KONI Super App

Technical Architecture Overview

Versi 1.0.0 | April 2026 | Draft

Agenda

No	Topik	Hal yang Dibahas
01	**Overview**	Apa itu KONI Super App, 5 modul utama, platform target
02	**Tech Stack**	Teknologi yang dipilih dan alasannya
03	**Architecture**	Microservices, API Gateway, inter-service communication
04	**Compro Platform**	Multi-repo CMS untuk ratusan cabang KONI
05	**Auth & Security**	SSO, RBAC, JWT, role hierarchy
06	**Database**	Database-per-service, MVP schema, JSONB
07	**Key Flows**	Order, SSO Login, CMS Update, Athlete, Booking
08	**Observability**	OpenTelemetry, Grafana LGTM, debugging workflow
09	**Deployment**	Per-server, Traefik, CI/CD, per-service pipeline
10	**Infra Spec**	Server requirements, VPS Indonesia, cost & scaling

01 — Overview

Apa itu KONI Super App?

Super App untuk Olahraga Indonesia

Platform digital terpadu untuk **Komite Olahraga Nasional Indonesia** yang menggabungkan 5 modul utama dalam satu ekosistem:

#	Modul	Deskripsi
1	**Marketplace**	Multi-vendor e-commerce — KONI toko sendiri + merchant eksternal
2	**Tiket Pesawat**	Hybrid: aggregator API pihak ketiga + deal khusus KONI
3	**Hotel**	Hybrid: aggregator API pihak ketiga + deal khusus KONI
4	**Database Atlet**	Profil komprehensif, tim, cabor, statistik, tournament
5	**GMS**	Game Management System — integrasi API Korea (standar internasional)

> **Ditambah:** Compro Platform — ratusan website company profile untuk KONI cabang seluruh Indonesia

Platform Target

Web-First, Mobile-Optimized:

- Web-only, **responsive design** (mobile-first approach)
- **PWA** (Progressive Web App) — installable, push notification
- **SSO** — satu akun untuk semua platform (Main App + semua Compro cabang)

3 Frontend Platform:

Platform	Domain	Fungsi
Main App	`app.koni.id`	Super app utama — marketplace, booking, athlete, admin
Compro Instances	`{slug}.koni.id`	Website per cabang — news, events, profil, galeri
Admin Dashboard	Built-in di kedua platform	CMS, user management, reports

User Roles — 9 Level Hierarchy

Role	Scope Akses	Deskripsi
`superadmin`	**Global**	Akses penuh seluruh sistem dan semua cabang
`admin`	**KONI Pusat**	Manajemen KONI pusat, melihat seluruh cabang
`admin-cabang`	**1 Cabang**	Mengelola satu cabang KONI (CMS, atlet, event)
`merchant`	**Marketplace**	Seller di marketplace, kelola toko & produk
`official-team`	**1 Tim**	Pelatih, manajer tim — kelola roster & jadwal
`player`	**Profil Sendiri**	Atlet terdaftar — profil, statistik, tim
`guardian`	**Linked Player**	Wali atlet di bawah umur
`game-admin`	**GMS**	Pengelola pertandingan & turnamen
`user`	**Publik**	Pengguna umum — browse, beli, booking

> JWT payload berisi: `sub`, `email`, `roles[]`, `tenant_id`, `permissions[]`

02 — Tech Stack

Teknologi yang dipilih dan alasannya

Technology Choices *(1/2 — Platform)*

Layer	Teknologi	Alasan singkat
Backend	FastAPI (Python)	Async, OpenAPI otomatis, typing kuat
API Gateway	Traefik	Routing, TLS, rate limit — satu pintu `api.koni.id`
Frontend	Next.js 15+ (React 19)	App Router, SSR/ISR, Server Components
Compro SDK	Paket NPM (TypeScript)	Komponen & API client bersama untuk semua compro
Database	PostgreSQL 16+ (per service)	JSONB, partitioning, ekosistem matang
Observability	OpenTelemetry + Grafana LGTM	Trace, log, metrik dalam satu dashboard
CI/CD	GitHub Actions + Docker	Build & deploy per service

Technology Choices *(2/2 — Data & integrasi)*

Layer	Teknologi	Alasan singkat
Cache & Session	Redis 7+	Session, cache, rate limit (**bukan** antrean event)
Message Broker	RabbitMQ	Event antar service — ACK, DLQ, retry
Object Storage	MinIO (S3-compatible)	Media atlet, produk, aset CMS
Payment	Midtrans	Bank, e-wallet, QRIS Indonesia
Shipping	RajaOngkir API	Ongkir multi-kurir domestik

03 — System Architecture

Microservices, API Gateway, Inter-Service Communication

10 Independent Microservices

Setiap service memiliki **database sendiri**, **deploy sendiri**, dan bisa **di-scale sendiri**:

#	Service	Port	Database	Tanggung Jawab
1	**Auth**	:8001	`auth_db`	SSO, JWT, users, roles, permissions
2	**Tenant**	:8002	`tenant_db`	Cabang KONI, subscription, billing
3	**CMS**	:8003	`cms_db`	Pages, news, events, gallery, themes
4	**Athlete**	:8004	`athlete_db`	Athletes, teams, sports, statistics
5	**GMS**	:8005	`gms_db`	Tournaments, matches, Korean API
6	**Marketplace**	:8006	`marketplace_db`	Products, merchants, cart, orders
7	**Booking**	:8007	`booking_db`	Flights, hotels, deal aggregation
8	**Media**	:8008	`media_db`	File upload, image processing, MinIO
9	**Notification**	:8009	`notification_db`	Email, push, in-app notification
10	**Payment**	:8010	`payment_db`	Midtrans, pembayaran, refund, webhook

Kenapa Microservices?

