Perancangan dan Implementasi Sistem Pemesanan Kantin Rumah Kayu ITERA Berbasis Cloud Run dan Cloud SQL pada Google Cloud Platform

TUGAS BESAR KOMPUTASI AWAN

Disusun Oleh:

ARYA PRATAMA	122140156
RUSTIAN AFENCIUS MARBUN	122140155
IKHSANNUDIN LATHIEF	122140137
IRMA AMELIA NOVIANTI	122140128
GIULIA PUSPO NEGORO	122140084
SILVA OKTARIA PUTRI	122140085

FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2025

BAB I

PENDAHULUAN

Kantin Rumah Kayu ITERA merupakan salah satu pusat kegiatan mahasiswa dan sivitas akademika yang setiap harinya melayani banyak pengunjung. Pada jam-jam sibuk, terutama saat jam istirahat, sering terjadi antrean panjang di depan tenant karena seluruh proses pemesanan masih dilakukan secara manual di kasir. Kondisi ini tidak hanya membuang waktu mahasiswa dan karyawan, tetapi juga berpotensi menimbulkan salah catat pesanan, pesanan tertukar, maupun keterlambatan penyajian karena informasi pesanan tidak tercatat dengan rapi.

Di sisi lain, pengelolaan kantin juga menjadi tantangan tersendiri bagi pemilik tenant. Pencatatan menu, perubahan harga, ketersediaan stok, serta riwayat pesanan biasanya masih dilakukan secara terpisah dengan media sederhana seperti kertas atau spreadsheet. Hal ini menyulitkan tenant dalam memantau pesanan secara real-time, mengevaluasi penjualan, maupun mengendalikan stok agar tidak berlebih atau justru habis saat permintaan tinggi. Admin pengelola kantin pun kesulitan mengawasi seluruh tenant secara terpusat karena tidak ada sistem yang menyatukan data.

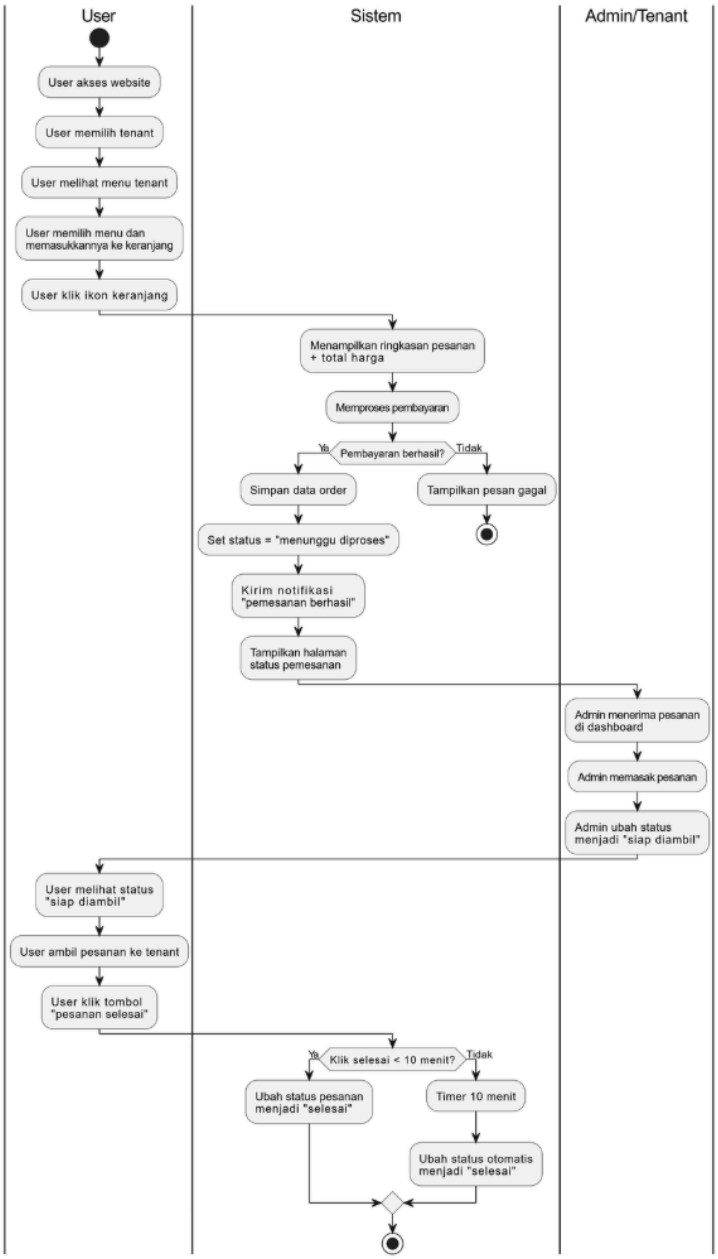
Berdasarkan permasalahan tersebut, dibutuhkan sebuah sistem pemesanan makanan berbasis web yang memungkinkan pengguna memilih tenant, melihat menu, memesan, dan melakukan pembayaran tanpa harus mengantre langsung di kantin. Sistem ini juga diharapkan dapat membantu pemilik tenant mengelola menu, pesanan, dan stok secara terstruktur, sekaligus memberikan alat bagi admin untuk mengelola dan memantau akun tenant secara terpusat. Dengan adanya digitalisasi proses pemesanan melalui aplikasi Kantin Rumah Kayu ITERA, diharapkan pengalaman makan di kantin menjadi lebih efisien, tertib, dan nyaman bagi seluruh pihak yang terlibat.

Laporan ini mendokumentasikan proses implementasi, konfigurasi, dan pengamanan infrastruktur untuk aplikasi "Pemesanan di Kantin Rumah Kayu ITERA". Proyek ini mengadopsi arsitektur layanan mikro atau microservices yang memisahkan sisi antarmuka pengguna (Frontend) dan logika bisnis (Backend). Frontend dibangun menggunakan kerangka kerja Vue.js, sedangkan Backend dikembangkan menggunakan Laravel. Seluruh infrastruktur dideploy di atas platform Google Cloud (GCP) dengan memanfaatkan layanan Cloud Run untuk komputasi tanpa server, Cloud SQL untuk manajemen basis data relasional, dan berbagai layanan pendukung keamanan serta jaringan lainnya. Tujuan utama dari konfigurasi ini adalah menciptakan lingkungan produksi yang aman, dapat diandalkan, dan terotomatisasi sesuai dengan standar industri.

BAB II

PERANCANGAN SISTEM

2.1 Gambaran Umum Aplikasi Kantin RK Itera



Pada gambar diatas adalah gambaran umum sistem pemesanan di Kantin RK ITERA. Pengguna terlebih dahulu mengakses website, memilih tenant yang diinginkan, lalu melihat daftar menu dari tenant tersebut. Menu yang ingin dibeli dimasukkan ke keranjang, kemudian

pengguna menekan ikon keranjang untuk melanjutkan. Sistem menampilkan ringkasan pesanan beserta total harga, lalu memproses pembayaran. Jika pembayaran berhasil, sistem menyimpan data order, mengubah status pesanan menjadi “menunggu diproses”, dan menampilkan halaman status pesanan sekaligus mengirim notifikasi bahwa pemesanan berhasil.

Di sisi lain, admin atau pemilik tenant akan menerima daftar pesanan baru di dashboard. Tenant kemudian menyiapkan/memasak pesanan tersebut dan, setelah siap, mengubah status pesanan menjadi “siap diambil”. Pengguna dapat memeriksa halaman status pesanan, melihat bahwa pesanannya sudah siap, lalu datang ke tenant untuk mengambil makanan. Setelah pesanan diambil, pengguna menekan tombol “pesanan selesai” di aplikasi. Jika pengguna tidak menekan tombol ini dalam waktu tertentu, sistem akan otomatis mengubah status pesanan menjadi “selesai”. Dengan alur ini, seluruh proses pemesanan, pengolahan, hingga konfirmasi selesai tercatat dengan jelas dan terstruktur bagi user maupun tenant.

2.2 Fitur Utama

2.2.1 Fitur Untuk User

Bagi user, sistem menyediakan fitur pemesanan makanan secara online. Melalui halaman tenant dan menu, pengguna dapat memilih makanan atau minuman yang diinginkan tanpa harus datang langsung dan mengantre di kantin. Proses pemesanan dibuat sederhana: user cukup memilih menu, menentukan jumlah, kemudian melakukan konfirmasi pesanan. Selain itu, sistem menyediakan fitur keranjang pemesanan. Keranjang ini memungkinkan pengguna mengelola banyak item sekaligus sebelum melakukan checkout. Dengan adanya keranjang, pengguna dapat menambah, mengurangi, atau menghapus menu secara fleksibel sehingga pesanan lebih rapi dan risiko kesalahan dapat dikurangi.

2.2.2 Fitur Untuk Pemilik Tenant

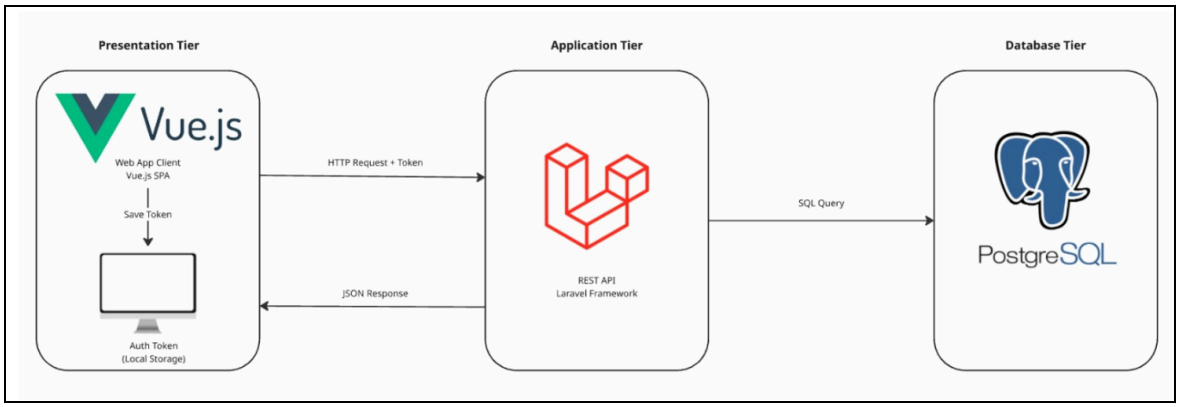
Untuk pemilik tenant, sistem menyediakan fitur manajemen pesanan yang terpusat. Tenant dapat melihat daftar pesanan yang masuk secara real-time, memantau status setiap pesanan, dan memperbarui status ketika pesanan sedang diproses atau sudah siap diambil. Hal ini membantu tenant menangani antrian pesanan dengan lebih terstruktur dibandingkan pemesanan manual. Sistem juga menyediakan fitur manajemen menu, yang memungkinkan tenant menambah, mengubah, atau menghapus menu dan harga secara mandiri. Perubahan tersebut langsung tercermin pada aplikasi pengguna sehingga informasi menu selalu terbaru. Selain itu, terdapat fitur pengaturan stok (set stock) yang membantu tenant mengontrol ketersediaan porsi yang dijual setiap hari. Dengan fitur ini, tenant dapat menandai menu yang habis sehingga aplikasi

tidak lagi menerima pesanan untuk menu tersebut, sehingga over-order dapat dihindari.

