara 0.05 hingga 0.2 untuk mengatur kecepatan pembelajaran model. Selain itu, n_estimators digunakan untuk menentukan jumlah pohon dalam ensemble dengan nilai antara 50 hingga 500. Parameter max_ leaf_nodes mengatur jumlah maksimum daun dalam setiap pohon, sedangkan min_samples_leaf memastikan jumlah minimum sampel pada setiap daun adalah 5 hingga 15. Tabel 5.5 Hasil inisialisasi model Parameter Sebelum Sesudah Direktori output Tidak tersedia Direktori

output/models dan output/performance dibuat Model machine learning Tidak ada BoostedDecisionTree diinisialisasi dengan random_state=42 Hyperparameter grid Tidak ada learning_rate, n_estimators, max_leaf_ nodes, min_samples_leaf terdefinisi Tabel 5.5 memperlihatkan hasil implementasi kode sumber yang bertujuan untuk menginisialisasi pipeline pelatihan model menggunakan algoritma Boosted Decision Tree. Proses ini mencakup pembuatan direktori output, model, dan performa, serta penentuan model machine learning beserta grid hyperparameter untuk optimasi. Tahapan selanjutnya adalah memuat data yang sudah diproses sebelumnya. Data difilter untuk menghapus nilai ekstrem (outliers) menggunakan Isolation Forest. Proses ini memastikan bahwa data yang digunakan untuk pelatihan lebih bersih dan konsisten. 52 Gambar 5.6 Penggalan kode sumber isolation forest Gambar 5.6 menjelaskan kalau Isolation Forest digunakan dengan tingkat kontaminasi sebesar 1%. Data yang terdeteksi sebagai outlier dihapus, dan hanya data inliers yang digunakan untuk pelatihan model. Tabel 5.6 Hasil isolation forest Parameter Sebelum Sesudah Total data 10452 10347 (1% outlier dihapus) Tabel 5.6 menjelaskan hasil penerapan algoritma Isolation Forest untuk mendeteksi dan menghapus outlier dari dataset awal yang berjumlah 10,452 entri. Algoritma ini menggunakan parameter contamination=0.01, yang menentukan bahwa 1% dari 44 total data akan dianggap sebagai outlier. Setelah algoritma dijalankan, sebanyak 105 data terdeteksi sebag i outlier dan dihapus dari dataset. Data yang tersisa, sebanyak 10,347 entri, dianggap valid (inliers) dan siap diguna 10 38 an untuk analisis atau pelatihan model. Berikutnya adalah membagi data menjadi subset pelatihan dan pengujian. Data dibagi denga 15 rasio 80:20, di mana 80% digunakan untuk pelatihan model, sementara sisanya untuk pengujian. Gambar 5.7 kode sumber train model Gambar 5.7 menunjukkan kode sumber untuk membagi data menggunakan fungsi train_test_split. Data pelatihan digunakan untuk melatih model, sedangkan data pengujian digunakan untuk mengevaluasi performa model. Parameter random_state=42 memastikan hasil

pembagian data konsisten setiap kali proses dilakukan. Selanjutnya, model dilatih menggunakan kombinasi hyperparameter yang telah disebutkan sebelumnya melalui GridSearch. Proses ini bertujuan untuk menemukan kombinasi parameter yang memberikan nilai MSE terendah. Tabel 5.7 tabel hasil train test split Parameter Jumlah data Total data 10347 Data pelatihan 8278 (80%) Data pen 15 ujian 2069 (20%) Tabel 5.7 merupakan hasil proses pembagian data menjadi data pelatihan dan pengujian menggunakan fungsi train_t 10 st_split. Dari total 10,347 data yang valid (setelah penghapusan outlier), data dibagi menjadi dua subset, yaitu 80% untuk pelatihan (sebanyak 8,278 data) dan 20% untuk 25 engujian (sebanyak 2,069 data). Proses ini dilakukan dengan parameter test_ size=0.2 dan random_state=42 untuk memastikan bahwa pembagian data konsisten setiap kali dijalankan. Data pelatihan 53 digunakan untuk melatih model prediksi APU_CPU dan APU_Memory, sementara data pengujian digunakan untuk mengevaluasi performa model yang dihasilkan. Pendekatan ini penting untuk menghindari overfitting dan memastikan bahwa model dapat melakukan generalisasi dengan baik pada data yang belum pernah dilihat sebelumnya. Gambar 5.8 Penggalan kode sumber kombinasi hperparameter Gambar 5.8 menunjukkan kode sumber proses Grid Search, di mana setiap kombinasi hyperparameter diuji, dan kombinasi dengan nilai MSE terendah dipilih untuk melatih ulang model. Proses ini memastikan bahwa model menggunakan pengaturan parameter yang optimal untuk memprediksi APU_CPU dan APU_Memory. Tabel 5.8 Hasil search best parameter Prediksi_APU Parameter terbaik MSE APU_CPU {'learning_ rate': 0.2, 'max_leaf_nodes': 10, 'min_samples_leaf': 5, 'n_ estimators': 500} 0.1001 APU_Memory {'learning_rate': 0.1, 'max_leaf_ nodes': 10, 'min_samples_leaf': 5, 'n_estimators': 500} 0.0121 Tabel 5.8 menjelaskan hasil dari proses pencarian parameter terbaik untuk prediksi APU_CPU dan APU_Memory menggunakan model Boosted Decision Tree dengan Grid Search berdasarkan nilai MSE. Proses ini mengevaluasi kombinasi berbagai hyperparameter, seperti learning_rate, n_

estimators, max_leaf_nodes, dan min_samples_leaf, untuk menghasilkan model yang memberikan prediksi paling akurat. Hasil Grid Search menunjukkan bahwa untuk prediksi APU_CPU, nilai MSE yang diperoleh adalah 0.1001, sedangkan untuk APU_Memory, nilai MSE lebih rendah, yaitu 0.0121. Parameter terbaik untuk model APU_CPU adalah {‘ learning_rate’: 0.2, ‘n_estimators’: 500, ‘max_leaf_nodes’: 10 , ‘min_samples_leaf’: 5}, sedangkan untuk APU_Memory adalah { ‘learning_rate’: 0.1, ‘max_leaf_nodes’: 10, ‘min_samples_leaf ’: 5, ‘n_estimators’: 500}. 54 Hasil ini menunjukkan bahwa mode l memiliki performa lebih baik dalam memprediksi APU_Memory dibandingkan APU_CPU, yang ditunjukkan oleh nilai MSE yang lebih kecil. Nilai MSE yang rendah mencerminkan tingkat kesalahan rata-rata antara prediksi model dan data aktual. Oleh karena itu, model dengan parameter terbaik ini digunakan untuk menghasilkan rekomendasi optimisasi sumber daya cloud secara lebih efektif. Gambar 5.9 penggalan kode sumber save model Langkah terakhir yang ditunjukkan pada Gambar 5.9 adalah menyimpan model yang telah dilatih dalam format .joblib, yang memungkinkan model digunakan kembali tanpa perlu melatih ulang. Tabel 5.9 Hasil save model Target Sebelum Sesudah APU_ CPU Tidak ada model file BoostedDecisionTree_CPU.joblib APU_Memory Tidak ada model file BoostedDecisionTree_Memory.joblib Tabel 5.9 menunjukkan perubahan kondisi sebelum dan sesudah pelatihan model dilakukan menggunakan kode program. Sebelum pelatihan, file model dalam format .joblib untuk APU_CPU dan APU_Memory belum ada karena proses pelatihan belum dilakukan. Setelah pelatihan selesai, model yang dilatih menggunakan algoritma Boosted Decision Tree disimpan dalam file .joblib, yaitu BoostedDecisionTree_CPU.joblib untuk APU_CPU dan BoostedDecisionTree_Memory.joblib untuk APU_Memory. File ini memungkinkan model digunakan langsung untuk proses prediksi tanpa perlu melakukan pelatihan ulang. 5.1.3 Hasil Optimisasi Proses optimisasi dilakukan untuk memberikan rekomendasi perubahan tipe mesin berdasarkan prediksi