Keuntungan:

Aspek	Dampak
Parallel development	Tim berbeda mengerjakan service berbeda **tanpa blocking**
Independent deployment	Update Marketplace **tidak pengaruhi** Auth atau CMS
Isolated scaling	CMS traffic tinggi? Scale CMS saja, bukan seluruh sistem
Fault isolation	GMS Service down? Marketplace & CMS **tetap jalan**
Tech flexibility	Setiap service bisa optimasi library/tool sendiri

Tantangan & Solusi:

Tantangan	Solusi yang Dipilih
Distributed debugging sulit	OpenTelemetry + Grafana (lihat Section 08)
Komunikasi antar service	REST (sync) + RabbitMQ (async events & tasks)
Data consistency lintas service	Eventual consistency + saga pattern
Service discovery	Traefik routing + private VLAN antar server

API Gateway — Traefik

Satu domain publik: `https://api.koni.id` — semua microservice **tidak** expose port sendiri ke internet; yang terbuka hanya **80/443 di Traefik**.

Routing (cuplikan):

Path (awalan)	Diteruskan ke
`/api/v1/auth/**`	Auth `:8001`
`/api/v1/cms/**`	CMS `:8003`
`/api/v1/marketplace/**`	Marketplace `:8006`
`/api/v1/payments/**`	Payment `:8010`

(Service lain mengikuti pola yang sama.)

Contoh 1 request lengkap:

1. Browser / app: `GET https://api.koni.id/api/v1/auth/me` + header `Authorization: Bearer …`

Inter-Service Communication

Dua pola komunikasi:

Pola	Mekanisme	Kapan Dipakai	Contoh
Synchronous	REST (HTTP)	Butuh response langsung	Marketplace **validasi user** ke Auth
Asynchronous	RabbitMQ	Reliable, retry, dead letter	`order.paid` **trigger email** ke Notification

Internal calls via private VLAN: `http://{server_ip}:{port}/internal/{resource}/{id}`

> Dilindungi **service key** — hanya bisa dipanggil dari private VLAN, bukan via API Gateway

04 — Compro Platform

Multi-repo CMS untuk ratusan cabang KONI

Arsitektur Compro — 3 Komponen

Setiap KONI cabang punya **website sendiri** (repo terpisah, deploy terpisah), terhubung ke **satu backend**:

Komponen	Repo	Fungsi
Compro Template	`koni-compro-template`	Starter project, di-fork per cabang baru
Compro SDK	`@koni/compro-sdk` (NPM)	Shared: UI, API client, auth, theming
Compro Instance	`koni-compro-{slug}`	Fork dari template, 1 repo per cabang

Onboarding: Create tenant → Fork template → Set config → Deploy → Admin isi konten

Compro — Dual Admin Access

Admin cabang mengelola konten dari **2 tempat** yang mengakses API yang sama:

Akses	URL	Fitur
Built-in Admin	`jatim.koni.id/admin`	CMS: pages, news, events, gallery, theme
Main App Admin	`app.koni.id/dashboard/tenant/koni-jatim`	CMS + athlete, users, reports

Fitur setiap Compro:

Halaman Publik	Admin Panel
News, profil cabang, galeri, jadwal event	CRUD halaman, berita, galeri, event
Profil atlet, pendaftaran online, SSO login	Theme editor, pengaturan site

Compro — ISR & Theming

ISR (Incremental Static Regeneration):

- Halaman di-generate **statis** (response < 50ms)
- Di-**revalidate on-demand** saat admin update konten

Dynamic Theming via CSS Custom Properties:

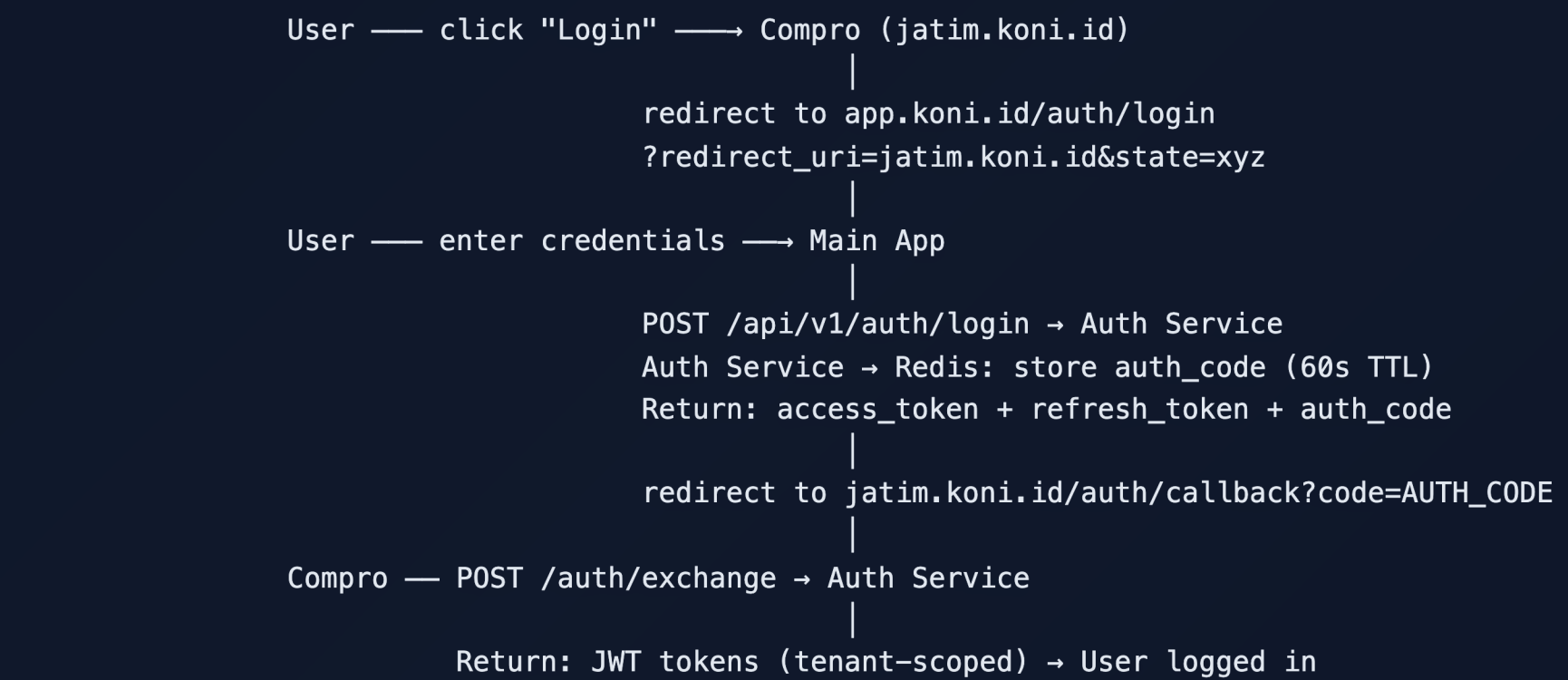
```
Admin ubah warna di theme editor → Disimpan ke CMS Service
    ↓
ThemeProvider (SDK) fetch config → inject CSS variables:
    --color-primary, --color-secondary, --font-heading
    ↓
Seluruh halaman compro berubah warna secara otomatis
```

