

DATA OUTPOST 2026

Building an analytics agent



Colton Padden

Member of Technical Staff, Vercel

@coltonpadden @cmpadden /in/colton-padden

Agenda

PART 1 · 40 MIN

Agent fundamentals

Building a harness

Agents at Vercel

Building with eve

Production grade agents

Demo

PART 2 · 80 MIN

Hands-on workshop

The company brain template

Connecting to Slack

Exploration

SOMEONE ASKS IN #ANALYTICS

“How did revenue do last month?”

∑ METRIC DEFINITIONS

“Revenue” can mean recorded payments or payments net of returns. The definition belongs in your dbt docs.

DATE RANGES

The data may end before today. The agent has to say when the requested period isn't available.

PERMISSIONS

A Slack thread has many people, and some of them can't see every table.

TRUST

Queries are read-only, and evals test the agent's behavior before each change.

Building a harness

har·ness /ˈhɑːr.nəs/ *noun*

1. A set of straps by which a horse is fastened to a cart. *Not this one.*

2. *AI* · also agent scaffolding

   An agent harness, also known as agent scaffolding, is the software infrastructure surrounding a large language model (LLM) that enables it to operate as an AI agent. It manages tool use, memory, state persistence, execution environments and feedback loops, as opposed to the model's internal reasoning.

SOURCE: WIKIPEDIA, “AGENT HARNESS” ↗

What you customize



INSTRUCTIONS

How it interprets your metric definitions and date ranges.



TOOLS

Typed functions: list models, read a definition, run read-only SQL.



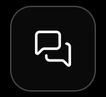
SKILLS

Procedures loaded only when a question needs them.



CONNECTIONS

The services it may reach, with credentials it never sees.



CHANNELS

Where people ask: Slack, Teams, the web.



POLICY

Who may ask, what needs approval, what's turned off.

When someone corrects the agent

	MEMORY	SEMANTIC UPDATE	CODE CHANGE
EXAMPLE	“I report in fiscal months”	“Revenue excludes returns”	“Add a cohort analysis tool”
LIVES IN	`agent/memory/`	dbt docs and metadata	Instructions, skills, tools
APPLIES TO	One person	Everyone, every tool	Everyone
REVIEW	None, applies next session	The data team, through a PR	A draft PR from the agent

A semantic update

#analytics

Bob 11:20 AM

Revenue should exclude returned orders.

company-brain Agent 11:21 AM

That's a metric definition, so it belongs in dbt. I opened a draft PR for the data team.

⇄ **Define net_revenue on orders**

acme/analytics#57 · draft, ready for review

dbt/dbt_project.yml

```
  orders:
    +meta:
      metrics:
        amount: "All recorded payments, regardless of status."
+       net_revenue: "SUM(amount) excluding returned and
+         return_pending orders. Use for “revenue”."
```

Same PR · evals/revenue-excludes-returns.eval.ts

```
+ await t.send("How did revenue do in March 2018?");
+ t.judge("Excludes returned orders from revenue.");
```

After the merge, the next data build regenerates the agent's catalog. Every tool and every person gets the new definition.

What the harness owns



CONTEXT

What the model sees each step, and compaction when it fills up.



EXECUTION

Running tools, retries, timeouts, and cancellation.



STATE

Sessions that survive crashes and redeploys.



IDENTITY

Who's asking, and whose credentials each call uses.



ISOLATION

A sandbox for code and commands, away from secrets.



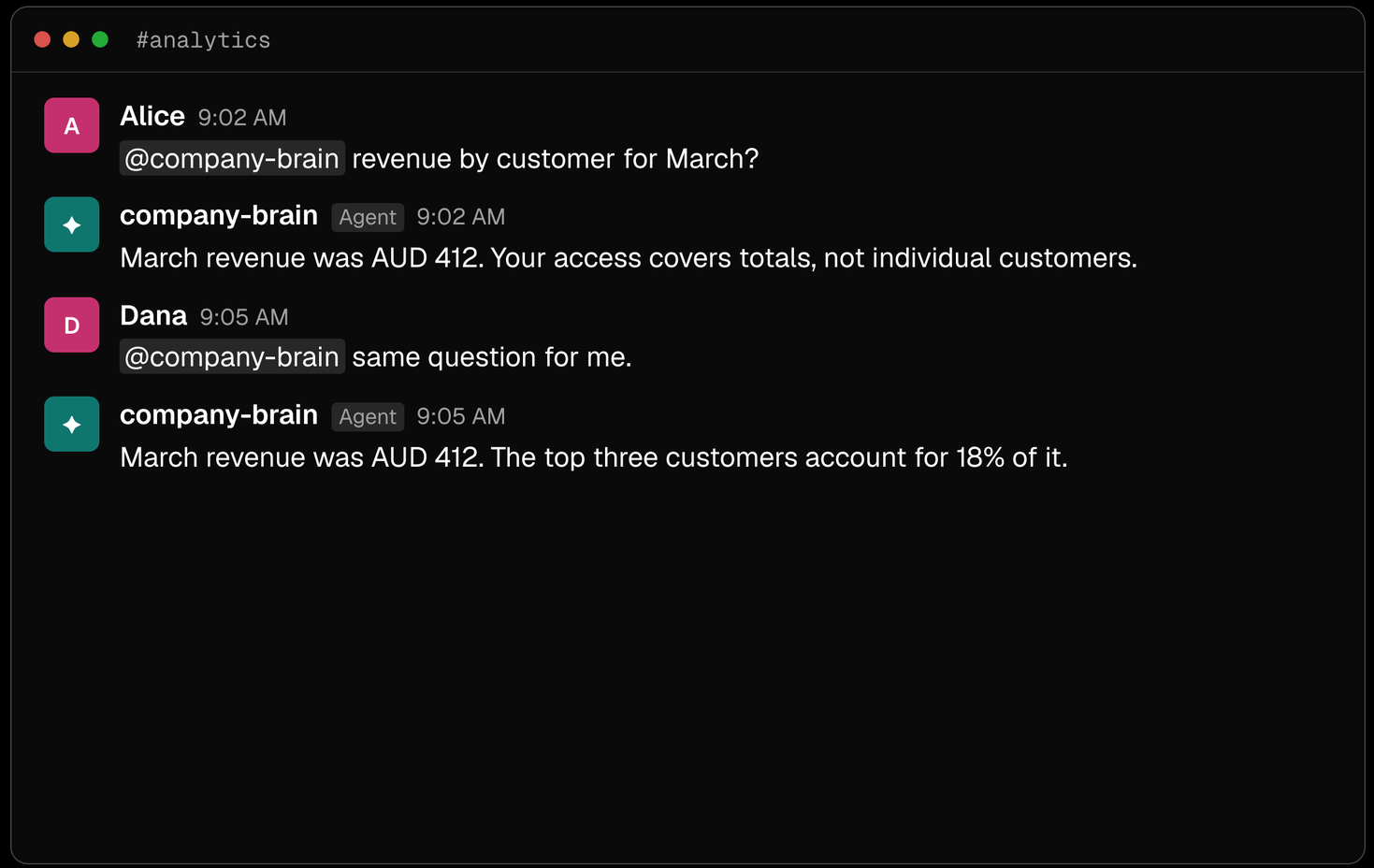
DELIVERY

Streaming events and posting replies to the right thread.

