

Decisions · Platform

ADR-0011: The API runs as Vercel functions inside the web project (staging first)

🕒 4 min read · Updated 13 October 2026

Why the API runs as Vercel functions inside the web project on staging, and how production will follow.

<https://docs.gurbanisoul.com/adr/0011-api-as-functions-in-the-web-project/>

Status: accepted (2026-09-25) — live on staging; production moves onto `all` (the whole API as one function) in the release after 1.3.9, with the `Render` single API kept deployed as its rollback target. *Amended 2026-09-26:* see Amendments. Supersedes the *hosting* part of ADR-0010 (per-context `Render` services); its slicing, generated routing and proofs stand.

Context

Running the five contexts as always-on `Render` services in production would cost about US\$35 a month; free `Render` instances sleep and share a 750-hour monthly cap, so they cannot serve production. The web is already on `Vercel Pro`, which runs Python functions on usage-based pricing out of the plan's monthly credit. The API is `stdlib` Python over a read-only `SQLite` file, and `serve.H` is a `BaseHTTPRequestHandler` — the handler shape `Vercel`'s Python runtime loads directly.

A trial as a *separate* `Vercel` project (`sggs-api`) passed the whole golden contract but showed two things: `Vercel`'s platform flood protection answered one burst-tested host with a browser challenge (which an API client cannot pass), and a project without a framework publishes every non-function file — the databases included — as static files. A second project would also put a second edge and firewall between the web and the API.

Decision

- **Functions in the web project.** At deploy time `tools/build_api_functions.py` generates, under `frontend/`: one function per context (`api/svc/<ctx>.py`) and `all` (the whole API on the full database), each fixing `SGGS_MODULES`, `SGGS_DB` and the commit, refusing to load when its slice lacks a declared table, and serving with `serve.H` unchanged. Slices are cut and proven by `tools/slice_db.py`. Nothing generated is committed. Every CI deployment carries all six functions.
- **Routing stays generated from the code.** `tools/gen_gateway.py` writes `frontend/vercel.json`'s rewrites; the function configuration (`bundles`, `region`) is added at deploy time together with the generated files, so a build without them — a `Vercel Git` preview — still succeeds. Per environment (`gateway/routes.json`),

`api_platform`: `vercel` rewrites each routed context's prefixes (and `/api/v1` twins) to its function and everything else to `all`; `api_platform`: `render` proxies `/api` to the Render single API. Rewrites are internal (one edge, one firewall), use unnamed groups so the handler sees the original path and query, and are host-conditioned for staging.

- **What ships is proven before it deploys.** After `vercel build`, `--verify-output` reads every function's bundled file map: its entry, the API code and its own database only — anything else from the web, or another context's database, fails the build — and no database or API source may appear in the static output. After deploy, `scripts/ci/verify_functions.py` checks every function's `/readyz` (`commit`, `contexts`, `X-Service`) and that the API's files are not downloadable; the gateway probes and the whole golden contract (266 records) replay through the staging site.
- **Region `pdx1`**, next to the Render API it replaces, so latency compares like for like.

Consequences

Staging no longer uses Render; the `sggs-staging-*` services, `sggs-api-staging` and the `RENDER_API_KEY` / `RENDER_DEPLOY_HOOK_STAGING` secrets can be removed. Web and API deploy atomically (no version skew; one rollback). Functions scale to zero after a few idle minutes, so the first request after a quiet spell pays a cold start (about a second in the trial); the p95 budget excludes cold starts, as ADR-0010's does. Vercel runs Python 3.12 with SQLite 3.40.0 (FTS5, trigram, bm25) — the Render image ran 3.40.1 — and the contract passed byte-for-byte on it. Moving production is a change of `production.api_platform` to `vercel`, released through the normal pipeline and gated on the performance baseline and the data canary (`docs/process/runbooks/services-production.md`).

Amendments

- **2026-09-26 — production's contexts move in one release.** Production went onto `all` with 1.3.10. Its per-context routing (`services-production` step 3) was merged and then held back (owner decision 2026-10-11: `production.services` in `gateway/routes.json` is empty) and goes live in **one** release for all five, a later one after the app is live on the App Store (1.4.0 keeps `all`): under the one-version policy every release is also an App Store update (owner decision). Each context is still gated on its own — `verify_functions --env production`, the golden contract before promotion, the performance gate per context — and rolls back alone by leaving the list. The Render single API stays deployed as the rollback until `services-production` step 4. The decision above is unchanged.