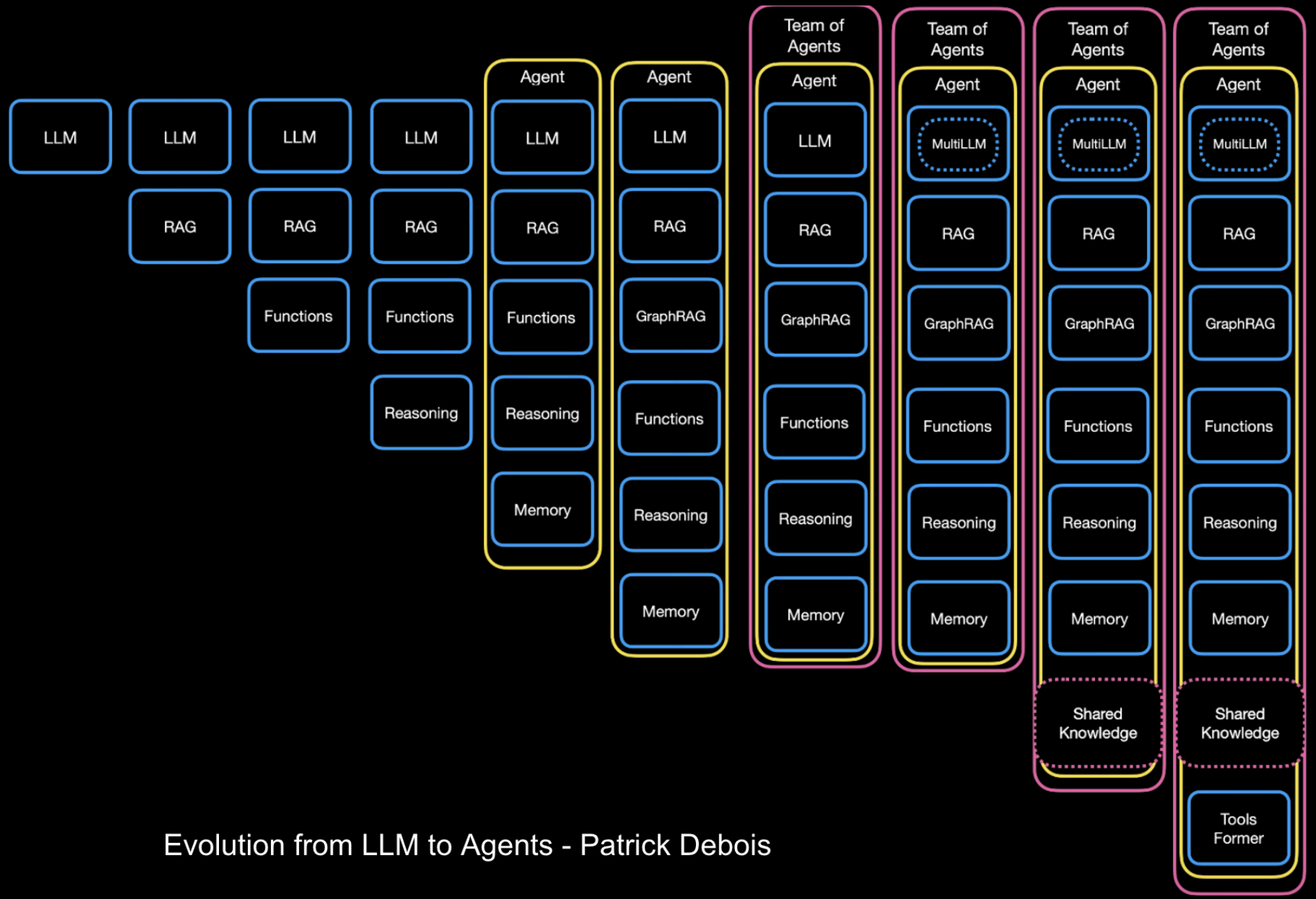
AI Coding Fabric

Context Development Lifecycle

Agentics Day 2026 - Patrick Debois



Evolution from LLM to Agents - Patrick Debois

Context is the new Code

Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

11:17 PM · Feb 2, 2025 · **6.8M** Views

1.4K 6K 33K 17K

Relevant View quotes

Post your reply Reply

Vibe coding

This also becomes the delivery mechanism for the self-serve POC conversation. Someone's friend sends them a URL or says "run `tessl init`" — they do it, open their editor, and their agent already knows what Tessl is and what to do with it. No guided setup required.

an example of AGENTS file:

```
# Tessl

Tessl is a skill management platform for AI coding assistants. Skills are
context files that guide you on how to work within this codebase —
conventions, patterns, architecture decisions, and task-specific
instructions.

## How to work with this project

Before starting any task, check if a relevant skill exists. Skills contain
the accumulated knowledge of how work gets done here — following them
produces better, more consistent results.

**Always:**
1. Run `tessl_list_skills` if you haven't oriented yourself yet
2. Fetch any skill relevant to the task with `tessl_get_skill` before
starting
3. Follow the skill's instructions throughout the task
4. If no skill exists for what you're doing, note it — a skill should
```

Product onboarding as a skill

I ❤ to think in parallels

2009 - What if Ops was like Dev

2025 - What if Context was like Code

Patrick Debois - Product Devrel Tessl

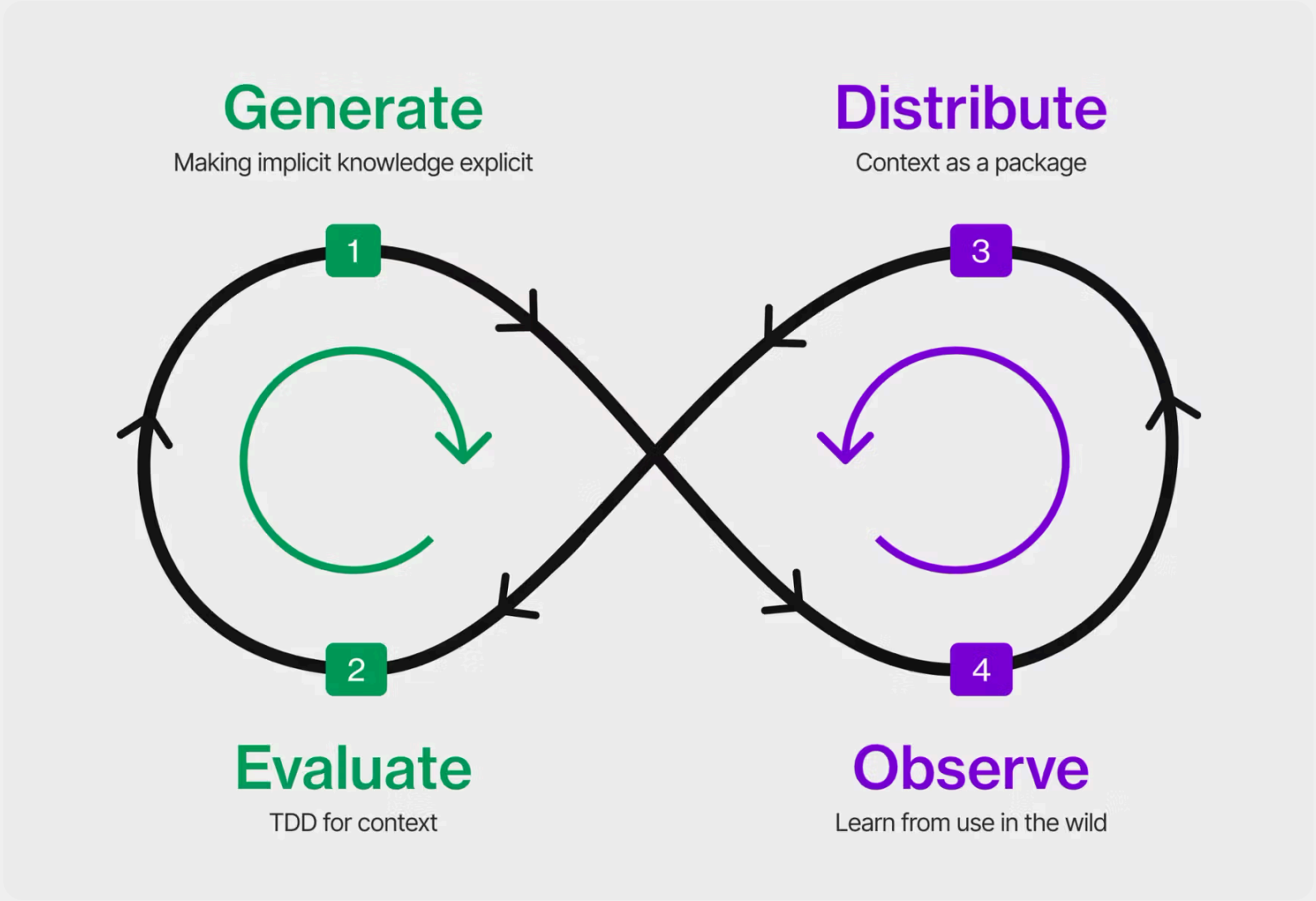
SDLC

Software Development Lifecycle

CDLC

Context Development Lifecycle

Context Development Lifecycle



https://tessl.io/blog/context-development-lifecycle-better-context-for-ai-coding-agents/

Today's Agenda

01 ✨ Generate — create & curate context

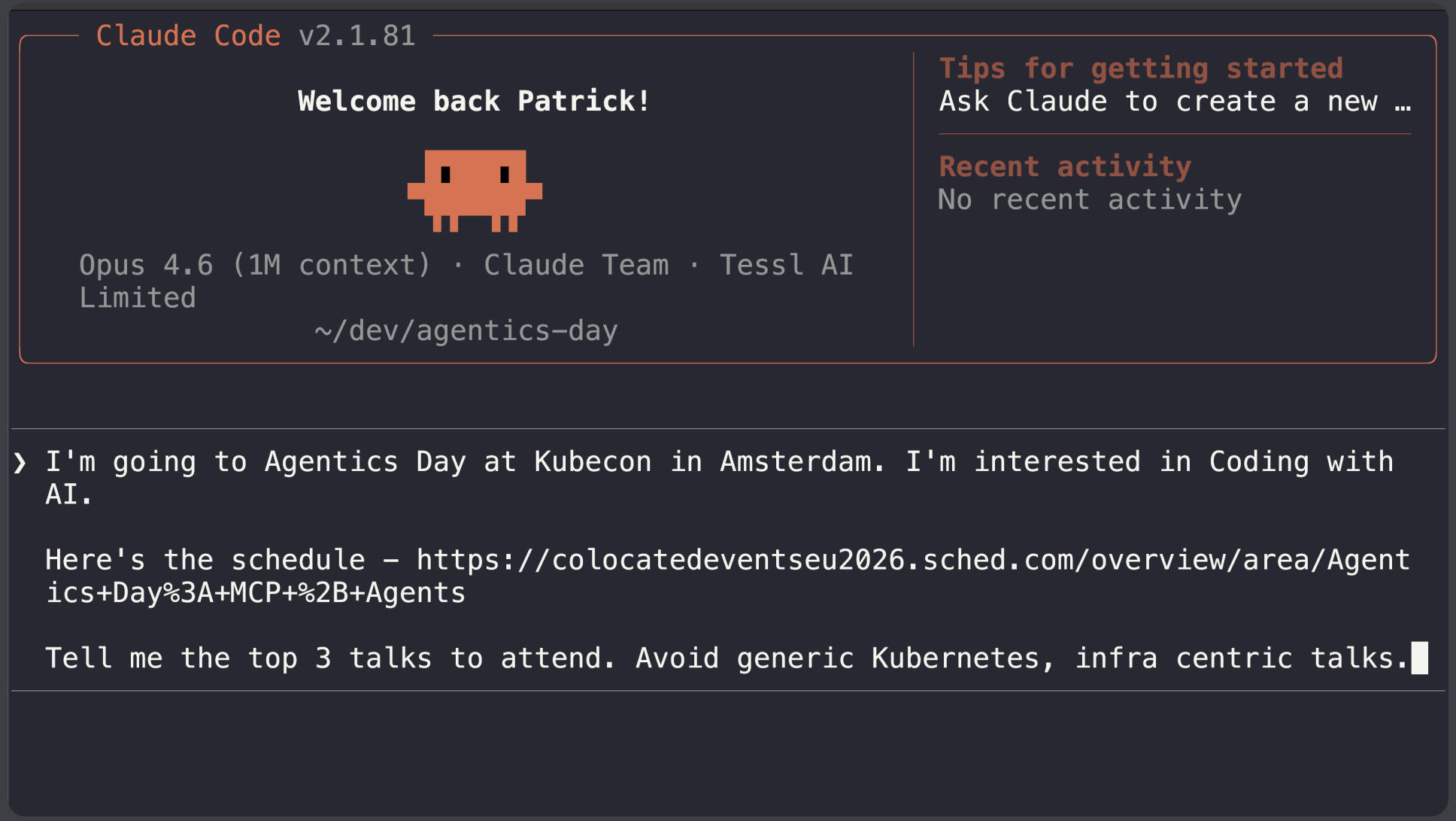
02 🔍 Evaluate — test & measure context quality

03 📦 Distribute — package & share context

04 👁️ Observe — monitor & improve in production


Generate

Prompts - Humans as context engine



https://colocatedeventseu2026.sched.com/overview/area/Agentics+Day%3A+MCP+%2B+Agents

Rules, Instructions - Agent.md

AGENTS.md

A simple, open format for guiding coding agents,
used by over 60k open-source projects.

