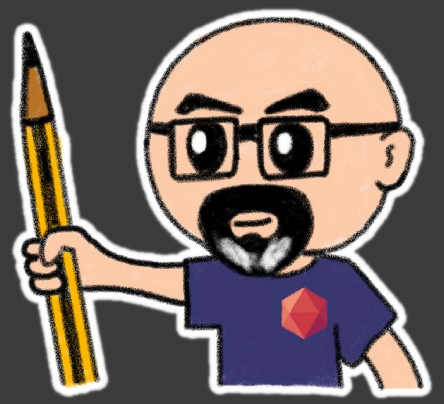
DEVOXX BELGIUM

Wrap, Reshape, or Redesign

Retrofitting Your APIs for a World of Agents

Horacio González 2026-09-26

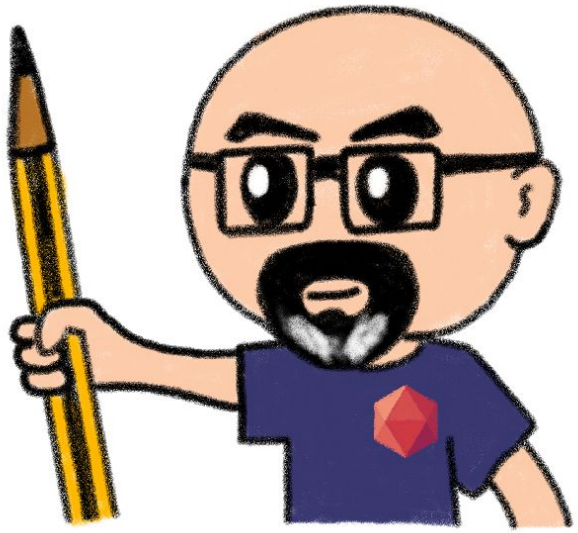


@ Lost In Brittany

clever cloud

Who are we?

Introducing myself and introducing Clever Cloud



Horacio Gonzalez

@LostInBrittany

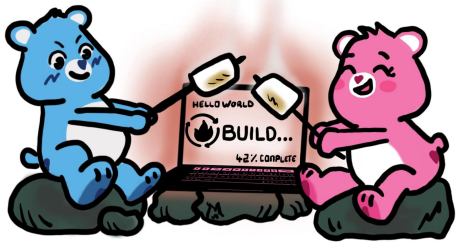
Spaniard Lost in Brittany

Head of DevRel

clever cloud



Finist Devs



Clever Cloud



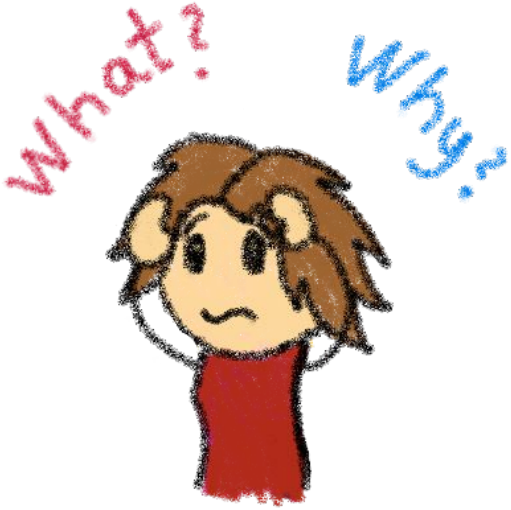
clever cloud

- European PaaS
- You push, we build, we run
- Applications and add-ons
- An API with a long history