Laptop agent vs company agent

	ON YOUR LAPTOP	FOR YOUR COMPANY
USERS	You	Everyone in the thread
CREDENTIALS	Yours, on disk	Each caller's, brokered
COMPUTE	A process you watch	Serverless functions that can stop at any time
WAITING	Blocks your terminal	Hours or days, no compute
SURFACE	Terminal	Slack, Teams, web, schedules

Multiplayer



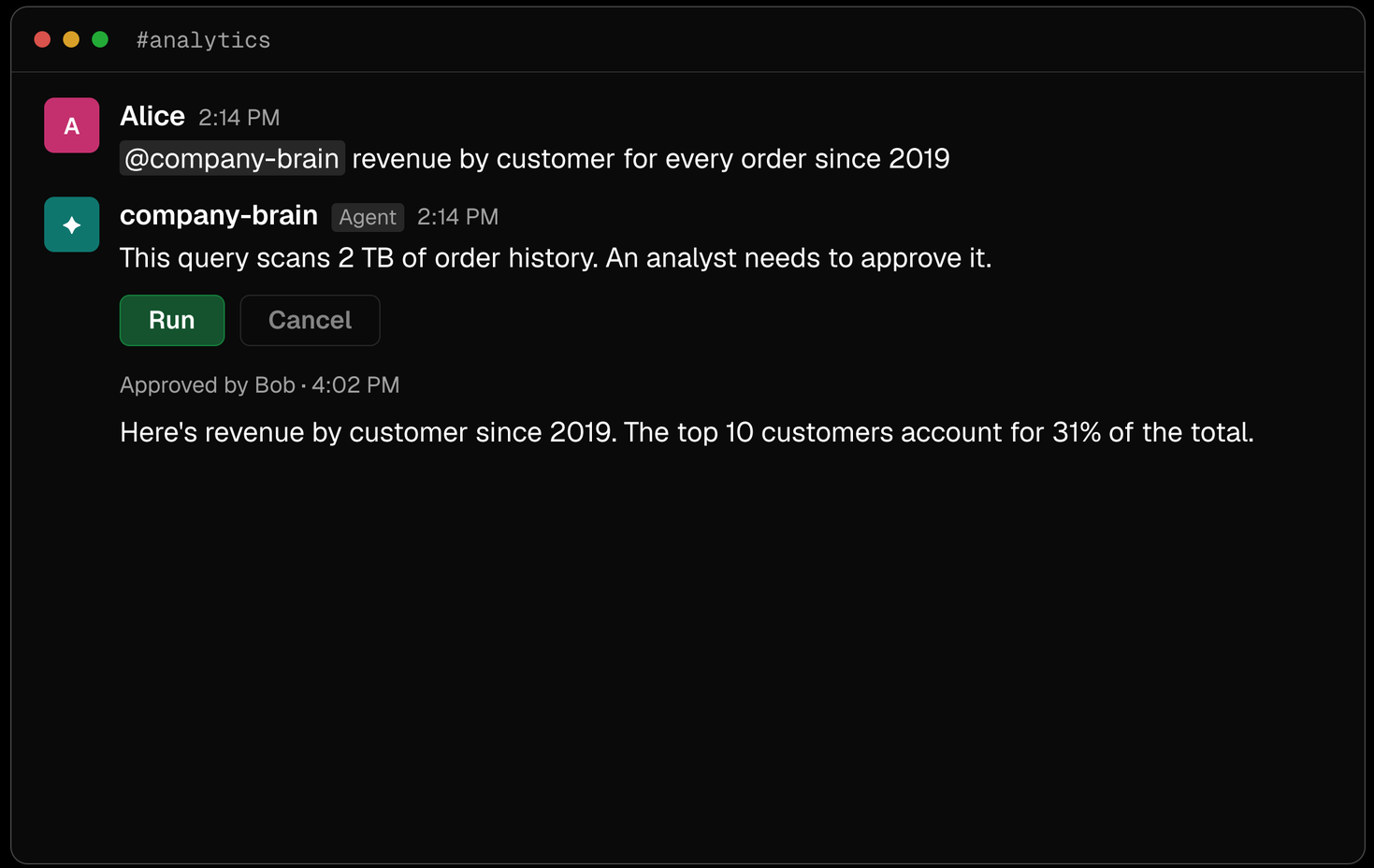
EACH TURN RUNS AS ITS SENDER

> Alice · Analyst
> `slack:T01:U_ALICE`
> `analytics_reader`

> Dana · CFO
> `slack:T01:U_DANA`
> `finance_reader`

One thread and one session. The harness picks credentials from the verified caller, and tokens never reach the model.

Human in the loop



1 · ASK

The agent asks for approval in the thread.

2 · PARK

The turn pauses. No function runs while it waits.