APU dan parameter penggunaan sumber daya. Proses ini mencakup pembentukan nilai APU menggunakan model Boosted Decision Tree, pemberian rekomendasi tipe mesin berdasarkan nilai APU, dan optimisasi konfigurasi tipe mesin menggunakan PSO. Berikut penjelasan tahapan optimisasi: 1) Prediksi APU 55 Nilai APU_CPU dan APU_Memory dihitung untuk mencerminkan rata-rata prediksi tingkat penggunaan sumber daya CPU dan memori. Prediksi ini dilakukan dengan menggunakan model Boosted Decision Tree yang telah dilatih sebelumnya. Model ini memanfaatkan berbagai parameter masukan, seperti rata-rata dan puncak penggunaan CPU dan memori, serta rata-rata dan puncak permintaan baca dan tulis Cassandra. Kode berikut digunakan untuk tahap ini: Gambar 5.10 penggalan kode sumber prediksi APU Gambar 5.10 menjelaskan bahwa nilai APU_CPU dan APU_Memory dihitung menggunakan data masukan berupa penggunaan rata-rata dan puncak CPU serta memori, ditambah dengan permintaan baca dan tulis Cassandra. Hasil prediksi ini memberikan gambaran tingkat penggunaan sumber daya untuk CPU dan memori. Tabel 5.10 Hasil predict APU Nilai Metrics Sebelum Sesudah Cpu_usage: 45 Memory_usage: 60 Peak_cpu: 85 Peak_memory: 90 Write_req: 100 Write_peak: 250 Read_req: 80 Pead_peak: 200 APU_CPU: 0 APU_ Memory: 0 APU_CPU: 70.75 APU_Memory: 88.50 Tabel 5.10 menunjukkan hasil prediksi APU_CPU dan APU_Memory berdasarkan input data yang diberikan ke model Boosted Decision Tree. Input data terdiri dari nilai rata-rata dan puncak penggunaan CPU, memori, serta jumlah permintaan baca dan tulis Cassandra yang telah dinormalisasi. Proses prediksi dimulai dengan membangun DataFrame untuk menyusun fitur-fitur yang diperlukan model. Setiap fitur mencerminkan kondisi penggunaan sumber daya dalam satu periode tertentu. Model kemudian memprediksi nilai APU_CPU dan APU_Memory secara terpisah menggunakan model yang telah dilatih sebelumnya. Hasil prediksi menunjukkan nilai APU_CPU sebesar 70.75 dan APU_Memory sebesar 88.50, yang memberikan indikasi tingkat penggunaan sumber daya secara keseluruhan. 2) Penentuan

rekomendasi machine type 56 Setelah nilai APU diperoleh, rekomendasi tipe mesin diberikan berdasarkan ambang batas APU yang telah ditentukan. Jika nilai APU_CPU atau APU_Memory melampaui 80%, maka sistem akan merekomendasikan peningkatan tipe mesin (upgrade). Sebaliknya, jika nilai APU berada di bawah 40%, maka sistem akan merekomendasikan penurunan tipe mesin (downgrade). Untuk nilai APU yang berada di antara 40%–80%, rekomendasinya adalah mempertahankan tipe mesin saat ini (maintain). Kode berikut digunakan untuk menentukan rekomendasi: Gambar 5.11 penggalan kode sumber get recommendation Gambar 5.11 menjelaskan bahwa rekomendasi diberikan berdasarkan threshold APU. Jika rekomendasi untuk CPU dan memori berbeda, misalnya CPU memerlukan peningkatan sementara memori memerlukan penurunan, maka rekomendasi “mixed_adjustment” akan diberikan untuk menangani kedu a kondisi tersebut secara bersamaan. Tabel 5.11 Hasil recommendation Parameter Sebelum Sesudah APU_CPU: 70.75 Tidak ada Maintain APU_ Memory: 88.50 Tidak ada Upgrade Tabel 5.11 menggambarkan perubahan kondisi rekomendasi sebelum dan sesudah prediksi APU dilakukan. Sebelum prediksi, nilai APU_CPU dan APU_Memory tidak tersedia, sehingga tidak ada rekomendasi yang dapat dihasilkan. Setelah prediksi menggunakan fungsi, nilai APU_CPU tercatat sebesar 70.75 dan APU_ Memory sebesar 88.50. Berdasarkan nilai ini, sistem merekomendasikan maintain karena nilai APU_CPU berada di antara 40–80%, dan upgrade memori karena nilai APU_Memory melebihi 80%. 3) Optimisasi dengan PSO Setelah rekomendasi awal ditentukan, proses optimisasi konfigurasi tipe mesin dilakukan menggunakan PSO. Dalam konteks ini, PSO digunakan untuk mencari konfigurasi CPU, memori, dan harga yang paling efisien berdasarkan prediksi APU. 57 Gambar 5.12 penggalan kode sumber PSO Gambar 5.12 menjelaskan bahwa algoritme PSO menggunakan parameter swarmsize=30 dan maxiter=100. Parameter swarmsize=30 menunjukkan jumlah partikel dalam populasi yang digunakan untuk mencari solusi optimal. Semakin besar jumlah partikel, semakin luas ruang

solusi yang dapat dieksplorasi, namun dengan waktu komputasi yang lebih lama. Sementara itu, maxiter=100 menunjukkan jumlah iterasi maksimum yang akan dijalankan PSO untuk mencari solusi optimal. Dengan iterasi yang cukup, PSO dapat menemukan konfigurasi CPU, memori, dan harga yang paling efisien. Tabel 5.12 Hasil PSO Parameter Sebelum Sesudah vCPUs 2 2 Memory (GB) 12 8 Price (USD) 51.67 41.72
Tabel 5.12 menunjukkan hasil optimisasi konfigurasi sumber daya menggunakan algoritma PSO. Sebelum optimisasi, nilai parameter seperti vCPUs, Memory, dan Price menggunakan nilai existing dari machine type yang digunakan. Setelah optimisasi, sistem merekomendasikan konfigurasi optimal dengan 2 vCPUs, 8 GB Memory, dan perkiraan biaya sebesar 41.72 USD. Proses PSO dijalankan dalam 30 partikel dan 100 iterasi, dengan mempertimbangkan batasan bawah dan atas untuk vCPUs, Memory, serta harga saat ini. Hasil ini memberikan solusi yang lebih efisien berdasarkan kinerja model dan kebutuhan sumber daya. 4) Penentuan nearest machine type Setelah PSO menghasilkan konfigurasi optimal, sistem mencari tipe mesin terdekat dari daftar kandidat yang tersedia. Tipe mesin terdekat dipilih berdasarkan efisiensi sumber daya dan harga. Proses ini memastikan bahwa 58 hasil optimisasi dapat diterapkan menggunakan tipe mesin yang sudah ada dalam daftar konfigurasi. Gambar 5.13 penggalan kode sumber nearest machine type Gambar 5.13 menjelaskan bahwa tipe mesin terdekat dipilih berdasarkan efisiensi sumber daya dan harga. Kandidat disaring untuk memastikan spesifikasi yang sesuai dengan hasil optimisasi dari PSO, lalu kandidat terbaik dipilih berdasarkan jarak sumber daya (CPU dan memori) serta harga. Tabel 5.13 Hasil nearest machine type Parameter Sebelum Sesudah vCPU: 2, Memory: 12GB Tidak diketahui N2-highmem-2
Tabel 5.13 menunjukkan hasil dari proses pencarian tipe mesin yang paling mendekati konfigurasi optimal berdasarkan nilai yang diberikan oleh algoritma PSO. Sebelum proses pencarian, spesifikasi tipe mesin seperti jumlah vCPU dan kapasitas memori tidak diketahui. Namun,

