Harper vs. Vercel + Supabase

Two routes to a modern application stack. Two vendors stitched across the public internet, or one runtime that fuses database, cache, application logic, and messaging into a single Node.js process, deployed globally as a peer mesh. Where each one wins.

harper

One Runtime. One Vendor. Global Mesh.

One in-memory Node.js process holds the data, the cache, the app logic, the message broker, and the vector index. Every API call is a function call. Every node is a peer.

Best when latency, global scale, lower TCO, on-premise control, or long-running agent workloads are first-class concerns.

Vercel + supabase

Two Planes. Two Vendors. Regional by Default.

Vercel functions for compute; managed Postgres with Auth, Storage, Realtime, and pgvector for data. A first-party marketplace integration handles billing and environment wiring.

Best when the team's existing fluency with Next.js, Postgres, and the Vercel deployment model is the real accelerator — not the stack itself.

	Harper	Vercel + Supabase
Architecture	One fused process. Data, cache, app logic, and messaging in one Node.js runtime per node.	Two planes. Functions for compute, Postgres for data, connected over the public internet.
Latency	Down to <1ms in-process data retrieval. 50 ms edge to client. ~100 ms global replication.	Edge functions ~106 ms P50. Node serverless warm ~246 ms, cold ~859 ms. Cross-region DB calls 500 ms–6 s.
Cold starts	None. Persistent in-memory process; logic and data co-resident.	Mitigated by Fluid Compute. Still material on Edge Functions and infrequent Node routes.
Data model	NoSQL document store. Key value store. Blob storage as a native field type. HNSW vector index with embeddings generated in-process.	Relational Postgres with JSONB. pgvector for embeddings. S3-compatible storage as a separate service. Embedding generation requires an external model API.
Real-time & scale	Native MQTT, WebSockets, SSE on the same Resource API. Horizontal mesh — every node reads, writes, and replicates peer-to-peer.	Realtime via Postgres logical replication over WebSockets. Vertical Postgres scale with read replicas on paid tiers.
Deployment	Fully managed with Harper Fabric. Deploy on your own infrastructure with Harper Pro.	Vercel serverless on AWS-backed regions. Supabase managed on AWS in ~13 regions.
Operational surface	One vendor, one platform, one log stream.	Two vendors, two dashboards, two billing systems, two log surfaces to correlate.

0 brokers to run

`MQTT, WebSockets, SSE native`

Real-time delivered by the same runtime that holds the data. Supabase Realtime covers Postgres change feeds. Anything beyond that means adding Ably, Pusher, or your own broker.

0 caches to invalidate

`Reads hit the same process`

Data lives in the runtime, so reads are already cache speed. No Redis, no Upstash, no invalidation logic, and no stale-cache bug at 2am.

1 vendor

`Database, cache, messaging, vectors`

Storage, retrieval, real-time, and embeddings in one runtime. Vercel + Supabase covers two of those. The rest are separate services, separate contracts, separate bills.

START FREE www.fabric.harper.fast

harper.fast · fabric.harper.fast · github.com/HarperFast/harper

Three things Harper does that the other stack can't.

The runtime difference shows up where the agentic era pushes hardest: agents that don't terminate, workloads that can't leave the building, and builders who aren't engineers. Each one breaks the function-plus-managed-database model. Each one is native to a unified runtime.

01 · LONG-RUNNING AGENTS

Durable agent workloads, not request/response stubs.

Vercel functions have execution time limits that reliably kill multi-step agents running 10–15 minutes for database queries, document summarization, and report generation. Supabase Edge Functions have similar Deno isolate constraints, plus a recursive call rate limit.

HARPER

Long-lived Node process. Agents run as durable workloads alongside the data they reason over, with MQTT, WebSockets, and in-process pub/sub for agent-to-agent messaging — no external broker, no polling Postgres for state.

02 · ON-PREMISE

Deploy anywhere. Same runtime, same security model.

Vercel and Supabase are SaaS. The architecture assumes data flows through a vendor-controlled cloud, with "regions" as the only knob to turn. For regulated industries, sovereign workloads, air-gapped facilities, or customer-hosted deployments, that's the end of the conversation.

HARPER

Harper Fabric for fully managed. Harper Pro to deploy the entire platform — database, APIs, runtime, vector search, agent infrastructure — on a public cloud region, a private data center, a telecom edge node, or an air-gapped government facility.

03 · CITIZEN DEVELOPERS

Enterprise governance the platform actually enforces.

Most citizen-developer platforms force a trade-off: easy tools that fail security review, or sanctioned tools so locked down the agility evaporates. The defaults are the controls, and the defaults are wrong.

HARPER

Citizen-built apps inherit the same security posture as Harper's production workloads at Verizon, Red Hat, and Western Union. One auth model, one deployment surface, one boundary to defend. IT can say yes without giving up the levers.

At-a-Glance: The Differences That Matter for Agents

Dimensions that don't fit on a generic feature matrix — but decide procurement.

	HARPER	VERCEL + SUPABASE
Agent Execution	Long-lived process. Durable workloads. Native A2A messaging.	Request/response, time-capped. Multi-step agents hit 504s at scale.
Credential Surface	Declarative config.yaml + schema.graphql. Agents commit files, not credentials.	Vercel project tokens, Supabase service-role keys, DB connection strings, env vars across dashboards.
Sovereign Deployment	Public cloud, private data center, edge node, or air-gapped — same runtime.	Vendor-controlled cloud. Region as the only knob.
Citizen-Dev Posture	One auth + RBAC model inherited by every app. Governance is a platform property.	Per-service auth. Governance bolted on per app, per integration, per dashboard.
Edge AI Inference	TensorFlow.js, ONNX, Transformers.js, Ollama in-process. Sub-50ms inference against co-located data.	No native GPU. Inference pushed off-platform. pgvector stores, but model execution lives elsewhere.
Open Source	Harper 5.0 core is Apache 2.0.	Supabase OSS. Vercel proprietary; Next.js features (ISR, server actions) are the lock-in.

START FREE www.fabric.harper.fast

harper.fast · fabric.harper.fast · github.com/HarperFast/harper