Think of AGENTS.md as a **README for agents**: a dedicated, predictable place to provide the context and instructions to help AI coding agents work on your project.

Explore Examples | View on GitHub

```
# AGENTS.md

## Setup commands
- Install deps: `pnpm install`
- Start dev server: `pnpm dev`
- Run tests: `pnpm test`

## Code style
- TypeScript strict mode
- Single quotes, no semicolons
- Use functional patterns where possible
```

Why AGENTS.md?

README.md files are for humans: quick starts, project descriptions, and contribution guidelines.

AGENTS.md complements this by containing the extra, sometimes detailed context coding agents need: build steps, tests, and conventions that might clutter a README or aren't relevant to human contributors.

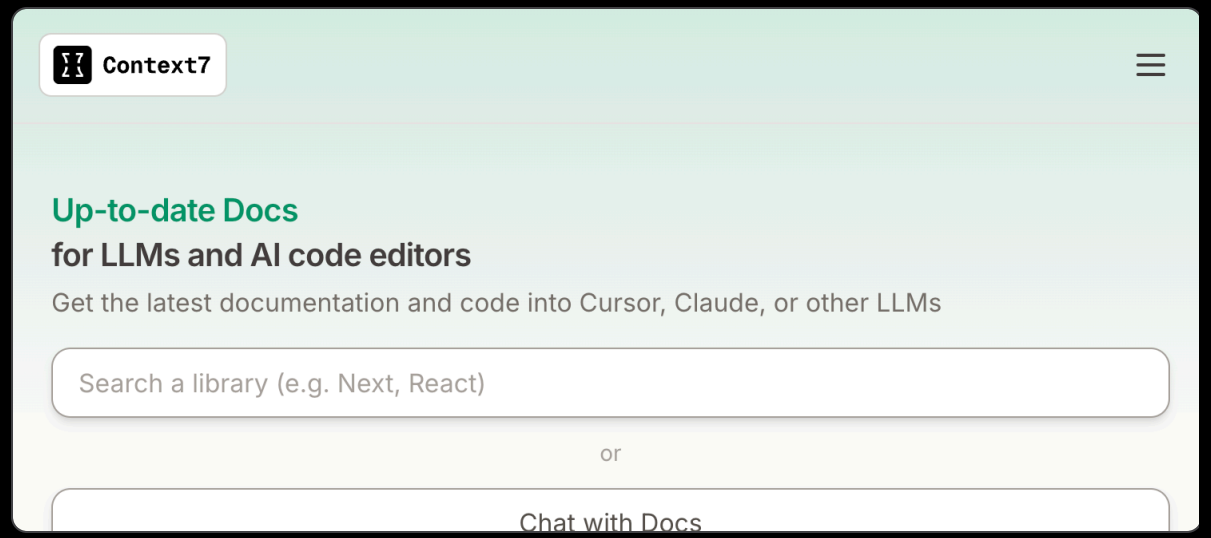
We intentionally kept it separate to:

- **Give agents a clear, predictable place for instructions.**
- **Keep READMEs concise and focused on human contributors.**
- **Provide precise, agent-focused guidance that complements existing README and docs.**

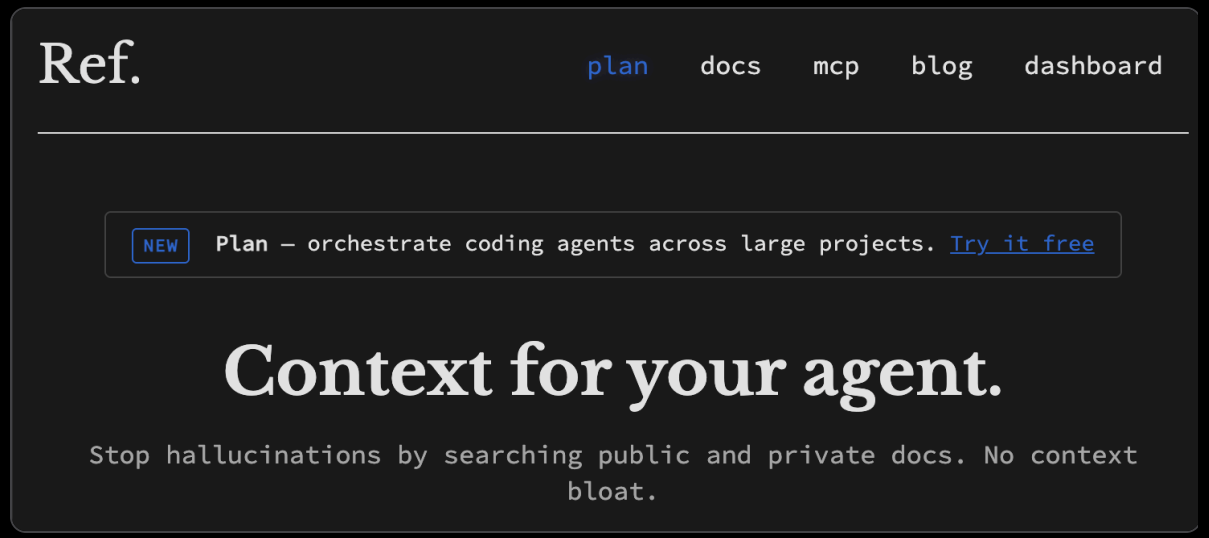
Rather than introducing another proprietary file, we chose a name and format that could work for anyone. If you're building or using coding agents and find this helpful, feel free to adopt it.