setelah menjalankan kode program, algoritma menemukan rekomendasi tipe mesin terdekat, yaitu N2- highmem-2, dengan spesifikasi 2 vCPUs dan 16 GB memori. 5.1.4 Hasil Analisis BigQuery dan GCP Report Proses analisis sumber daya cloud dilakukan menggunakan BigQuery dan laporan dari GCP untuk mengidentifikasi sumber daya yang tidak terpakai secara efisien. Analisis ini mencakup tiga kategori utama, yaitu snapshot lama yang tidak digunakan, IP address dan disk yang tidak terpakai, serta project GCP yang hanya mengaktifkan API monitoring. Berikut adalah hasil dari setiap kategori analisis. 5.1.4.1 Snapshot lama yang tidak digunakan Gambar 5.14 penggalan kodesumber fungsi get_old_ snapshots Gambar 5.14 menunjukkan fungsi utama get_old_snapshots yang dirancang untuk mengidentifikasi snapshot lama berdasarkan kriteria ukuran dan tanggal. Fungsi ini memulai proses dengan mendapatkan daftar project melalui google_client.get_list_projects, kemudian menjalankan fungsi 59 list_snapshots secara paralel menggunakan utils.parallel_ jobs. Hasil snapshot dari berbagai project kemudian difilter menggunakan fungsi filter_snapshots, dengan kriteria ukuran minimum dan usia maksimum snapshot yang ditentukan melalui parameter input. Hasil akhir diformat sebagai laporan JSON dan dikirim ke Google Sheets menggunakan google_client.report_to_gsheet. Gambar 5.15 penggalan kodesumber fungsi generate_date Gambar 5.15 menampilkan kode fungsi generate_date, yang bertanggung jawab untuk menghitung tanggal maksimum berdasarkan parameter usia snapshot, termasuk tahun, bulan, dan hari. Fungsi ini menghitung waktu mundur dari tanggal saat ini dengan mengonversi parameter waktu menjadi durasi dalam hari menggunakan timedelta. Tanggal yang dihasilkan digunakan sebagai batas waktu untuk menentukan snapshot mana yang termasuk dalam kriteria lama. Gambar 5.16 kodesumber fungsi list_ snapshots Gambar 5.16 adalah kode untuk fungsi list_snapshots, yang digunakan untuk mengambil daftar snapshot dari setiap project menggunakan klien Google Compute API. Pada fungsi ini, kredensial diinisialisasi menggunakan google_client.get_credential, dengan lingkup

akses yang mencakup seluruh platform cloud. Snapshot yang berhasil diambil dimasukkan ke dalam daftar dengan atribut seperti nama, ukuran, tanggal pembuatan, lokasi penyimpanan, dan tautan langsung. 60 Gambar 5.17 Penggalan kodesumber fungsi filter_snapshots Gambar 5.17 menunjukkan fungsi filter_snapshots, yang menyaring daftar snapshot berdasarkan kriteria ukuran minimum dan usia maksimum yang telah dihitung. Setiap snapshot diperiksa dengan membandingkan tanggal pembuatan (diubah menjadi format datetime) dengan tanggal maksimum yang dihitung sebelumnya. Snapshot yang memenuhi kriteria kemudian dimasukkan ke dalam daftar hasil untuk disusun dalam laporan akhir. Gambar 5.18 Hasil get old snapshot Gambar 5.18 menunjukkan hasil dari fungsi get_old_ snapshots, yang digunakan untuk mengidentifikasi snapshot lama dengan ukuran besar yang masih tersimpan dalam berbagai proyek GCP. Data yang ditampilkan mencakup informasi seperti nama snapshot, ukuran dalam GB, potensi penghematan biaya dalam USD, lokasi penyimpanan, tanggal pembuatan, dan tautan akses langsung ke snapshot terkait. Laporan ini menyajikan snapshot dari berbagai proyek dengan ukuran mulai dari 50 GB hingga 500 GB, serta potensi penghematan biaya yang bervariasi. Informasi ini memberikan gambaran rinci tentang snapshot yang tidak lagi diperlukan, berdasarkan kriteria usia dan ukuran yang telah ditentukan, sehingga dapat membantu dalam pengambilan keputusan untuk menghapus snapshot yang 61 tidak terpakai guna mengoptimalkan penggunaan sumber daya cloud dan mengurangi biaya operasional. 5.1.4.2 IP address dan disk yang tidak terpakai Gambar 5.19 kodesumber fungsi get_detached_ips_disks Gambar 5.19 menunjukkan fungsi get_ detached_ips_disks, yang bertujuan untuk mengidentifikasi IP address dan disk yang tidak terpakai dengan memanfaatkan BigQuery. Fungsi ini memulai proses dengan menginisialisasi credentials dan client BigQuery untuk project billing- production-infra-***. Fungsi ini menghitung tanggal previous Sunday berdasarkan waktu saat ini, lalu menggunakannya sebagai batas waktu untuk melakukan query terhadap data rekomendasi penggunaan

sumber daya dari tabel recommendations_export. Data hasil query mencakup informasi tentang jenis rekomendasi, sumber daya target, lokasi, deskripsi, serta estimasi biaya yang dapat dihemat. Gambar 5.20 kodesumber query get detached ips disks Gambar 5.20 menampilkan kode query SQL yang digunakan dalam fungsi get_detached_ips_disks. Query ini memfilter data berdasarkan kolom recommender, dengan hanya memilih rekomendasi dari IdleResourceRecommender untuk IP address dan disk. Query juga memastikan bahwa data diambil hanya dari partisi tabel dengan waktu tertentu, yang dihitung dari previous Sunday. Kolom-kolom seperti recommender, recommender_subtype, dan primary_impact. cost_projection. cost_in_local_currency diambil untuk memberikan informasi detail tentang rekomendasi dan potensi penghematan biaya. 62 Gambar 5.21 Penggalan kodesumber detached ips disk report to gsheet Gambar 5.21 memperlihatkan bagian kode yang mengolah hasil query BigQuery menjadi laporan. Setiap baris data diolah untuk mendapatkan informasi project, tipe sumber daya, rekomendasi, nama sumber daya, kode mata uang biaya, unit biaya, dan deskripsi. Informasi ini kemudian disusun dalam format laporan dengan header dan isi data, lalu dikirim ke Google Sheets menggunakan fungsi google_client.report_ to_gsheet. Gambar 5.22 Hasil analisis deattach ip and disks Gambar 5.22 menunjukkan hasil analisis terhadap IP address dan disk yang tidak terpakai di GCP. Tabel ini berisi informasi proyek, jenis sumber daya, rekomendasi, nama sumber daya, potensi penghematan biaya dalam satuan USD dan IDR, serta deskripsi yang menjelaskan alasan rekomendasi, seperti penghapusan IP address atau disk yang tidak digunakan. Analisis ini bertujuan untuk membantu identifikasi sumber daya yang tidak efisien, sehingga dapat dihapus untuk mengurangi biaya operasional yang tidak diperlukan. 63 5.1.4.3 Project GCP yang hanya mengaktifkan API monitoring Gambar 5.23 Penggalan kodesumber fungsi list assets Gambar 5.23 menunjukkan fungsi list_assets, yang dirancang untuk mendapatkan daftar tipe asset yang ada pada setiap project