Watch it fail

Before we name anything

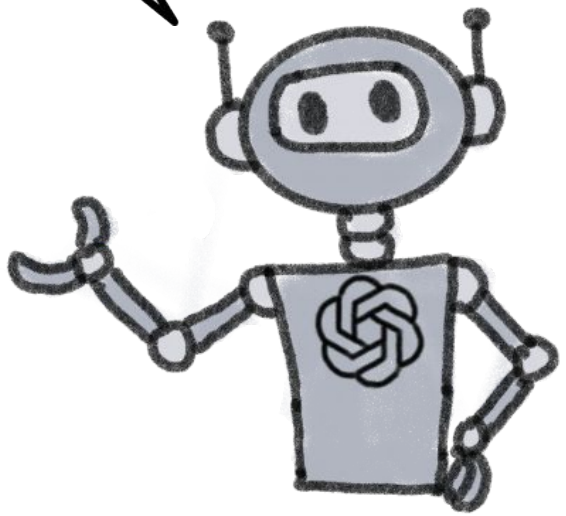


A simple ask

Set Java version to 25 on my application



```
Environment variable CC_JAVA_VERSION
updated successfully, Java version
set to 25
```



The follow-up

Is the application running
on Java 25 now?



No

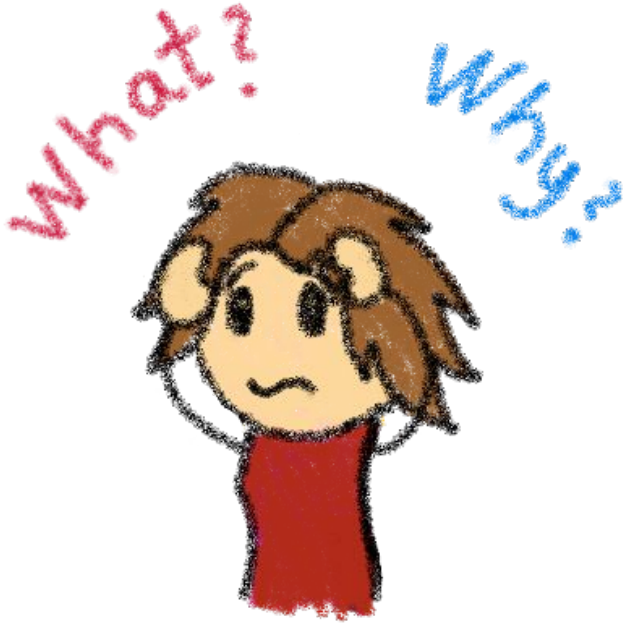
HTTP 200 ☑

Schema validation passed ☑

Authentication succeeded ☑

The API did exactly what it promised ☑

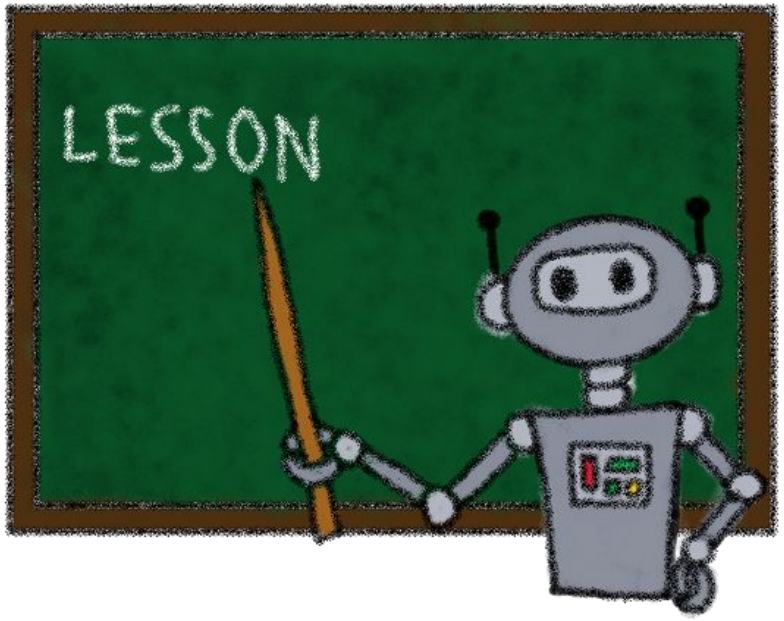
The user got what they asked for ☒