05 — Authentication & Security

SSO, RBAC, JWT Strategy

SSO Flow — Satu Akun untuk Semua

Login **selalu** di `app.koni.id`, redirect kembali ke compro:



Token Strategy & RBAC

Token Types:

Token	Tipe	Lifetime	Storage
Access Token	JWT (RS256)	15 menit	Memory
Refresh Token	Opaque UUID	7 hari	HttpOnly Cookie
Auth Code	Opaque UUID	60 detik	Redis

Role Hierarchy:

```
superadmin → admin → admin-cabang → official-team → player / guardian
                   ↘ merchant                      ↘ game-admin
user (pengguna umum — browse, beli, booking)
```

> JWT payload: `sub`, `email`, `roles[]`, `tenant_id`, `permissions[]`

06 — Database Design

Database-per-Service Pattern

10 Database, 42 Tabel

Setiap microservice punya **PostgreSQL database sendiri** — tidak ada cross-database foreign key:

```
PostgreSQL Instance
 │
 ├── auth_db          6 tabel   users, roles, permissions, user_roles,
 │                               role_permissions, refresh_tokens
 ├── tenant_db        3 tabel   tenants, subscriptions, subscription_invoices
 ├── cms_db           6 tabel   cms_themes, cms_pages, cms_sections,
 │                               cms_news, cms_events, cms_galleries
 ├── athlete_db       4 tabel   sports, tenant_sports, teams, athletes
 ├── gms_db           4 tabel   tournaments, matches, match_participants,
 │                               gms_sync_logs
 ├── marketplace_db   8 tabel   merchants, products, product_images, carts,
 │                               cart_items, orders, order_items, reviews
 ├── booking_db       3 tabel   bookings, booking_passengers, koni_deals
 ├── media_db         2 tabel   media_files, media_folders
 ├── notification_db  3 tabel   notifications, notification_templates,
 │                               notification_logs
 └── payment_db       3 tabel   payments, payment_methods, refunds
```

Cross-Service & JSONB

Cross-Service Data References (ID reference, bukan FK constraint):

ID	Dimiliki oleh	Digunakan oleh
`user_id`	Auth Service	Semua service lain
`tenant_id`	Tenant Service	CMS, Athlete, GMS
`athlete_id`	Athlete Service	GMS (match participants)

Validasi via REST: `GET /internal/users/{id}`, `GET /internal/tenants/{id}`

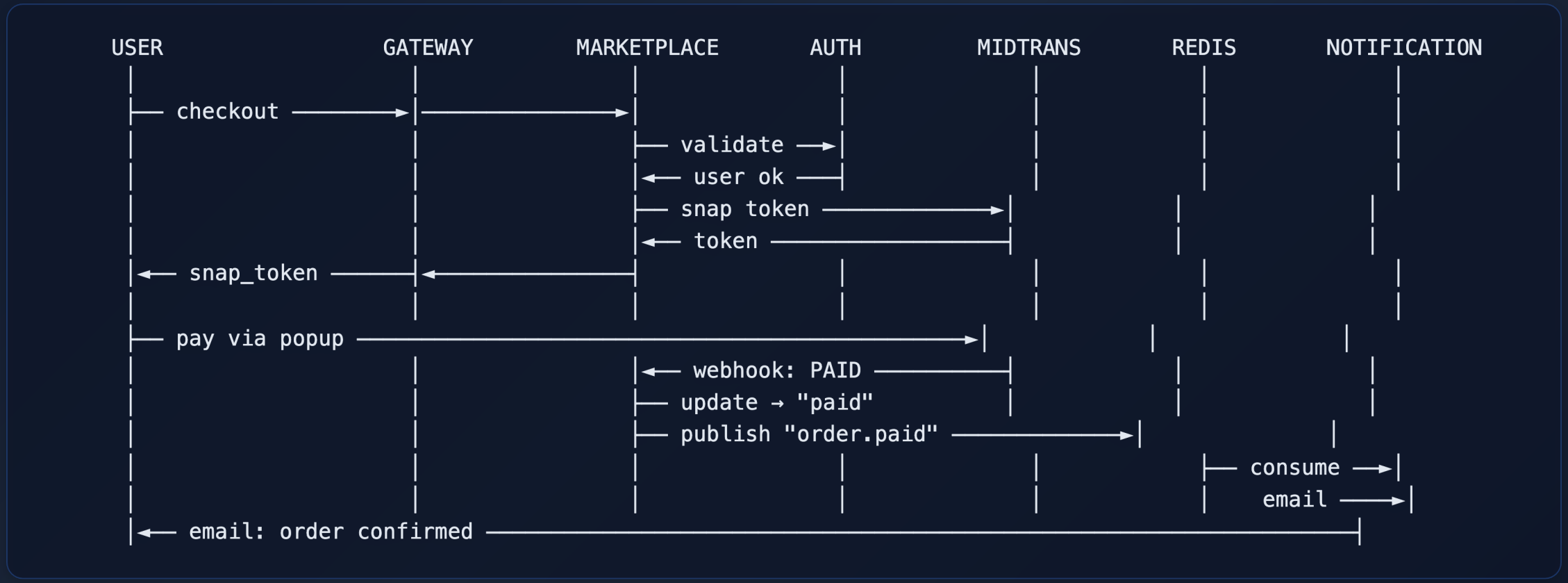
JSONB untuk data yang berbeda-beda per cabor:

