AGENTIC DEV DAYS · STOCKHOLM · 2026

Vibe engineering taught me Rust.

Building Aura — an agentic LLM gateway — with Claude Code

Marcus Elwin

AI Lead @ Kustom · AI Blogger @ UmaiTech

WHO AM I

Marcus Elwin

- AI Lead @ Kustom — agentic commerce at scale
- OSS @ UmaiTech — blog & open source stuff
- 10+ years in AI — retail, fintech, legal tech
- KTH alum — DS → ML → PM → AIEng→ AI Lead

FIND ME AT

Personal blog

LinkedIn

AGENDA

What we'll cover

01 **Vibe Engineering** >> Vibe Coding

02 **Rust vs Python vs TypeScript** — for AI engineering

03 **Meet Aura + live demo** — an agentic LLM gateway in Rust

04 **Rust learnings** — what surprised me, what hurt

05 **What's next** — for Aura & for the space

~20 min · questions welcome at the end

01

Vibe Engineering

The discipline behind the vibes.

```
$ git log --oneline
a8f3d2 feat: add routing
c4e1b7 test: cover fallback
9d2a0e docs: update PRD
1b6f5c chore: scaffold
```

Same tools. Different discipline.

VIBE CODING

× Prompt & hope

× One giant PR

× No plan, just flow

× Trust the AI

× Skip the tests

× Language-agnostic guessing

Fast for throwaway demos.

>>

VIBE ENGINEERING

✓PRD first, prompt second

✓Bite-sized commits

✓Architecture diagrams

✓Verify, don't trust

✓Tests as a contract

✓Expertise still matters

What ships to production.

I know which house I would live in.

Six principles I keep returning to

01

Plan before you prompt

Write the PRD.
Then let the AI code.

02

Bite-sized PRs

Easier to review,
verify, and revert.

03

Draw the system

Mermaid diagrams make
patterns visible.

04

Keep it simple

MVC beats design-pattern
gymnastics every time.

05

Know your language

AI won't rescue you from
unfamiliar territory.

06

Mix your tools

Research in one, plan in
another, refine in a third.

Full list: 10 principles on ***@MarqKwesi***

“

Taste still matters in AI software engineering.

→AI generates. You decide what ships.

→Code taste = architectural judgment you can't prompt for.

→The bottleneck moved from typing to deciding.

umai-tech.com/blog/taste-still-matters-in-ai-software-engineering

02

Languages

Rust, Python, TypeScript — pick your poison.

```
// which tool for which job?

match use_case {
  HotPath ⇒ Rust,
  Research ⇒ Python,
  Agents ⇒ TypeScript,
```

Rust vs Python vs TypeScript

Honest tradeoffs, not a winner.

🦀 Rust

Performance	●●●
Type safety	●●●
AI ecosystem	●○○
Learning curve	●●●

USE FOR

Gateways, inference, hot-path infra, CLIs

"Rust won't let you ship a race condition."

🐍 Python

Performance	●○○
Type safety	●○○
AI ecosystem	●●●
Learning curve	●○○

USE FOR

ML training, research, prototypes, data work

"Every AI library lives here first."

🔷 TypeScript

Performance	●●○
Type safety	●●○
AI ecosystem	●●○
Learning curve	●●○

USE FOR

Agents, web apps, full-stack, edge

"Where LangGraph.js and Mastra live."

When to reach for what

🦀 Rust when...

hot path · <10ms overhead · single-binary deploys · high concurrency · memory-safe without GC

🐍 Python when...

training · research · heavy ML libs · rapid iteration · pipeline glue · Jupyter-shaped problems

🔷 TypeScript when...

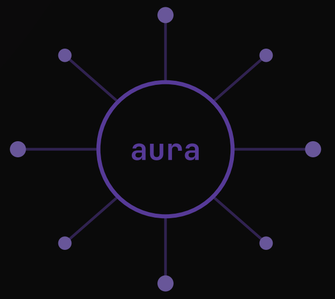
agent frameworks · web apps · typed full-stack · browser runtime · edge deployment

The honest answer is usually "all three".

03

Meet Aura-LLM

Built in Rust. Because latency matters.