2.2.3 Fitur Untuk Admin

Pada sisi admin, sistem menyediakan fitur manajemen pemilik tenant. Admin dapat membuat, mengubah, dan menonaktifkan akun tenant sesuai kebutuhan operasional kantin. Dengan demikian, struktur pengelolaan tenant menjadi lebih tertib karena setiap tenant memiliki akun resmi yang tercatat di sistem. Fitur ini juga memungkinkan admin mengawasi aktivitas tenant secara terpusat, baik dari sisi jumlah pesanan maupun status operasional. Admin dapat memastikan bahwa hanya tenant yang terdaftar dan aktif yang dapat menerima pesanan melalui sistem. Dengan adanya pengelolaan tenant yang jelas, tanggung jawab setiap tenant terhadap pesanan dan layanan kepada pengguna menjadi lebih transparan.

2.3 Arsitektur Diagram



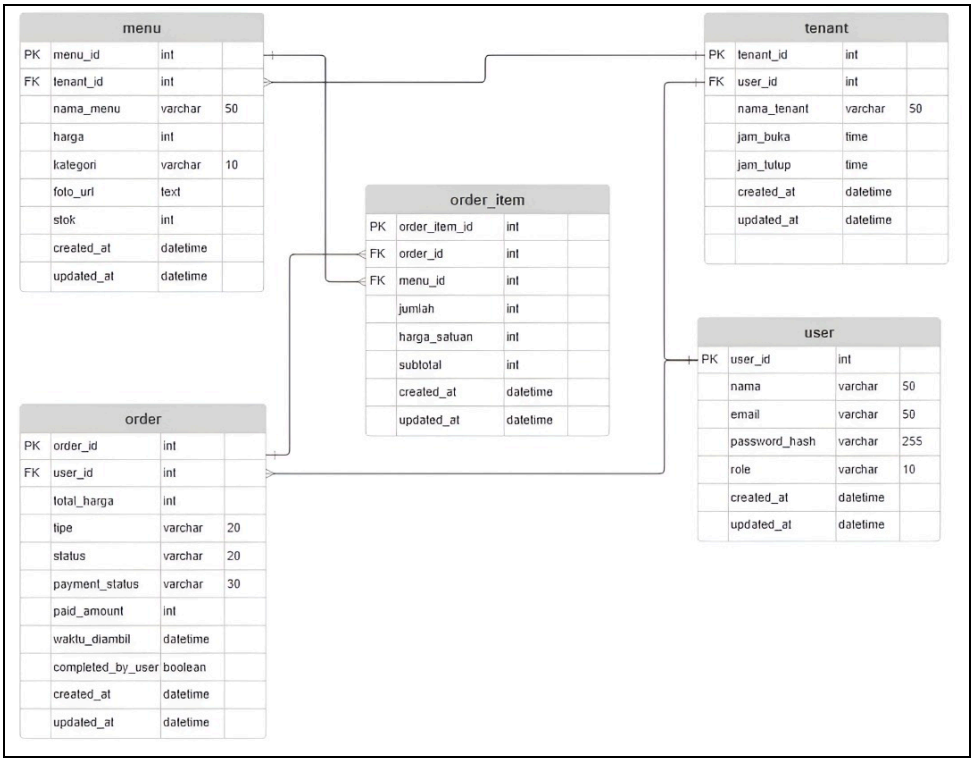
Arsitektur sistem pada gambar ini menerapkan konsep three-tier architecture yang membagi sistem ke dalam tiga lapisan utama, yaitu Presentation Tier, Application Tier, dan Database Tier. Pendekatan ini digunakan untuk memisahkan tanggung jawab antar komponen sistem sehingga proses pengembangan, pengelolaan, dan pemeliharaan aplikasi dapat dilakukan secara lebih terstruktur, aman, dan efisien.

Pada Presentation Tier, aplikasi frontend dibangun menggunakan Vue.js dalam bentuk Single Page Application. Lapisan ini berfungsi sebagai antarmuka utama bagi pengguna untuk melakukan interaksi dengan sistem, seperti melihat menu, melakukan pemesanan, dan mengirimkan permintaan ke server. Proses autentikasi dilakukan dengan mekanisme token, di mana token yang diterima setelah proses login disimpan pada local storage dan kemudian disertakan dalam setiap HTTP request yang dikirimkan ke backend sebagai bentuk validasi akses.

Lapisan Application Tier diimplementasikan menggunakan REST API berbasis framework Laravel yang berperan sebagai pusat pengolahan logika bisnis aplikasi. Backend menerima permintaan dari frontend, melakukan validasi token dan data, memproses aturan bisnis yang berlaku, serta mengelola alur transaksi sebelum mengakses basis data. Hasil pemrosesan kemudian dikirimkan kembali ke frontend dalam bentuk respons JSON, sehingga komunikasi antar lapisan dapat berjalan secara terstandar dan terpisah dengan jelas.

Pada Database Tier, sistem menggunakan PostgreSQL sebagai basis data relasional untuk menyimpan seluruh data aplikasi, termasuk data pengguna, menu, dan transaksi pemesanan. Akses ke basis data dibatasi hanya melalui lapisan aplikasi, sehingga frontend tidak memiliki akses langsung terhadap data. Pembatasan ini bertujuan untuk menjaga keamanan, integritas, serta konsistensi data yang tersimpan di dalam sistem.

2.4 Perancangan Basis Data



Skema basis data pada sistem pemesanan makanan kantin dirancang untuk mendukung seluruh proses bisnis secara terintegrasi, mulai dari pengelolaan data tenant, penyediaan menu, pengelolaan pengguna, hingga pencatatan transaksi pemesanan. Perancangan basis data menggunakan model relasional dengan menerapkan prinsip normalisasi, sehingga setiap tabel memiliki fungsi yang jelas dan tidak terjadi duplikasi data. Hubungan antar tabel dibangun menggunakan kunci primer dan kunci asing untuk menjaga konsistensi serta integritas data.

Tabel tenant digunakan untuk menyimpan data penjual atau unit kantin yang terdaftar dalam sistem. Informasi yang disimpan meliputi identitas tenant, nama tenant, serta jam operasional berupa jam buka dan jam tutup. Tabel ini menjadi dasar dalam mendukung konsep multi-tenant, di mana satu sistem dapat digunakan oleh beberapa penjual. Relasi antara tabel tenant dan tabel menu bersifat satu ke banyak, yang berarti satu tenant dapat memiliki lebih dari satu menu yang ditawarkan.

Tabel menu berfungsi untuk menyimpan data menu makanan dan minuman yang disediakan oleh masing-masing tenant. Data yang dicatat meliputi nama menu, harga, kategori, stok, serta URL gambar menu. Setiap menu dihubungkan dengan satu tenant melalui atribut tenant_id sebagai kunci asing. Dengan struktur ini, sistem dapat menampilkan menu berdasarkan tenant tertentu serta mendukung pengelolaan data menu secara terpisah antar penjual.

Tabel user digunakan untuk menyimpan data pengguna aplikasi. Informasi yang tersimpan mencakup nama pengguna, alamat email, kata sandi yang telah dienkripsi, serta peran pengguna dalam sistem. Tabel ini berperan dalam proses autentikasi dan otorisasi pengguna, serta menjadi referensi utama dalam setiap transaksi pemesanan yang dilakukan oleh pengguna.

Tabel order digunakan untuk mencatat data transaksi pemesanan secara keseluruhan. Informasi yang disimpan meliputi total harga pesanan, tipe pemesanan, status pesanan, status pembayaran, jumlah pembayaran, serta waktu pengambilan pesanan. Selain itu, tabel ini juga mencatat status penyelesaian pesanan oleh pengguna. Tabel order memiliki relasi satu ke banyak dengan tabel order_item, yang memungkinkan satu transaksi terdiri dari beberapa item menu.

Tabel order_item berfungsi untuk menyimpan rincian item dalam setiap transaksi pemesanan. Data yang dicatat meliputi menu yang dipesan, jumlah item, harga satuan, dan subtotal. Tabel ini berperan sebagai penghubung antara tabel order dan tabel menu, sehingga hubungan banyak ke banyak antara pesanan dan menu dapat dikelola dengan baik. Informasi pada tabel ini digunakan sebagai dasar dalam perhitungan total harga pesanan.

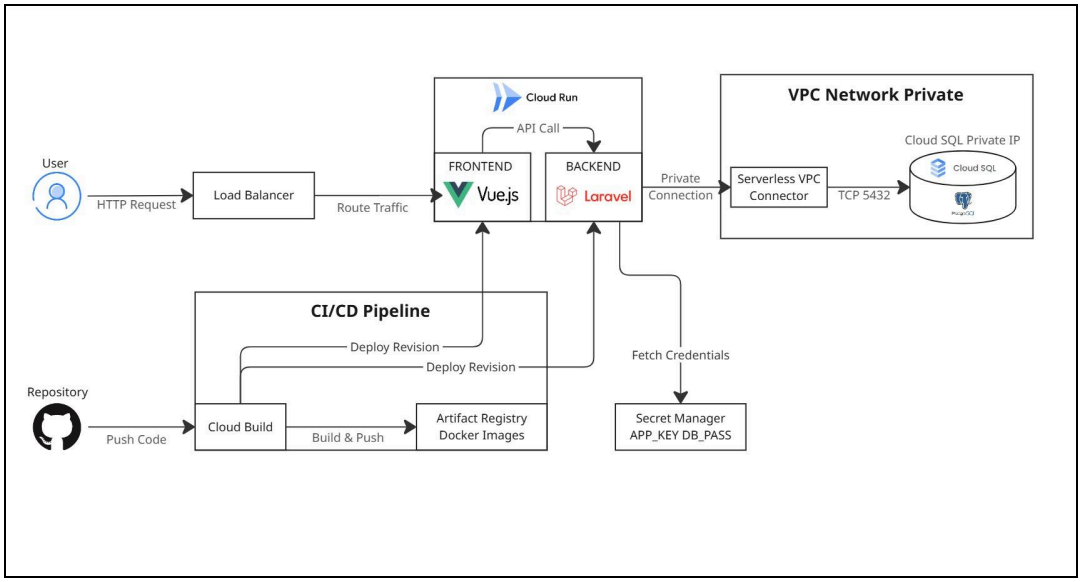
Skema basis data yang dirancang telah sesuai dengan kebutuhan sistem pemesanan makanan kantin. Setiap tabel saling terhubung secara logis dan mendukung alur proses bisnis yang jelas. Dengan penerapan struktur relasional dan prinsip normalisasi, skema basis data ini mampu menjaga konsistensi data, meminimalkan redundansi, serta mendukung pengelolaan sistem secara efektif dan berkelanjutan.

