

Edge-delivered. Database-enforced.

How the platform is hosted, from the browser to the database. A **global Cloudflare edge** serves and protects the application, and a **dedicated Supabase backend on AWS** holds your data, your users and your business logic, with security enforced at the database engine itself.



SOVEREIGN
— AUTO ENGINE —

2 layers

Edge delivery and backend, each with a distinct security role

TLS.

Automated certificates, terminated at the Cloudflare edge

RLS.

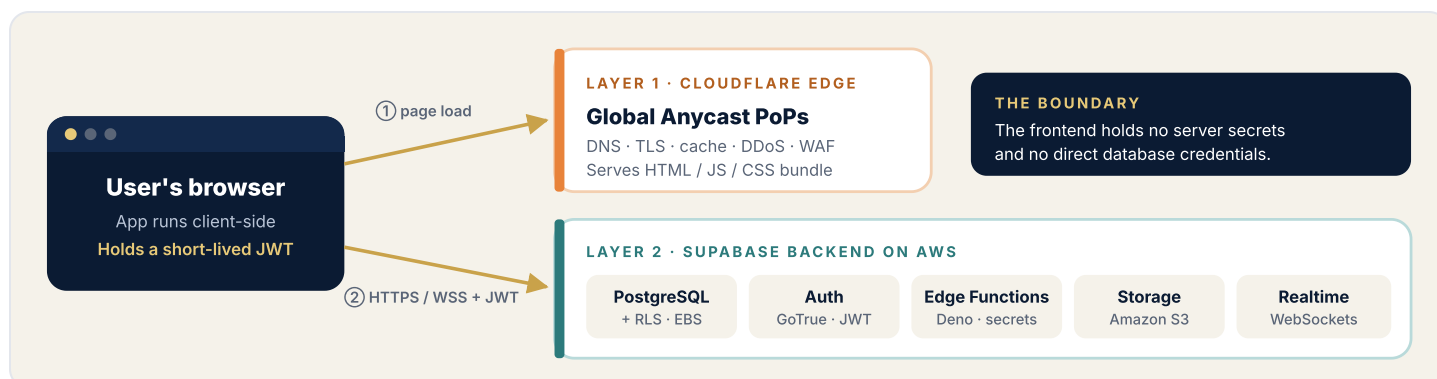
Row-Level Security enforced inside PostgreSQL

0 secrets

Server credentials never shipped to the browser

AT A GLANCE

Two layers, one clear security boundary



1 Frontend & Edge Layer

CLOUDFLARE

Handles global traffic delivery, security, and serving the client application bundle to browsers.

Client bundles & static assets

When an app is published, the compiled frontend build (HTML, JS, CSS, static images) is deployed to an edge storage tier and distributed across Cloudflare's global Anycast network.

DNS & routing

When users hit your domain or the platform's default subdomain, DNS resolves to the Cloudflare edge points of presence (PoPs) closest to the visitor.

SSL/TLS termination

Automated certificate issuance (Let's Encrypt / Google Trust Services) and TLS termination happen at the Cloudflare edge, before traffic reaches origin infrastructure.

Edge caching & security

Cloudflare caches static assets to minimize origin load, absorbs DDoS attacks, and applies basic web application firewall (WAF) filtering.

Execution boundary

Once loaded from the edge, the frontend runs **entirely client-side** in the user's browser. It holds **no server secrets** and **no direct database credentials**. Everything sensitive lives behind the backend layer.

WHY IT MATTERS

Authorization is decided by the database, not the browser. Every request carries a signed, short-lived token, and PostgreSQL Row-Level Security decides what each user can read or change. Server secrets stay in backend functions and never reach the client.





2 Backend & Database Layer

AWS VIA SUPABASE

The platform provisions a dedicated backend environment built on Supabase, whose underlying infrastructure runs on Amazon Web Services (AWS).

PostgreSQL database · Compute & EBS on AWS

- The core relational engine runs on AWS virtual compute instances backed by **encrypted Amazon EBS** storage volumes.
- Enforces all data constraints, transactions and **Row-Level Security (RLS)** policies directly at the database-engine level.

Authentication · GoTrue

- Handles user registration, password hashing, OAuth callbacks and token signing (JWTs).
- The browser receives **short-lived JWT access tokens** carrying user claims, passed in Authorization headers to authorize database queries and function calls.

Edge Functions / server compute · Deno on AWS

- Executes server-side business logic, third-party API calls and private integrations.
- Houses **platform secrets and service-role credentials** that must never be exposed to the browser.

Object storage · Amazon S3

- File uploads, profile avatars, documents and exported media reside in managed S3 buckets.
- Access is **gated by PostgreSQL RLS rules**, the same policies that protect the data.

Realtime engine

A cluster running on AWS manages WebSocket connections and broadcasts database changes (INSERT, UPDATE, DELETE) down to subscribed frontend clients, so screens update the moment the record does.

TRAFFIC FLOW

From first click to returned data

01

Initial page load

The user requests your domain. A Cloudflare edge PoP terminates SSL and returns the cached HTML/JS/CSS bundle.

Browser → Cloudflare edge

02

API & data requests

The browser-resident app makes HTTPS/WSS requests directly to the Supabase endpoint on AWS, carrying the authenticated user's session JWT.

Browser → Supabase on AWS

03

Authorization & execution

The AWS backend validates the JWT, evaluates RLS policies in Postgres or runs server-side logic in an edge function, and returns the data payload.

JWT → RLS / function → response

WHERE THINGS LIVE

Responsibility by layer

Concern	Frontend & Edge (Cloudflare)	Backend & Database (AWS via Supabase)
Application code	Compiled client bundle, cached globally	Server-side logic in Deno edge functions
Encryption in transit	TLS terminated at the edge PoP	HTTPS / WSS directly to the Supabase endpoint
Identity	Carries a short-lived JWT only	GoTrue issues and signs JWTs; backend validates them
Data access control	None held client-side	Row-Level Security in PostgreSQL; S3 gated by RLS
Secrets & credentials	Never present	Held server-side in edge functions
Attack mitigation	DDoS absorption and basic WAF filtering	Engine-level constraints and policies

This overview describes the hosting architecture at a summary level. Infrastructure providers, certificate authorities and service components are subject to change. Contact Sovereign for current security documentation.