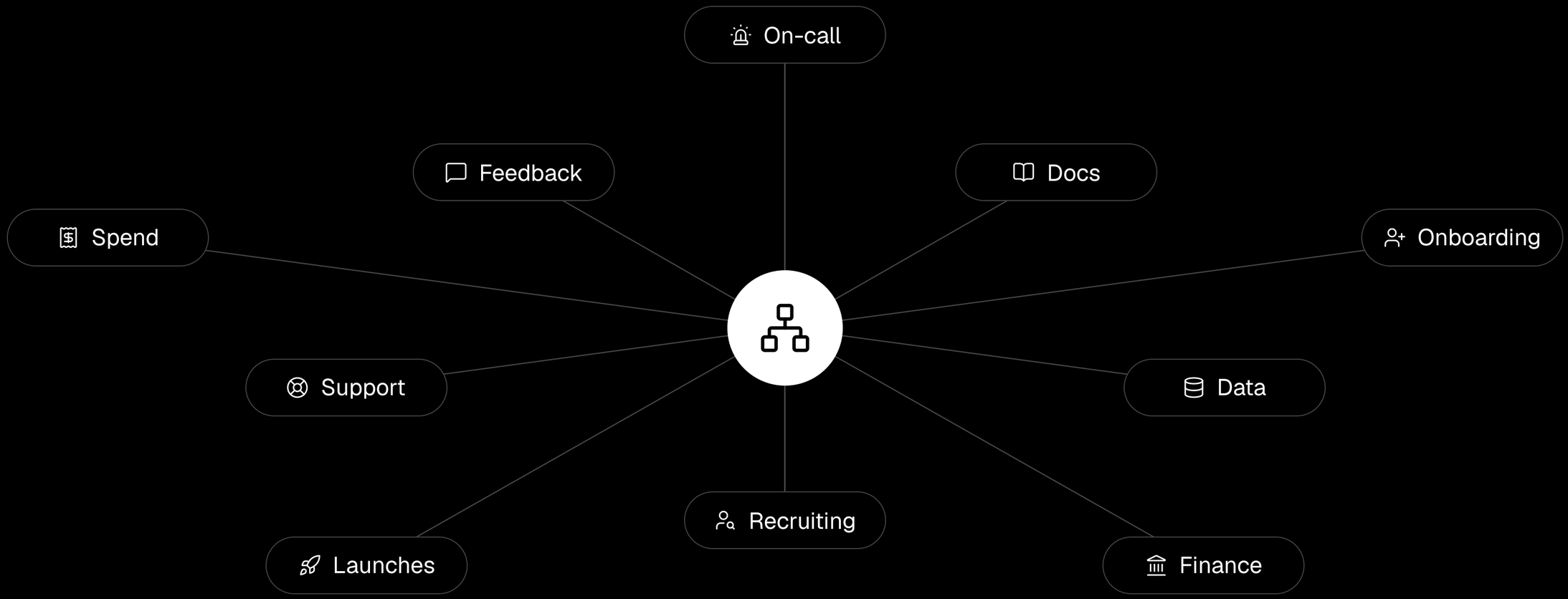
3 · ANSWER

Someone allowed to approve clicks Run, possibly hours later.

4 · RESUME

The session resumes on a new function and runs the query.

Adoption of agents



Lessons learned

CONFIGURATION AS FILES

Domain knowledge lives on disk and is reviewed like code.

MULTIPLAYER BY DEFAULT

A Slack thread has many callers, each with their own access.

DURABLE EXECUTION

Checkpoint every step, and wait without holding a function.

SHARED CAPABILITIES

Teams share tools and skills through extensions.

Types of agents

Types of agents

Company brain | Software factory | Agentic apps

A Slack agent that answers your team's business questions from your documented dbt models.

Answers in Slack

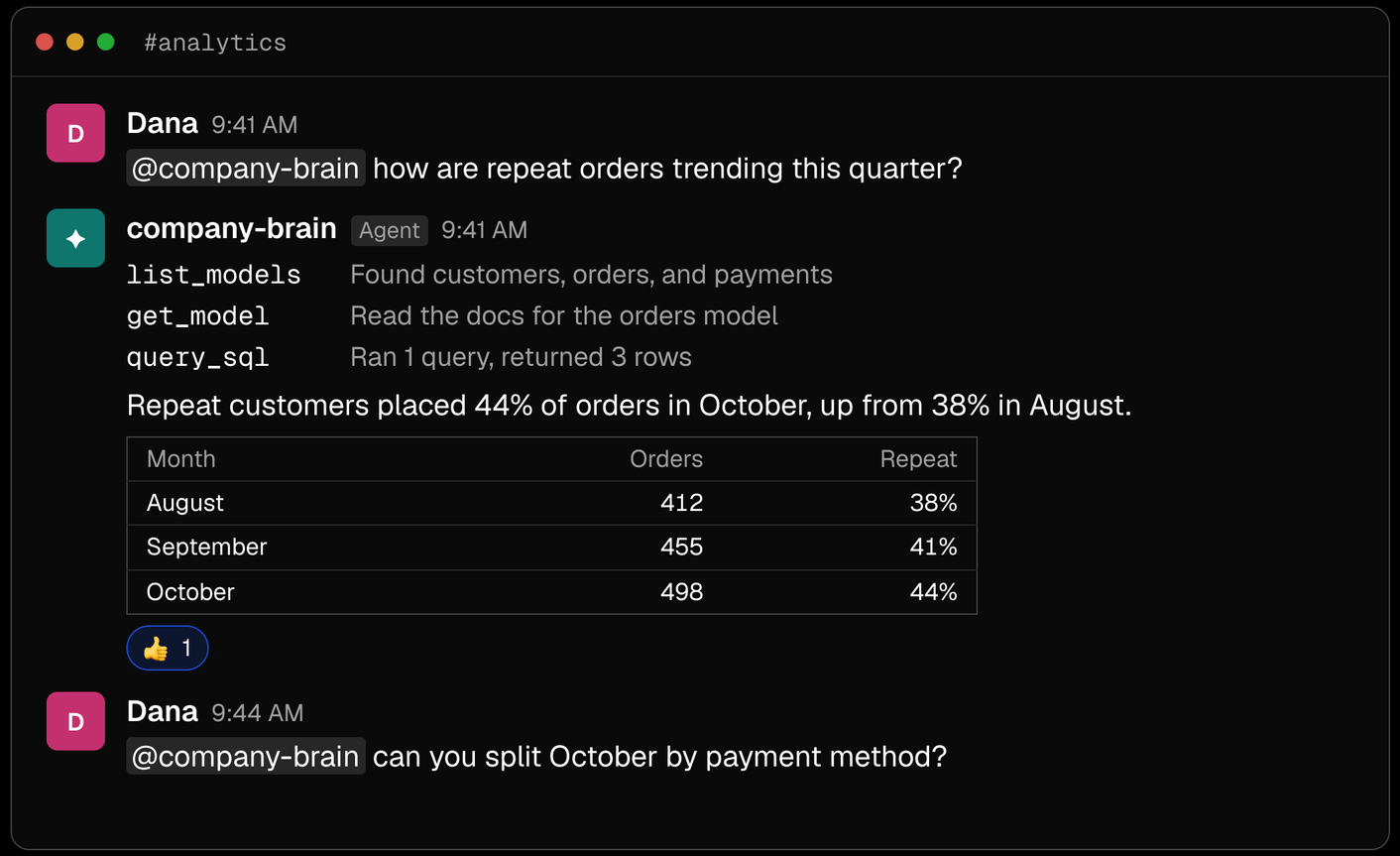
Only for the people and channels you allow.

Grounded in dbt

Reads your model docs before it writes a query.

Read-only SQL

Queries DuckDB or MotherDuck, and never writes to them.



Types of agents

Company brain | **Software factory** | Agentic apps

An agent that turns issues into reviewed draft pull requests, from triage and planning to implementation.