The variable is set

The running instances never saw it

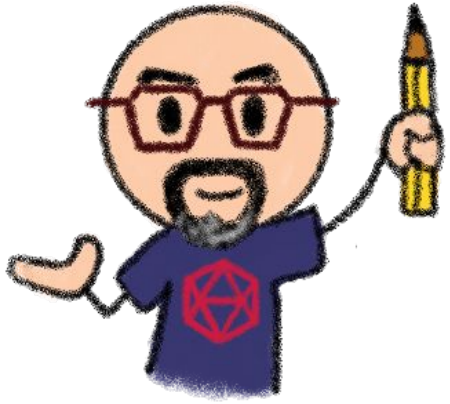
Not a dumb model



The agent did the correct thing

Any developer would know to **restart the instances**

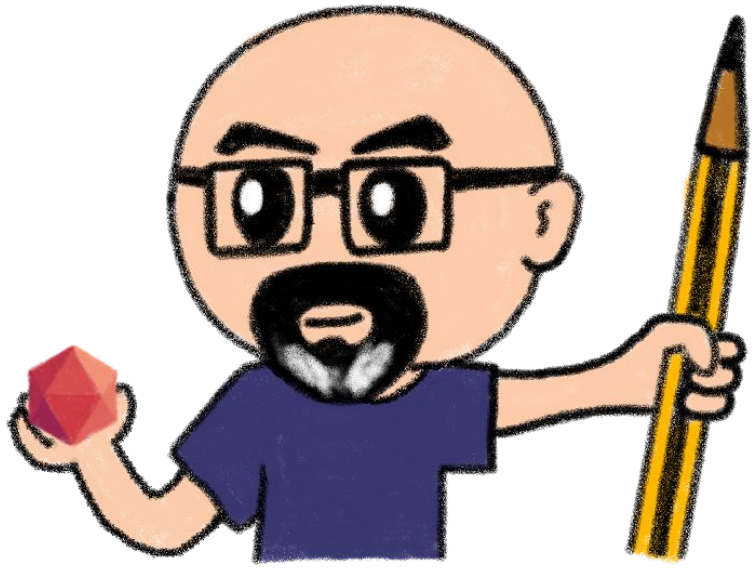
The API never said so



Agents are the users your API was never designed for

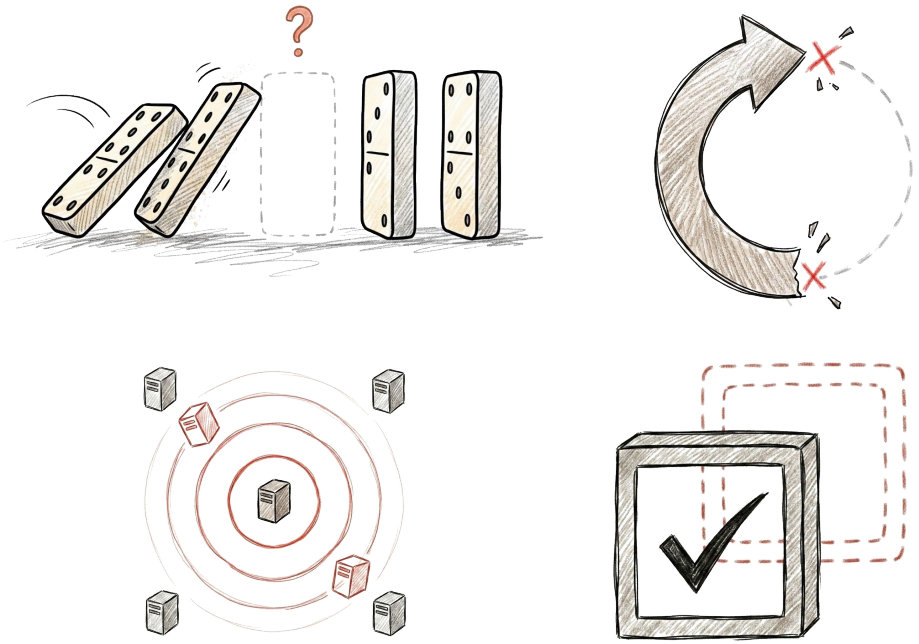
The next 40 min

- One platform: ours, to keep it real
- Three failures, each worse than the last
- Three different answers