BAB III

IMPLEMENTASI DI GOOGLE CLOUD

3.1 Arsitektur Sistem

Secara garis besar, topologi infrastruktur yang dibangun menghubungkan pengguna akhir dengan aplikasi melalui protokol HTTPS yang aman. Pengguna mengakses aplikasi melalui domain kustom yang dilayani oleh Global External Application Load Balancer. Load balancer ini bertugas mendistribusikan lalu lintas ke layanan Cloud Run Frontend. Frontend kemudian berkomunikasi dengan Cloud Run Backend melalui REST API yang diamankan. Backend adalah satu-satunya komponen yang diizinkan untuk mengakses basis data Cloud SQL melalui jaringan privat Virtual Private Cloud (VPC). Seluruh kredensial sensitif disimpan secara terpusat di Secret Manager, dan proses pembaruan aplikasi ditangani secara otomatis oleh Cloud Build.



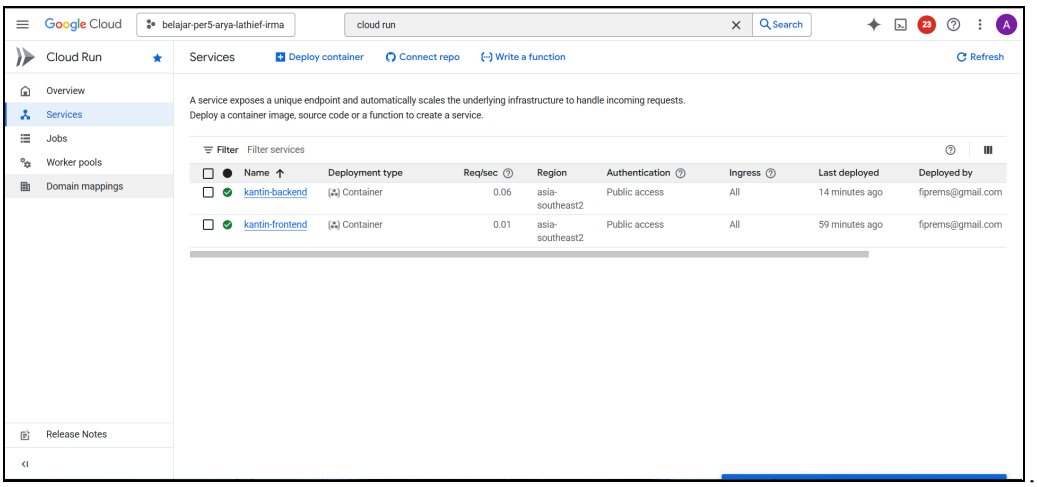
3.1 Konfigurasi Layanan Komputasi (Cloud Run)

Implementasi komputasi dilakukan menggunakan Google Cloud Run karena sifatnya yang serverless, yang memungkinkan aplikasi untuk melakukan scaling otomatis dari nol hingga ribuan instance berdasarkan permintaan lalu lintas.

3.1.1 Deployment Frontend dan Backend

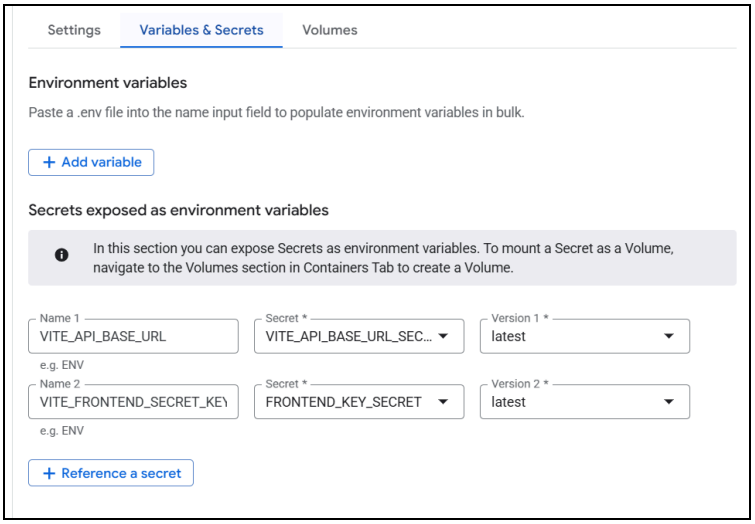
Terdapat dua layanan utama yang berjalan di Cloud Run, yaitu layanan Frontend untuk antarmuka pengguna dan layanan Backend untuk API. Pemisahan ini dilakukan agar

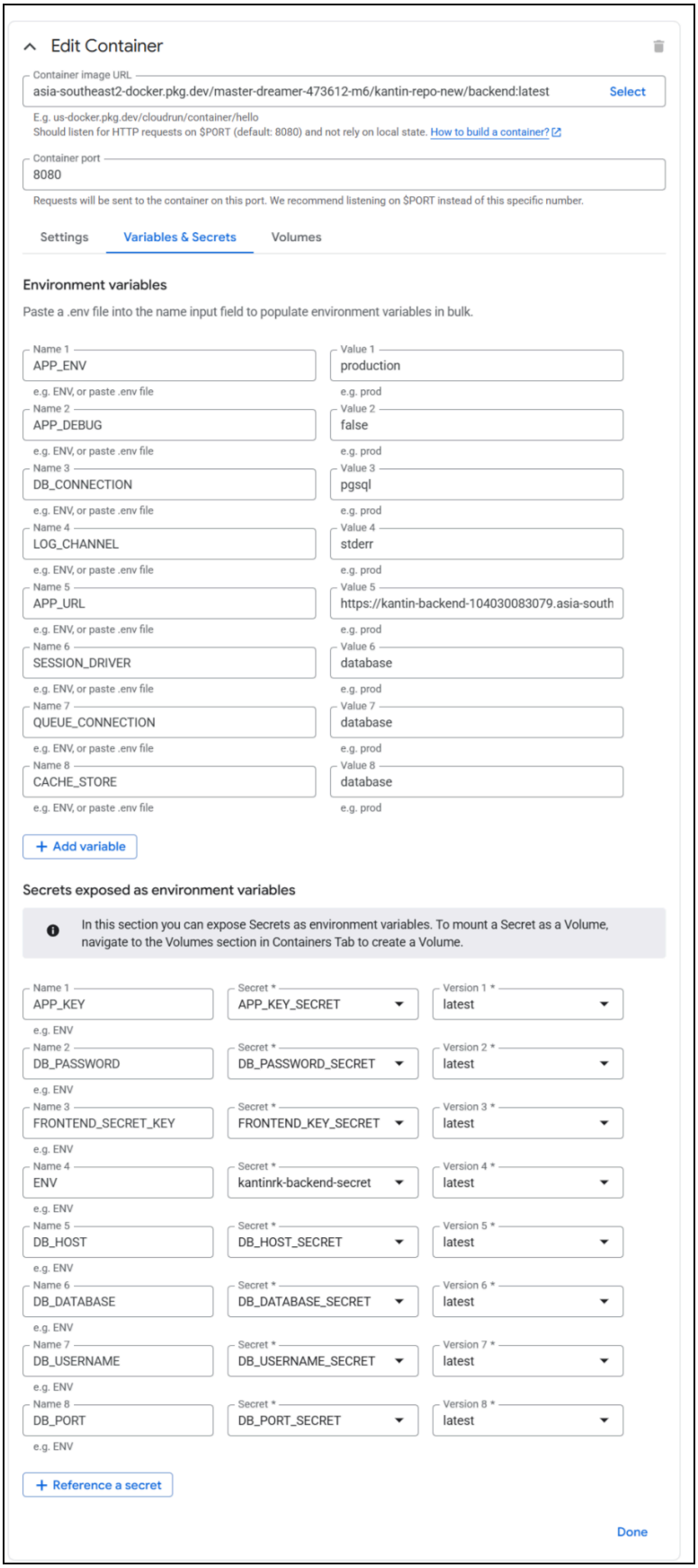
pengembangan dan skalabilitas masing-masing komponen dapat dikelola secara independen. Layanan Backend dikonfigurasi menggunakan container Docker yang menjalankan PHP dan Nginx, sementara Frontend menyajikan aset statis hasil build Vue.js



3.1.2 Manajemen Variabel Lingkungan

Konfigurasi aplikasi tidak lagi disimpan dalam kode sumber (hardcoded), melainkan disuntikkan melalui Environment Variables pada level revisi Cloud Run. Hal ini mencakup konfigurasi koneksi database, mode debug, dan URL API. Pendekatan ini memungkinkan perubahan konfigurasi tanpa perlu mengubah kode program dan melakukan build ulang pada container image.





Edit Container

Container image URL
asia-southeast2-docker.pkg.dev/master-dreamer-473612-m6/kantin-repo-new/backend:latest Select

E.g. us-docker.pkg.dev/cloudrun/container/hello
Should listen for HTTP requests on $PORT (default: 8080) and not rely on local state. How to build a container?

Container port
8080

Requests will be sent to the container on this port. We recommend listening on $PORT instead of this specific number.

Settings | Variables & Secrets | Volumes

Environment variables

Paste a .env file into the name input field to populate environment variables in bulk.

Name	Value
Name 1: APP_ENV	Value 1: production
Name 2: APP_DEBUG	Value 2: false
Name 3: DB_CONNECTION	Value 3: pgsql
Name 4: LOG_CHANNEL	Value 4: stderr
Name 5: APP_URL	Value 5: https://kantin-backend-104030083079.asia-south
Name 6: SESSION_DRIVER	Value 6: database
Name 7: QUEUE_CONNECTION	Value 7: database
Name 8: CACHE_STORE	Value 8: database

e.g. ENV, or paste .env file
e.g. prod

+ Add variable

Secrets exposed as environment variables

In this section you can expose Secrets as environment variables. To mount a Secret as a Volume, navigate to the Volumes section in Containers Tab to create a Volume.

Name	Secret *	Version *
Name 1: APP_KEY	Secret *: APP_KEY_SECRET	Version 1 *: latest
Name 2: DB_PASSWORD	Secret *: DB_PASSWORD_SECRET	Version 2 *: latest
Name 3: FRONTEND_SECRET_KEY	Secret *: FRONTEND_KEY_SECRET	Version 3 *: latest
Name 4: ENV	Secret *: kantinrk-backend-secret	Version 4 *: latest
Name 5: DB_HOST	Secret *: DB_HOST_SECRET	Version 5 *: latest
Name 6: DB_DATABASE	Secret *: DB_DATABASE_SECRET	Version 6 *: latest
Name 7: DB_USERNAME	Secret *: DB_USERNAME_SECRET	Version 7 *: latest
Name 8: DB_PORT	Secret *: DB_PORT_SECRET	Version 8 *: latest