GitHub and Linear

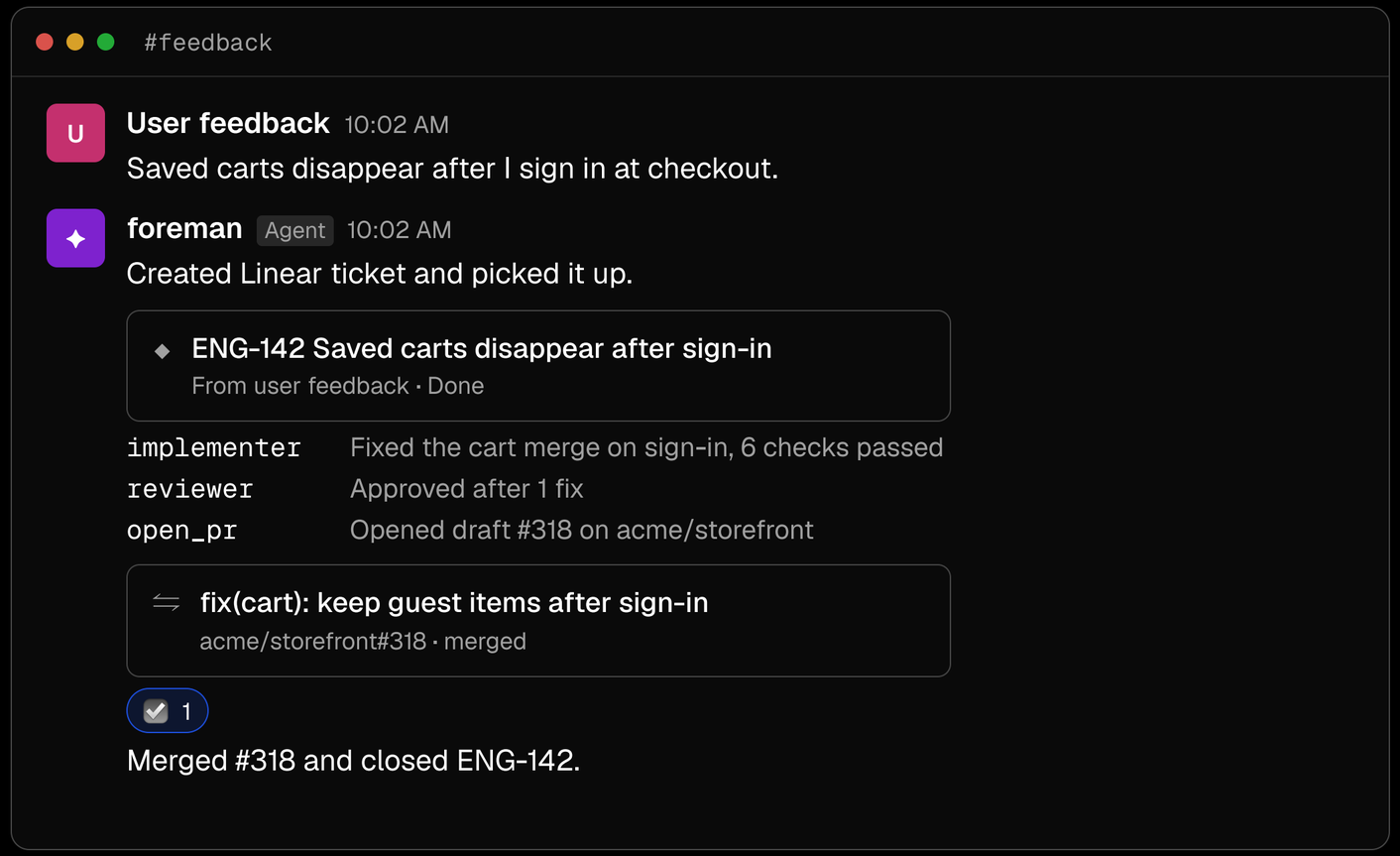
Picks up tasks where your team already files them.

Four stations

Classifier, analyst, implementer, and reviewer subagents work each task.

Draft pull requests

Every task ends in a reviewed draft on your repository.



Types of agents

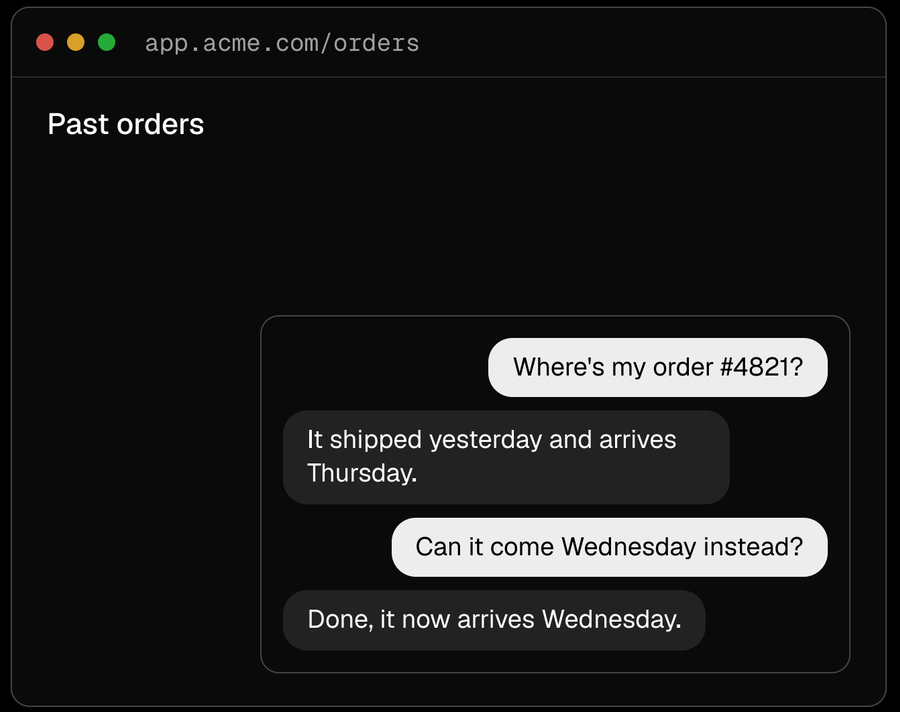
Company brain | Software factory | **Agentic apps**

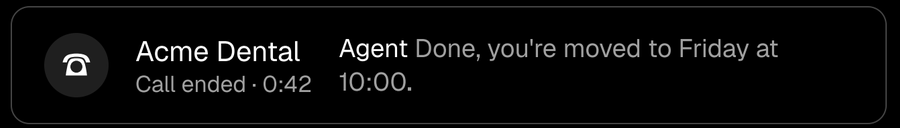
Agents inside your product, over web, voice, or iMessage, with memory for every user.

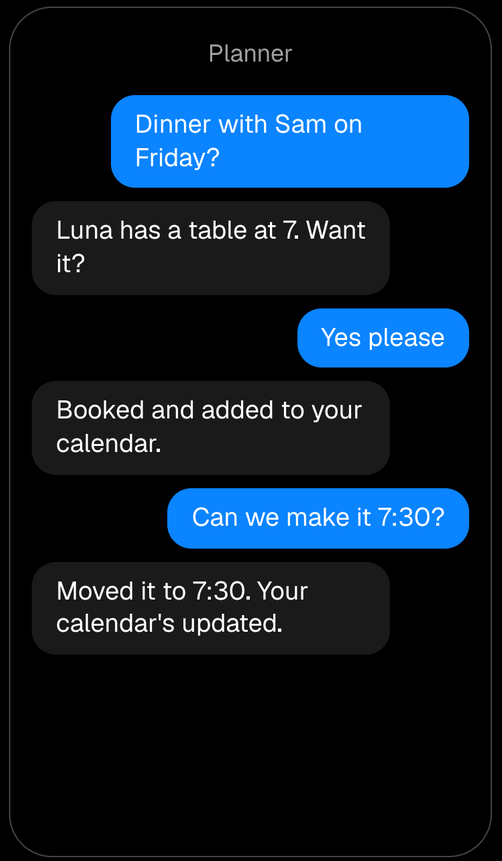
Web chat
Inside your app, with long-term memory for each user.