GCP. Dengan menggunakan klien asset_v1, fungsi ini membuat permintaan API untuk mencatat semua tipe asset dalam project tertentu. Parameter parent digunakan untuk menunjukkan cakupan project yang relevan. Jika terjadi kesalahan saat mengambil data asset, fungsi ini akan memberikan pesan kesalahan sebagai umpan balik. Gambar 5.24 kodesumber fungsi get cost project Gambar 5.22 menjelaskan fungsi get_cost_ project, yang bertujuan untuk mendeteksi project GCP yang hanya memiliki layanan API monitoring aktif tanpa layanan lain. Fungsi ini menggunakan client BigQuery untuk menjalankan query yang menghitung biaya bulanan berdasarkan dataset daily_gcp_billing_*** . Hasil query ini kemudian difilter untuk menemukan project yang hanya menggunakan API monitoring selama periode waktu tertentu, tanpa melibatkan layanan lain dengan biaya tambahan. Data yang diperoleh mencakup project_id dan biaya penggunaan. 64 Gambar 5.25 kodesumber report to gsheet fungsi get cost project Gambar 5.25 memperlihatkan bagian kode yang memproses data hasil query dengan menambahkan catatan tambahan tentang asset yang digunakan pada setiap project. Proses ini melibatkan fungsi list_assets untuk mendeteksi aset unik yang dimiliki oleh project tersebut. Hasilnya dirangkum dalam catatan (notes) untuk setiap project dan disimpan ke dalam laporan yang diunggah ke Google Sheets. Gambar 5.26 Hasil monitoring costs project Hasil pada Gambar 5.26 menunjukkan data project di Google Cloud Platform (GCP) yang hanya mengaktifkan API monitoring. Tabel tersebut berisi kolom project_id, biaya penggunaan (invoice_usd), dan catatan (notes) yang berisi daftar aset terkait yang digunakan oleh project tersebut. Biaya yang tercatat merepresentasikan total pengeluaran dari project terkait, yang sebagian besar hanya menggunakan layanan monitoring seperti Cloud Monitoring tanpa adanya penggunaan layanan lain. Informasi ini memberikan gambaran tentang project - project yang memiliki aktivitas terbatas, sehingga dapat diidentifikasi untuk penghematan biaya atau penyesuaian konfigurasi berdasarkan analisis kebutuhan sumber daya. 65

5.1.5 Hasil DevOps Proses DevOps untuk pipeline Machine learning dan analisis sumber daya diimplementasikan menggunakan konfigurasi CI/CD GitLab. Pipeline yang diterapkan dibagi ke dalam tiga bagian utama, yaitu pipeline merge request, pipeline Machine learning, dan pipeline analisis sumber daya. Setiap bagian memiliki stages yang saling mendukung untuk memastikan proses berjalan secara otomatis, konsisten, dan efisien. 1) Pipeline Unit Testing Pipeline ini bertujuan untuk memastikan bahwa setiap perubahan kode yang diusulkan melalui merge request telah diuji secara otomatis sebelum digabungkan ke dalam branch utama. Pipeline ini terdiri dari tahapan berikut: A. Stage setup Gambar 5.27 Penggalan kode sumber stage setup Pada Gambar 5.27 ditunjukkan stage setup yang terdapat job setup-job yang memastikan bahwa lingkungan pengembangan Python dibuat menggunakan virtual environment, dan semua dependencies yang diperlukan di-install untuk kebutuhan stage selanjutnya. B. Stage unit test Gambar 5.28 kode sumber unit testing gcp reports 66 Gambar 5.29 kode sumber unit testing machine learning Gambar 5.28 dan Gambar 5.29 menunjukkan bahwa stage ini terdiri dari dua job, yaitu unit_test_gcp_reports dan unit_test_machine_learning, yang berfungsi untuk menjalankan pengujian unit pada kode untuk modul gcp_reports dan recommendation_vm. Hasil pengujian disimpan dalam format JUnit XML, yang memungkinkan integrasi dengan sistem pelaporan pengujian. Gambar 5.30 Tampilan pipeline unit testing Gambar 5.30 merupakan hasil tampilan dari penerapan stage setup dan unit test yang telah dijelaskan di atas. Pada pipeline ini, kedua stage berjalan sesuai alur yang ditentukan, di mana setup-job berhasil mempersiapkan lingkungan pengembangan, diikuti oleh unit_test_machine_learning dan unit_tests_gcp_reports yang sukses menjalankan pengujian unit tanpa adanya kegagalan. Tampilan ini menunjukkan bahwa setiap job telah diselesaikan dengan status success, memastikan bahwa konfigurasi dan pengujian unit pada kedua modul berfungsi dengan baik. 2) Pipeline Machine learning Pipeline ini

dirancang untuk menjalankan alur kerja Machine learning dari awal hingga akhir, termasuk pengumpulan data, pemrosesan, dan pelatihan model. Stage stagenya adalah sebagai berikut: A. Stage Setup 67 Tahap ini sama dengan pipeline merge request yang terdapat pada gambar 5.27, yaitu memastikan lingkungan Python disiapkan dengan benar. B. Stage collect_data Gambar 5.31 kode sumber stage collect data Gambar 5.31 menunjukkan kode sumber untuk stage collect_data dalam pipeline Machine learning. Pada stage ini, job yang bernama collect_ data dijalankan untuk menginisialisasi proses pengumpulan data menggunakan script collect_data.py. Script ini bertugas mengambil data metrik yang diperlukan dari Datadog dan menyimpannya ke dalam direktori output ($OUTPUT_DIR). Hasil dari stage ini disimpan sebagai artifacts yang akan digunakan pada tahap-tahap selanjutnya dalam pipeline Machine learning. C. Stage Preprocess_data Gambar 5.32 Penggalan kode sumber stage preprocess data Gambar 5.32 menunjukkan kode sumber untuk stage preprocess_data dalam pipeline Machine learning. Pada stage ini, terdapat job bernama preprocess_data, yang menjalankan script data_ preprocessing.py. Script ini berfungsi untuk mengolah data mentah yang telah dikumpulkan sebelumnya menjadi bentuk data yang lebih terstruktur dan siap digunakan pada tahapan berikutnya dalam pipeline. Hasil dari proses ini disimpan di direktori output 68 ($OUTPUT_DIR) sebagai artifacts untuk memastikan data dapat diakses oleh stage berikutnya. D. Stage Calculate APU Gambar 5.33 Penggalan kode sumber calculate apu job Gambar 5.33 menunjukkan kode sumber untuk stage calculate_ apu dalam pipeline Machine learning. Pada stage ini terdapat job bernama calculate_apu, yang menjalankan script calculate_apu.py. Script ini bertugas menghitung nilai APU berdasarkan data yang telah diproses sebelumnya. Nilai APU yang dihasilkan akan menjadi dasar untuk memberikan rekomendasi optimisasi sumber daya pada tahapan berikutnya. Hasil perhitungan disimpan dalam direktori visualisasi ($VISUALIZATION_ DIR) sebagai artifacts untuk mendukung proses lanjutan. E. Stage