Data	Contoh Catur	Contoh Sepakbola
Athlete stats	`{"rating_elo": 2100}`	`{"goals": 45, "assists": 12}`
Match score	`{"result": "1–0"}`	`{"home": 3, "away": 1}`
Match stats	`{"moves": 42}`	`{"possession": "58%"}`

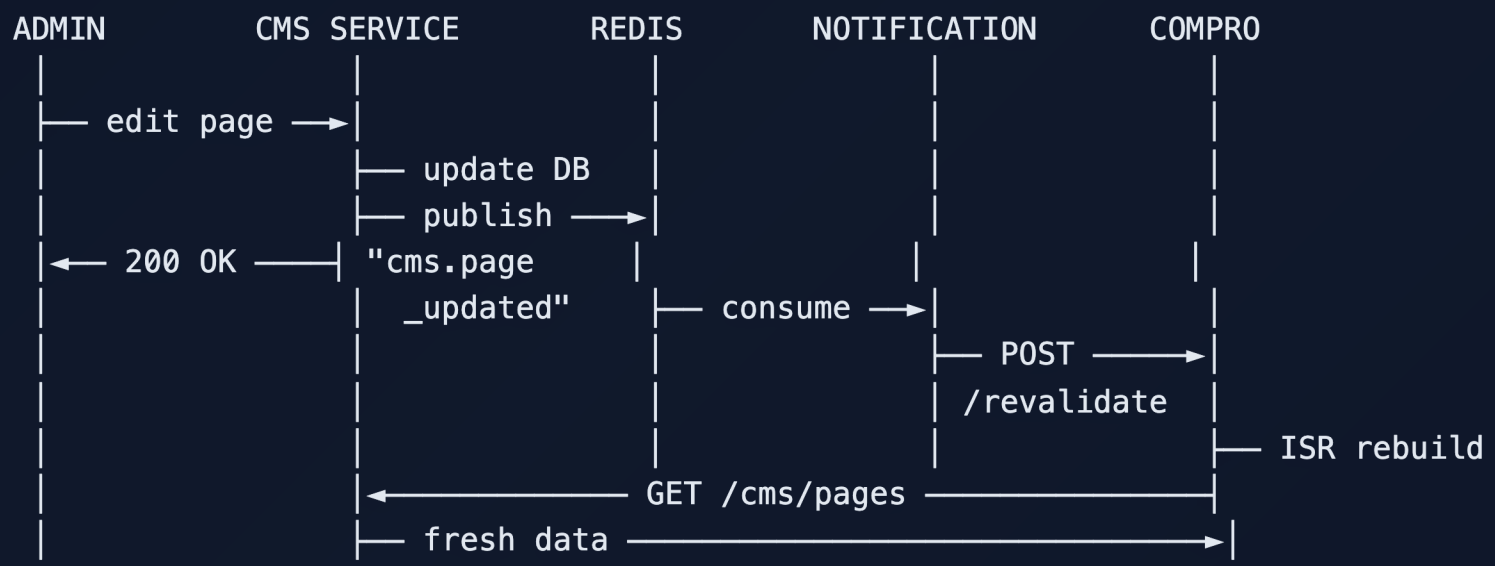
07 — Key Flows

Order, SSO, CMS Update, Athlete Registration, Booking

Flow: Marketplace Order

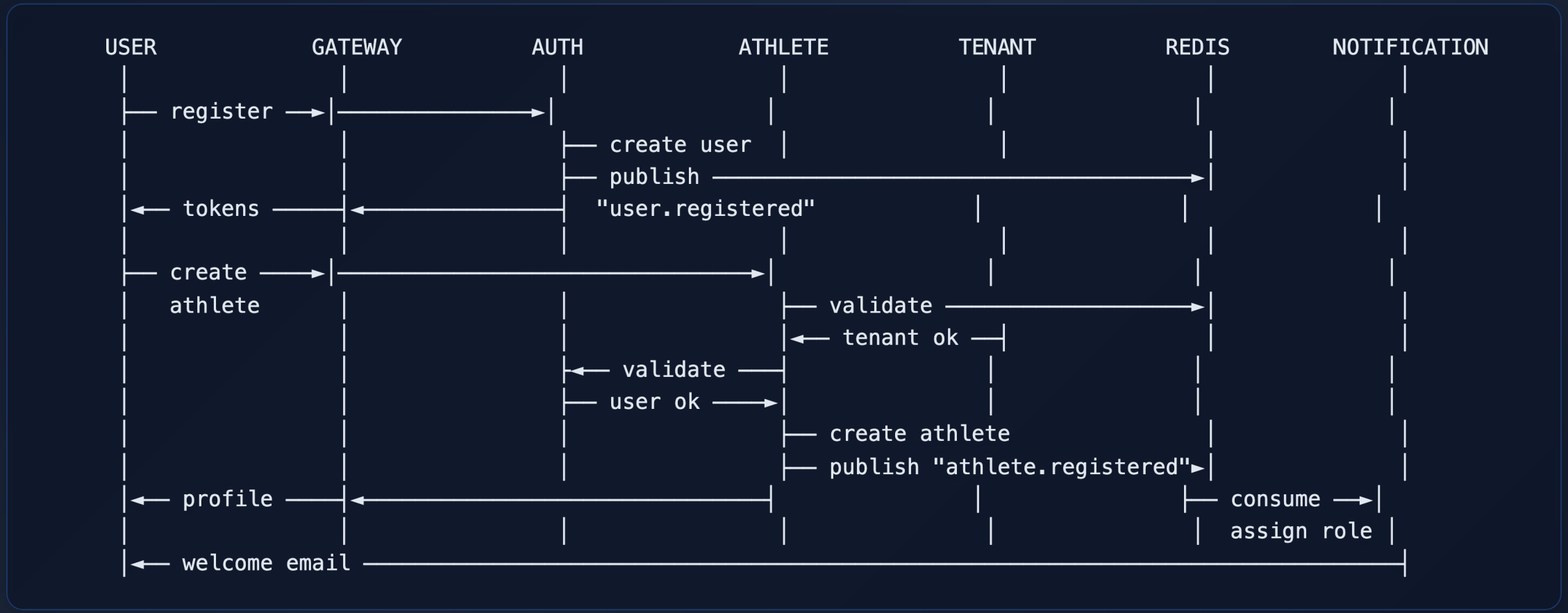


Flow: CMS Update & Compro Revalidation

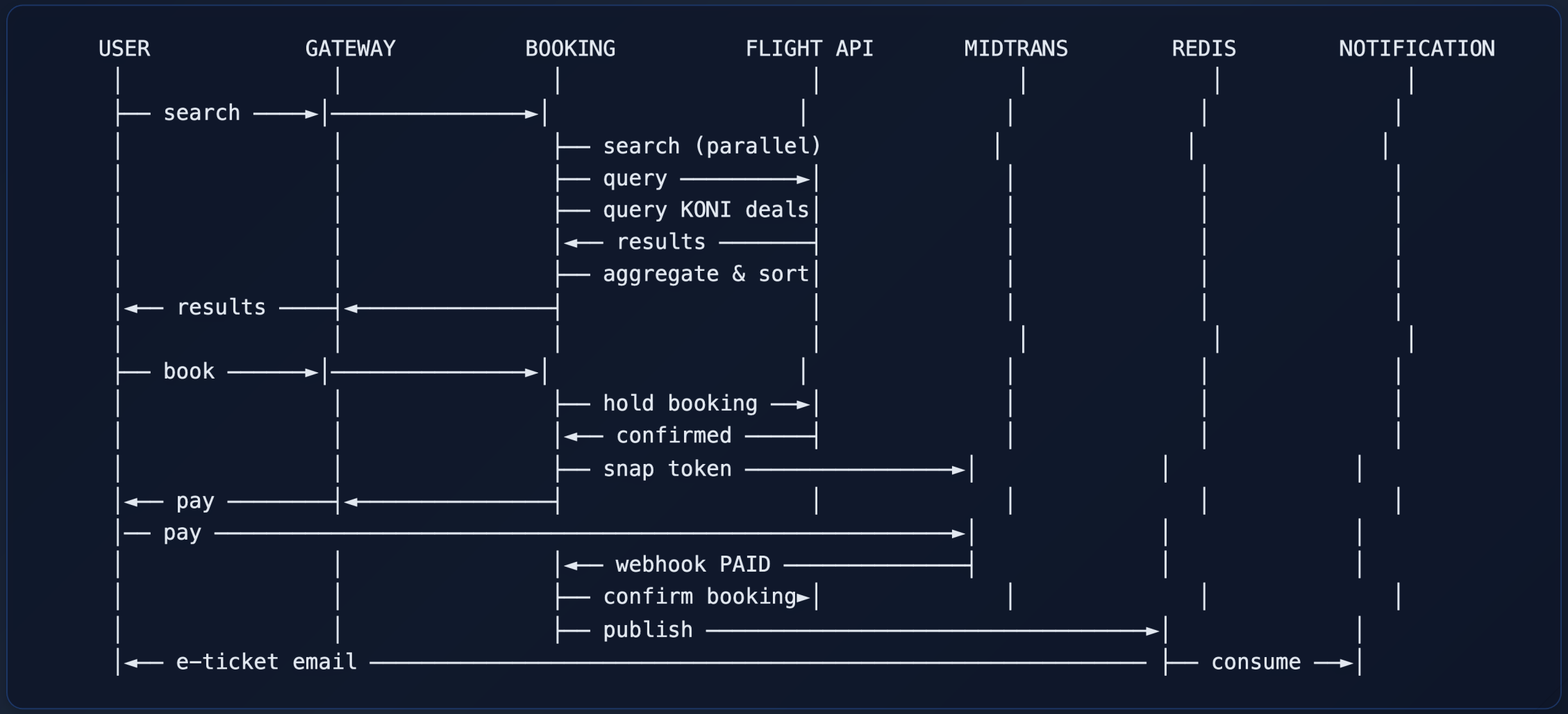


> Halaman publik terupdate dalam **hitungan detik** setelah admin save

Flow: Athlete Registration



Flow: Booking (Flight)



08 — Observability & Debugging

Cara tercepat menemukan dan memperbaiki bug

Grafana LGTM Stack + OpenTelemetry

Komponen	Fungsi
OpenTelemetry	Instrumentasi otomatis — collect traces, logs, metrics
Loki	Log aggregation — structured JSON logs
Tempo	Distributed tracing — trace request lintas service
Prometheus	Metrics — latency, error rate, throughput
Grafana	Unified dashboard — satu tempat untuk semua data

Kenapa LGTM, bukan ELK? RAM jauh lebih ringan (Loki ~256MB vs Elasticsearch ~2-4GB), unified dashboard (logs + traces + metrics dalam satu tempat), dan menggunakan standard **OpenTelemetry** yang open-source.