Agent.md

Context from code and docs



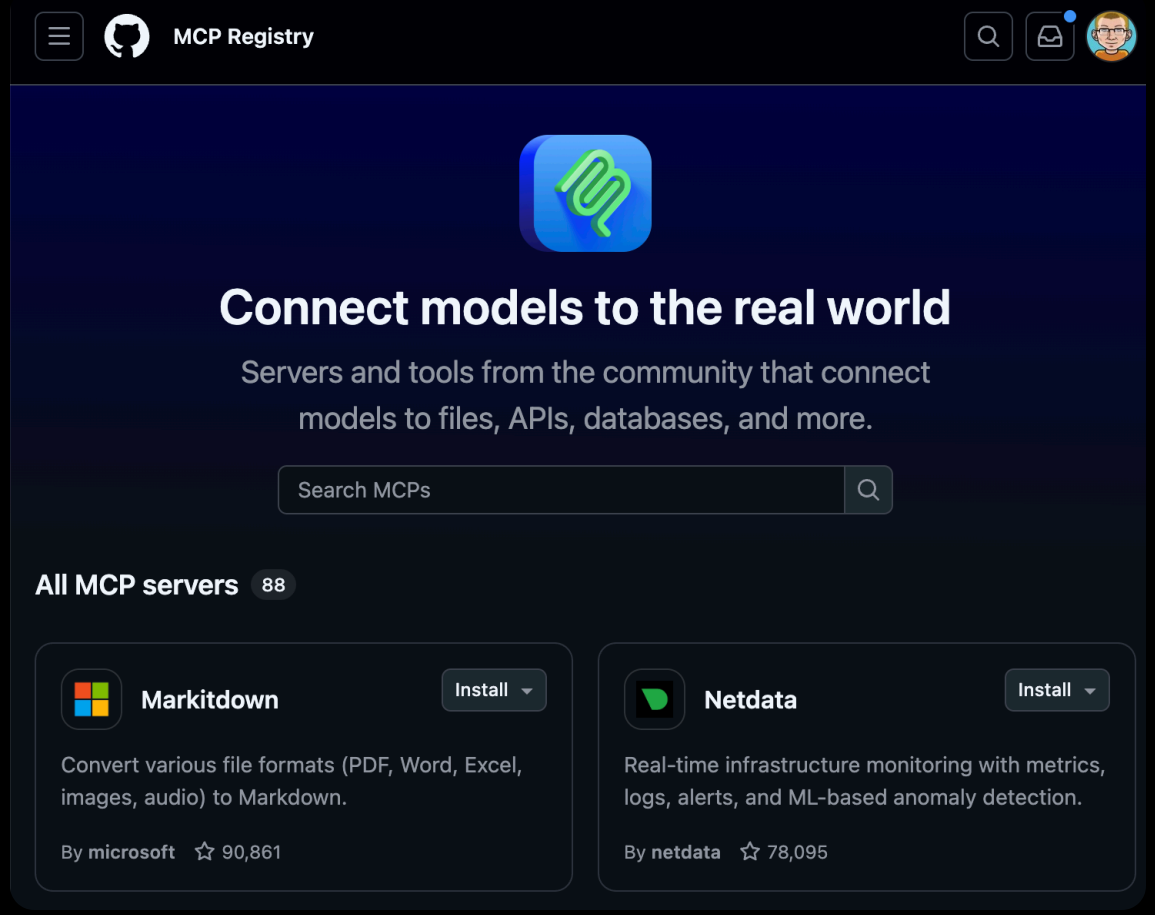
https://context7.com/



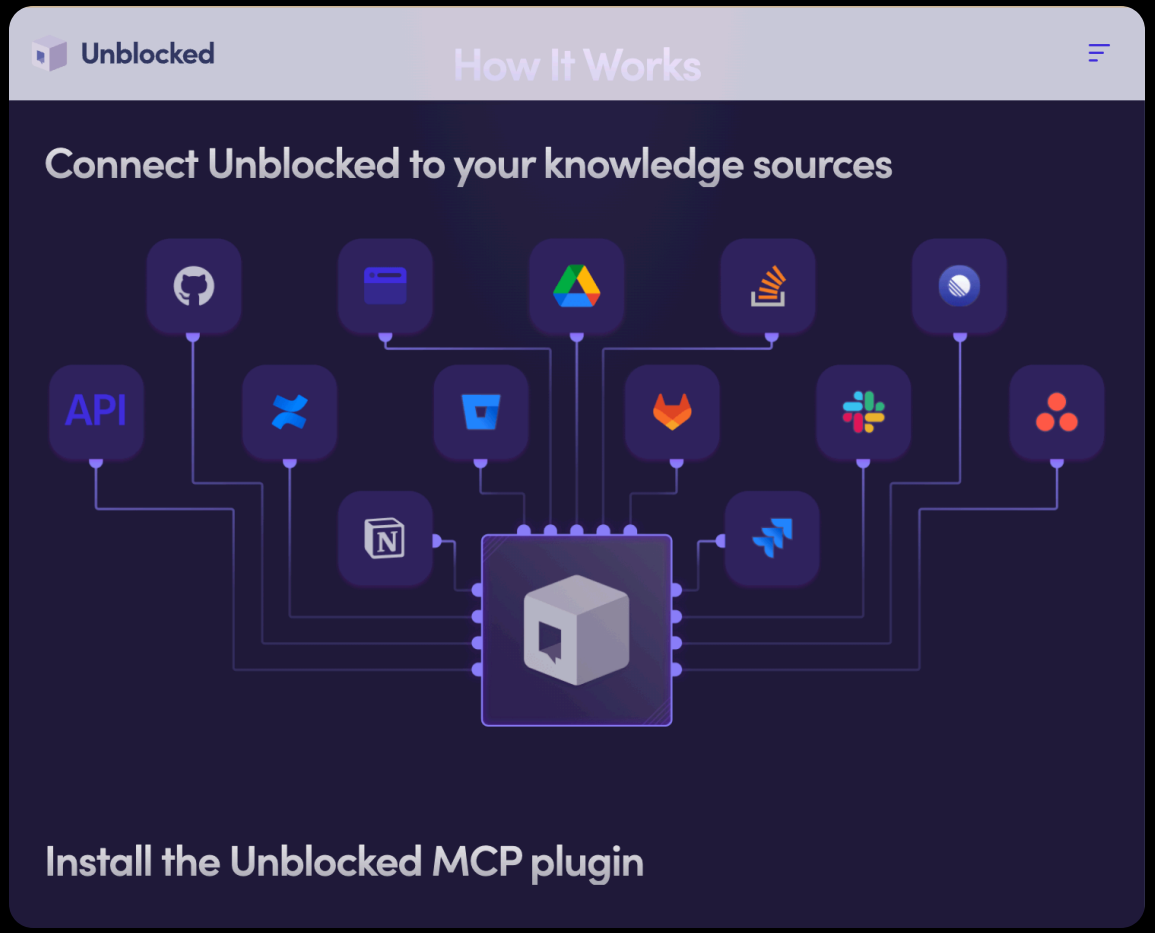
https://ref.tools/

https://github.com/andrewyng/context-hub

Context Connectors

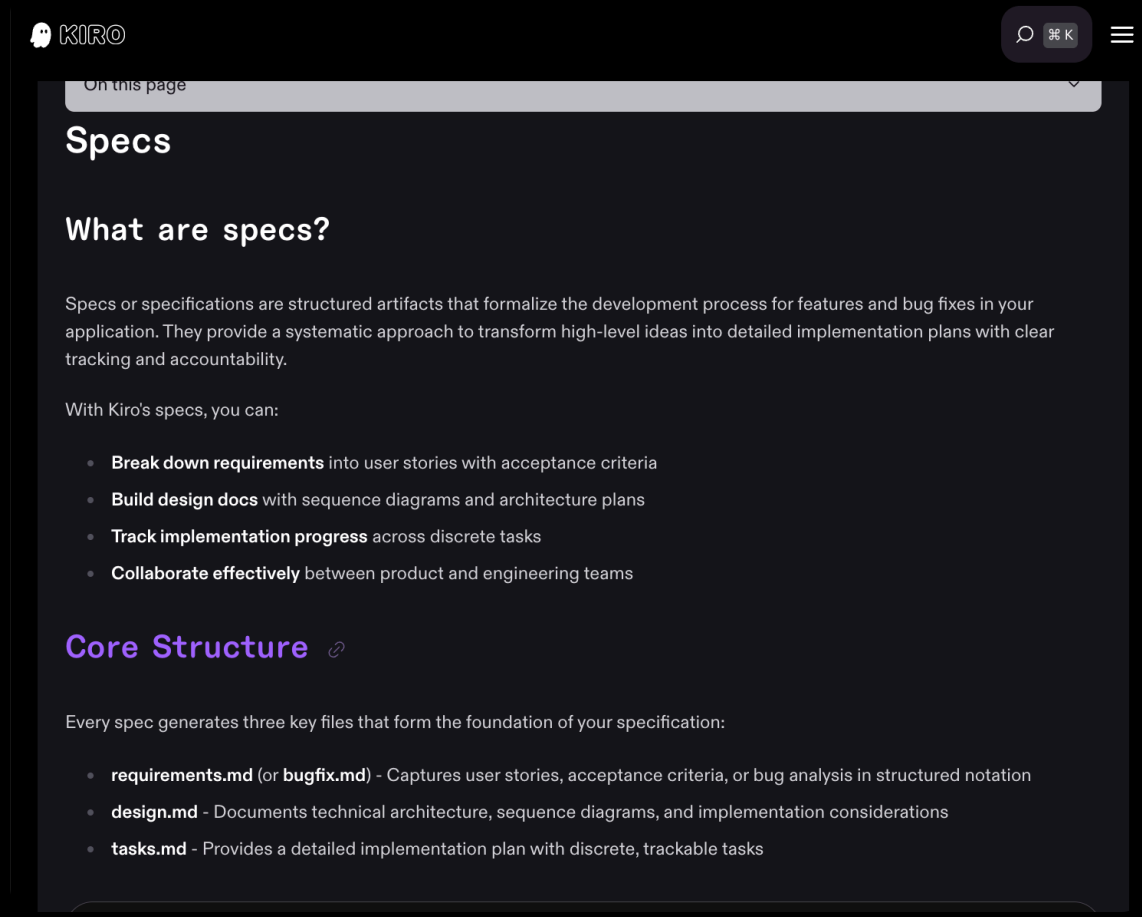


Github MCP - https://github.com/mcp

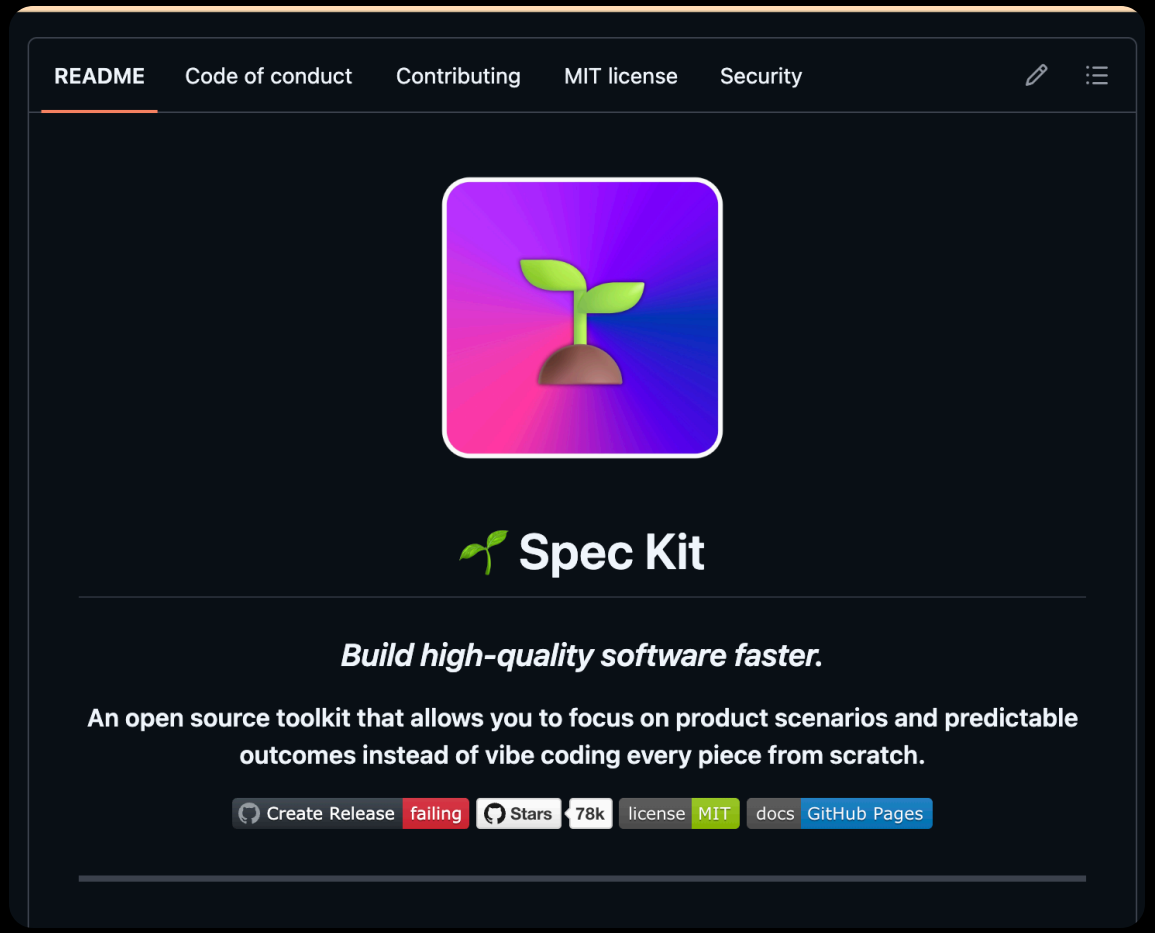


https://getunblocked.com/unblocked-mcp/

Spec Driven Development



Kiro - http://kiro.dev/docs/specs/



Gitub Speckit - https://github.com/github/spec-kit

Today's Agenda

01 ✅ Generate — create & curate context

02 🔍 **Evaluate — test & measure context quality**

03 📦 Distribute — package & share context

04 👁️ Observe — monitor & improve in production


Evaluate

Skill Markdown

SKILL.md | Quality | Evals | Security

Terraform Style Guide

Generate and maintain Terraform code following HashiCorp's official style conventions and best practices.

Reference: HashiCorp Terraform Style Guide

Code Generation Strategy

When generating Terraform code:

1. Start with provider configuration and version constraints
2. Create data sources before dependent resources
3. Build resources in dependency order
4. Add outputs for key resource attributes
5. Use variables for all configurable values

File Organization

File	Purpose

https://docs.tessl.io/evaluate/evaluating-skills

Example Structure

```
# terraform.tf
terraform {
  required_version = ">= 1.7"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# variables.tf
variable "environment" {
  description = "Target deployment environment"
  type        = string

  validation {
    condition     = contains(["dev", "staging", "prod"], var.environment)
    error_message = "Environment must be dev, staging, or prod."
  }
}

# locals.tf
locals {
  common_tags = {
    Environment = var.environment
    ManagedBy   = "Terraform"
  }
}

# main.tf
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
```

https://docs.tessl.io/evaluate/evaluating-skills

Skill review ~ Linting

Validation 100%

Warnings & errors only

Checks the skill against the spec for correct structure and formatting. All validation checks must pass before discovery and implementation can be scored.

Validation — 11 / 11 Passed

Validation for skill structure

Criteria	Description	Result
skill_md_line_count	SKILL.md line count is 354 (<= 500)	Pass
frontmatter_valid	YAML frontmatter is valid	Pass
name_field	'name' field is valid: 'terraform-style-guide'	Pass
description_field	'description' field is valid (165 chars)	Pass
compatibility_field	'compatibility' field not present (optional)	Pass
allowed_tools_field	'allowed-tools' field not present (optional)	Pass
metadata_version	'metadata' field not present (optional)	Pass
metadata_field	'metadata' field not present (optional)	Pass
license_field	'license' field not present (optional)	Pass
frontmatter_unknown_keys	No unknown frontmatter keys found	Pass

https://docs.tessl.io/evaluate/evaluating-skills

Skill eval ~ Grammarly

Discovery 75%

Based on the skill's description, can an agent find and select it at the right time? Clear, specific descriptions lead to better discovery.