e.g. ENV

+ Reference a secret

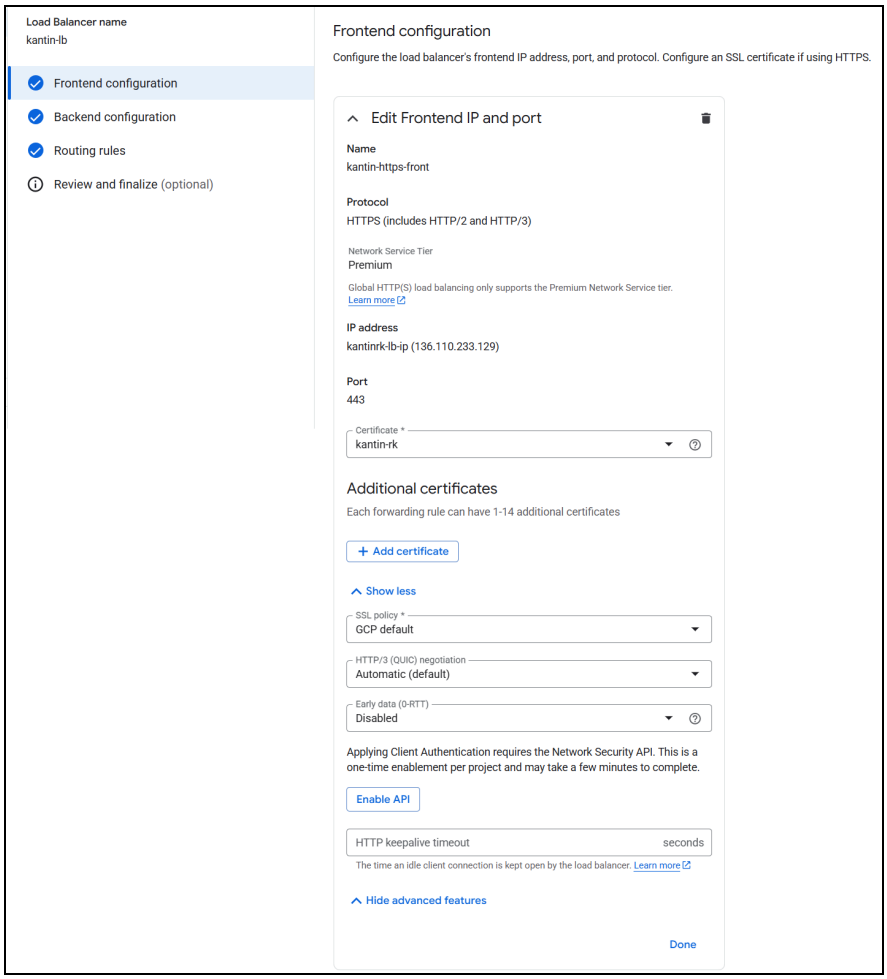
Done

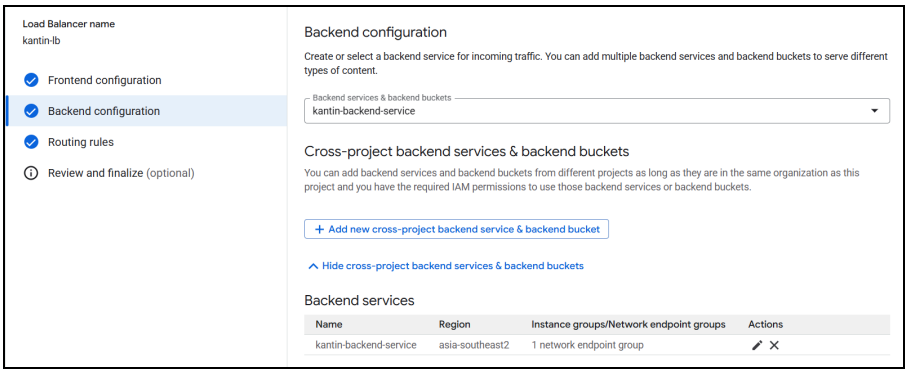
3.2 Manajemen Jaringan dan Domain

Untuk memastikan aplikasi dapat diakses dengan mudah dan aman oleh pengguna, diterapkan konfigurasi jaringan tingkat lanjut menggunakan Load Balancer dan sertifikat SSL terkelola.

3.2.1 Application Load Balancer

Google Cloud Global External Application Load Balancer digunakan sebagai pintu gerbang utama lalu lintas internet. Load Balancer ini dikonfigurasi untuk meneruskan permintaan HTTP dan HTTPS ke layanan Cloud Run melalui Serverless Network Endpoint Group (NEG). Penggunaan Load Balancer ini memungkinkan penggunaan alamat IP statis global anycast yang mempercepat akses pengguna dari berbagai lokasi geografis.





3.2.2 Pemetaan Domain dan SSL

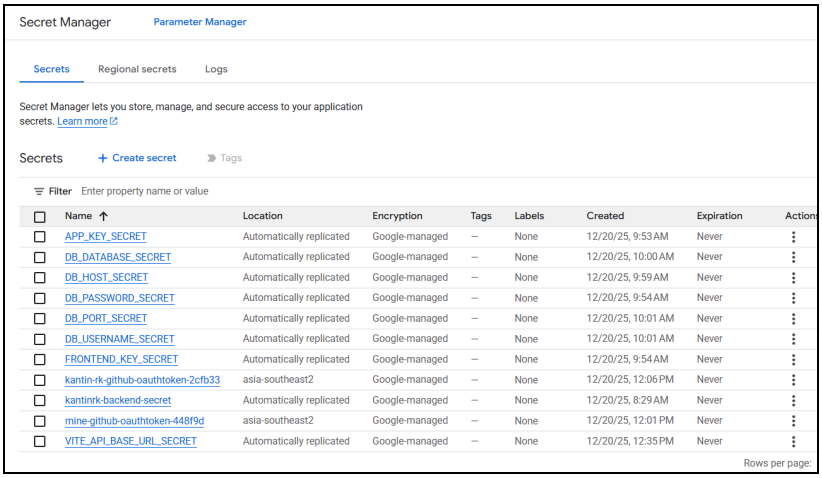
Aplikasi dapat diakses melalui domain kustom kantin-rk.my.id. Sertifikat SSL/TLS disediakan dan dikelola secara otomatis oleh Google (Google-managed Certificate). Hal ini menjamin bahwa seluruh komunikasi antara browser pengguna dan server terenkripsi dengan standar keamanan modern. Status sertifikat yang aktif menandakan bahwa domain telah terverifikasi dan aman untuk digunakan.

3.3 Implementasi Keamanan

Keamanan merupakan aspek paling krusial dalam proyek ini. Penerapan keamanan dilakukan secara berlapis (defense in depth) mulai dari level aplikasi, manajemen identitas, hingga penyimpanan data rahasia.

3.3.1 Manajemen Rahasia dengan Secret Manager

Guna menghindari kebocoran data sensitif, seluruh kredensial penting seperti kata sandi basis data (DB_PASSWORD), kunci aplikasi Laravel (APP_KEY), dan kunci rahasia komunikasi antar layanan (FRONTEND_SECRET_KEY) disimpan di Google Secret Manager. Cloud Run dikonfigurasi untuk mereferensikan nilai-nilai ini pada saat runtime, sehingga tidak ada teks rahasia yang terlihat pada dashboard konfigurasi biasa.



3.3.2 Keamanan API dan Middleware

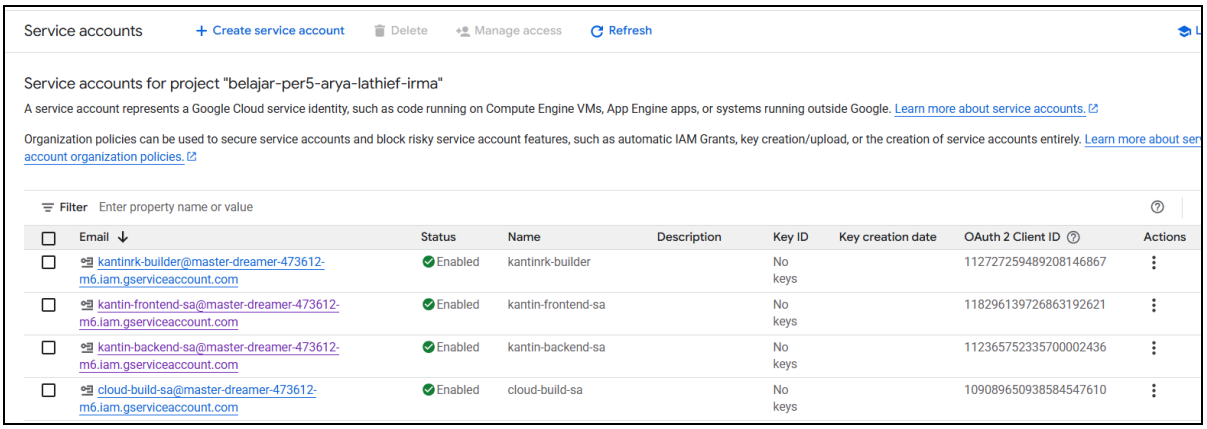
Komunikasi antara Frontend dan Backend dilindungi menggunakan mekanisme kunci rahasia kustom. Middleware EnsureFrontendRequest telah diimplementasikan pada Backend Laravel untuk memvalidasi keberadaan header X-Kantin-Key pada setiap permintaan masuk. Jika permintaan tidak menyertakan kunci yang valid, server akan menolak akses dengan kode status 403 Forbidden. Hal ini mencegah akses API langsung melalui browser atau pihak ketiga yang tidak berwenang. Selain itu, konfigurasi CORS (Cross-Origin Resource Sharing) pada Laravel telah diperketat untuk hanya menerima permintaan dari domain Frontend yang sah.

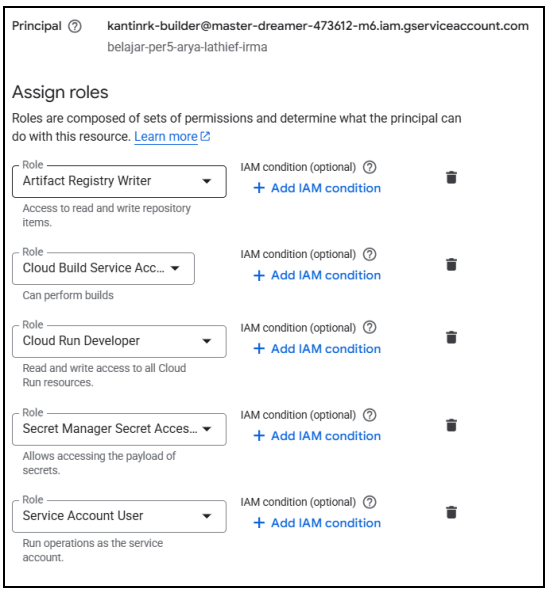
backend > config > cors.php

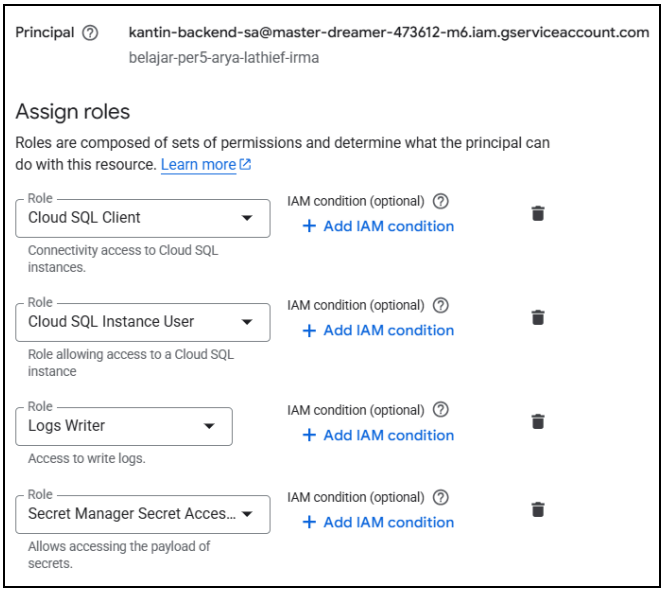
```php
<?php

return [
    'paths' => ['api/*', 'sanctum/csrf-cookie'],

    'allowed_methods' => ['*'],

    'allowed_origins' => [
        'https://kantin-rk.my.id',
        'https://www.kantin-rk.my.id',
    ],

    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => true,
];
```