The semantic gap

From one failure to the shape of the problem



Silent contract

Humans read docs

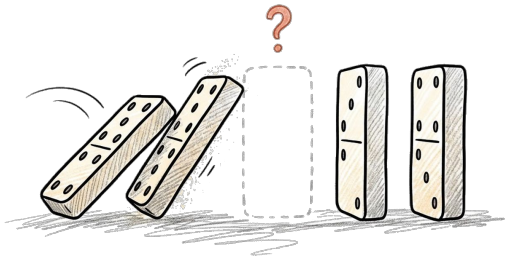
Humans infer conventions

Humans recognise danger

None of it was ever written down

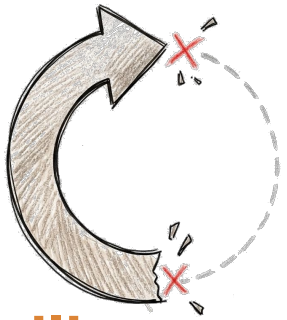


Four questions



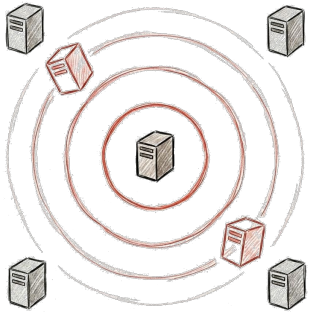
Dependencies

What else must happen, in what order?



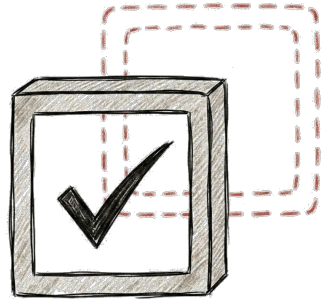
Reversibility

Can I undo this?



Blast radius

What else does this touch?



