marcle.ai - One-Page App Summary

Evidence source: README.md, docker-compose.yml, frontend/*, backend/app/*

What it is

A self-hosted web app that combines a public homelab status dashboard with an authenticated Q&A app at /ask. It serves static frontend pages through nginx and proxies API traffic to a FastAPI backend.

Who it’s for

- Primary persona: a self-hosting homelab operator (owner/operator context appears throughout README and frontend copy).
- Secondary users: authenticated Ask users who submit questions and receive email answers.

What it does

- Publishes service health states (healthy, degraded, down, unknown) via /api/status and a live dashboard.
- Tracks incidents and exposes overview/service detail APIs with cache age, last incident, and history.
- Runs concurrent background checks on configured services with check-type profiles and auth references.
- Provides admin APIs/UI to create/update/toggle services, bulk enable/disable, and view audit logs.
- Supports configurable outbound notifications with endpoint filters and test dispatch.
- Implements Google OAuth login for Ask, per-user points, and rate-limited question submission.
- Posts Ask questions to Discord webhook and accepts secure answer webhooks that email users.

How it works (architecture)

- Frontend container (nginx): serves /, /admin, /ask static pages; proxies /api/* and /healthz to backend:8000.
- Backend container (FastAPI): status + admin endpoints in app.main, Ask routes in app.routers.ask.
- Data layer: JSON files for services, observations, notifications, and audit log under /data; SQLite for Ask DB.
- Flow: browser polls status/overview -> backend serves cached payload from refresh loop -> observations update.
- Ask flow: Google OAuth callback -> session cookie + SQLite user record -> question insert and points decrement -> Discord webhook -> answer webhook -> email send.
- Dedicated distributed session store/queue worker: Not found in repo (sessions and rate limits are in-memory).

How to run (minimal)

- Copy env template: cp .env.example .env
- Set required values in .env (at minimum ADMIN_TOKEN to enable admin; Ask vars for OAuth/Discord/SMTP if used).
- Start containers: docker compose up --build
- Open: http://localhost:9182 (status), /ask (Ask app), /admin (admin panel).