This is a solid description with clear 'what' and 'when' clauses and good distinctiveness for Terraform-specific work. The main weaknesses are moderate specificity (lacks concrete action examples) and limited trigger term coverage (missing common user phrases like 'IaC' or '.tf files').

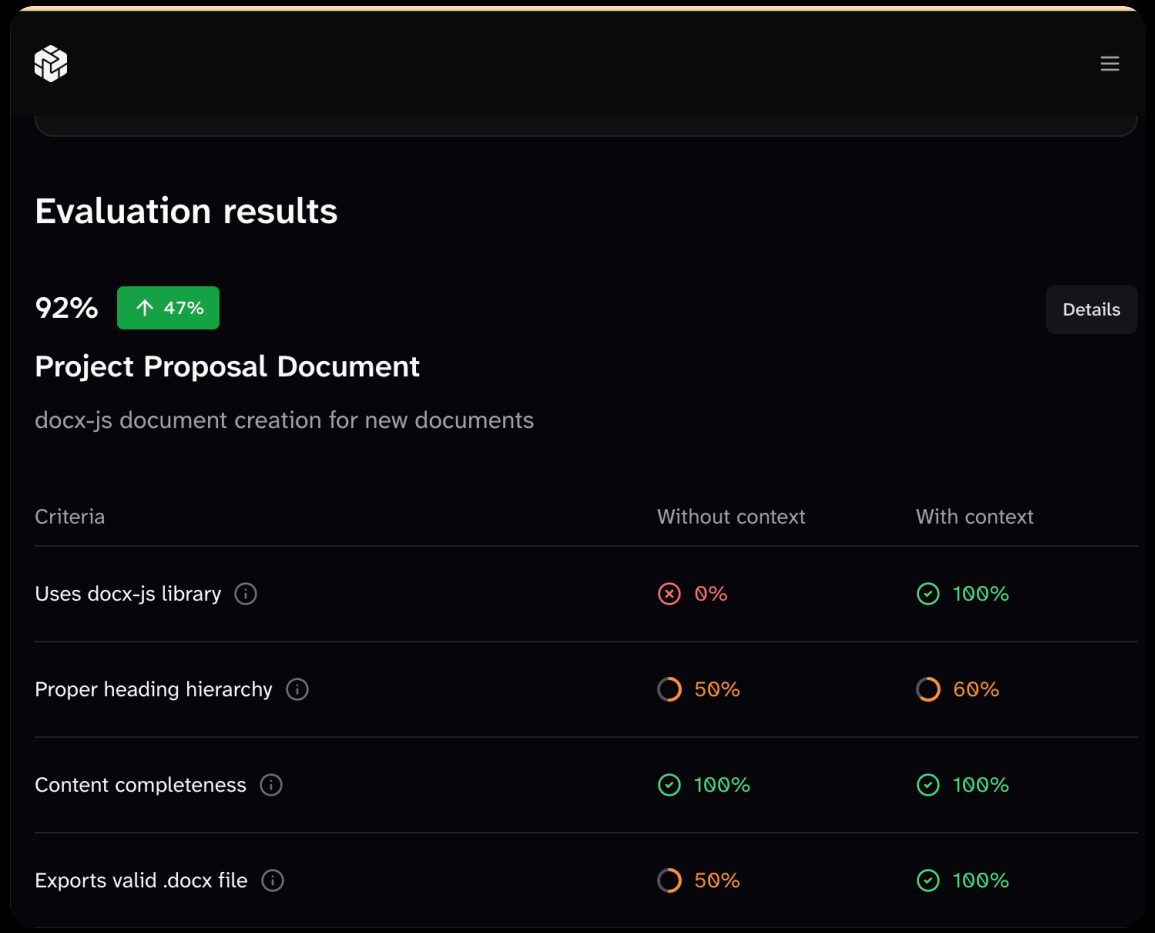
Suggestions

- Add specific concrete actions like 'create modules, define resources, configure providers, set up backends' to improve specificity
- Expand trigger terms to include common variations: 'infrastructure as code', 'IaC', '.tf files', 'terraform modules', 'cloud infrastructure'

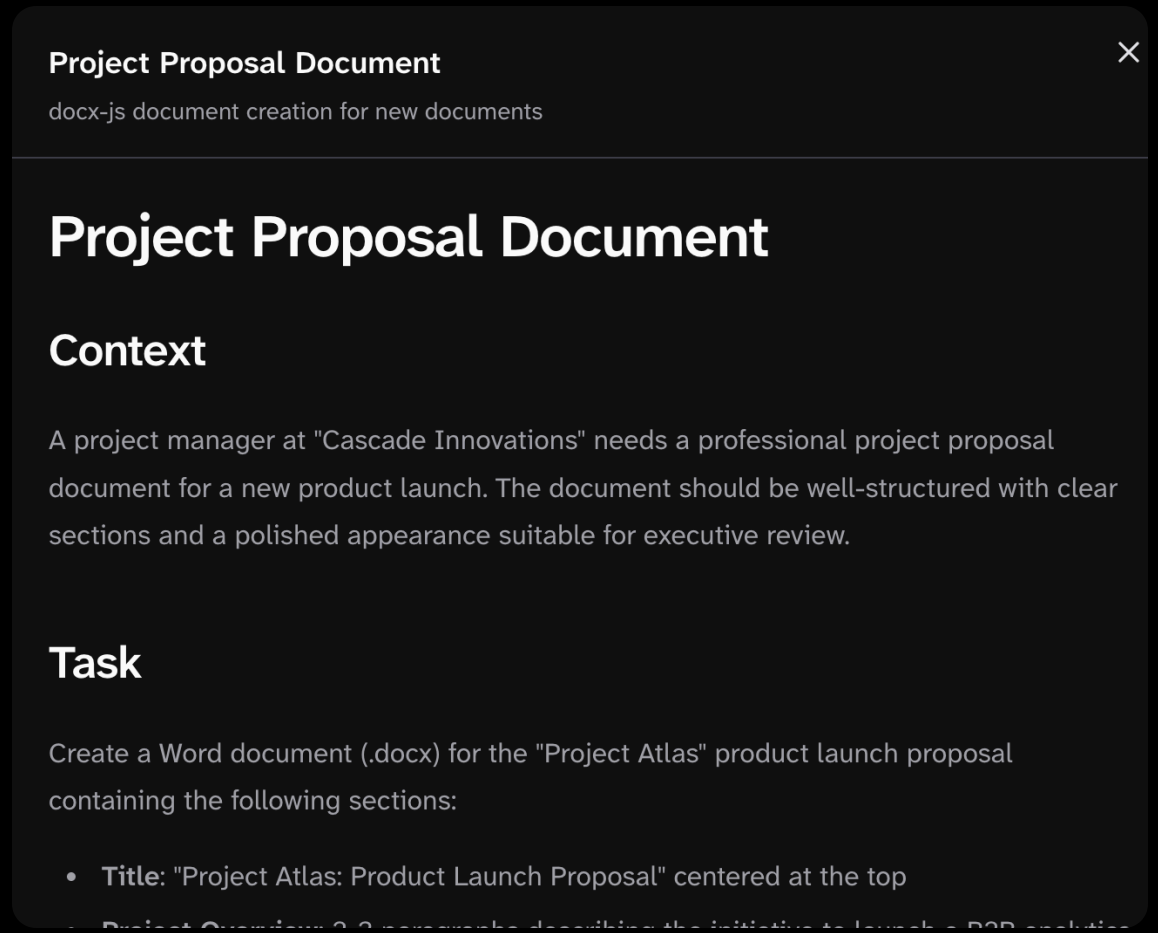
Dimension	Reasoning	Score
Specificity	Names the domain (Terraform HCL) and mentions style conventions and best practices, but doesn't list specific concrete actions like 'create modules', 'define resources', 'configure providers', or 'set up state backends'.	2 / 3
Completeness	Clearly answers both what ('Generate Terraform HCL code following HashiCorp's official style conventions and best practices') and when ('Use when writing, reviewing, or generating Terraform configurations') with explicit trigger guidance.	3 / 3

https://docs.tessl.io/evaluate/evaluating-skills

Task eval ~ Unit Tests



https://tessl.io/registry/skills/github/anthropics/skills/docx/evals



https://tessl.io/registry/skills/github/anthropics/skills/docx/evals

With your Repo ~ E2E Tests

scenario.json

Generated by `tessl scenario download`. Defines the fixture for the eval run.

```
{
  "type": "coding",
  "fixture": {
    "type": "commit",
    "repoUrl": "https://github.com/org/repo.git",
    "ref": "24829180ba1fafb86b...",
    "exclude": ["*.mdc", "*.md", "tile.json", "tessl.json", ".tessl/"]
  }
}
```

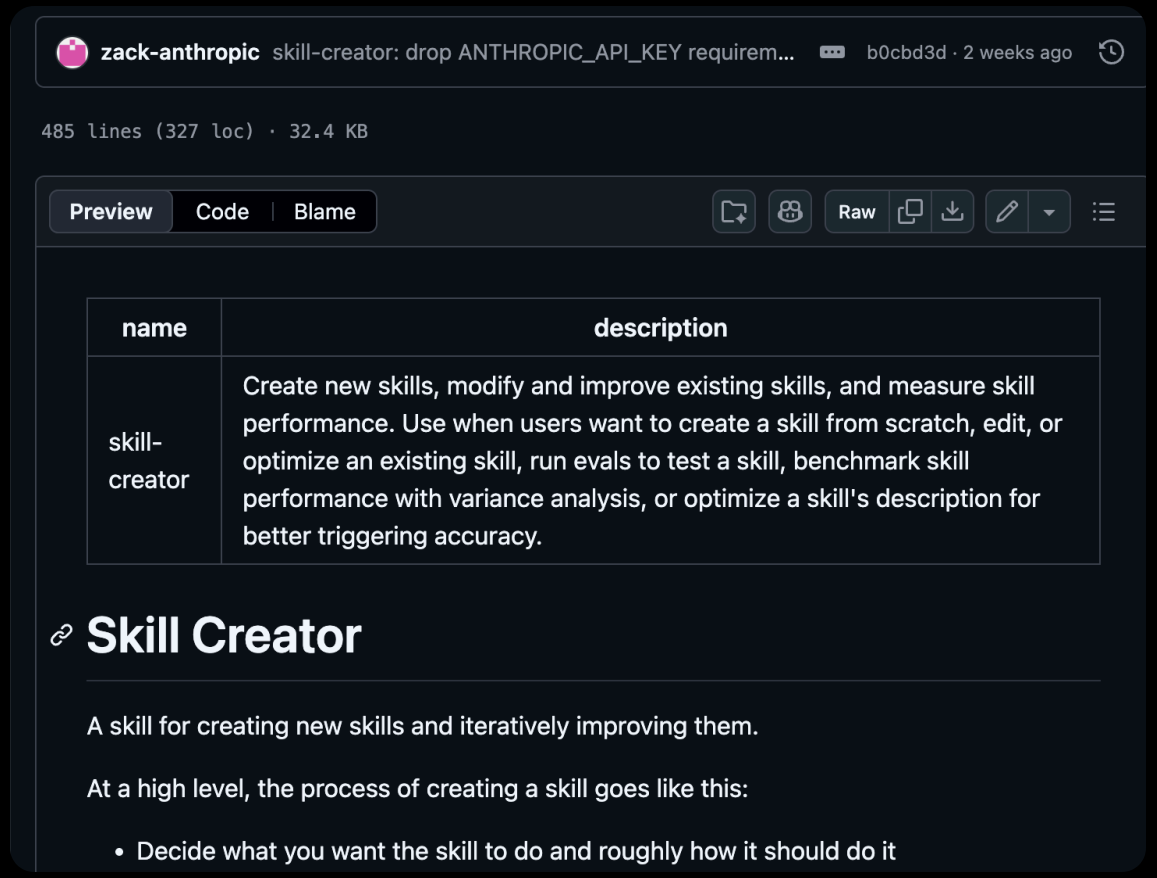
- `fixture.ref` — the parent commit hash (the starting state for the agent)
- `fixture.exclude` — context patterns stripped for baseline; also used as the default `--context-pattern` at run time
- `fixture.repoUrl` — full clone URL