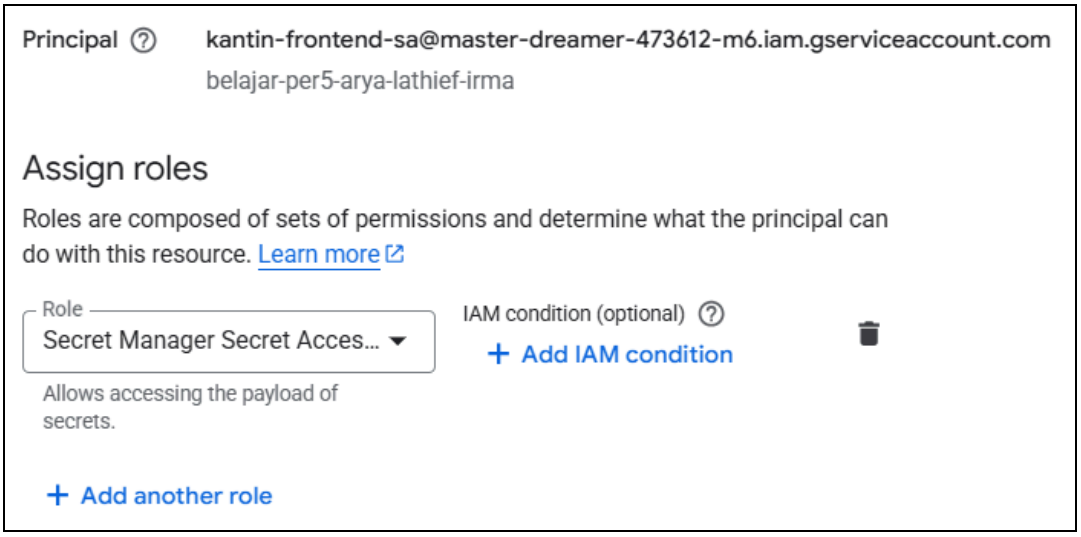
3.3.3 Manajemen Identitas dan Akses (IAM)

Prinsip hak akses minimal atau Least Privilege diterapkan pada Service Accounts. Layanan Backend berjalan menggunakan Service Account khusus yang hanya memiliki peran spesifik, yaitu Cloud SQL Client untuk koneksi database, Secret Manager Secret Accessor untuk membaca kredensial, dan Logs Writer untuk pencatatan log. Pembatasan ini meminimalisir dampak kerusakan apabila terjadi insiden keamanan pada salah satu layanan.







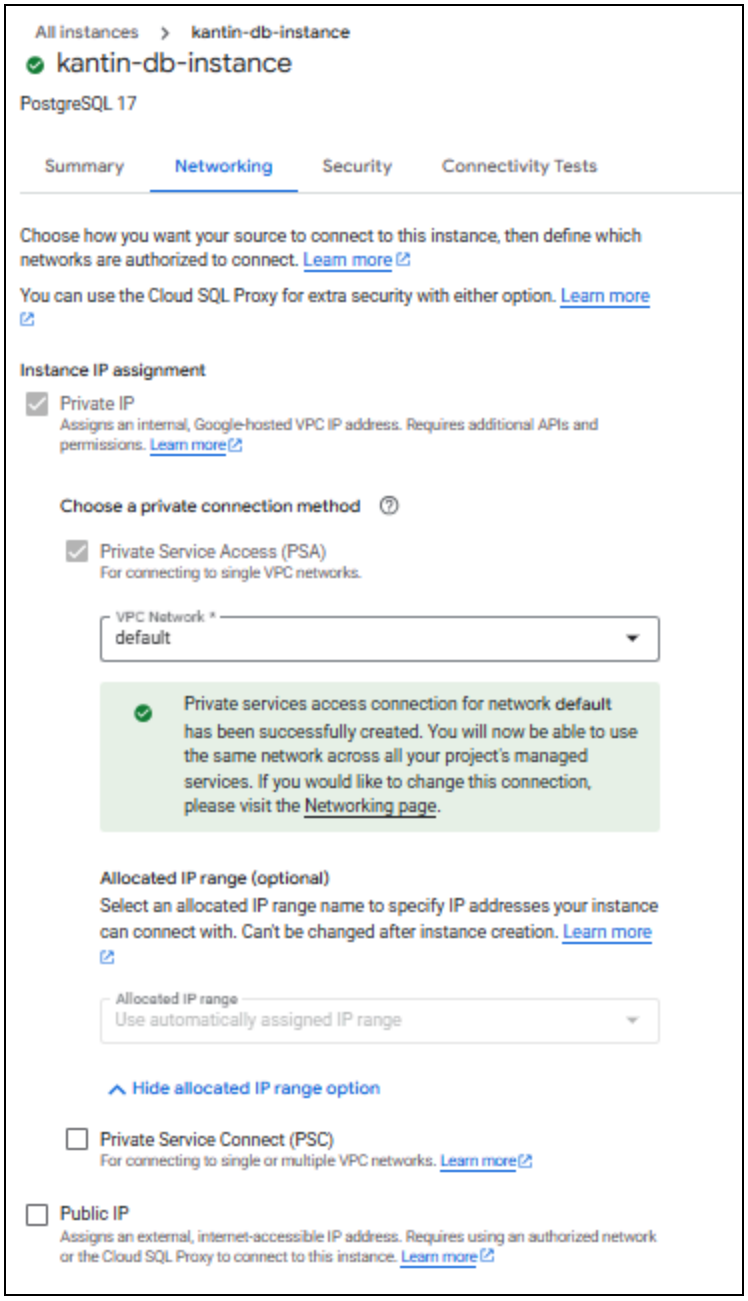


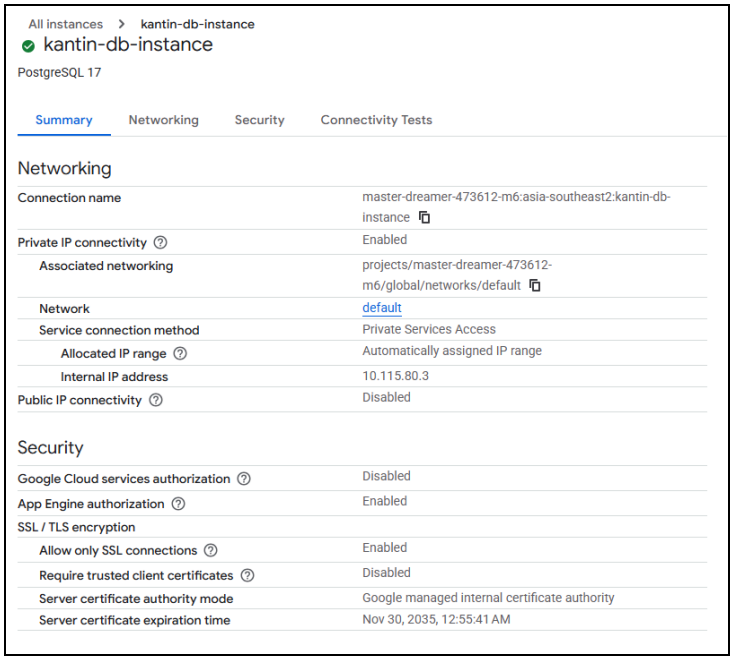
3.4 Konfigurasi Basis Data dan Jaringan Privat

Basis data relasional dikelola menggunakan Google Cloud SQL. Untuk meningkatkan keamanan data, akses publik (Public IP) ke basis data telah dinonaktifkan sepenuhnya.

3.4.1 Virtual Private Cloud (VPC) Peering

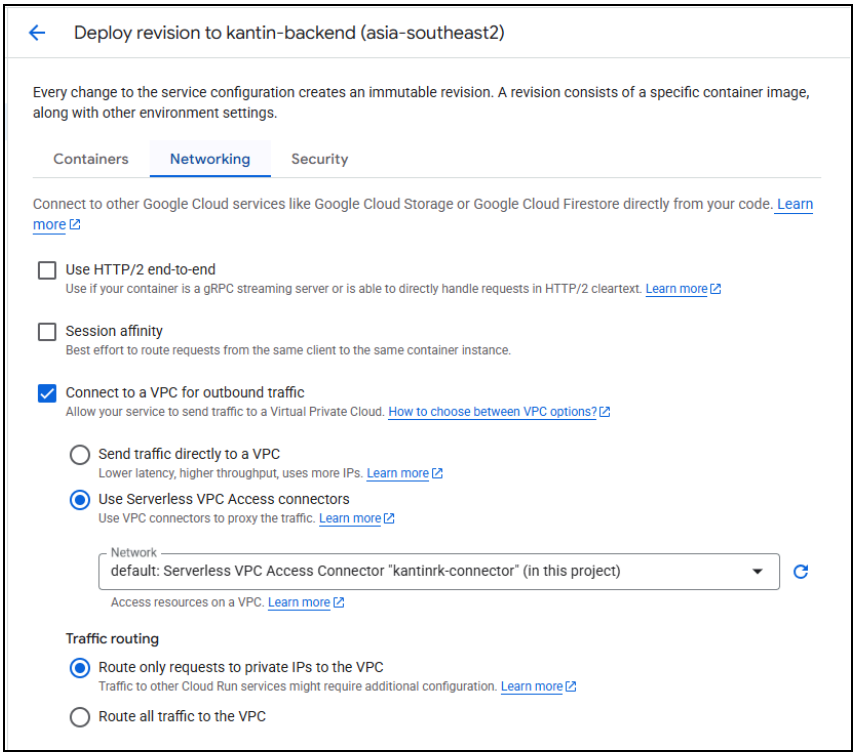
Layanan Cloud Run Backend terhubung ke Cloud SQL melalui jaringan privat VPC menggunakan Serverless VPC Access Connector. Mekanisme ini memastikan bahwa lalu lintas data antara aplikasi dan basis data tidak pernah keluar ke internet publik, melainkan tetap berada di dalam jaringan internal Google yang terisolasi.


All instances > kantin-db-instance
kantin-db-instance
PostgreSQL 17
Summary
Networking
Security
Connectivity Tests
Choose how you want your source to connect to this instance, then define which networks are authorized to connect. Learn more
You can use the Cloud SQL Proxy for extra security with either option. Learn more
Instance IP assignment
Private IP
Assigns an internal, Google-hosted VPC IP address. Requires additional APIs and permissions. Learn more
Choose a private connection method
Private Service Access (PSA)
For connecting to single VPC networks.
VPC Network *
default
Private services access connection for network default has been successfully created. You will now be able to use the same network across all your project's managed services. If you would like to change this connection, please visit the Networking page.
Allocated IP range (optional)
Select an allocated IP range name to specify IP addresses your instance can connect with. Can't be changed after instance creation. Learn more
Allocated IP range
Use automatically assigned IP range
Hide allocated IP range option
Private Service Connect (PSC)
For connecting to single or multiple VPC networks. Learn more
Public IP
Assigns an external, internet-accessible IP address. Requires using an authorized network or the Cloud SQL Proxy to connect to this instance. Learn more



3.4.2 Koneksi Database Privat

Backend Laravel dikonfigurasi untuk menghubungi basis data menggunakan alamat IP internal (Private IP). Hal ini membuktikan keberhasilan implementasi VPC Peering dan menjamin integritas serta kerahasiaan data transaksi yang tersimpan.