Voice
Calls that run tools and ask for approval.

iMessage
Threads that share memory with web and Slack.







Templates

vercel.com/templates/eve

Products Resources Enterprise Pricing

Contact Log In Sign Up

eve

Jumpstart your app development process with pre-built solutions from Vercel and our community.

Search templates... Filters

Foreman

eve Software Factory



eve Incident Response Agent

eve GitHub Agent

eve Software Factory

Software factory built on eve: AI agents work each stage of

eve Incident Response Agent

sre investigates production issues using a hypothesis-driven

eve GitHub Maintainer

GitHub maintainer eve agent for your inbox. Weekly issue

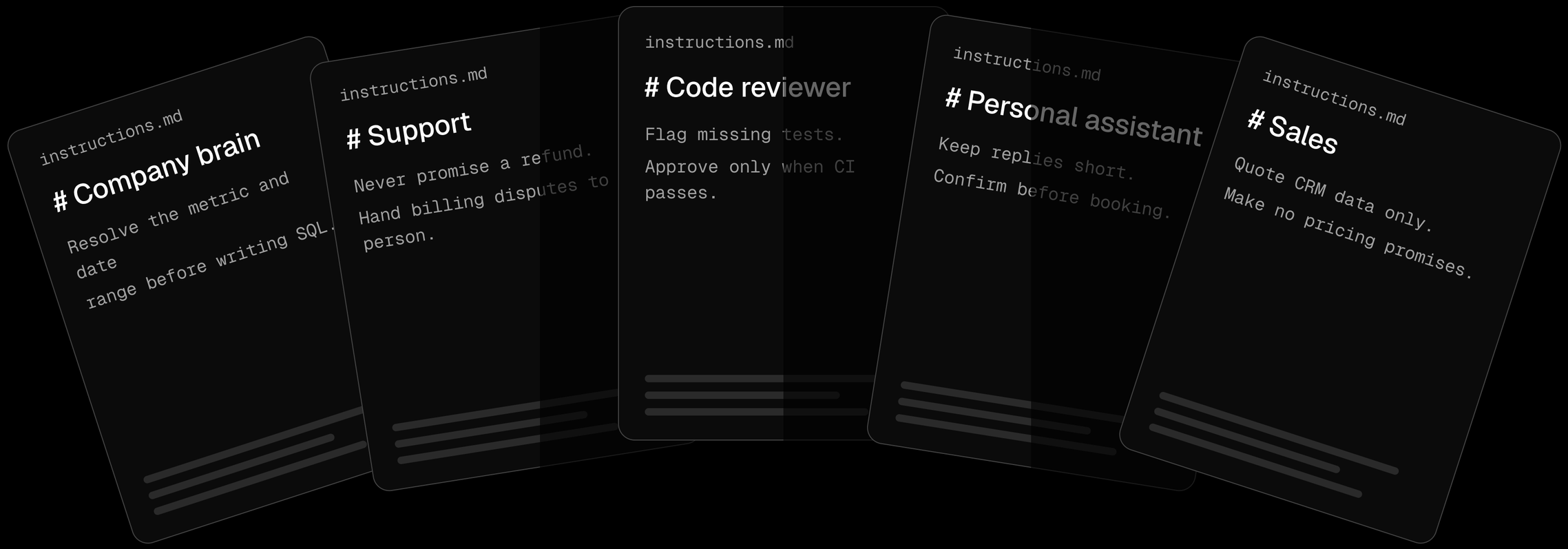
Building with eve

Building an eve agent

company-brain/agent/

agent.ts	the model and its settings	TYPESCRIPT
instructions.md	how to answer a business question	MARKDOWN
tools/	list_models, get_model, query_sql	TYPESCRIPT
skills/	procedures, loaded when needed	MARKDOWN
channels/	Slack, and the terminal	TYPESCRIPT
connections/	MCP and OpenAPI services	TYPESCRIPT
sandbox/	where commands run	TYPESCRIPT
schedules/	when it wakes up on its own	TYPESCRIPT

Instructions



instructions.md

agent/instructions.md · company brain, trimmed

You help employees answer business questions using company data. Translate everyday language into the relevant models, fields, and calculations yourself.

1. Use list_models, then get_model to read definitions.
2. Resolve the metric, row grain, date field, units, filters, and join cardinality before writing SQL. Ask a clarifying question when an ambiguous term would change the answer.
3. For date-sensitive questions, inspect MIN/MAX of the date field. Never silently substitute a historical period.
4. Answer from successful tool results only.

WHO IT SERVES

Employees who don't know the schema.

HOW IT REASONS

Read model definitions before writing SQL. Ask when a term like “revenue” is ambiguous.

WHAT IT WON'T DO

Answer for a period it doesn't have, or from a failed query.

Tools

- query_sql()
- send_sms()
- create_issue()
- create_refund()
- search_docs()
- open_pr()
- create_event()
- page_oncall()
- send_email()

Tools

agent/tools/query_sql.ts

```ts
import { defineTool } from "eve/tools";
import { z } from "zod";
import { requireModelAccess } from "../lib/access.ts";
import { querySql } from "../lib/query.ts";

export default defineTool({
  description:
    "Execute one read-only DuckDB SELECT against documented models.",
  inputSchema: z.object({ sql: z.string().min(1).max(16_384) }),
  async execute({ sql }, ctx) {
    requireModelAccess(ctx.session.auth.current);
    return querySql(sql, ctx.abortSignal);
  },
});
```

THE FILE NAME IS THE TOOL NAME

tools/query_sql.ts becomes query_sql.

TYPED INPUT

A Zod schema the model must satisfy.

POLICY IN CODE

Access checks and read-only SQL parsing run on every call.

Skills

- LOADED

monthly-revenue.md

description

Build the monthly revenue review for finance.