task.md

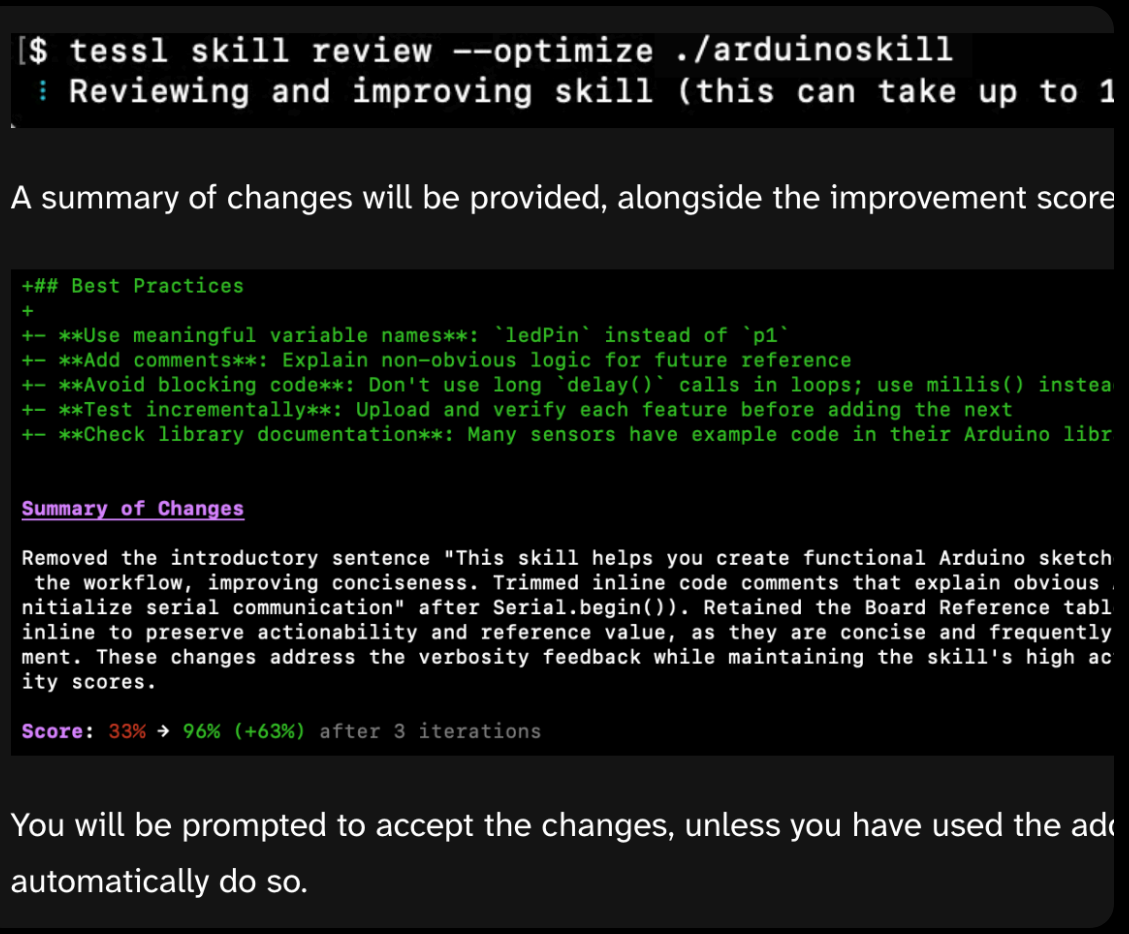
Free-form markdown. This is the **only file the agent sees** — it has no access to `criteria.json`. Typically structured with Problem, Expected Behavior, and Acceptance Criteria sections. You can edit this freely before running.

https://docs.tessl.io/evaluate/evaluating-your-codebase

Skill optimize ~ Code Actions

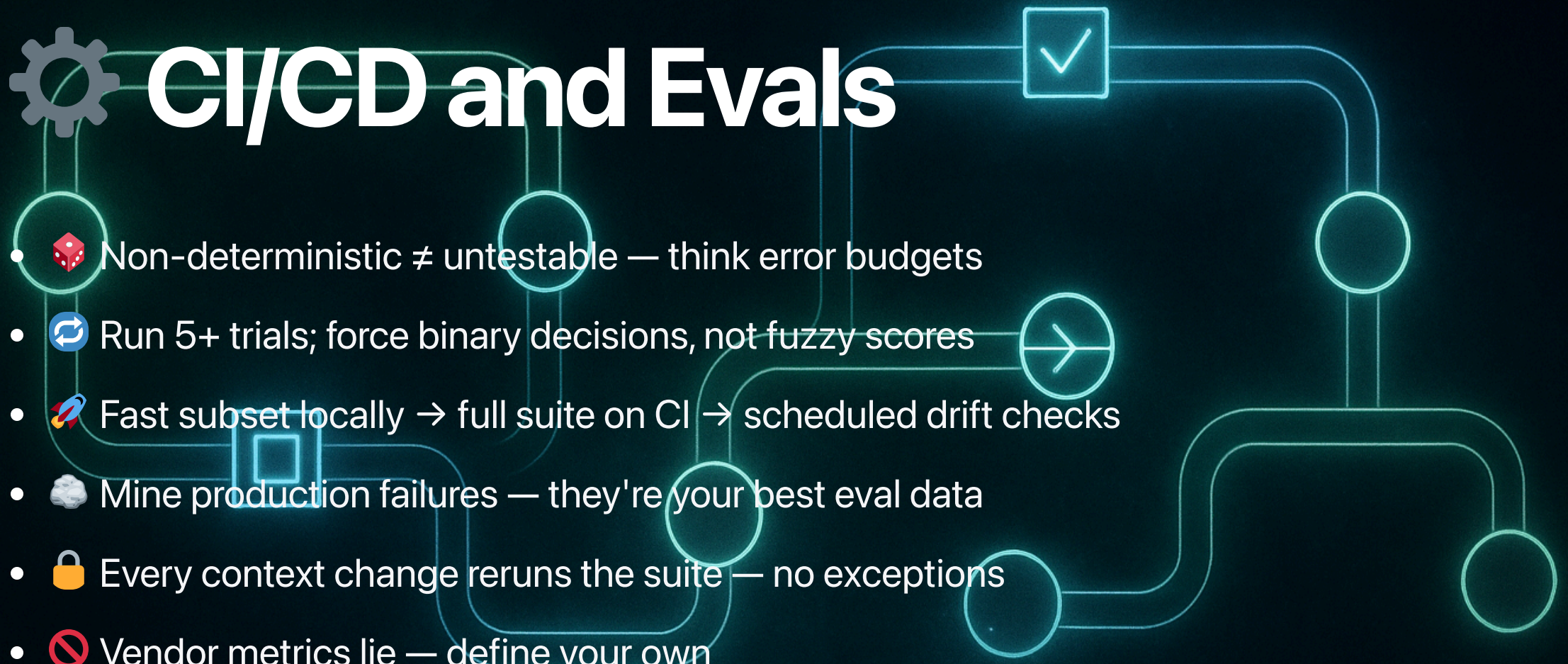


https://github.com/anthropics/skills/blob/main/skills/skill-creator/SKILL.md



https://docs.tessl.io/evaluate/optimize-a-skill-using-best-practices

⚙️ CI/CD and Evals



- 🎲 Non-deterministic ≠ untestable — think error budgets
- 🔁 Run 5+ trials; force binary decisions, not fuzzy scores
- 🚀 Fast subset locally → full suite on CI → scheduled drift checks
- 🪨 Mine production failures — they're your best eval data
- 🔒 Every context change reruns the suite — no exceptions
- 🚫 Vendor metrics lie — define your own

https://tessl.io/blog/cicd-for-context-in-agentic-coding-same-pipeline-different-rules/

Today's Agenda

01 ✅ Generate — create & curate context

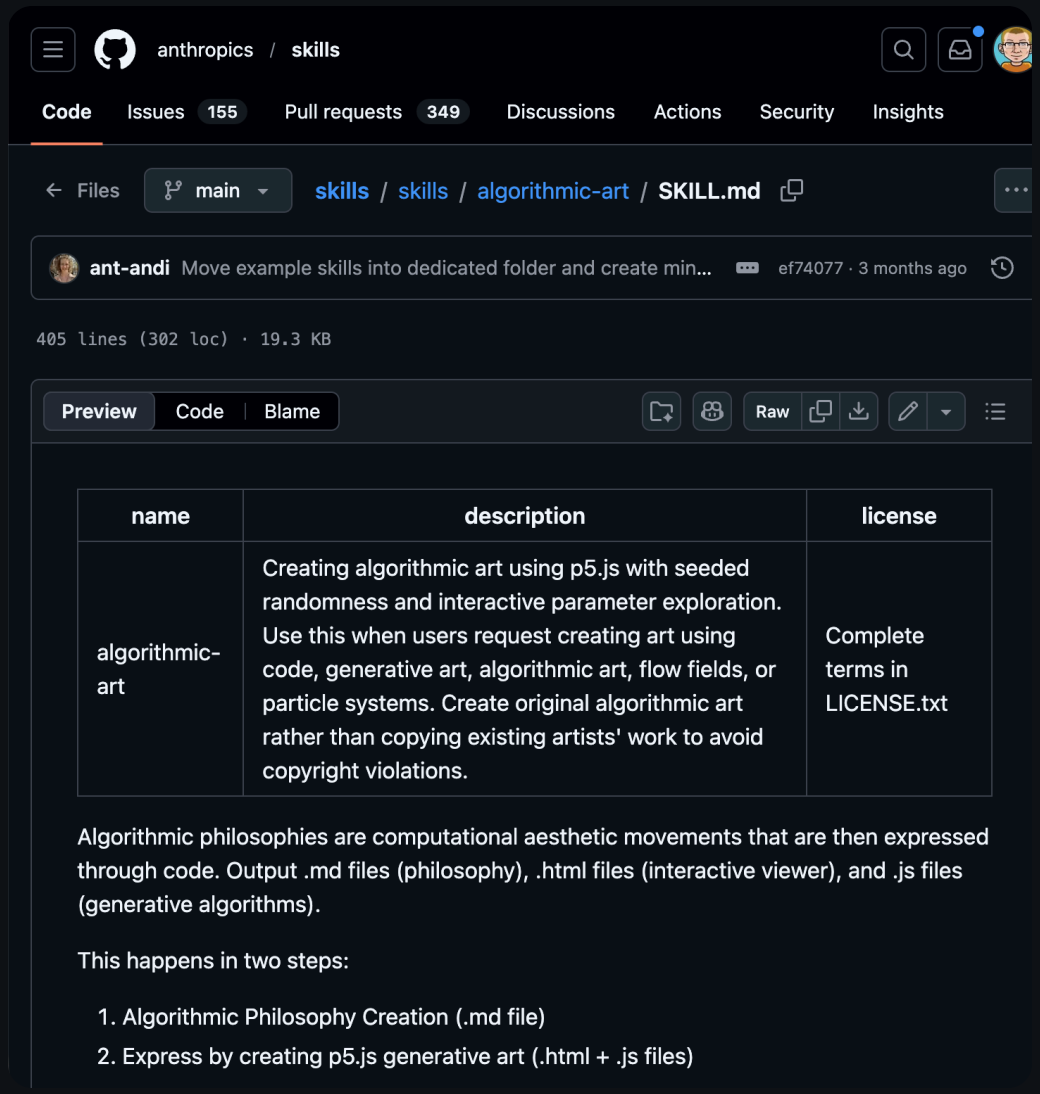
02 ✅ Evaluate — test & measure context quality

03 📦 **Distribute — package & share context**

04 👁️ Observe — monitor & improve in production


Distribute

Share on github



name	description	license
algorithmic-art	Creating algorithmic art using p5.js with seeded randomness and interactive parameter exploration. Use this when users request creating art using code, generative art, algorithmic art, flow fields, or particle systems. Create original algorithmic art rather than copying existing artists' work to avoid copyright violations.	Complete terms in LICENSE.txt

Algorithmic philosophies are computational aesthetic movements that are then expressed through code. Output .md files (philosophy), .html files (interactive viewer), and .js files (generative algorithms).