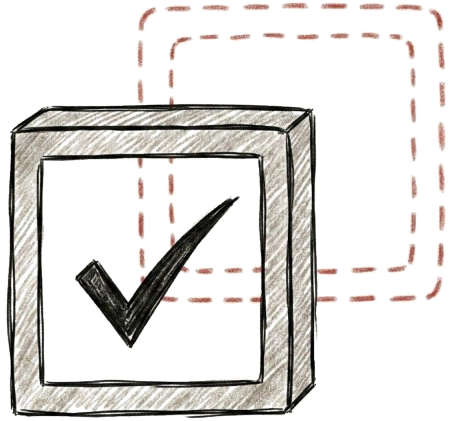
Completion

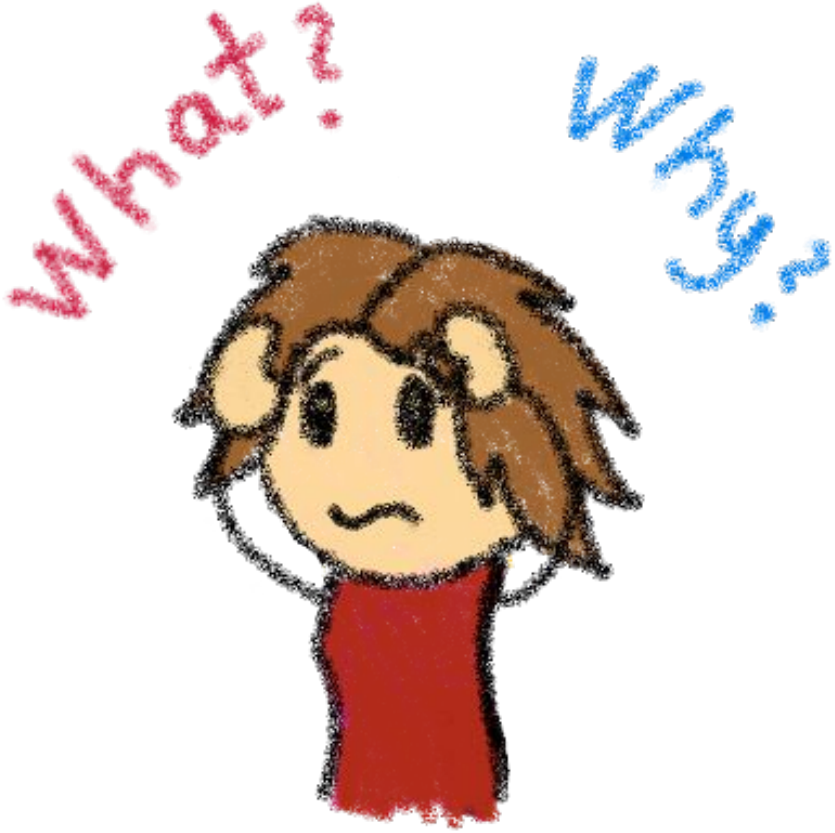
When this returns, is it done?