Aura LLM Gateway

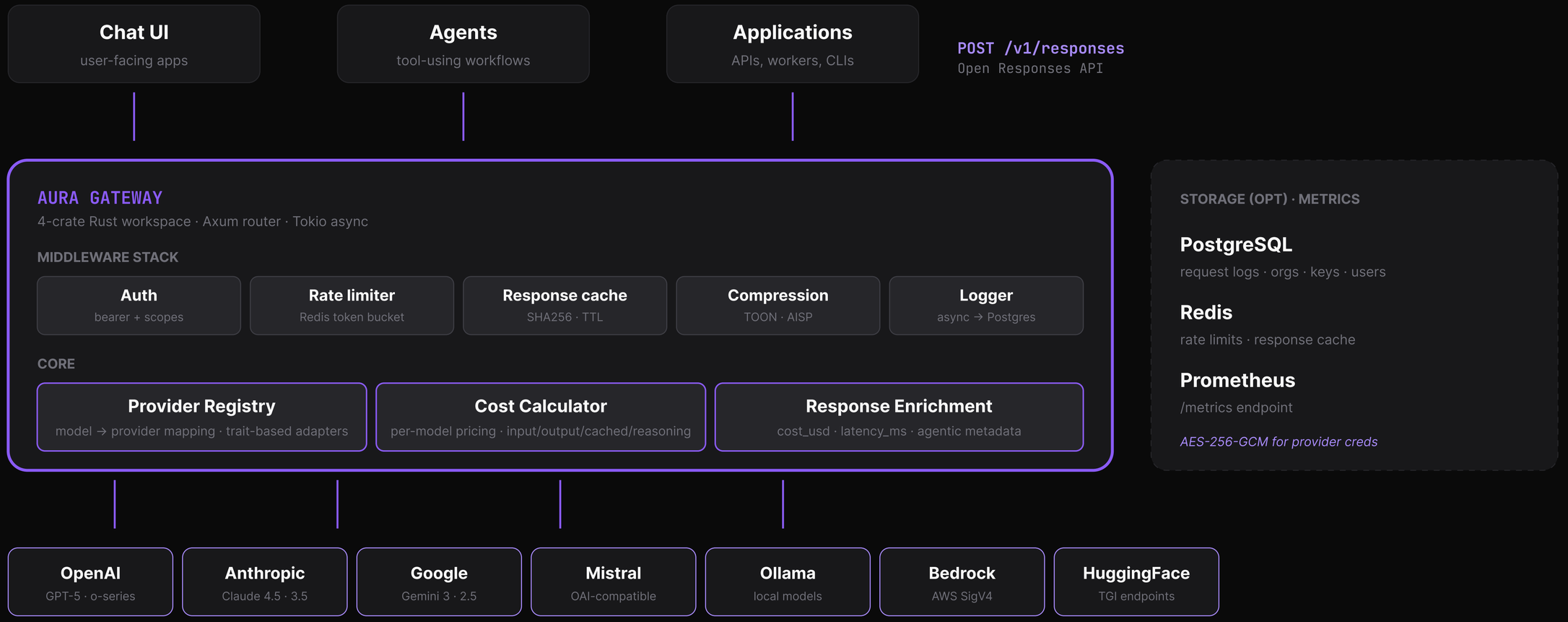
An agentic router for the Open Responses API era.

- ✓ **Open Responses API — not just OpenAI-compatible; agentic-native spec**
- ✓ **7 providers — OpenAI, Anthropic, Google, Mistral, Ollama, Bedrock, HF**
- ✓ **Agentic metadata — tool calls, reasoning items, requires_action**
- ✓ **Cost tracking — automatic per-request USD on every response**
- ✓ **Multi-tenant — org / team / project / end-user hierarchy**
- ✓ **Encrypted credentials — AES-256-GCM envelope encryption**
- ✓ **Rate limit + cache — Redis-backed token bucket & TTL cache**
- ✓ **Prompt compression — TOON, AISP (40–60% token savings)**

Rust	Axum	Tokio	SQLx	Redis	Prom.
LANGUAGE	WEB FRAMEWORK	ASYNC RUNTIME	POSTGRES (OPT)	CACHE (OPT)	/METRICS

Architecture

Axum router · provider registry · cost calculator · pluggable storage



Rust, without being a Rust dev

I'm a Python person. I built a Rust gateway with **Claude Code**.

WHAT WORKED

+Compiler errors are a teacher, not a wall

+Types catch provider schema drift early

+Single static binary, no runtime deps

+Tokio async handles SSE streams cleanly

+Claude Code is fluent in idiomatic Rust

+Refactors feel safe — the compiler has my back

WHAT HURT

—Borrow checker + async = pain spikes

—Lifetimes took weeks to internalize

—Crate ecosystem thinner for AI work

—No `pip install` shortcuts

—Compile times break flow state

—Debugging async traits is a trip

Claude Code didn't replace Rust knowledge. It made it reachable.

03 · MEET AURA

Live demo.

One endpoint, three providers, full agentic metadata.

aura · /v1/responses

```
$ curl -X POST localhost:8080/v1/responses \
  -H "Authorization: Bearer $AURA_KEY" \
  -d '{
    "model": "claude-sonnet-4-5",
    "input": [{
      "role": "user",
      "content": "Search the web for..."
    }],
    "tools": [{"type": "web_search"}],
    "user": "customer_123"
  }'
```

WATCH FOR →

· Provider resolved from model name (no routing config)
· Response enriched with cost_usd, latency_ms, agentic{}

WHAT AURA ADDS

USAGE + COST

```
input_tokens: 1,842
output_tokens: 318
cost_usd: 0.00732
```

AGENTIC METADATA

```
provider: "anthropic"
latency_ms: 523
has_tool_calls: true
tools_used: ["web_search"]
requires_action: false
request_id: "aura_550e..."
```

Every response. Every provider. Same shape.

04

Learnings

What I'd tell past-me.

Six things I'd tell past-me

#1 Pick languages by latency budget, not hype

<10ms overhead? Rust. Else, ship in what your team knows.

#2 Observability first, features second

You can't fix what you can't see. OTEL from day one.

#3 Typed provider schemas save you at 3am

LLM APIs drift. Strong types catch it before users do.

#4 Build fallback chains before you need them

Every provider goes down. Retry & switch is table stakes.

#5 Cost tracking is a product feature

Not a nice-to-have. Surface it to users and PMs.

#6 Vibe code prototypes. Vibe engineer production.

Different modes. Different rigor. Don't confuse them.

What's next for Aura

- → Multi-node load balancer
- → Automated pricing scraper (cron)
- → Webhooks & async callbacks
- → Admin dashboard (React UI)
- → More providers via the trait system

Thank you.

Questions? Let's talk after.

WRITE-UP	LINKEDIN	X / TWITTER	REPO
umai-tech.com	linkedin.com/in/MarcusElwin	@MarqKwesi	UmaiTech/aura-llm-gateway