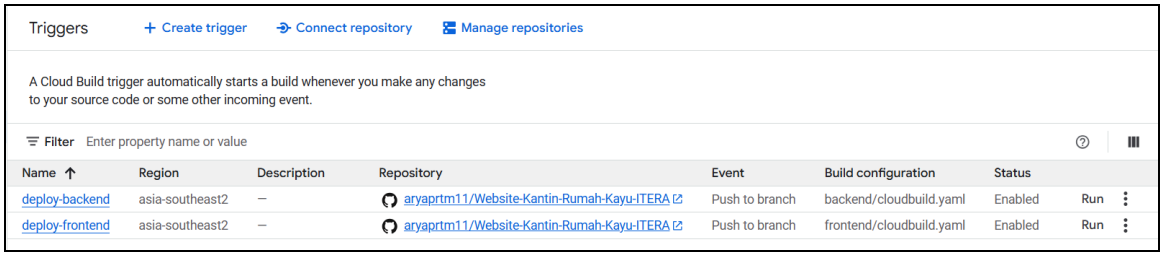


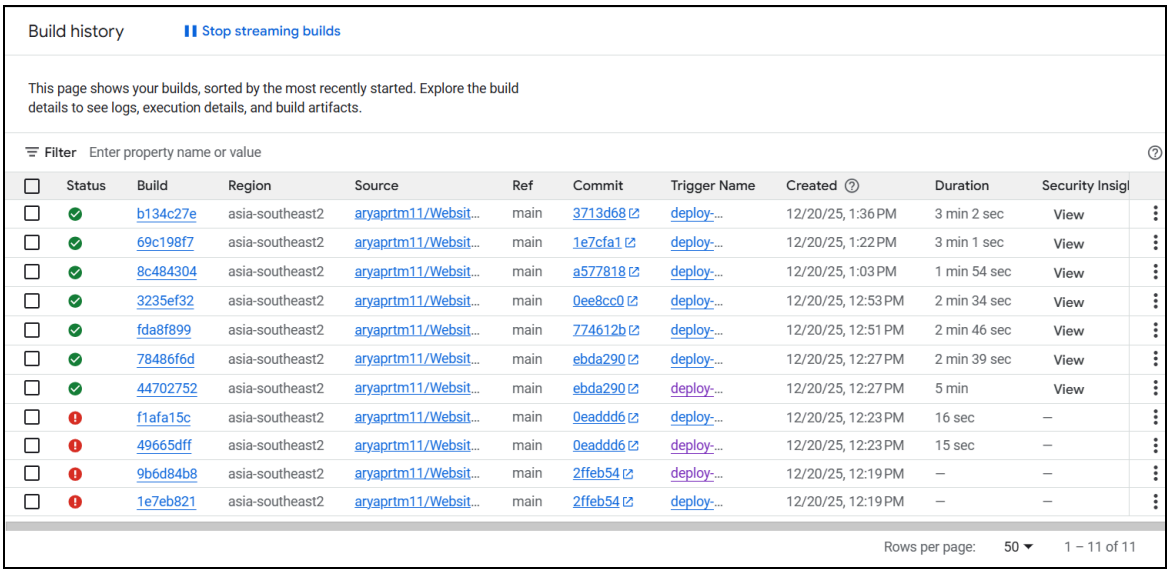
3.5 Otomatisasi Deployment (CI/CD)

Proses deployment kode dari repositori ke lingkungan produksi dilakukan secara otomatis menggunakan pipelien Continuous Integration dan Continuous Deployment (CI/CD) dengan Google Cloud Build.

3.5.1 Pipeline Cloud Build

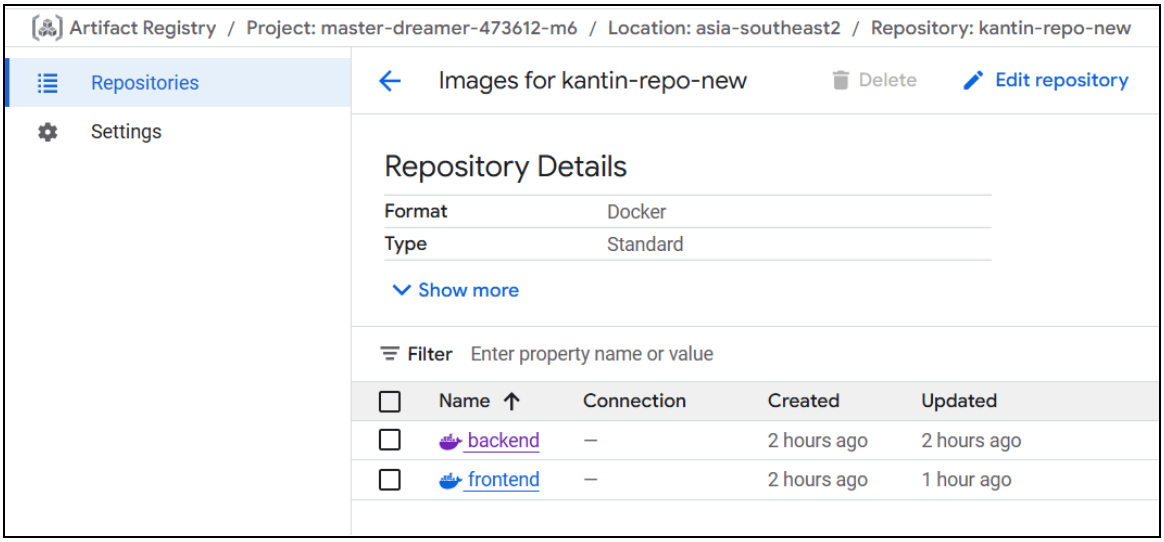
Dua pemicu (trigger) Cloud Build telah dibuat secara terpisah untuk Frontend dan Backend. Setiap kali terjadi perubahan kode (push) pada cabang utama repositori Git, Cloud Build akan secara otomatis menjalankan serangkaian tugas. Tugas tersebut meliputi pembuatan file konfigurasi lingkungan produksi, pembangunan Docker image, penyimpanan image ke Artifact Registry, dan deployment revisi baru ke Cloud Run.





3.5.2 Artifact Registry

Seluruh container image yang dihasilkan oleh proses build disimpan secara terpusat dan terversioning di Google Artifact Registry region Jakarta (asia-southeast2). Hal ini memudahkan manajemen versi aplikasi dan proses rollback jika diperlukan.



BAB IV
PERFORMANCE TESTING DAN COST ANALYSIS

Bab ini membahas hasil pengujian performa serta analisis biaya dari implementasi sistem pemesanan Kantin RK ITERA yang dibangun menggunakan arsitektur serverless di Google Cloud Platform. Pengujian performa dilakukan untuk mengevaluasi kemampuan sistem dalam menangani beban akses yang mendekati kondisi nyata di lingkungan kampus, sedangkan analisis biaya bertujuan untuk memperkirakan kebutuhan anggaran operasional sistem baik pada lingkungan pengembangan maupun produksi.

4.1 Performance Testing

4.1.1 Tujuan Pengujian

Pengujian performa bertujuan untuk mengevaluasi sejauh mana sistem pemesanan Kantin RK ITERA mampu menangani beban akses pengguna yang mendekati kondisi nyata di lingkungan kampus. Fokus utama pengujian meliputi waktu respons (latency), throughput sistem, serta tingkat kegagalan permintaan (error rate). Hasil pengujian ini digunakan sebagai dasar untuk menilai kesiapan sistem sebelum digunakan secara penuh dan sebagai acuan dalam menentukan kebutuhan optimasi infrastruktur.

4.1.2 Lingkungan dan Metode Pengujian

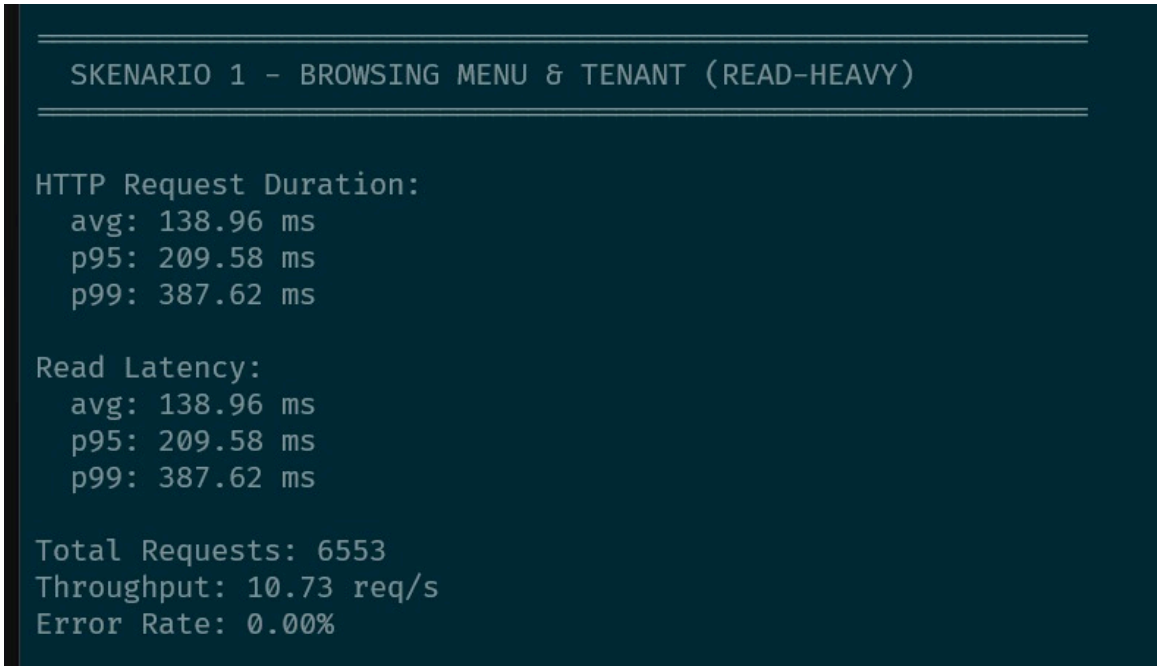
Pengujian dilakukan pada lingkungan yang menyerupai konfigurasi produksi. Frontend aplikasi dibangun menggunakan Vue.js dan di deploy pada Cloud Run, sedangkan backend menggunakan Laravel API yang juga berjalan pada Cloud Run. Basis data dikelola menggunakan Cloud SQL PostgreSQL dengan koneksi privat melalui Virtual Private Cloud (VPC). Pengujian beban dilakukan menggunakan tools *k6*, dengan generator beban ditempatkan pada region yang sama (asia-southeast2) untuk meminimalkan pengaruh latensi jaringan eksternal. Setiap skenario dijalankan dalam durasi tertentu dengan jumlah virtual user (VU) yang berbeda sesuai karakteristik beban yang disimulasikan.