Correlation ID — Kunci Debugging

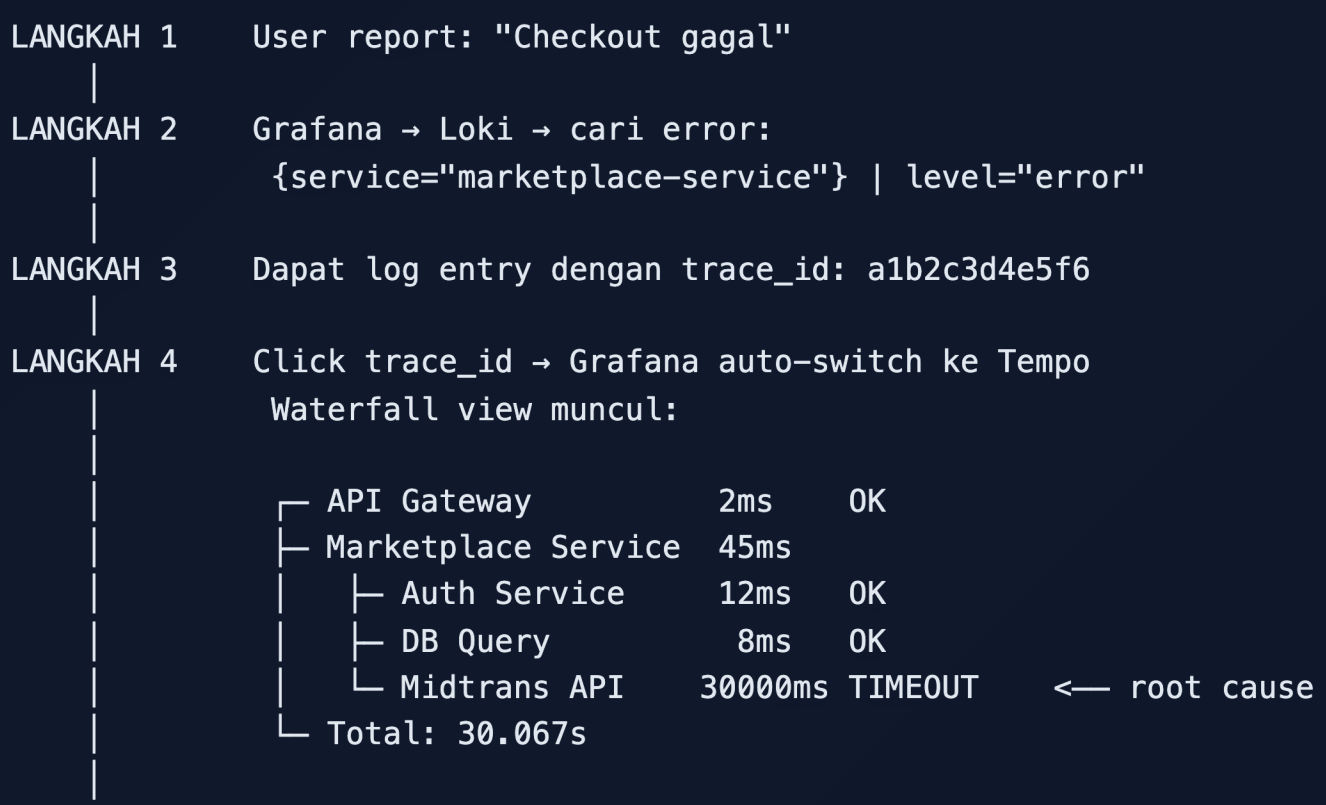
Setiap request mendapat **unique trace_id** yang dipropagasi ke **semua service**:

```
{
  "level": "error",
  "service": "marketplace-service",
  "trace_id": "a1b2c3d4e5f6",
  "event": "checkout_failed",
  "error": "midtrans_timeout"
}
```

Query di Grafana — satu trace_id menampilkan semua logs lintas service:

```
{service=~".+"} | json | trace_id = "a1b2c3d4e5f6"
```

Debugging Workflow — 5 Langkah



09 — Deployment & CI/CD

Per-Server Deployment, Traefik, Per-Service Pipeline

Deployment — Per Server, Per Service

Setiap service punya **Dockerfile sendiri** dan di-deploy **independen per server**:

App Server 1	Auth, Tenant, CMS, Athlete, GMS + Traefik + Next.js
App Server 2	Marketplace, Booking, Media, Notification, Payment + MinIO
DB Server	PostgreSQL (10 databases) + Redis + RabbitMQ
Observability	OTel Collector, Loki, Tempo, Prometheus, Grafana

Compro instances di-deploy di **VPS terpisah** (1 VPS bisa hosting beberapa compro via Nginx):

Compro Server	Nginx reverse proxy → compro-jatim, compro-jabar, compro-dki, ...

CI/CD — Independent Per Service

Setiap service punya **pipeline sendiri** — hanya service yang berubah yang di-build & deploy:

```
Git Push: koni-marketplace-service
    │
    ├── 1. Lint & Type Check
    ├── 2. Run Tests (pytest)
    ├── 3. Docker Build (marketplace-service only)
    ├── 4. Push to ghcr.io/koni/marketplace-service
    └── 5. Deploy: SSH → docker pull & restart container
```

Branch	Deploy Target	Trigger
`develop`	Staging	Otomatis setiap push
`main`	Production	Manual trigger (approval)

> Update Marketplace **tidak perlu** rebuild Auth, CMS, atau service lain

10 — Infrastructure Specification

Spesifikasi server, network, domain, SSL, dan estimasi biaya

Server Requirements — Development

Untuk development & testing (1-5 user), **semua jalan di 1 mesin** via Docker Compose:

Komponen	Spesifikasi	Catatan
vCPU	4 cores	Cukup untuk 10 service + DB + Redis + RabbitMQ
RAM	8 GB (min) — 16 GB (ideal)	8 GB ketat tapi jalan, 16 GB nyaman
Storage	50 GB SSD	OS + Docker images + database
OS	Ubuntu 24.04 / macOS / Windows (WSL2)	Docker Desktop atau Docker Engine

> Semua service, database, Redis, MinIO, dan observability stack dalam **satu** `docker-compose up`