Train Model Gambar 5.34 Penggalan kode sumber train model job Gambar 5.34 menunjukkan kode sumber untuk stage train_model dalam pipeline Machine learning. Pada stage ini terdapat job bernama train_ model, yang bertugas melatih model Machine learning menggunakan data hasil preprocessing dan nilai APU yang telah dihitung sebelumnya. Proses pelatihan model ini bertujuan untuk menghasilkan model yang mampu memberikan rekomendasi optimisasi sumber daya berdasarkan pola penggunaan yang terdeteksi. 69 Hasil pelatihan model disimpan dalam direktori $OUTPUT_DIR sebagai artifacts yang akan digunakan pada tahapan berikutnya. Gambar 5.35 Penggalan Kodesumber optimize resource job Gambar 5.35 menunjukkan kode sumber untuk stage optimize_resources dalam pipeline Machine learning. Pada stage ini terdapat job bernama resource_optimize, yang bertugas menjalankan script resources_optimize.py. Job ini menggunakan hasil model yang telah dilatih sebelumnya, yang disimpan di direktori $MODELS_DIR, serta data biaya sumber daya dari $COST_DATA_PATH. Proses optimisasi bertujuan untuk memberikan rekomendasi perubahan konfigurasi sumber daya, seperti upgrade, downgrade, atau maintain tipe mesin virtual, berdasarkan hasil prediksi APU dan data biaya. Tahapan ini menjadi langkah terakhir dalam pipeline, di mana hasil optimisasi disimpan dan dapat digunakan oleh tim untuk implementasi perubahan tipe mesin virtual yang lebih efisien secara biaya dan sumber daya. Gambar 5.36 Tampilan pipeline machine learning 1 Gambar 5.37 Tampilan pipeline machine learning 2 Gambar 5.36 dan gambar 5.37 menunjukkan tampilan hasil dari penerapan pipeline Machine learning secara keseluruhan, mencakup seluruh tahapan mulai dari pengumpulan data, pra-pemrosesan, perhitungan APU, pelatihan model, hingga siap digunakan untuk proses optimisasi sumber daya. 3) Pipeline Resources Analysis 70 Pipeline ini bertujuan untuk melakukan analisis sumber daya secara otomatis menggunakan data yang dikumpulkan. Hasil dari analisis ini berupa rekomendasi optimisasi seperti penyesuaian tipe mesin virtual (VM), deteksi sumber daya yang tidak digunakan, serta

evaluasi biaya proyek. Pipeline ini memiliki dua tahap utama: A. Stage setup Stage ini berfungsi sama seperti stage setup yang digunakan pada pipeline merge request maupun pipeline Machine learning, sebagaimana dijelaskan sebelumnya pada Gambar 5.27. Tahap ini memastikan lingkungan pengembangan Python siap digunakan oleh seluruh job di stage berikutnya. B. Stage resources analysis Stage ini dirancang untuk menjalankan berbagai job yang berfungsi mendukung analisis sumber daya, seperti memberikan rekomendasi VM, menganalisis biaya proyek, mendeteksi disk dan IP yang tidak terpakai, serta mengidentifikasi snapshot lama. Detail masing-masing job akan dijelaskan di bawah ini. Gambar 5.38 Penggalan kode stage recommendation vm Gambar 5.38 menunjukkan konfigurasi job generate-recommendation-vm dalam tahap resource_analysis, yang bertugas menjalankan analisis sumber daya menggunakan skrip main.py dengan parameter $COMMAND. Job ini mengaktifkan lingkungan python virtual environment, memastikan PYTHONPATH telah dikonfigurasi dengan benar, dan menjalankan skrip yang menghasilkan rekomendasi tipe VM berdasarkan model machine learning yang telah dilatih serta data penggunaan sumber daya terkini. Eksekusi job ini bergantung pada setup-job untuk memastikan bahwa lingkungan eksekusi telah siap, dan hanya dijalankan berdasarkan schedule dengan variabel $COMMAND yang ditetapkan sebagai get-resources- recommendation. 71 Gambar 5.39 Penggalan kode sumber pipeline cost project job Gambar 5.39 menunjukkan konfigurasi job generate_cost_project_job dalam tahap resource_analysis, yang bertugas menjalankan analisis biaya proyek menggunakan skrip main.py dengan parameter $COMMAND yang ditetapkan sebagai cost-project dan variabel $MAX_REQUESTS. Job ini mengaktifkan python virtual environment, menjalankan skrip untuk mengidentifikasi proyek dengan biaya yang tidak optimal berdasarkan data yang tersedia, serta menampilkan pesan status eksekusi. Eksekusi job ini bergantung pada setup-job untuk memastikan lingkungan eksekusi telah siap dan hanya dijalankan berdasarkan schedule dengan variabel

$COMMAND dan $MAX_REQUESTS yang telah ditetapkan. Gambar 5.40 Penggalan kode sumber pipeline deattach ip disk job Gambar 5.40 menunjukkan konfigurasi job deattach_ip_disk_job dalam tahap resource_ analysis, yang bertugas menjalankan analisis terhadap disk dan IP address yang tidak terpakai menggunakan perintah python main.py deattach-ip-disk. Job ini mengaktifkan python virtual environment, mencetak pesan status eksekusi, dan memproses data yang diperoleh melalui integrasi dengan BigQuery untuk mengidentifikasi sumber daya yang tidak aktif agar dapat dihapus guna mengurangi biaya operasional. Eksekusi job ini bergantung pada setup-job untuk memastikan lingkungan telah siap dan hanya dijalankan berdasarkan schedule dengan variabel $COMMAND yang telah ditetapkan sebagai deattach-ip-disk. 72 Gambar 5.41 Penggalan kode sumber old snapshot job Gambar 5.41 menunjukkan konfigurasi job old_snapshots_job dalam tahap resource_analysis, yang bertugas menjalankan analisis terhadap snapshot lama menggunakan perintah python main.py old-snapshots dengan beberapa parameter, seperti -y untuk tahun, -m untuk bulan, -d untuk hari, dan -ds untuk ukuran disk. Parameter ini memberikan fleksibilitas dalam menentukan kriteria snapshot yang akan dianalisis guna mengoptimalkan penggunaan sumber daya cloud. Job ini mengaktifkan python virtual environment, dan memproses data untuk mengidentifikasi snapshot lama yang memenuhi kriteria agar dapat dihapus. Eksekusi job ini bergantung pada setup-job untuk memastikan lingkungan telah siap dan hanya dijalankan berdasarkan schedule dengan variabel $COMMAND yang telah ditetapkan sebagai old-snapshots, serta beberapa variabel tambahan seperti $JOBS, $YEARS, $MONTHS, $DAYS, dan $DISK_SIZE. Gambar 5.42 Tampilan resources analysis 73 Gambar 5.43 sample tampilan pipeline saat running Gambar 5.42 dan 5.43 menunjukkan hasil implementasi pipeline schedule untuk beberapa tugas otomatisasi dalam optimisasi sumber daya cloud. Pipeline ini terdiri dari beberapa tugas utama yang dijalankan berdasarkan jadwal (cron schedule) yang telah ditentukan. Tugas utama dalam

pipeline schedule ini meliputi lima job utama, yaitu Generate new model ML, Get old snapshot report, Get detect unused project, Get deattach IP and disks and Get recommendation VM resources. Kelima job tersebut masing- masing bertugas menjalankan analisis dan optimisasi sumber daya, seperti memberikan rekomendasi tipe mesin, menganalisis biaya proyek, mendeteksi disk dan IP yang tidak digunakan, serta mengidentifikasi snapshot lama. Semua job dirancang untuk memastikan efisiensi dan optimalisasi penggunaan sumber daya cloud secara menyeluruh. 5.2 Pembahasan Bagian ini membahas hasil pengujian aplikasi yang dikembangkan untuk mendeteksi dan mengoptimalkan penggunaan sumber daya cloud. Pengujian dilakukan dengan tiga metode utama: A/B testing, white box testing, dan evaluasi performa model menggunakan confusion matrix. Analisis hasil dilakukan secara naratif, dengan mengacu pada data hasil uji tanpa mengulangi penjelasan kode atau tangkapan layar dari subbab sebelumnya. Metode A/B Testing diterapkan untuk membandingkan biaya penggunaan sumber daya cloud sebelum dan sesudah optimisasi. Optimisasi dilakukan melalui integrasi Machine Learning menggunakan Boosted Decision Tree dan algoritme Particle Swarm Optimization (PSO). Tujuan utama pengujian ini adalah untuk 74 menilai efektivitas pendekatan optimisasi dalam mengurangi biaya operasional cloud. Analisis dilakukan dengan menghitung penghematan biaya dalam bentuk nominal dan persentase, yang menunjukkan efisiensi dari metode yang digunakan. Selanjutnya, pengujian performa model dilakukan menggunakan confusion matrix untuk mengevaluasi akurasi prediksi. Matriks ini mengidentifikasi prediksi model yang benar dan salah melalui nilai True Positive (TP), False Positive (FP), True Negative (TN), dan False Negative (FN). Berdasarkan matriks ini, nilai evaluasi seperti accuracy, precision, recall, dan F1-score dihitung menggunakan formula yang sesuai. Pengukuran ini memberikan gambaran tentang ketepatan model dalam memberikan rekomendasi optimisasi yang sesuai dengan kondisi aktual. Selain itu, metode white box testing digunakan untuk