4.1.3 Skenario Pengujian

Untuk merepresentasikan pola penggunaan aplikasi kantin, tiga skenario pengujian diterapkan, yaitu skenario baca dominan (read-heavy), skenario tulis dominan (write-heavy), dan skenario beban campuran (mixed load). Ketiga skenario ini dirancang agar dapat menggambarkan perilaku pengguna pada kondisi normal maupun jam sibuk.

4.1.4 Hasil Pengujian Skenario 1 – Browsing Menu & Tenant (Read-Heavy)

Skenario pertama mensimulasikan aktivitas pengguna saat menjelajahi daftar tenant dan menu makanan. Operasi yang diuji pada skenario ini didominasi oleh permintaan baca (read), sehingga berfokus pada performa endpoint pengambilan data tenant dan menu. Berdasarkan hasil pengujian, waktu respons rata-rata berada pada kisaran 138,96 ms dengan nilai p95 sebesar 209,58 ms dan p99 sebesar 387,62 ms. Total permintaan yang berhasil diproses mencapai 6.553 request dengan throughput sekitar 10,73 request per detik. Selama pengujian berlangsung, tidak ditemukan error, sehingga error rate tercatat sebesar 0%.

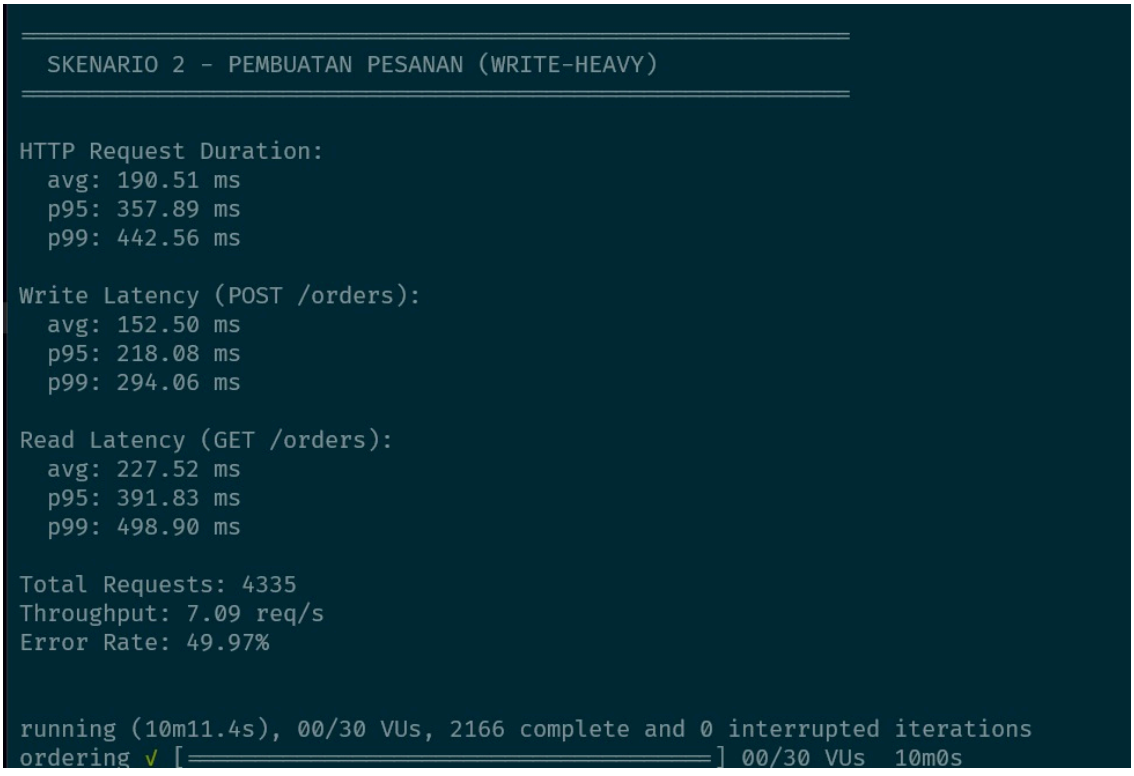


Gambar 4.1 Skenario Read-Heavy

Hasil tersebut menunjukkan bahwa sistem mampu menangani beban baca dengan sangat baik. Latency yang relatif rendah dan tidak adanya error mengindikasikan bahwa query baca ke basis data serta arsitektur backend telah berjalan secara efisien dan stabil.

4.1.5 Hasil Pengujian Skenario 2 – Pembuatan Pesanan (Write-Heavy)

Skenario kedua berfokus pada aktivitas pembuatan pesanan yang melibatkan operasi tulis ke basis data. Pada skenario ini, sistem diuji dengan beban tulis yang relatif tinggi dan berkelanjutan untuk mengamati kestabilan sistem dalam menangani transaksi. Hasil pengujian menunjukkan bahwa waktu respons rata-rata keseluruhan berada pada kisaran 190,51 ms. Untuk operasi penulisan pesanan (POST /orders), latency rata-rata tercatat sebesar 152,50 ms, sedangkan operasi pembacaan data pesanan (GET /orders) memiliki latency rata-rata sebesar 227,52 ms. Total permintaan yang diproses berjumlah 4.335 request dengan throughput sekitar 7,09 request per detik.

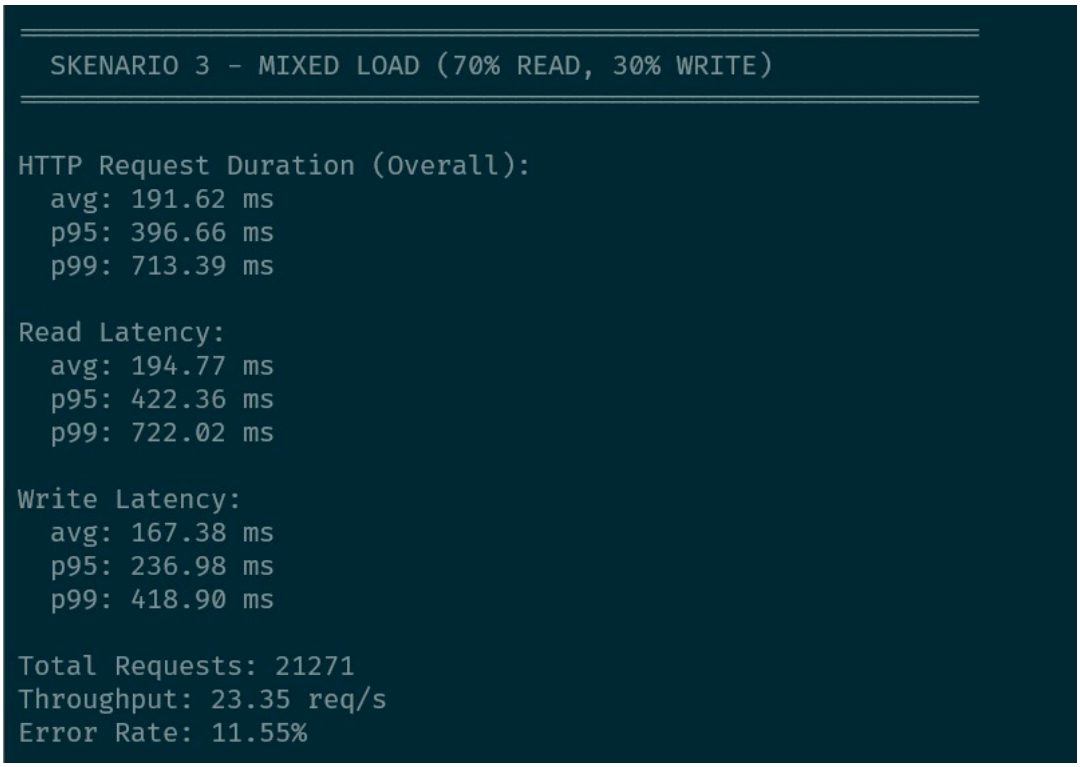


Gambar 4.2 Skenario pengujian Write-Heavy

Namun demikian, pada skenario ini ditemukan tingkat kegagalan yang cukup tinggi, dengan error rate mencapai 49,97%. Kondisi ini menunjukkan bahwa meskipun request yang berhasil diproses memiliki waktu respons yang relatif baik, sistem belum stabil ketika menghadapi beban tulis yang intensif. Hal ini mengindikasikan adanya bottleneck, kemungkinan pada pengelolaan koneksi basis data, kapasitas backend, atau mekanisme transaksi aplikasi.

4.1.6 Hasil Pengujian Skenario 3 – Mixed Load

Skenario ketiga mensimulasikan kondisi jam makan siang, di mana pengguna secara bersamaan melakukan aktivitas melihat menu dan melakukan pemesanan. Beban pada skenario ini merupakan kombinasi antara operasi baca dan tulis. Hasil pengujian menunjukkan bahwa sistem mampu menangani throughput yang lebih tinggi, yaitu sekitar 23,35 request per detik dengan total 21.271 request selama pengujian. Waktu respons rata-rata berada pada kisaran 191,62 ms, dengan nilai p95 mendekati 396,66 ms dan p99 mencapai 713,39 ms. Error rate pada skenario ini tercatat sebesar 11,55%.



Gambar 4.3 Skenario Mixed Load

Secara umum, performa sistem pada beban campuran masih tergolong cukup baik untuk aplikasi kantin. Namun, meningkatnya latency pada persentil tinggi serta masih adanya error menunjukkan bahwa sistem memerlukan optimasi lanjutan agar dapat beroperasi lebih stabil pada kondisi puncak.

4.1.7 Analisis Bottleneck dan Rekomendasi

Berdasarkan ketiga skenario pengujian, dapat disimpulkan bahwa sistem telah stabil untuk operasi baca, namun masih menghadapi kendala pada operasi tulis dengan beban tinggi. Bottleneck utama diperkirakan berasal dari keterbatasan koneksi basis data dan konfigurasi autoscaling backend. Untuk meningkatkan performa dan stabilitas sistem, direkomendasikan penerapan connection pooling pada Cloud SQL, penyesuaian parameter autoscaling Cloud Run seperti concurrency dan jumlah maksimum instance, serta optimasi query dan transaksi pada proses pembuatan pesanan. Pengujian ulang setelah proses optimasi juga diperlukan untuk memastikan penurunan error rate hingga berada pada batas yang dapat diterima.

5.2 Cost Analysis

4.2.1 Tujuan dan Ruang Lingkup Analisis Biaya

Analisis biaya dilakukan untuk memberikan gambaran menyeluruh mengenai kebutuhan anggaran operasional dalam menjalankan aplikasi Kantin RK ITERA di Google Cloud Platform. Analisis ini tidak hanya berfokus pada besaran biaya, tetapi juga mempertimbangkan efisiensi arsitektur serverless yang digunakan, potensi optimasi biaya, serta risiko finansial yang mungkin muncul seiring dengan peningkatan jumlah pengguna dan trafik sistem. Ruang lingkup analisis biaya mencakup seluruh komponen utama infrastruktur, yaitu layanan komputasi Cloud Run (frontend dan backend), basis data Cloud SQL PostgreSQL, Cloud HTTP(S) Load Balancing, Cloud Storage, serta Cloud Logging dan Monitoring. Estimasi biaya dibedakan menjadi dua lingkungan, yaitu lingkungan Development dan Production, agar dapat menggambarkan kebutuhan biaya pada fase pengembangan maupun operasional penuh.