Back to the demo

- The call returned success
- The application was unchanged

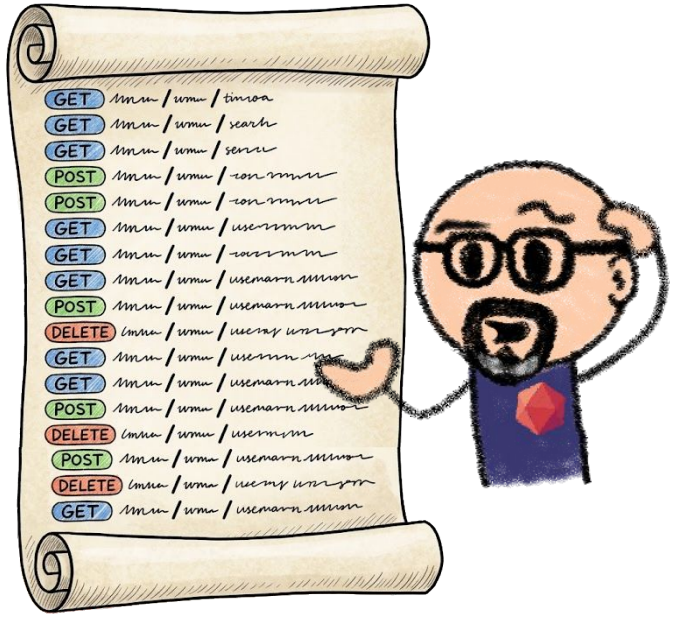
A **Completion** failure





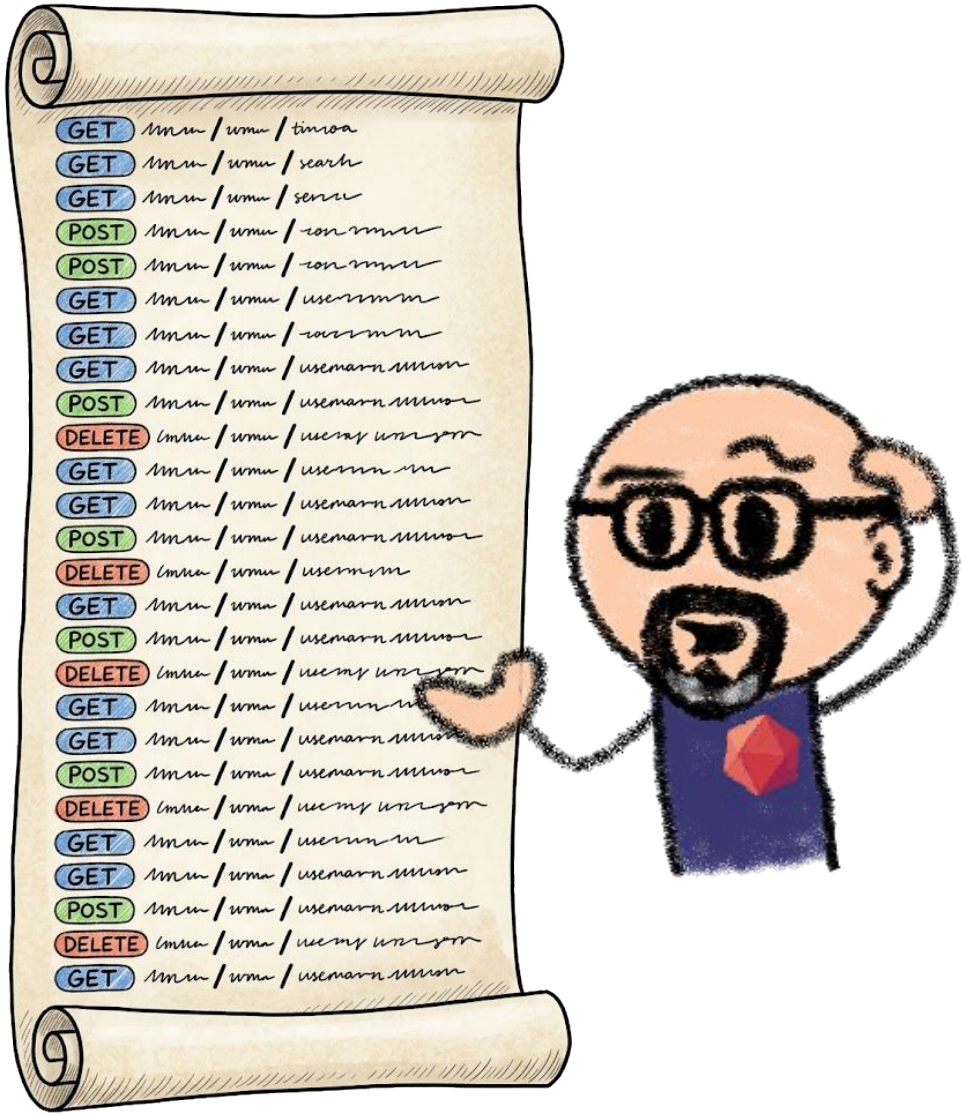
Our own spec

310 operations, 0 descriptions...
267 with no summary either



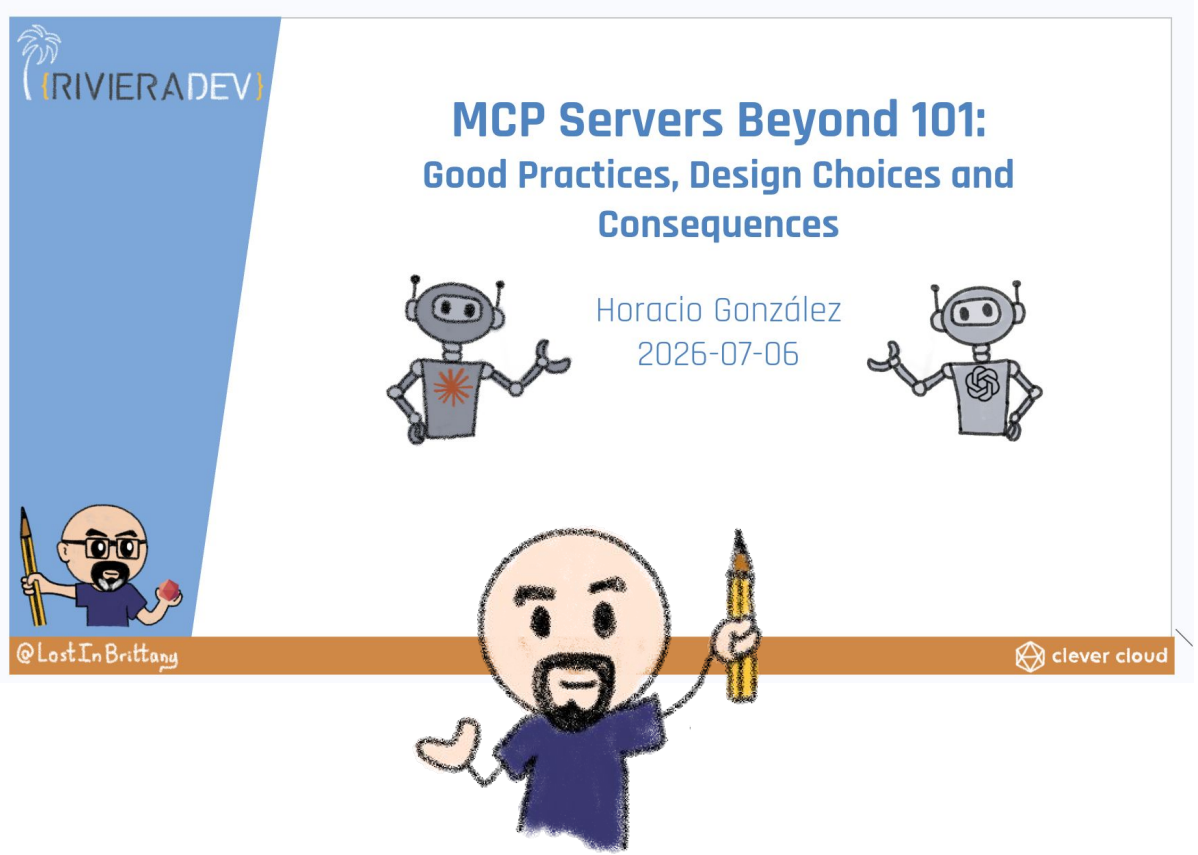
A confession