Server Requirements — Production

Untuk live dengan banyak cabang, **dipisah ke beberapa server** (IDCloudHost / Biznet Gio):

Server	vCPU	RAM	Storage	Isi
App Server 1	8	16 GB	100 GB	Auth, Tenant, CMS, Athlete, GMS + Traefik + Next.js
App Server 2	8	16 GB	100 GB	Marketplace, Booking, Media, Notif, Payment + MinIO
DB Server	4	16 GB	200 GB	PostgreSQL (10 DB) + Redis + RabbitMQ
Observability	4	8 GB	100 GB	Grafana, Loki, Tempo, Prometheus
Compro Server	4	8 GB	50 GB	Nginx + beberapa compro cabang

> **Provider:** IDCloudHost / Biznet Gio (datacenter Indonesia, latency rendah)

Compro Deployment — VPS + Nginx

Semua compro cabang di-deploy di **VPS Indonesia**, dikelola dengan **Nginx** sebagai reverse proxy:

Aspek	Detail
Server	1 VPS bisa hosting **10-20 compro** (tergantung traffic)
Reverse Proxy	Nginx — routing domain ke container/port yang tepat
SSL	Let's Encrypt via Certbot (auto-renewal)
ISR	Next.js standalone mode + on-demand revalidation
Deploy	Git push → GitHub Actions → Docker build → deploy ke VPS

Contoh Nginx config per cabang:

```
jatim.koni.id  → localhost:3001 (compro-jatim)
jabar.koni.id  → localhost:3002 (compro-jabar)
dki.koni.id    → localhost:3003 (compro-dki)
```

> **Scaling:** Jika 1 VPS penuh, tambah VPS baru untuk cabang berikutnya

Cost Estimation & Scaling Path

Estimasi Biaya Bulanan (IDCloudHost / Biznet Gio — Indonesia):

Fase	Cabang	Infrastruktur	Biaya/bulan
Staging	—	1 VPS (4 vCPU, 8 GB)	~Rp 200K
Production v1	< 50	5 VPS (lihat tabel)	~Rp 1.5 - 2 juta
Production v2	50 - 200	7+ VPS + managed DB	~Rp 5 - 8 juta
Production v3	200+	Kubernetes cluster	~Rp 12 - 20 juta

Scaling Milestones:

Trigger	Action
DB connections > 100	PgBouncer connection pooling
Server CPU > 70%	Horizontal scaling (load balancer)
200+ cabang	Migrasi ke **Kubernetes**
DB > 100 GB	Read replicas + table partitioning

Provider: IDCloudHost · Biznet Gio (datacenter lokal Indonesia, latency rendah)

Terima Kasih

KONI Super App — Technical Architecture

Dokumentasi lengkap tersedia di `TECHNICAL_DOCUMENTATION.md`
14 sections | ER diagrams, flow charts, API specs, infra specs

Appendix

Referensi tambahan

Appendix: Event Catalog

Event	Publisher	Consumer	Trigger
`user.registered`	Auth	Notification	User baru daftar
`user.role_changed`	Auth	Tenant, CMS, Athlete	Role diubah
`tenant.created`	Tenant	CMS, Notification	Cabang baru
`tenant.subscription_paid`	Tenant	Notification	Bayar subscription
`cms.page_updated`	CMS	Notification → Compro	Admin edit halaman
`cms.news_published`	CMS	Notification	Berita dipublish
`athlete.registered`	Athlete	Auth, Notification	Atlet baru daftar
`tournament.created`	GMS	Athlete, Notification	Turnamen baru
`match.completed`	GMS	Athlete (stats)	Pertandingan selesai
`order.paid`	Marketplace	Notification	Pembayaran berhasil
`order.shipped`	Marketplace	Notification	Pesanan dikirim
`booking.confirmed`	Booking	Notification	Booking dikonfirmasi

Appendix: Tech Stack Versions

Teknologi	Minimum Version	Catatan
Python	3.12+	asyncio improvements, typing
FastAPI	0.115+	Latest stable
SQLAlchemy	2.0+	Async support
Pydantic	2.0+	V2 performance
Next.js	15+	App Router, Server Components
PostgreSQL	16+	JSONB improvements
Redis	7+	Cache & session
RabbitMQ	3.13+	Message broker, management UI
Traefik	3.0+	Docker provider, metrics
OpenTelemetry	1.20+	Auto-instrumentation
Grafana	11+	Unified dashboard
Loki	3.0+	Log aggregation
Tempo	2.0+	Distributed tracing
Docker	27+	Compose V2

Penjelasan Teknologi

Teknologi yang mungkin belum familiar

Apa itu ISR (Incremental Static Regeneration)?

ISR adalah fitur **Next.js** yang membuat halaman website super cepat tapi tetap bisa diupdate kapan saja.

Analogi: Seperti papan pengumuman yang sudah dicetak (cepat dibaca siapa saja), tapi bisa diganti isinya oleh admin kapan saja tanpa harus cetak ulang seluruh papan.

Tanpa ISR	Dengan ISR
Setiap user buka halaman → server query database → render HTML → kirim	Halaman sudah jadi HTML statis → langsung kirim
Response time: **200-500ms**	Response time: **< 50ms**
Server beban tinggi jika banyak pengunjung	Server ringan — hanya kirim file HTML
Konten selalu real-time	Konten diupdate saat admin save (hitungan detik)

Apa itu Traefik?

Traefik = **reverse proxy** di depan semua service: client hanya kenal `api.koni.id`, Traefik yang meneruskan ke container/service di belakang layar.

Analogi: Resepsionis hotel — tamu sebut tujuan, resepsionis arahkan ke "kamar" (microservice) yang benar.

Contoh aturan (konsep): *"Jika host `api.koni.id` dan path diawali `/api/v1/auth`, kirim ke pool `auth-service`."*
Di Docker, sering diisi lewat **label** pada container (Traefik baca dari Docker socket) — tanpa edit file Nginx manual tiap kali tambah service.

