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. Artikel ini mengidentifikasi tantangan	JETIR: Journal Of Emerging Technologies and Innovative Research. Volume 10, Issue 6. ISSN: 2349-5162

			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/10.1109/TCC.2020.3015769*
4	Balalaie, A., Heydarnoori, 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.2016.64
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)*

	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.

2.2.1 *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. 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.

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

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

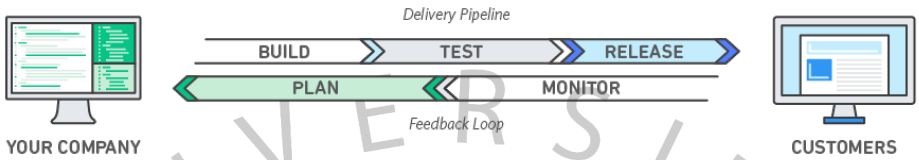
b. *BigQuery*

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

2.2.4 *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 CI*

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). GitLab mendukung berbagai bahasa pemrograman, seperti Python, Java, Ruby, dan Node.js. 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 .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.

2.2.6 *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. 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*.

2.2.7 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

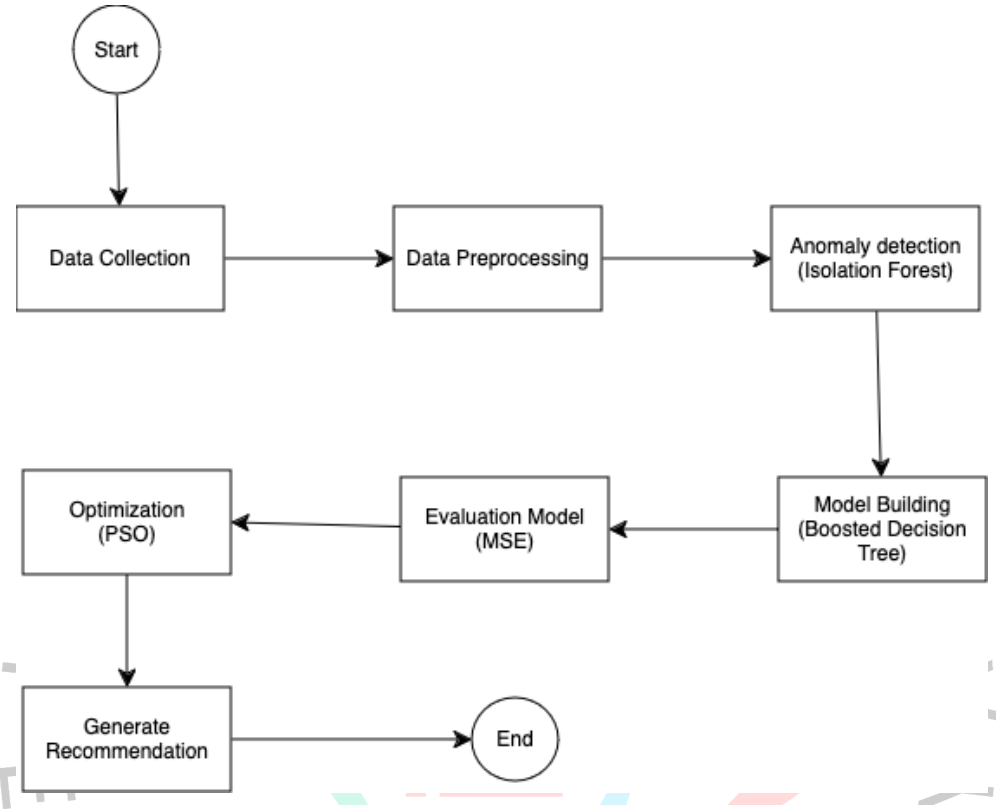
2.2.8 Application Programming Interface (API)

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

pola penggunaan CPU dan memori pada VM Cassandra guna menghasilkan rekomendasi terkait jenis *machine type* yang lebih efisien.



Gambar 2.8 Machine Learning Flow

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.

2.2.10 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.

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:

$$x' = \frac{\mathrm{x} - x_{min}}{\mathrm{x}_{max} - x_{min}}$$

di mana x' adalah nilai hasil normalisasi, x adalah nilai aktual, x_{max} adalah nilai maksimum, dan x_{min} adalah nilai minimum dari metrik tersebut.

Normalisasi ini 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.

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" titik 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:

$$\text{Score} = -\frac{E(h(x))}{c(n)}$$

   di mana:
   $E(h(x))$ adalah rata-rata kedalaman pohon untuk titik data x
   $c(n)$ adalah nilai normalisasi 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* yang 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 (Friedman, 2001). Prediksi akhir dari model *Boosted Decision Tree* dapat dinyatakan sebagai:

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

   di mana:
   $F(x)$ adalah prediksi akhir dari model,
   M adalah jumlah pohon,
   α_m adalah bobot dari pohon ke- m ,
   $h_m(x)$ 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).

2.2.13 Evaluasi Model

Evaluasi model dilakukan dengan menggunakan metrik *Mean Squared Error (MSE)*. MSE adalah metrik evaluasi yang paling umum digunakan dalam model regresi, karena mengukur rata-rata dari kuadrat perbedaan antara nilai prediksi 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 berikut:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

di mana:
y_i: nilai aktual
$\hat{y}_i$: nilai prediksi
n : jumlah data

MSE yang lebih rendah 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 permintaan 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 type* VM yang paling efisien.

PSO bekerja dengan konsep partikel yang mewakili calon solusi 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:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

di mana:
$v_i(t+1)$: Kecepatan partikel i pada iterasi $t+1$.
w: Bobot inersia, mengontrol kontribusi kecepatan sebelumnya.
c_1 dan c_2 : Koefisien untuk faktor kognitif dan sosial.
r_1 dan r_2 : Nilai acak antara 0 dan 1.
p_i : Posisi terbaik yang ditemukan oleh partikel i .
g : Posisi terbaik global yang ditemukan oleh 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

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 tulis 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 Requests* (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.

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.

UNIVERSITAS PEMBANGUNAN JAYA