This happens in two steps:

1. Algorithmic Philosophy Creation (.md file)
2. Express by creating p5.js generative art (.html + .js files)

Anthropic Example Skill - https://github.com/anthropics/skills/blob/main/skills/algorithmic-art/SKILL.md

Package managers & Versioning

Why APM

AI coding agents need context to be useful — standards, prompts, skills, plugins — but today every developer sets this up manually. Nothing is portable nor reproducible. There's no manifest for it.

APM fixes this. Declare your project's agentic dependencies once in `apm.yml`, and every developer who clones your repo gets a fully configured agent setup in seconds — with transitive dependency resolution, just like npm or pip.

```yaml
# apm.yml — ships with your project
name: your-project
version: 1.0.0
dependencies:
  apm:
    # Skills from any repository
    - anthropics/skills/skills/frontend-design
    # Plugins
    - github/awesome-copilot/plugins/context-engineering
    # Specific agent primitives from any repository
    - github/awesome-copilot/agents/api-architect.agent.md
    # A full APM package with instructions, skills, prompts, hooks...
    - microsoft/apm-sample-package
```

```
git clone <org/repo> && cd <repo>
```

APM - https://github.com/microsoft/apm

Tessl Docs Website Press Kit

That directory is linked to `tessl.json`. This tile provides documentation for the FastAPI framework, which you, or your agent, can now query using the `query_library_docs` tool when you need information about FastAPI usage, features, and best practices.

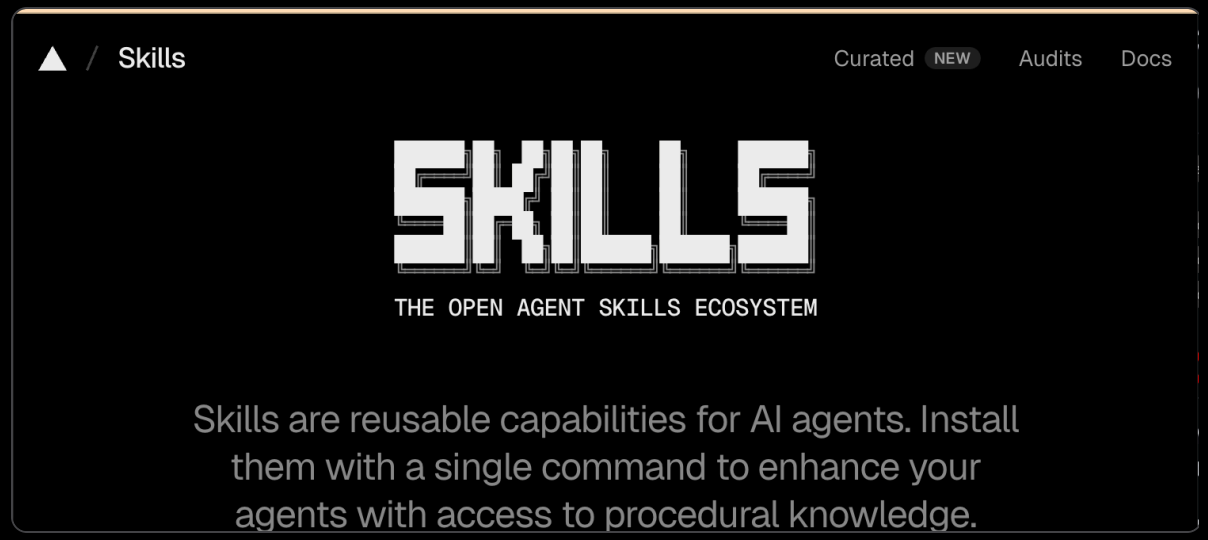
Inside the tile directory, you'll find a manifest file called `tile.json` that contains metadata about the tile, including the package version:

```
{
  "name": "tessl/pypi-fastapi",
  "version": "0.116.0",
  "docs": "docs/index.md",
  "describes": "pkg:pypi/fastapi@0.116.1",
  "summary": "FastAPI framework, high performance, easy to learn, fast
  "private": true
}
```

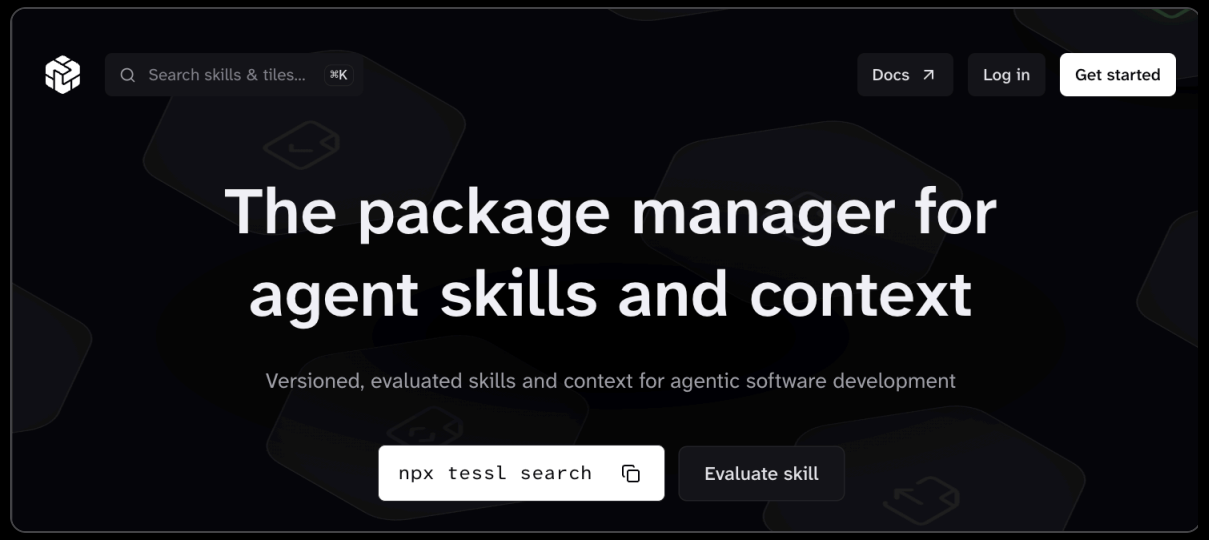
- If a tile describes a package, you must have (at a minimum)
  - `describes` - a valid PURL ↗ to identify the package