memastikan bahwa setiap fungsi inti dalam aplikasi berjalan sesuai desain. Fungsi-fungsi ini meliputi praproses data, pemuatan model, prediksi APU, hingga optimisasi dengan PSO. Pengujian memastikan bahwa semua fungsi bebas dari kesalahan logika dan mampu menangani data dengan format yang sesuai. Hasil dari ketiga metode pengujian ini memberikan gambaran komprehensif tentang kinerja aplikasi yang dikembangkan. Dari segi penghematan biaya, aplikasi berhasil menunjukkan efektivitas dalam mengurangi biaya operasional cloud. Ketepatan prediksi model tercermin dari evaluasi confusion matrix, yang menghasilkan metrik akurasi yang tinggi. Sementara itu, kelancaran fungsi utama dalam pipeline memastikan aplikasi dapat diandalkan dan efisien untuk memenuhi kebutuhan optimisasi sumber daya cloud secara menyeluruh. 5.2.1 Pengujian A/B Testing Pada bagian ini, pengujian A/B Testing dilakukan untuk membandingkan biaya penggunaan sumber daya sebelum dan sesudah optimisasi dengan Machine Learning Boosted Decision Tree yang diintegrasikan dengan PSO. Pengujian ini bertujuan untuk mengevaluasi efektivitas metode PSO dalam mengurangi biaya penggunaan sumber daya cloud. Data yang digunakan meliputi biaya sebelum optimisasi (Cost Before) dan biaya setelah optimisasi (Cost After PSO) dalam waktu 10 bulan. Tabel 5.14 Hasil pengujian A/B Testing untuk Cost Saving Month Cost Before (USD) Cost After PSO (USD) Cost Saving (USD) Cost Saving (%) 2024-01 3,438.84 3,068.73 370.11 10.76% 2024-02 3,649.59 3,478.43 171.16 4.69% 2024-03 3,601.85 3,402.95 198.90 5.52% 2024-04 2,933.41 2,779.58 153.83 5.24% 2024-05 2,933.41 2,571.07 362.34 12.35% 2024-06 3,498.75 3,190.32 308.43 8.81% 2024-07 3,219.12 2,972.41 246.71 7.66% 2024-08 3,580.26 3,254.36 325.90 9.10% 2024-09 3,765.48 3,440.27 325.21 8.64% 2024-10 3,543.82 3,212.34 331.48 9.35% Tabel 5.14 menunjukkan hasil pengujian A/B Testing untuk setiap bulan dalam rentang waktu pengujian. Cost Before mencerminkan total biaya penggunaan sumber daya pada PT. XYZ sebelum optimisasi, sedangkan Cost After PSO 75 dihitung berdasarkan rekomendasi tipe mesin menggunakan

algoritme machine learning. Kolom Cost Saving (USD) menunjukkan jumlah penghematan biaya dalam dolar, sementara kolom Cost Saving (%) mengindikasikan persentase penghematan terhadap biaya awal, sebagaimana dijelaskan pada Bab 3.2.1 pada bagian metode pengujian A/B Testing dan bab 4.3.1 pada bagian rancangan pengujian A/B Testing. Hasil pengujian ini menunjukkan bahwa sistem berhasil memberikan penghematan biaya yang signifikan setiap bulan. Selama periode 10 bulan pengujian, total biaya sebelum optimisasi mencapai $33,164.53, sedangkan total biaya setelah implementasi sistem machine learning berkurang menjadi $30,371.25, menghasilkan total penghematan sebesar $2,793.28. Penghematan biaya tertinggi terjadi pada Mei 2024, sebesar 12.35% atau $362.34, sedangkan penghematan terendah tercatat pada februari 2024, sebesar 4.69% atau $171.16. Secara keseluruhan, rata-rata penghematan biaya selama periode pengujian mencapai 8.12%, menunjukkan bahwa metode optimisasi yang digunakan efektif dalam mengurangi biaya penggunaan sumber daya cloud. Selain cost saving, analisis juga dilakukan untuk mengukur persentase optimisasi secara keseluruhan berdasarkan validitas rekomendasi yang dihasilkan oleh sistem. Persentase optimisasi dihitung berdasarkan kombinasi rekomendasi maintain, downgrade, dan upgrade, termasuk kondisi di mana tipe mesin sudah tidak dapat diubah lebih lanjut karena keterbatasan. Tabel 5.15 Pengujian A/B Testing untuk cost optimization Month Total Recommendation Valid Optimizations Percentage Optimization (%) 2024-01 36 33 92% 2024-02 39 39 100% 2024-03 38 35 92% 2024-04 34 34 100% 2024-05 34 32 91% 2024-06 34 32 94% 2024-07 34 32 94% 2024-08 34 31 91% 2024-09 34 32 92% 2024-10 41 38 93% Tabel 5.15 menunjukkan persentase optimisasi berdasarkan rekomendasi yang valid, termasuk kondisi di mana tipe mesin sudah berada pada batas minimum atau maksimum kapasitas. Persentase optimisasi dihitung dengan membandingkan jumlah rekomendasi valid dengan total rekomendasi yang dihasilkan oleh sistem dalam 10 bulan, sesuai yang dijelaskan pada Bab 3.2.1 pada bagian metode

pengujian A/B Testing dan bab 4.3.1 pada bagian rancangan pengujian A/B Testing. Hasil pengujian ini menunjukkan bahwa rata-rata persentase optimisasi untuk seluruh periode pengujian mencapai 94%, yang mengindikasikan bahwa metode yang digunakan mampu memberikan rekomendasi valid untuk sebagian besar kasus. Hal ini memastikan bahwa sistem tidak hanya mengurangi biaya tetapi juga memberikan optimisasi yang sesuai dengan kondisi penggunaan sumber daya cloud, menjadikan solusi lebih efektif dan efisien. 76 5.2.2 Pengujian White box Pada pengujian white box, pengujian difokuskan pada verifikasi fungsi-fungsi inti aplikasi, termasuk proses data preprocessing, penyimpanan token, pemuatan model Machine Learning, dan pengolahan input dari pengguna. Pengujian dilakukan untuk memastikan bahwa setiap fungsi berjalan lancar tanpa error atau ketidaksesuaian terhadap desain awal aplikasi. Hasil pengujian menunjukkan bahwa seluruh alur aplikasi telah berfungsi sesuai dengan rancangan. Setiap tahapan, mulai dari data preprocessing hingga proses prediksi, berhasil dijalankan dengan baik dan memenuhi kebutuhan aplikasi. Hal ini menandakan bahwa semua komponen internal telah diimplementasikan dengan benar dan mampu menjalankan fungsi-fungsi yang diperlukan secara efisien. Gambar 5.44 Hasil unit testing Gambar 5.45 Laporan unit test machine learning 77 Gambar 5.46 Laporan unit test gcp report Gambar 5.44 - Gambar 5.46 menampilkan hasil pengujian unit test di pipeline Gitlab untuk modul Machine Learning dan modul G 26 P Reports. Dari hasil pengujian, terlihat bahwa seluruh unit tests berhasil dijalankan tanpa ada kegagalan, error, atau tes yang dilewati (skipped). Proses pengujian memakan w ktu total 34.17 detik, dengan masing-masing modul menunjukkan tingkat keberhasilan 100%, memastikan bahwa setiap fungsi berjalan sebagaimana mestinya. 5.2.3 Pengujian Confusion Matrix Gambar 5.47 Hasil Confusion Matrix Gambar 5.47 menunjukkan hasil confusion matrix yang diperoleh dari pengujian model menggunakan library scikit-learn. Hasil ini menggambarkan performa model dalam membedakan kelas True (1) dan False (0) pada data uji