1. Confirm the month
2. Query orders
3. Report drivers

churn-analysis.md

description

Explain why customers cancelled.

cohort-retention.md

description

Compare retention across signup cohorts.

forecast.md

description

Project next quarter's revenue.

postmortem.md

description

Draft a postmortem from an incident thread.

board-metrics.md

description

Pull the metrics for the board deck.

agent/skills/

Skills

agent/skills/monthly-revenue.md

```
---
description: Build the monthly revenue review for finance.
---

1. Confirm the month, and whether returns are excluded.
2. Query orders.amount by status for that month and the
   month before.
3. Report the total, the change, and the top three drivers.
4. Note any days missing from the data.
```

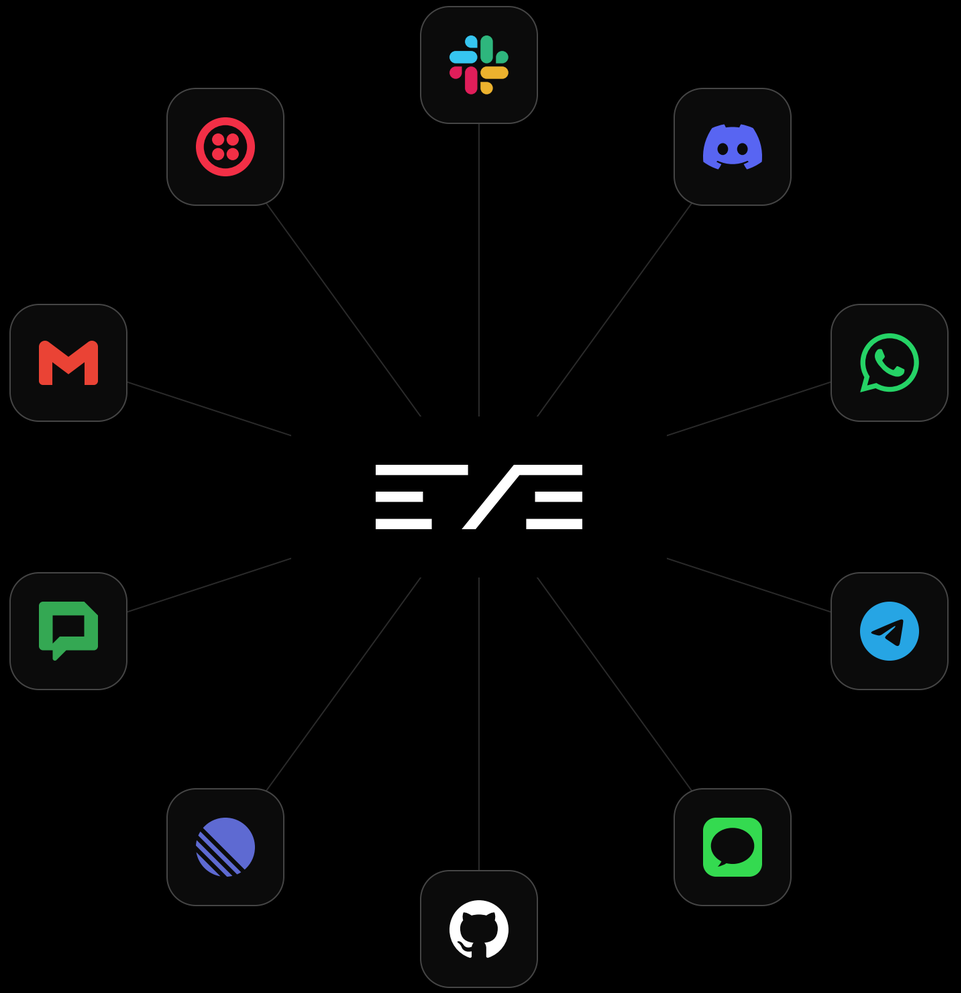
LOADED ON DEMAND

Only the description sits in context until a question needs it.

AGENT SKILLS STANDARD

A skill written for another client works as-is.

Channels



Channels

agent/channels/slack.ts · company brain

```ts
import { connectSlackCredentials } from "@vercel/connect/eve";
import { defaultSlackAuth, slackChannel } from "eve/channels/slack";
import { canReadModels } from "../lib/access.ts";

export default slackChannel({
  credentials: connectSlackCredentials("slack/company-brain"),
  onAppMention(ctx, message) {
    const auth = defaultSlackAuth(message, ctx);
    return canReadModels(auth, process.env) ? { auth } : null;
  },
  onDirectMessage(ctx, message) {
    const auth = defaultSlackAuth(message, ctx);
    return canReadModels(auth, process.env) ? { auth } : null;
  },
});
```

BUILT-IN CHANNELS

Slack, Teams, iMessage, WhatsApp, Discord, the web.