4.2.2 Asumsi Dasar Perhitungan Biaya

Faktor	Development	Production
Pengguna aktif bulanan	±3.000 akun	±30.000 akun
Request HTTP	±30.000/bulan	±900.000/bulan
Lalu lintas keluar (egress internet)	±30 GB/bulan	±250 GB/bulan
Jam aktif rata-rata Cloud Run	±100 jam/bulan (sering idle, scale-to-zero)	730 jam/bulan (min 1 instance aktif)
Database	1 instance kecil (uji coba)	1 instance menengah dengan backup harian
Region harga	`asia-southeast2` (Jakarta)	`asia-southeast2`

Gambar 4.4 Asumsi Dasar Perhitungan Biaya

Perhitungan biaya didasarkan pada sejumlah asumsi yang mencerminkan kondisi penggunaan aplikasi di lingkungan kampus. Pada lingkungan development, diasumsikan sistem digunakan oleh tim pengembang dan tester dengan jumlah pengguna terbatas serta trafik yang relatif rendah. Cloud Run pada lingkungan ini sering berada dalam kondisi idle dan memanfaatkan fitur scale-to-zero. Sebaliknya, pada lingkungan production diasumsikan aplikasi melayani puluhan ribu pengguna aktif dengan trafik yang jauh lebih tinggi, khususnya pada jam operasional kantin. Untuk menjaga ketersediaan layanan, minimal satu instance Cloud Run tetap aktif sepanjang waktu. Seluruh perhitungan biaya mengacu pada harga layanan Google Cloud di region asia-southeast2 (Jakarta) dengan pendekatan estimasi konservatif. Pendekatan ini bertujuan agar hasil analisis tetap relevan meskipun terjadi fluktuasi trafik atau perubahan kecil pada pola penggunaan aplikasi.

4.2.3 Estimasi Biaya Lingkungan Development

Layanan	Spesifikasi & Asumsi	Estimasi Biaya / Bulan
Cloud Run Frontend	0,25 vCPU / 512 MB, autoscale 0→1, ±100 jam aktif	**≈ Rp120.000**
Cloud Run Backend	0,5 vCPU / 1 GB, autoscale 0→2, ±100 jam aktif	**≈ Rp280.000**
Cloud SQL PostgreSQL (kecil)	shared-core kecil, 20 GB SSD + backup basic	**≈ Rp300.000**
Cloud HTTP(S) Load Balancing	1 forwarding rule + ±30 GB egress internet	**≈ Rp200.000**
Cloud Storage	±10 GB file gambar/menu & aset statis	**≈ Rp30.000**
Cloud Logging & Monitoring	Volume log kecil, sebagian masih dalam free tier	**≈ Rp30.000**
Perkiraan Total Development		**≈ Rp960.000 / bulan**

Gambar 4.5 Estimasi Biaya Lingkungan Development

Lingkungan development digunakan sebagai sarana pengembangan, pengujian, dan validasi fitur sebelum sistem diterapkan ke lingkungan produksi. Dengan memanfaatkan arsitektur serverless, layanan Cloud Run pada lingkungan ini dapat berhenti sepenuhnya ketika tidak ada permintaan, sehingga biaya komputasi dapat ditekan secara signifikan. Berdasarkan estimasi, biaya operasional lingkungan development berada pada kisaran Rp950.000 hingga Rp1.000.000 per bulan. Komponen biaya terbesar berasal dari Cloud Run backend dan Cloud SQL PostgreSQL berukuran kecil yang digunakan untuk keperluan uji coba data. Biaya layanan lain seperti Cloud Storage, Load Balancer, serta Logging dan Monitoring relatif kecil karena volume data dan trafik yang masih terbatas.

Dengan mempertimbangkan free tier Cloud Run dan kuota gratis Cloud Logging, biaya riil pada lingkungan development berpotensi lebih rendah, bahkan dapat mendekati Rp800.000 per bulan apabila aktivitas pengujian tidak berlangsung secara intensif. Hal ini menunjukkan bahwa arsitektur yang digunakan cukup efisien untuk mendukung proses pengembangan tanpa membebani anggaran secara berlebihan.

4.2.4 Estimasi Biaya Lingkungan Production

Layanan	Spesifikasi & Asumsi	Estimasi Biaya / Bulan
Cloud Run Frontend	1 vCPU / 1 GB, `min_instances=1`, autoscale sampai ~10 instance	**≈ Rp900.000**
Cloud Run Backend	2 vCPU / 2 GB, `min_instances=1`, autoscale sampai ~15 instance	**≈ Rp2.400.000**
Cloud SQL PostgreSQL (menengah)	setara 2 vCPU, 8 GB RAM, 50 GB SSD, backup harian	**≈ Rp1.700.000**
Cloud HTTP(S) Load Balancing	1 rule, ±250 GB egress internet / bulan	**≈ Rp1.000.000**
Cloud Storage	±50 GB aset statis dan file pendukung	**≈ Rp150.000**
Cloud Logging & Monitoring	±30 GB log tersimpan + metrik dasar	**≈ Rp250.000**
Perkiraan Total Production		**≈ Rp6.400.000 / bulan**

Gambar 4.6 Estimasi Biaya Lingkungan Production

Lingkungan production dirancang untuk melayani seluruh pengguna Kantin RK ITERA secara aktif dan berkelanjutan. Pada lingkungan ini, sistem harus mampu menangani lonjakan trafik, menjaga ketersediaan layanan, serta memastikan keamanan dan konsistensi data transaksi. Estimasi biaya operasional lingkungan production berada pada kisaran Rp6.400.000 per bulan. Biaya terbesar berasal dari Cloud Run backend yang menangani logika bisnis dan transaksi pemesanan, serta Cloud SQL PostgreSQL berkapasitas menengah yang berfungsi sebagai basis data utama. Layanan Cloud HTTP(S) Load Balancing juga memberikan kontribusi biaya yang signifikan, terutama dari sisi lalu lintas keluar (egress internet) yang meningkat seiring bertambahnya jumlah pengguna.

Komponen biaya lain seperti Cloud Storage serta Cloud Logging dan Monitoring melengkapi kebutuhan operasional sistem. Meskipun tidak menjadi penyumbang biaya terbesar, komponen ini tetap berperan penting dalam menjaga keandalan, keamanan, dan observabilitas sistem. Secara keseluruhan, estimasi biaya tersebut dinilai wajar dan proporsional untuk aplikasi skala kampus yang melayani ribuan hingga puluhan ribu pengguna aktif.

4.2.5 Analisis Distribusi Biaya dan Efisiensi Arsitektur

Dari hasil estimasi, dapat disimpulkan bahwa sebagian besar biaya operasional terkonsentrasi pada komponen komputasi backend dan basis data. Hal ini sejalan dengan karakteristik aplikasi yang bersifat transaksional dan memerlukan pemrosesan data secara real-time. Penggunaan arsitektur serverless memberikan keuntungan signifikan dari sisi efisiensi biaya, karena sistem hanya menggunakan sumber daya komputasi saat diperlukan. Dibandingkan dengan pendekatan berbasis server tradisional yang memerlukan kapasitas tetap, Cloud Run memungkinkan sistem menyesuaikan konsumsi sumber daya secara dinamis berdasarkan beban aktual. Namun demikian, fleksibilitas ini juga menuntut pengelolaan konfigurasi yang tepat. Tanpa pengaturan autoscaling, concurrency, dan logging yang optimal, biaya dapat meningkat secara tidak terkontrol ketika terjadi lonjakan trafik.

4.2.6 Strategi Optimalisasi Biaya

Untuk menjaga efisiensi biaya dalam jangka panjang, beberapa strategi optimalisasi dapat diterapkan. Autoscaling Cloud Run dapat diatur secara lebih agresif dengan meningkatkan concurrency per instance dan menetapkan nilai min_instances nol untuk layanan non-kritis. Selain itu, ukuran instance Cloud SQL perlu dievaluasi secara berkala agar sesuai dengan kebutuhan aktual sistem. Penerapan Committed Use Discounts (CUD) juga dapat dipertimbangkan setelah pola penggunaan sistem stabil. Dengan komitmen penggunaan jangka menengah, biaya layanan Cloud SQL dan Cloud Run dapat ditekan secara signifikan. Selain itu, penggunaan Cloud CDN dan pengaturan cache pada aset statis dapat mengurangi biaya egress dari load balancer. Pengelolaan logging juga menjadi aspek penting dalam optimalisasi biaya. Log debug sebaiknya dibatasi pada lingkungan development, sementara pada lingkungan production diterapkan retensi log yang lebih ketat untuk mencegah pembengkakan biaya Cloud Logging.

4.2.7 Risiko Biaya dan Mitigasi

Meskipun estimasi biaya telah disusun secara konservatif, beberapa risiko tetap perlu diperhatikan. Lonjakan trafik mendadak, misalnya akibat acara kampus atau promosi besar, dapat menyebabkan peningkatan jumlah instance Cloud Run dan koneksi basis data yang berdampak langsung pada biaya. Risiko ini dapat dimitigasi dengan pembatasan max_instances serta penerapan connection pooling pada database. Risiko lain berasal dari egress internet yang tinggi akibat pengiriman aset statis berukuran besar. Penggunaan Cloud CDN dan kompresi data dapat menjadi solusi untuk menekan biaya tersebut. Selain itu, risiko over-provisioning sumber daya dapat dihindari dengan pemantauan metrik secara rutin dan penerapan prinsip rightsizing.

4.2.8 Kesimpulan Analisis Biaya

Berdasarkan hasil analisis, dapat disimpulkan bahwa aplikasi Kantin RK ITERA memiliki kebutuhan biaya operasional yang relatif efisien untuk skala aplikasi kampus. Dengan estimasi sekitar satu juta rupiah per bulan untuk lingkungan development dan sekitar enam juta rupiah per bulan untuk lingkungan production, sistem ini dinilai layak secara finansial. Melalui penerapan autoscaling, rightsizing, serta praktik FinOps sederhana, biaya operasional masih dapat dioptimalkan lebih lanjut tanpa mengorbankan performa dan keandalan sistem. Hal ini menunjukkan bahwa arsitektur cloud yang digunakan tidak hanya mendukung kebutuhan teknis sistem, tetapi juga selaras dengan pertimbangan efisiensi biaya jangka panjang.

BAB V
KESIMPULAN

Proyek infrastruktur komputasi awan untuk aplikasi pemesanan di Kantin Rumah Kayu ITERA ini telah berhasil diimplementasikan dengan memenuhi standar keamanan dan operasional yang tinggi. Penggunaan layanan serverless Cloud Run yang dipadukan dengan keamanan jaringan privat VPC, manajemen rahasia terpusat, serta otomatisasi CI/CD membuktikan bahwa arsitektur ini siap untuk menangani beban kerja produksi dengan aman dan efisien. Seluruh konfigurasi telah diuji dan berjalan sesuai dengan spesifikasi yang diharapkan.