yang terdiri dari 361 sampel. Berikut adalah nilai-nilai yang diperoleh dari confusion matrix: a) True Positives (TP) : 169 (kelas 1 yang diprediksi benar) b) True Negatives (TN) : 168 (kelas 0 yang diprediksi benar) c) False Positives (FP) : 22 (kelas 0 yang diprediksi salah) d) False Negatives (FN) : 2 (kelas 1 yang diprediksi salah) 78 1. Accuracy Perhitungan Hasil (%) $accuracy$ == TP +TN TP +TN+ FP +FN $accuracy$ = 169+168 169+168+=22 +2 $accuracy$ = 337 361 $accuracy$ = 0.93 93 % 2. Precision Perhitungan Hasil (%) $preci$ ion + == TN TN +FN $precisi$ n (== TP TP +FP $precision$ + = = 168 168+22 $precision$ + == 168 190 pr $ecision$ + = =0.88 $precision$ (= = 169 169+2 $precision$ (== 169 171 pr $ecision$ (= =0.98 Kelas 0 = 88 % Kelas 1 = 98 % 3. Recall Perhitungan Hasil (%) $recall$ + == TN TN +FP $recall$ (== T 19 TP +FN $reca$ l + == 168 168+2 Kelas 0 = 98 % Kelas 1 = 88 % 79 $recall$ + == 168 170 $recall$ + = =0. 98 $recall$ (== 169 169+22 $recall$ (== 1 9 191 $recall$ (= =0.88 4. F1-score Perhitungan Hasil (%) $F1$=$scor$ e = =2=x =$precision$=x=$recal$ l $precision$ +$recall$ $F1$=Sco re = =2=x 0.88=x=0.98 0.88+0.98 $F1$=$Score$ = 1.72 1.86 $F1$=$Score$ = 0,93 93 % Hasil perhitungan classification report menggunakan library sklearn ditampilkan pada Gambar 5.48. Gambar 5.48 Hasil Classification report 80 BAB VI PENUTUP Bab ini membahas kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut. Kesimpulan menjelaskan hasil penelitian secara ringkas dan informatif, sedangkan

saran memberikan masukan untuk penelitian lanjutan atau pengembangan sistem di masa mendatang. 6.1 Kesimpulan Penelitian ini berhasil mengembangkan sistem automasi DevOps untuk optimalisasi biaya cloud di e-commerce. Berikut adalah kesimpulan utama dari penelitian ini: 1. Sistem DevOps untuk optimalisasi biaya cloud telah dikembangkan dengan pipeline otomatisasi di GitLab CI/CD, yang mencakup pengumpulan data, preprocessing, pelatihan model, dan optimasi sumber daya. Pipeline ini memastikan proses berjalan otomatis, mengurangi ketergantungan pada pekerjaan manual, dan meningkatkan efisiensi operasional. 2. Evaluasi performa Machine Learning menggunakan Boosted Decision Tree menunjukkan hasil yang baik, dengan akurasi 93%, presisi kelas positif 88%, recall 98%, dan F1-score 93%. Untuk kelas negatif, presisi mencapai 98%, recall 88%, dan F1-score 93%, membuktikan model mampu memberikan rekomendasi akurat berdasarkan penggunaan CPU dan memori. 3. Sistem berhasil mengintegrasikan Particle Swarm Optimization (PSO) untuk menentukan konfigurasi tipe mesin optimal berdasarkan threshold sumber daya 40%-80%. PSO memberikan rekomendasi peningkatan (upgrade) atau penurunan (downgrade) tipe mesin dengan efisiensi tinggi. 4. Integrasi dengan BigQuery mendukung deteksi sumber daya yang tidak terpakai, seperti IP address, snapshots, dan disk. Rekomendasi yang dihasilkan mampu mengurangi pemborosan sumber daya dan mengoptimalkan biaya operasional. 5. Implementasi DevOps dengan pipeline CI/CD memastikan otomatisasi dalam pengolahan data, pelatihan model, dan penerapan rekomendasi. Sistem ini meningkatkan konsistensi proses dan mengurangi waktu pengerjaan manual dengan mempercepat siklus pengembangan serta implementasi. 6. Sistem ini memberikan rekomendasi optimasi yang berpotensi menghemat biaya cloud hingga 8,12% selama periode pengujian 10 bulan (Januari–Oktober 2024), dengan validasi persentase optimasi mencapai 94%. Hasil ini menunjukkan bahwa sebagian besar rekomendasi yang dihasilkan sistem efektif dalam meningkatkan efisiensi penggunaan sumber daya. Kesimpulan ini menunjukkan bahwa sistem yang dikembangkan

dapat diadopsi untuk mengelola dan mengoptimalkan biaya cloud secara lebih efisien dalam lingkungan e-commerce. 81 6.2 Saran Untuk pengembangan lebih lanjut, berikut adalah beberapa saran yang dapat dipertimbangkan: 1. Penggunaan data penggunaan sumber daya dapat diperluas dengan mengintegrasikan data dari lebih banyak cluster Cassandra agar model lebih general dan robust terhadap variasi data. 2. Menerapkan algoritma optimasi sumber daya lain, seperti Random Forest, Gradient Boosting, atau kombinasi metaheuristic algorithms untuk mengevaluasi performa lebih lanjut dibandingkan algoritma yang digunakan saat ini. 3. Pengguna dapat diberikan fitur tambahan pada aplikasi untuk mengunggah file CSV atau JSON, sehingga sistem dapat memproses data secara batch, tidak hanya melalui input manual.

REPORT #24668451

Results

Sources that matched your submitted document.

IDENTICAL CHANGED TEXT

INTERNET SOURCE
1. **0.51%** repository.unpad.ac.id
https://repository.unpad.ac.id/bitstreams/8e1a58a6-7782-4495-97da-aedf64017...

INTERNET SOURCE
2. **0.47%** repository.uinjkt.ac.id
https://repository.uinjkt.ac.id/dspace/bitstream/123456789/77017/1/AMMAR%2...

INTERNET SOURCE
3. **0.35%** repo.itera.ac.id
https://repo.itera.ac.id/assets/file_upload/SB2308040050/119140128_4_160228...

INTERNET SOURCE
4. **0.35%** eprints.upnyk.ac.id
http://eprints.upnyk.ac.id/36995/2/SKRIPSI%20FULL_PUTRA%20BAGASPATI%20..

INTERNET SOURCE
5. **0.31%** repository.uinjkt.ac.id
https://repository.uinjkt.ac.id/dspace/bitstream/123456789/66765/1/ALDI%20ZU..

INTERNET SOURCE
6. **0.27%** digilib.esaunggul.ac.id
https://digilib.esaunggul.ac.id/public/UEU-Undergraduate-19818-Bab1.Image.M...