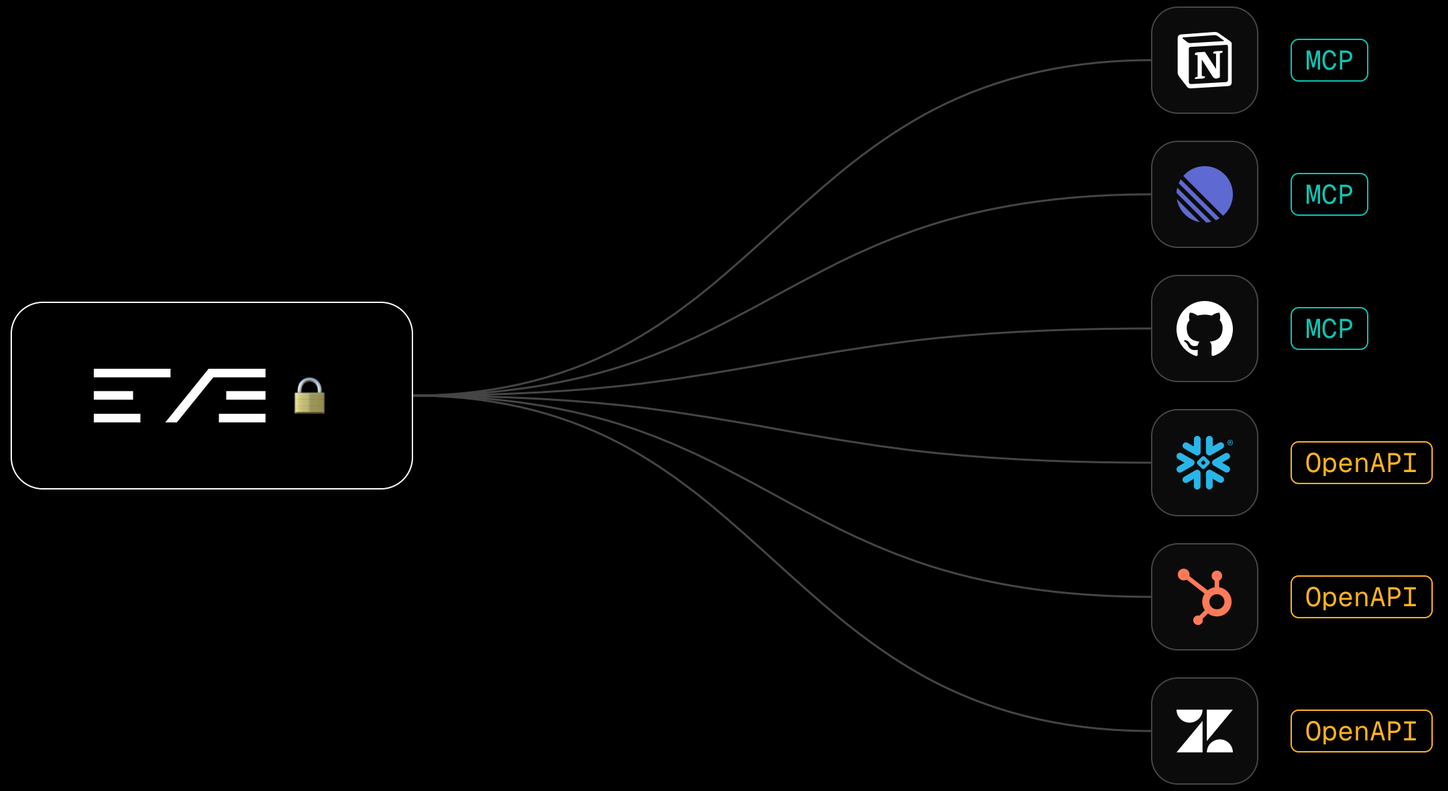
ADMISSION

Each handler decides who may start a turn. Return null to ignore.

ALLOW LISTS

Only listed people and channels can reach the data.

Connections



Connections

agent/connections/warehouse.ts

```ts
import { defineMcpClientConnection } from "eve/connections";

export default defineMcpClientConnection({
  url: process.env.WAREHOUSE_MCP_URL!,
  description: "The analytics warehouse.",
  auth: (ctx) => ({
    principalType: "user",
    getToken: async () => ({
      token: await warehouseToken(ctx.session.auth.current),
    }),
  }),
});
```

MCP AND OPENAPI

Every tool on the server, filtered to what you allow.

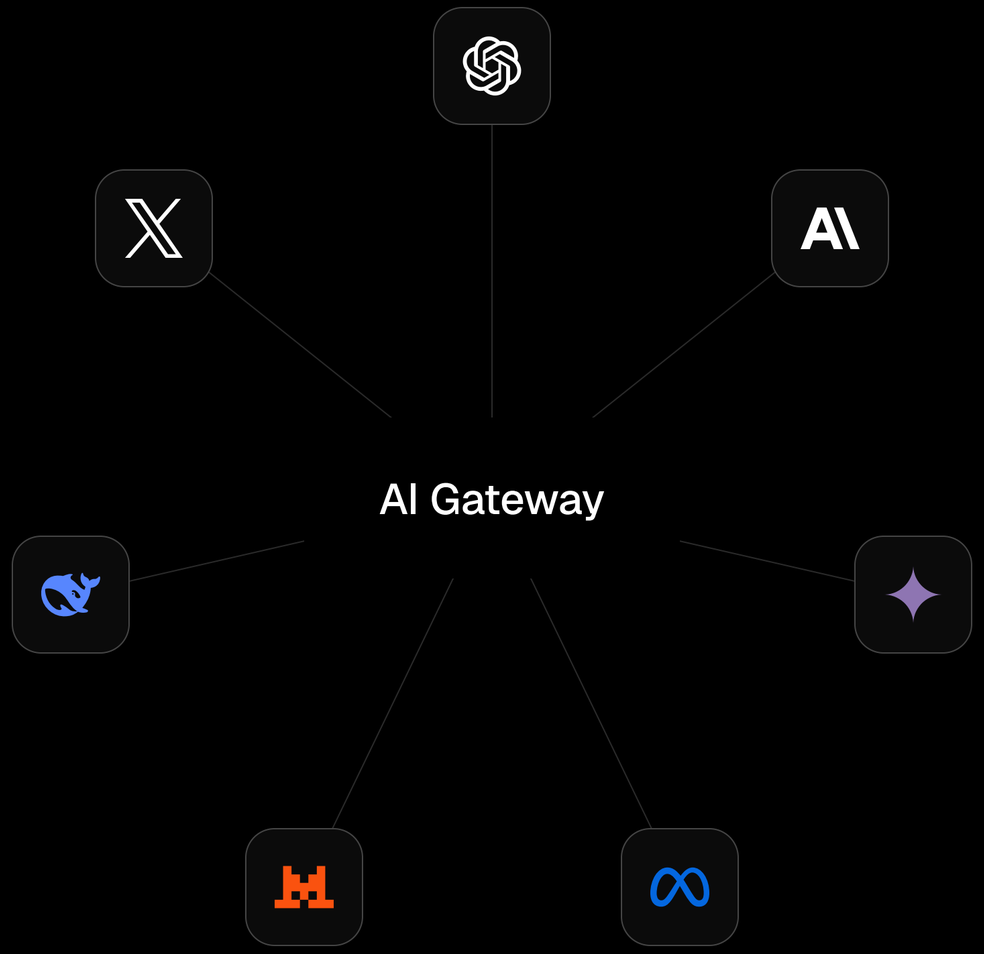
CREDENTIALS STAY IN THE RUNTIME

Per user or per app. Tokens never enter the conversation.

SUBAGENTS

Delegate focused work to a child agent with its own tools.

Models and AI Gateway



Models and AI Gateway

agent/agent.ts

```ts
import { defineAgent } from "eve";

export default defineAgent({
  // Routed through AI Gateway
  model: "anthropic/claude-sonnet-5",
  reasoning: "high",
  compaction: { threshold: 0.75 },
  build: { externalDependencies: ["@duckdb/node-api"] },
});
```

ONE-LINE EDIT

Swap providers by changing a string.

AI GATEWAY

One endpoint for every model, with provider fallback.

PER-TASK MODELS

Subagents and compaction can each use a different model.

Identity and sandboxes

ANY OIDC PROVIDER

Verify callers with any OIDC provider or your own authenticator.

AGENT OR USER

Each connection acts as the agent itself, or as the signed-in user.

A SANDBOX PER SESSION

Commands and generated code run isolated, away from credentials.

PICK THE PROVIDER

Vercel Sandbox or Docker. The company brain uses just-bash.

Workflow tools



Workflow tools

agent/tools/publish_report.ts

```ts
export default defineWorkflowTool({
  description: "Publish the monthly revenue report.",
  inputSchema: z.object({ month: z.string() }),
  async execute({ month }, ctx) {
    "use workflow";
    const report = await buildReport(month);
    const answer = await ctx.ask({
      prompt: `Publish ${month}? ${report.summary}`,
      display: "confirmation",
      options: [
        { id: "publish", label: "Publish" },
        { id: "cancel", label: "Cancel" },
      ],
    });
    if (answer.optionId !== "publish") return { published: false };
    return { published: true, url: await publish(report) };
  },
});
```

FIXED ORDER OF STEPS

Your code sets the order of steps.

DURABLE WAITS

ctx.ask parks the run for approval, across restarts.

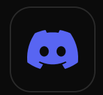
AGENTIC AND DETERMINISTIC

Combine model-driven steps and fixed steps in one run.