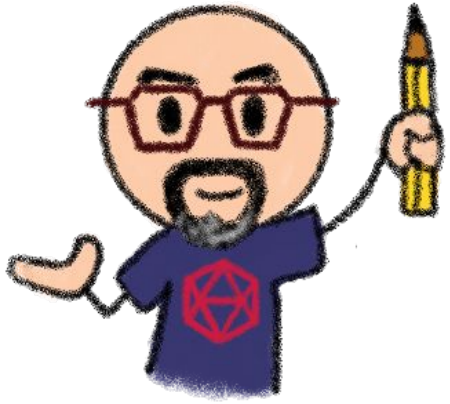
- We wrote this
- It was fine
- **Because humans were reading it**



Not an MCP talk

- It does not decide what the right tools are
- That is an API design problem
- **And it is yours**



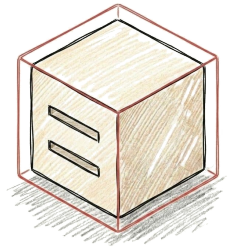


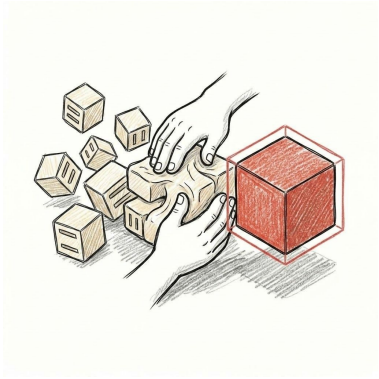
Schema validation prevents malformed decisions