https://docs.tessl.io/use/make-your-agents-smarter-with-documentation

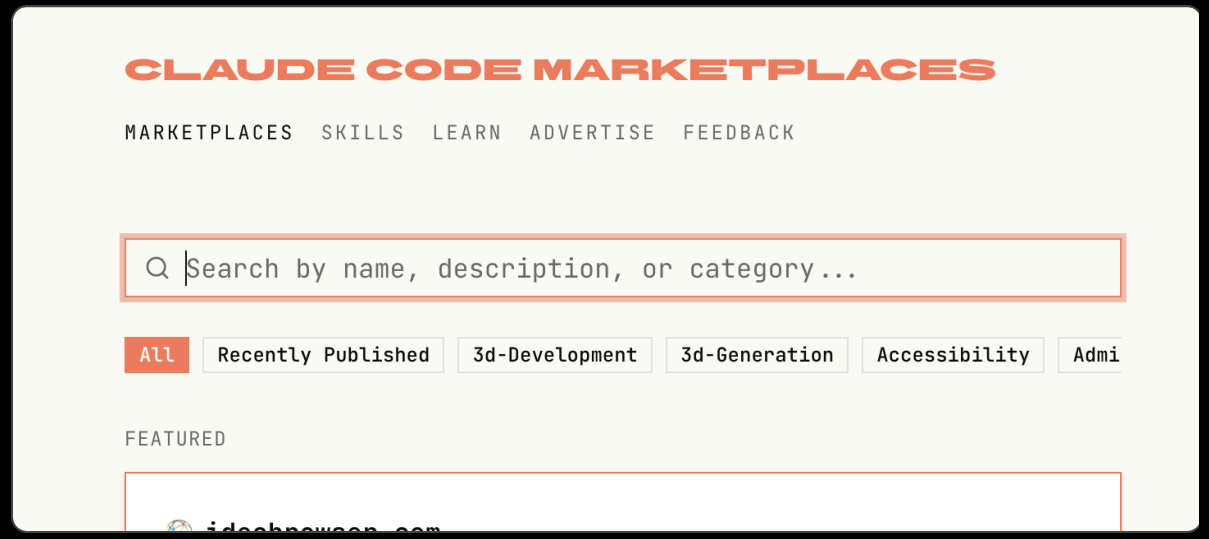
Marketplaces & Registries



Skills.sh - https://skills.sh

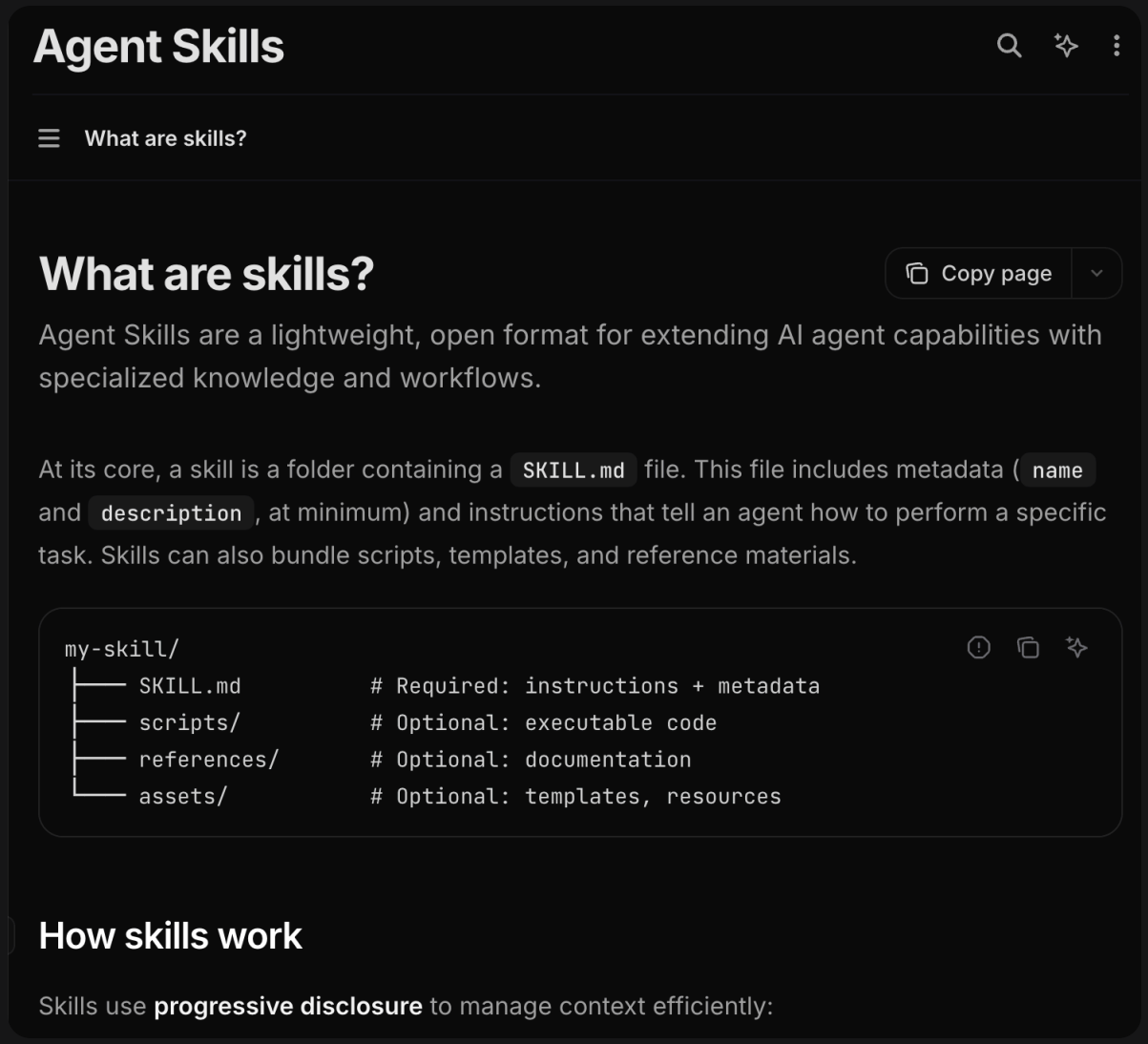


Tessl Registry - https://tessl.io/registry



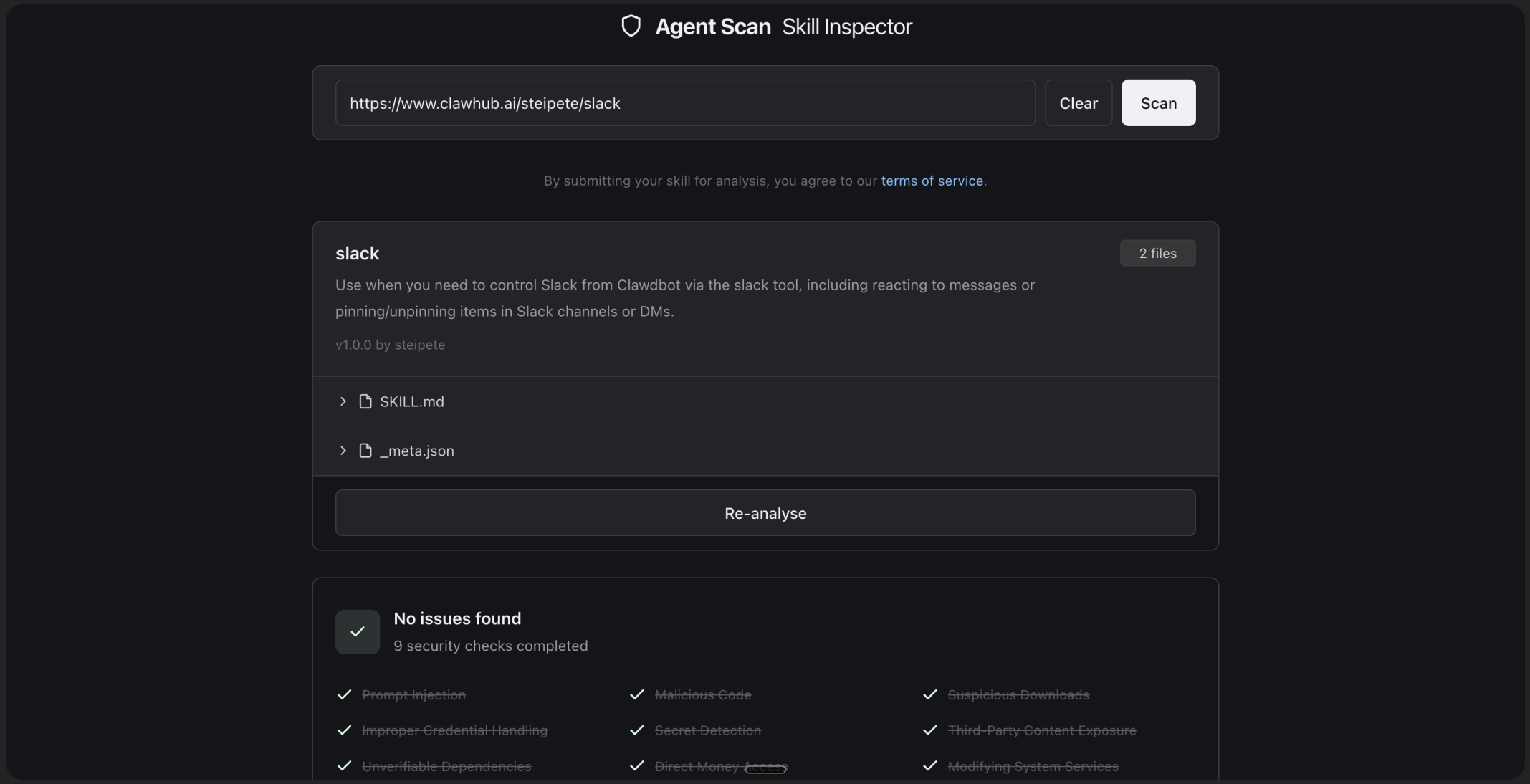
Claude Marketplaces - https://claudemarketplaces.com/

Skills - context package standard ?



https://agentskills.io/what-are-skills

Context security scan



Snyk Skill Scan - https://labs.snyk.io/experiments/skill-scan/

Context provenance

README MIT license

AISBOM - AI Software Bill of Materials

JSON Spec for Transparency Obligations of the EU AI Act, including LLM / foundation models

Version 0.1 (December 11, 2023)

> **Note**
>
> - This JSON file is intended as a means to address the transparency requirements in the upcoming EU AI Act (focus on Article 13 & 52).
> - The file is an illustrative example as the basis for discussion and feedback.
> - To use the file, copy the template and insert the values of the AI System at hand, using the descriptions given in the template as a guidance).
> - The file is not a formal JSON Schema, but we may adopt the schema in the future for improved automated processing.

Call to action

- Please share your feedback in GitHub Discussions.
- See the call for contributions at the end of this document.

https://github.com/aai-institute/AI-SBOM

```
$ git-ai blame src/auth.ts
a731df7 Lois Tam          1) export async function login(email, password) {
a731df7 Lois Tam          2)   const user = await db.findByEmail(email)
a731df7 Lois Tam          3)   if (!user) throw new AuthError('not found')
b9c4e22 Cursor | Opus 4   4)   const valid = await bcrypt.compare(
b9c4e22 Cursor | Opus 4   5)     password, user.passwordHash
b9c4e22 Cursor | Opus 4   6)   )
b9c4e22 Cursor | Opus 4   7)   if (!valid) throw new AuthError('bad creds')
a731df7 Lois Tam          8)   return issueSession(user)
a731df7 Lois Tam          9) }
```

AI Blame

Git AI links each line of AI-code to the agent, model and prompt that generate
and their agents understand the "why" behind every line.

https://usegitai.com/

Today's Agenda

01 ✅ Generate — create & curate context

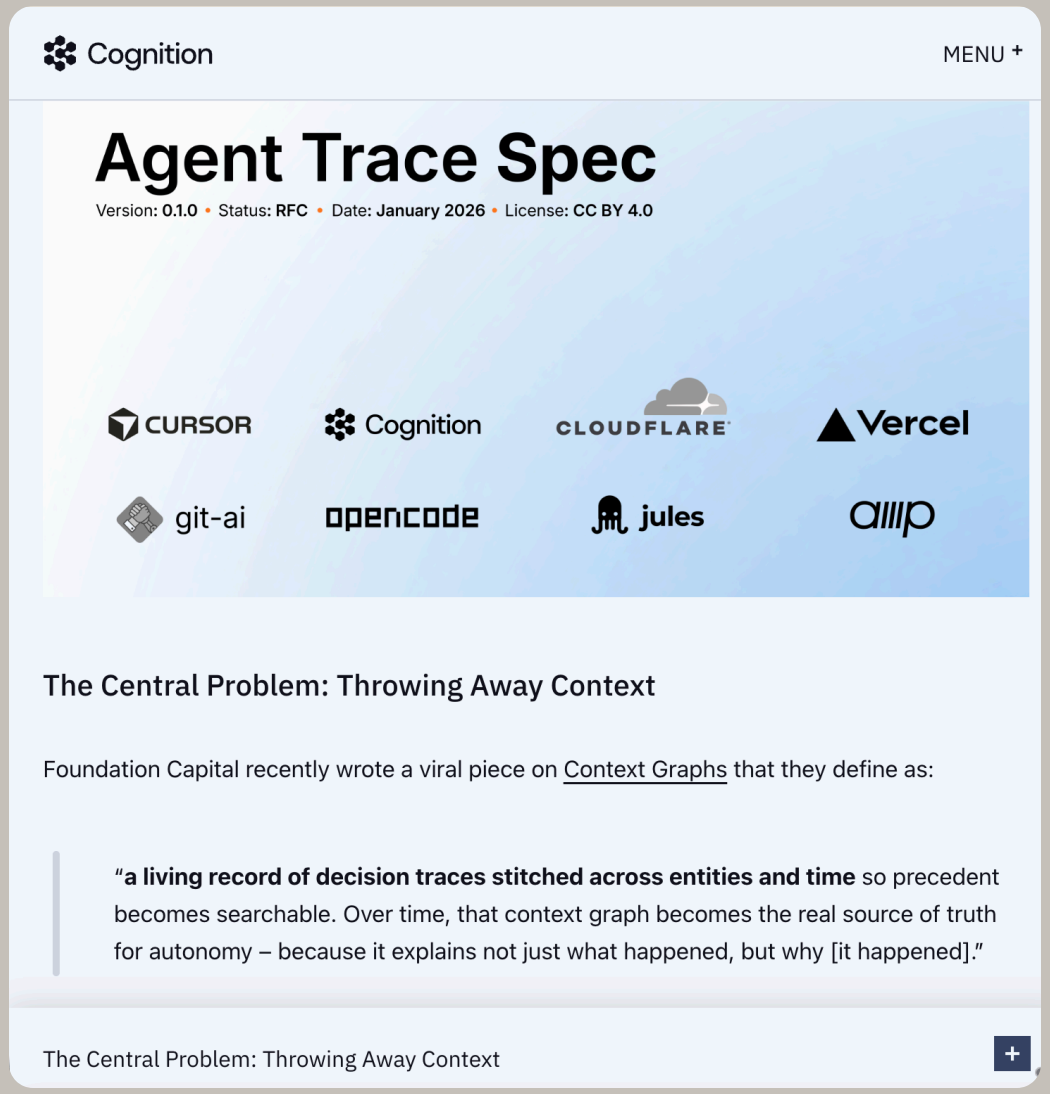
02 ✅ Evaluate — test & measure context quality