INTERNET SOURCE
7. **0.21%** www.divasoft.net
https://www.divasoft.net/blog/10-aplikasi-terbaik-untuk-dipakai-belajar-coding...

INTERNET SOURCE
8. **0.21%** repository.uinjkt.ac.id
https://repository.uinjkt.ac.id/dspace/bitstream/123456789/65121/1/MUHAMMA...

INTERNET SOURCE
9. **0.2%** eprints.upj.ac.id
https://eprints.upj.ac.id/id/eprint/7763/20/8.%20BAB%20I.pdf

REPORT #24668451

INTERNET SOURCE
10. **0.2%** jurnal.umt.ac.id
https://jurnal.umt.ac.id/index.php/jt/article/viewFile/9099/4575

INTERNET SOURCE
11. **0.19%** biztech.proxsisgroup.com
https://biztech.proxsisgroup.com/devops-kunci-untuk-mempercepat-dan-meng…

INTERNET SOURCE
12. **0.17%** repository.upi.edu
http://repository.upi.edu/70476/1/S_PGSD_1702504_Chapter1.pdf

INTERNET SOURCE
13. **0.16%** repo.darmajaya.ac.id
http://repo.darmajaya.ac.id/15859/7/7%20BAB%203.pdf

INTERNET SOURCE
14. **0.16%** etheses.uinsgd.ac.id
https://etheses.uinsgd.ac.id/49462/4/4_bab1.pdf

INTERNET SOURCE
15. **0.16%** digilibadmin.unismuh.ac.id
https://digilibadmin.unismuh.ac.id/upload/42160-Full_Text.pdf

INTERNET SOURCE
16. **0.14%** www.trivusi.web.id
https://www.trivusi.web.id/2022/04/evaluasi-sistem-dengan-confusion-matrix.h…

INTERNET SOURCE
17. **0.14%** eprints.ums.ac.id
https://eprints.ums.ac.id/32237/2/BAB%20I.pdf

INTERNET SOURCE
18. **0.13%** digilib.unila.ac.id
http://digilib.unila.ac.id/14764/1/BAB1.pdf

INTERNET SOURCE
19. **0.13%** eprints.ums.ac.id
https://eprints.ums.ac.id/32514/2/BAB%20I.pdf

INTERNET SOURCE
20. **0.13%** www.exactitudeconsultancy.com
https://www.exactitudeconsultancy.com/id/laporan/37408/pasar-e-commerce/

REPORT #24668451

INTERNET SOURCE
21. **0.13%** eprints.ums.ac.id
https://eprints.ums.ac.id/36823/5/BAB%20I.pdf

INTERNET SOURCE
22. **0.13%** www.puskomedia.id
https://www.puskomedia.id/blog/mengapa-pengujian-pengguna-penting-dalam..

INTERNET SOURCE
23. **0.12%** www.codepolitan.com
https://www.codepolitan.com/blog/apa-itu-gitlab-keunggulan-dan-cara-kerjany..

INTERNET SOURCE
24. **0.12%** jurnal.amikom.ac.id
https://jurnal.amikom.ac.id/index.php/joism/article/download/1928/699/9956

INTERNET SOURCE
25. **0.12%** kc.umn.ac.id
https://kc.umn.ac.id/id/eprint/34481/4/BAB_III.pdf

INTERNET SOURCE
26. **0.12%** ojs.kalbis.ac.id
http://ojs.kalbis.ac.id/index.php/kalbisiana/article/download/429/342

INTERNET SOURCE
27. **0.12%** ukm.stiepas.ac.id
https://ukm.stiepas.ac.id/wp-content/uploads/2023/10/buku-Pengantar-Ecomm..

INTERNET SOURCE
28. **0.1%** bios.sinergis.org
https://bios.sinergis.org/bios/article/download/128/62/

INTERNET SOURCE
29. **0.1%** rumahcoding.co.id
https://rumahcoding.co.id/menggunakan-pandas-untuk-manipulasi-data-di-pyt...

INTERNET SOURCE
30. **0.1%** sis.binus.ac.id
https://sis.binus.ac.id/2024/10/31/what-is-confusion-matrix/

INTERNET SOURCE
31. **0.09%** repository.unsri.ac.id
http://repository.unsri.ac.id/165835/3/RAMA_56201_09011382025150_00030479...

REPORT #24668451

INTERNET SOURCE
32. 0.09% www.ptbsb.id
https://www.ptbsb.id/artikel/bahasa-pemrograman-python/

INTERNET SOURCE
33. 0.09% eprints.upj.ac.id
https://eprints.upj.ac.id/id/eprint/6067/10/BAB%20III.pdf

INTERNET SOURCE
34. 0.09% kc.umn.ac.id
https://kc.umn.ac.id/18095/5/BAB_II.pdf

INTERNET SOURCE
35. 0.08% esairina.medium.com
https://esairina.medium.com/memahami-confusion-matrix-accuracy-precision-...

INTERNET SOURCE
36. 0.08% pdfs.semanticscholar.org
https://pdfs.semanticscholar.org/31f3/f348b12fa6a88245cc227d3c653f28165a4b...

INTERNET SOURCE
37. 0.08% ejournal.indo-intellectual.id
https://ejournal.indo-intellectual.id/index.php/ifi/article/download/1387/813/10...

INTERNET SOURCE
38. 0.08% jurnal.uts.ac.id
https://jurnal.uts.ac.id/index.php/JINTEKS/article/download/4702/2203/15823

INTERNET SOURCE
39. 0.08% repository.uinjkt.ac.id
https://repository.uinjkt.ac.id/dspace/bitstream/123456789/55620/1/MACIA%20...

INTERNET SOURCE
40. 0.07% www.editage.com
https://www.editage.com/insights/is-jetir-issn-2349-5162-reliable-under-ugc-ca...

INTERNET SOURCE
41. 0.07% eprints.upj.ac.id
https://eprints.upj.ac.id/id/eprint/2319/4/8.%20BAB%20I.pdf

INTERNET SOURCE
42. 0.07% azuralabs.id
https://azuralabs.id/blog-programming/implementasi-docker-dalam-lingkungan...

REPORT #24668451

INTERNET SOURCE

43. **0.07%** repository.its.ac.id
https://repository.its.ac.id/88352/1/05111740000153-Undergraduate_Thesis.pdf

INTERNET SOURCE

44. **0.07%** jurnal.pknstan.ac.id
https://jurnal.pknstan.ac.id/index.php/JMKP/article/download/2922/1437/14853

INTERNET SOURCE

45. **0.06%** blog.myskill.id
https://blog.myskill.id/tips-karir/panduan-lengkap-menjadi-cloud-computing-e…

INTERNET SOURCE

46. **0.06%** eprints.poltekkesjogja.ac.id
http://eprints.poltekkesjogja.ac.id/11999/5/5.%20Chapter3.pdf.pdf

INTERNET SOURCE

47. **0.06%** repository.mediapenerbitindonesia.com
http://repository.mediapenerbitindonesia.com/312/1/%28%2BISBN%29K%2022..

INTERNET SOURCE

48. **0.05%** journal.universitaspahlawan.ac.id
https://journal.universitaspahlawan.ac.id/index.php/jpdk/article/download/100…

INTERNET SOURCE

49. **0.04%** www.anakciremai.com
https://www.anakciremai.com/2016/03/makalah-ilmu-sosial-tentang-wanita-tu…

QUOTES

INTERNET SOURCE

1. **0.13%** jurnal.uts.ac.id
https://jurnal.uts.ac.id/index.php/JINTEKS/article/download/4702/2203/15823

INTERNET SOURCE

2. **0.12%** bios.sinergis.org
https://bios.sinergis.org/bios/article/download/128/62/