Extensions and the registry



```
eve add channel/slack
```



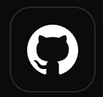
```
eve add channel/discord
```

◇

```
eve add channel/web
```



```
eve add extension/link
```



```
eve add extension/github
```

◇

```
eve add @acme/analytics
```

Evals


revenue-excludes-returns
0.94
relative-dates
0.88
no-pii-in-channels
1.00
aggregate-before-join
0.62
slack-allowlist
1.00

Evals

evals/revenue-excludes-returns.eval.ts

```ts
import { defineEval } from "eve/evals";
import { orders } from "./fixtures";

export default defineEval({
  description: "Revenue excludes returned orders.",
  async test(t) {
    const session = await t.session({ stubs: orders });
    await session.send("How did revenue do in March 2018?");
    t.calledTool("get_model");
    t.calledTool("query_sql");
    t.judge("Excludes returned orders from revenue.").atLeast(0.8);
  },
});
```

REAL TURNS

Evals talk to your agent over the same HTTP API as your users.

FIXED TOOL RESULTS

Stubs pin what tools return, so a changing warehouse doesn't change the test.

WHAT IT DID

Assert on tool calls: definitions before SQL.

HOW WELL

A judge model scores the answer. Run with --repeat to measure consistency.

Production grade agents

Architecture

RUNTIME

Durable execution, state persistence, event streaming

Workflow SDK

Postgres

Checkpointed steps, park between messages, resume on delivery

SESSION TURN ○ ○ ○ TURN ○ ○ TURN ○ ○ ○ ○ ○ = CHECKPOINTED STEP

AI SDK

Any provider API

Model calls, streaming

Sandbox SDK

Docker

Isolated execution

Connection SDK

Snowflake API

MCP and HTTP endpoints

Tools and subagents

Functions, child agents

CHANNEL

Where your agent gets surfaced

Chat SDK

- Slack
- Discord
- Web Chat
- Google Chat
- Microsoft Teams
- WhatsApp
- API
- Cron
- Twilio
- Linear
- GitHub
- Telegram
- iMessage
- SMS
- Email
- Notion

VERCEL MANAGED / SELF-HOSTED

Or deploy anywhere: Postgres-backed durability, Docker sandboxes, no managed services.

One session, step by step

KIND	STEP	0S — ILLUSTRATIVE — 19S
USER	“How did revenue do last month?”	
TOOL	`list_models`	
TOOL	`get_model · orders`	
TOOL	`query_sql · MIN/MAX(order_date)`	
MODEL	Ask a clarifying question	
PARKED	`Waiting for a reply`	
USER	“March, excluding returns”	
TOOL	`query_sql · monthly totals`	
MODEL	Answer in the thread	

MODEL · STEP 9 OF 9

Answer in the thread

Leads with the number and the period. SQL only if asked.

CONTEXT · 10 ITEMS

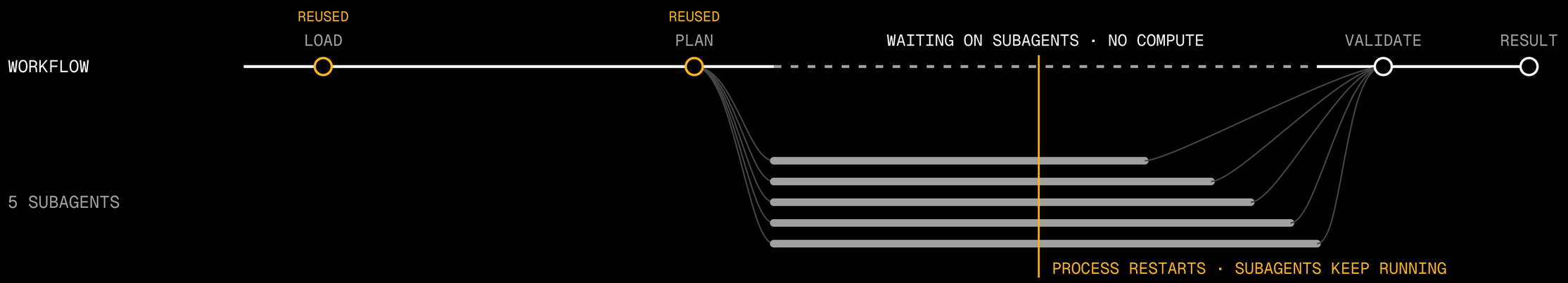
+ 4 earlier

```
Models: customers, orders
Definition: orders
Window: Jan 1 to Apr 9, 2018
Reply: clarifying question
User: March, excluding returns
Result: March and February totals
```

AVAILABLE CAPABILITIES

`list/get_model` dbt catalog	`query_sql` read-only SQL
`skills` loaded on demand	`slack` thread = session

Parking and resuming



CHECKPOINTED STEPS

Each step saves its result as soon as it finishes.

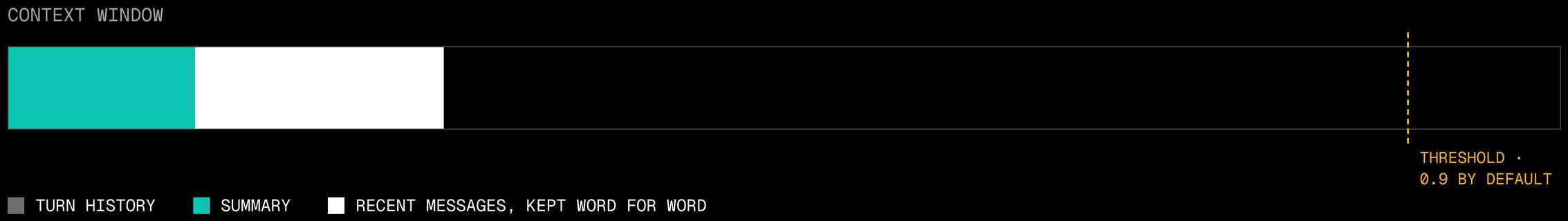
WAITS RELEASE COMPUTE

While subagents work, or a person approves, it holds no compute.

RESTARTS REPLAY

If the server dies, a new one reuses the finished steps.

Compaction



Query results pile up in context with every question.

At the threshold, older turns are summarized: decisions, progress, constraints.

The session moves to a fresh workflow run, with the same ID and stream.

Out of the box

Webhook verification	✓ EVE	Thread-to-session routing	✓ EVE
Crash recovery	✓ EVE	Approvals without held compute	✓ EVE
Context compaction	✓ EVE	Stream reconnects	✓ EVE
Safe redeploys	✓ EVE	Per-caller credentials	✓ EVE



Let’s build!

Two ways to get started

FROM THE CLI

Scaffold an agent and run it locally.

~/src

```
npx eve@latest init my-agent
cd my-agent
npm run dev
```

FROM VERCEL

Choose a template and name your agent.