03 ✅ Distribute — package & share context

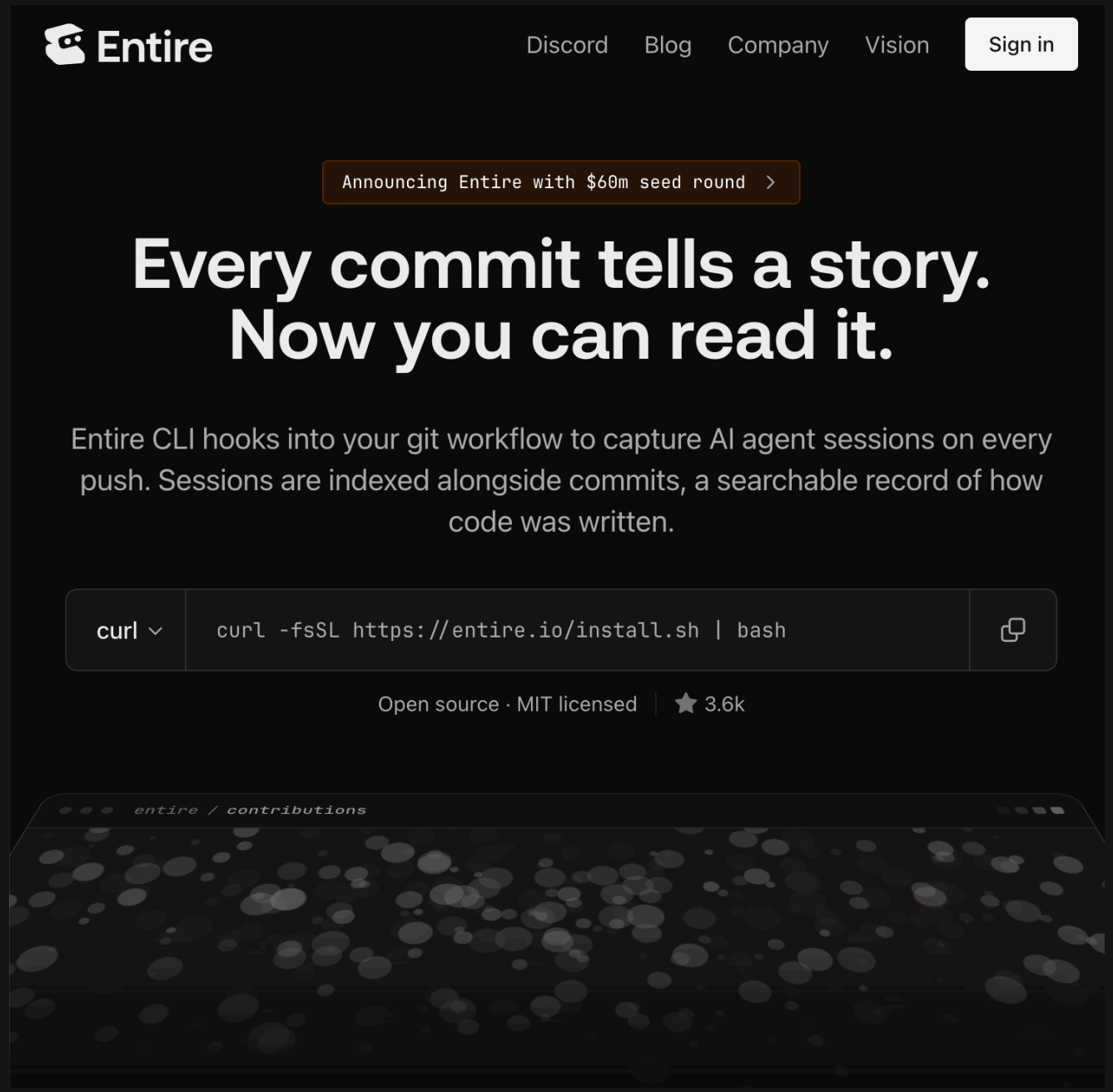
04 👁️ **Observe — monitor & improve in production**


4.Observe

Standardized Agent logs



https://cognition.ai/blog/agent-trace



https://entire.io/

Context from production logs

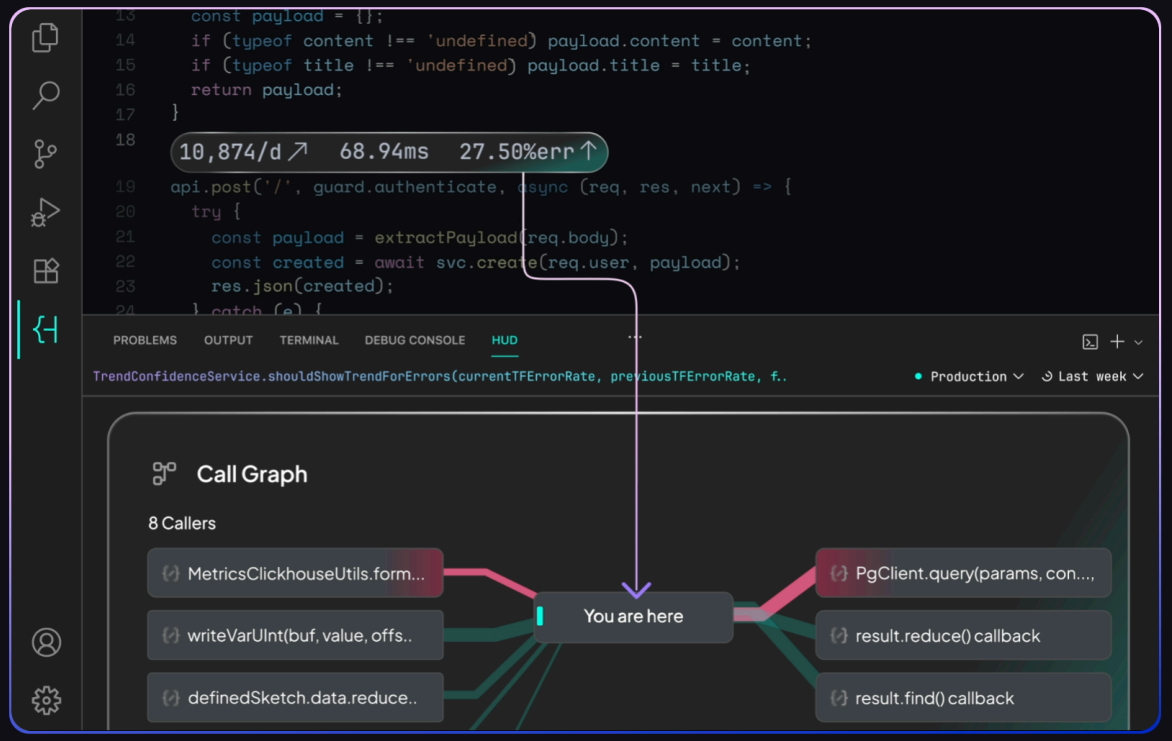
{Hud Blog Docs About us Book a demo Log in ↗

Where code meets reality

Install once. No config. Zero maintenance.

Understand code behavior with function level data

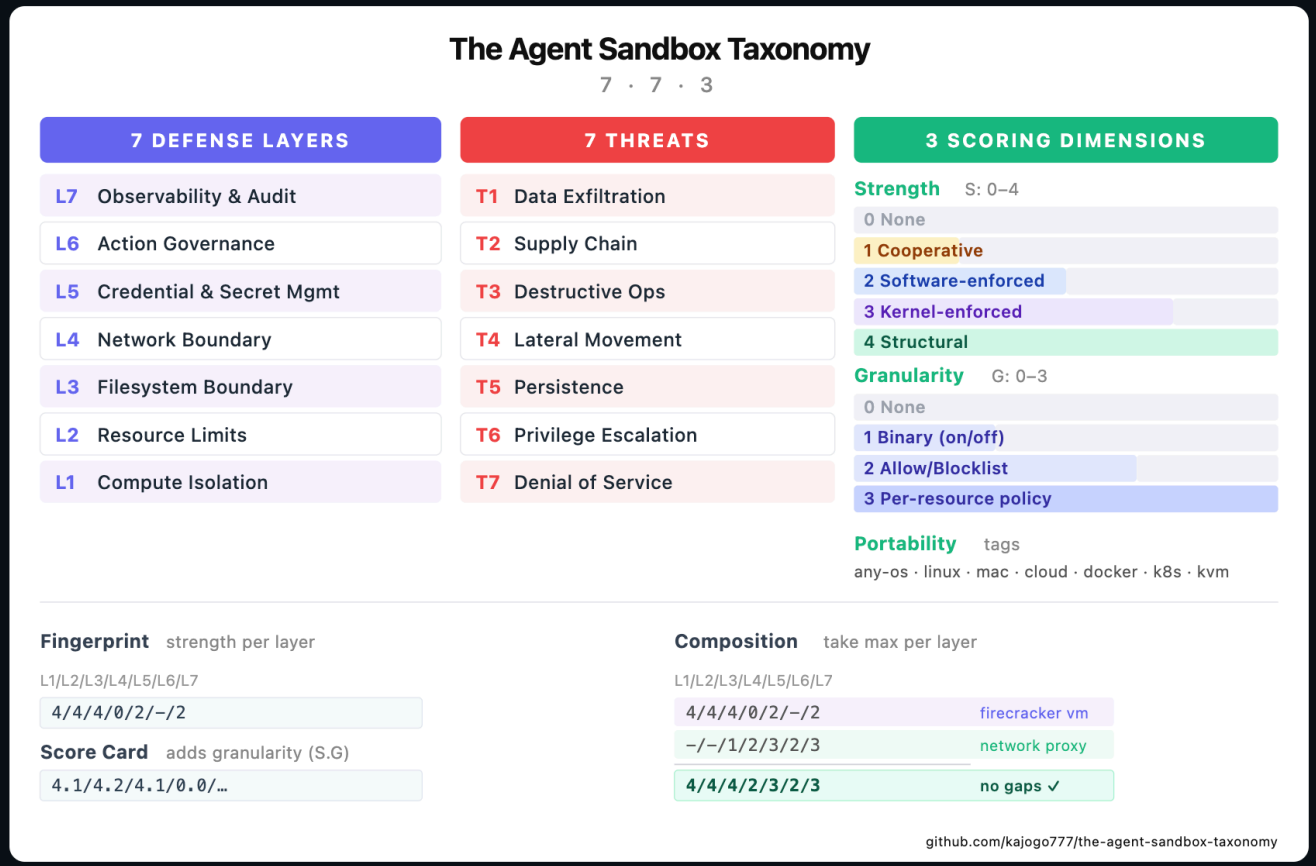
Hud gathers errors and performance data at the service and function level. It connects, at runtime, the business impact and the root cause in the code. Engineers use this data in the IDE to understand how their code behaves in reality.



https://www.hud.io/

Agent sandboxing

An open taxonomy and scoring framework for evaluating AI agent sandboxes. It decomposes sandboxing into **7 defense layers**, maps them against **7 threat categories**, and scores each mechanism on **3 dimensions** (strength, granularity, portability), producing comparable fingerprints for any product. Includes score cards for 23 sandbox tools, a composition framework for stacking complementary products, and a decision checklist for choosing the right sandbox stack.



https://github.com/kajogo777/the-agent-sandbox-taxonomy/tree/main

Context Filter ~ WAF

README

context-filter (macOS)

DYLD_INSERT_LIBRARIES library to detect and warn about prompt injection in Claude Code instruction files on macOS

The Problem

When you clone a repository, Claude Code automatically loads `CLAUDE.md` and skill files into context **before any hooks fire**. These files can contain prompt injection attacks that manipulate Claude's behavior.

This library intercepts file reads at the syscall level using `DYLD_INSERT_LIBRARIES`, scans for injection patterns, and prepends a warning to suspicious files—all before the content reaches Node.js/Claude.

Two detection layers:

- **Regex patterns** (built-in, always active) — fast pattern matching against known injection signatures based on Lasso Security's research. Configurable via `cf-module/config/patterns.json`.
- **ML scanner sidecar** (optional) — when the scanner daemon is running, the library also sends file content to a prompt injection ML model (LLM Guard or NeMo Guardrails) for deeper detection. Falls back to regex-only if the daemon is not running.

Context filter - https://github.com/jedi4ever/context-filter

Today's Agenda

01 ✅ Generate — create & curate context

02 ✅ Evaluate — test & measure context quality

03 ✅ Distribute — package & share context

04 ✅ Observe — monitor & improve in production

Unfinished thoughts

- What will chaos engineering look like for context ?
- And context analytics & Context SEO
- How will A/B testing work ?
- Who will be the next Github or Kubernetes for context ?
- Will Agents not do all this autonomously ?
- Will Agents invent their own context language ?


Context is the fuel.
Coding agents are the engine
— Patrick Debois

🙏 Thanks for listening !

Connect on LI for the slides send me feedback