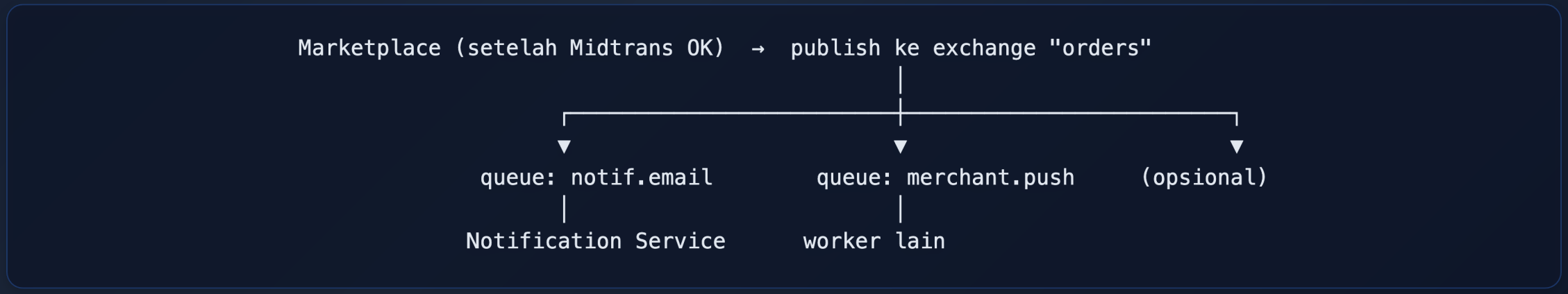
Fitur	Manfaat di KONI
TLS	Let's Encrypt otomatis untuk `api.koni.id`
Routing	Path → service & port di VLAN
Rate limit / CORS	Satu tempat untuk kebijakan akses

Apa itu RabbitMQ?

Message broker — antar service tidak saling “telepon” langsung untuk pekerjaan yang boleh **tunda**; mereka **titip pesan** ke antrean. Pesan **disimpan** sampai consumer selesai memproses (**ACK**).

Analogi: Kantor pos — surat tidak hilang walau penerima sedang offline.

Contoh singkat di KONI



Konsep	Arti singkat

Redis vs RabbitMQ — Siapa Kerja Apa?

	Redis	RabbitMQ
Untuk	Session login, cache API, rate limit, kode SSO sementara	Event bisnis: `order.paid`, kirim email, side-effect antar service
Kalau restart	Cache boleh kosong lagi	Pesan penting **tetap ada** sampai berhasil diproses
Redis Streams	**Tidak dipakai** untuk event — satu pola saja: RabbitMQ	—

> **Prinsip:** Redis = cepat & boleh hilang; RabbitMQ = **tidak boleh hilang** (pembayaran, notifikasi).

Meilisearch — Opsional (Roadmap)

Tidak wajib di awal. Search marketplace & atlet cukup pakai **PostgreSQL** (`pg_trgm` + indeks).

Aspek	PostgreSQL dulu	+ Meilisearch nanti
Typo	Terbatas	Jauh lebih toleran
Ketik cepat di search box	Bisa berat	Dirancang untuk respons sangat cepat
Filter banyak sekaligus	SQL kompleks	Facet bawaan

Kapan ditambah? Latensi search naik, keluhan typo, atau data sudah besar (mis. 50+ cabang / banyak ribuan dokumen).

Cara pasang: container Meilisearch + worker konsumsi **RabbitMQ** (sinkron dari PostgreSQL) — tanpa ubah kontrak API utama (bisa feature flag).

Apa itu OpenTelemetry?

OpenTelemetry (OTel) adalah **standar open-source** untuk mengumpulkan data monitoring dari semua service.

Analogi: GPS tracker di setiap paket pengiriman — lacak perjalanan request dari awal sampai selesai.

Data	Fungsi	Contoh
Traces	Lacak perjalanan request	Auth (12ms) → Marketplace (45ms) → Midtrans (2s)
Logs	Rekam kejadian & error	`"error": "midtrans_timeout"`
Metrics	Ukur performa sistem	Error rate 2.3%, response 120ms

> **Kenapa OTel?** Standar industri, vendor-neutral, auto-instrument (minimal ubah kode).

Apa itu Grafana, Loki & Tempo?

Tiga tool ini membentuk **"ruang kontrol"** untuk monitoring seluruh sistem:

Tool	Fungsi	Analogi
Grafana	Dashboard visual — semua data di satu layar	TV monitor di ruang kontrol
Loki	Simpan & cari log dari semua 10 service	Filing cabinet yang bisa dicari instan
Tempo	Visualisasi trace — perjalanan satu request	Peta rute pengiriman paket

Cara kerja: User report bug → Grafana cari error di Loki → Dapat trace_id → Klik → Tempo tampilkan perjalanan request → Root cause ditemukan

> **Kenapa bukan ELK?** Loki ~256MB RAM vs Elasticsearch ~2-4GB. Lebih hemat & sudah include tracing.

RajaOngkir — Cek Ongkir Marketplace

RajaOngkir adalah API untuk mengecek ongkos kirim dari **17+ kurir domestik** Indonesia (JNE, J&T, SiCepat, Anteraja, POS, dll).

Paket	Harga	Request/Hari	Fitur
Starter	**Rp 0 (Gratis)**	100	Cek ongkir domestik + internasional
Pro	Rp 149.000/bulan	25.000	Response cepat + semua fitur
Enterprise	Rp 249.000/bulan	50.000	Buat resi otomatis + pickup + payment

Flow di marketplace KONI:

```
Buyer pilih produk → Masukkan alamat → RajaOngkir API cek ongkir
→ Tampilkan pilihan kurir & harga → Buyer pilih → Checkout
```

> **Rekomendasi:** Mulai dengan **Starter (gratis)** saat MVP, upgrade ke **Pro** saat order > 100/hari