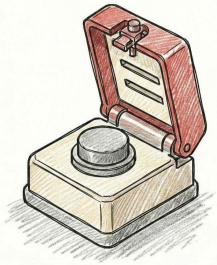
It doesn't prevent bad decisions

The framework

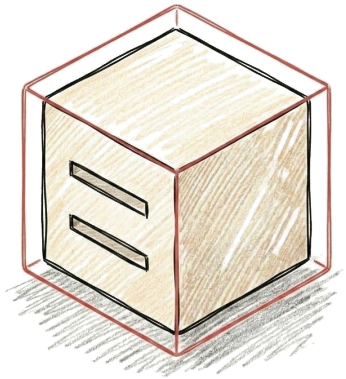
And its shelf life





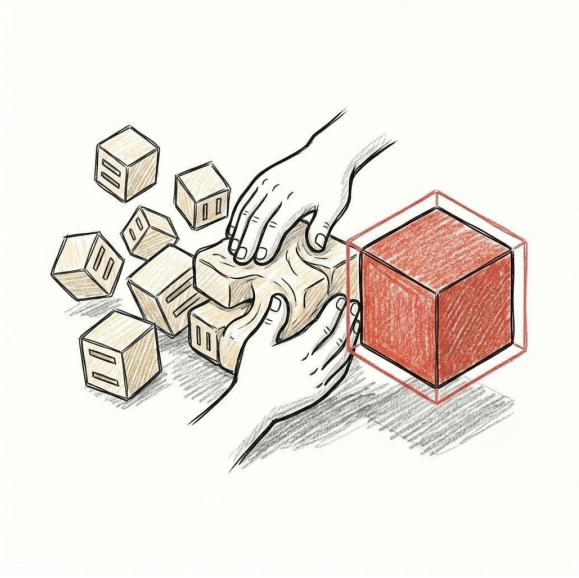


Three options



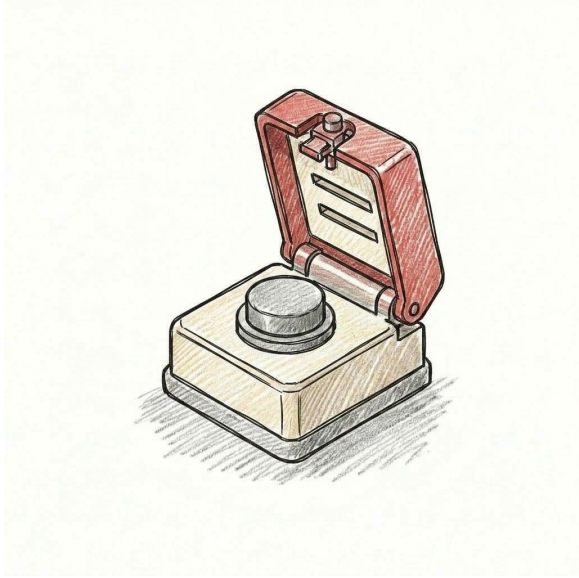
WRAP

Expose as-is



RESHAPE

Intent-level tools



REDESIGN

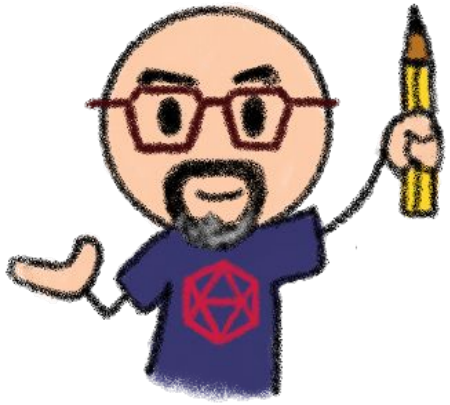
The capability was wrong

How big is the gap?

Use	When
WRAP	knowledge is already explicit
RESHAPE	knowledge exists, scattered
REDESIGN	there are unstated invariants

The options are defined by **semantic gap**,
not by how much work they cost



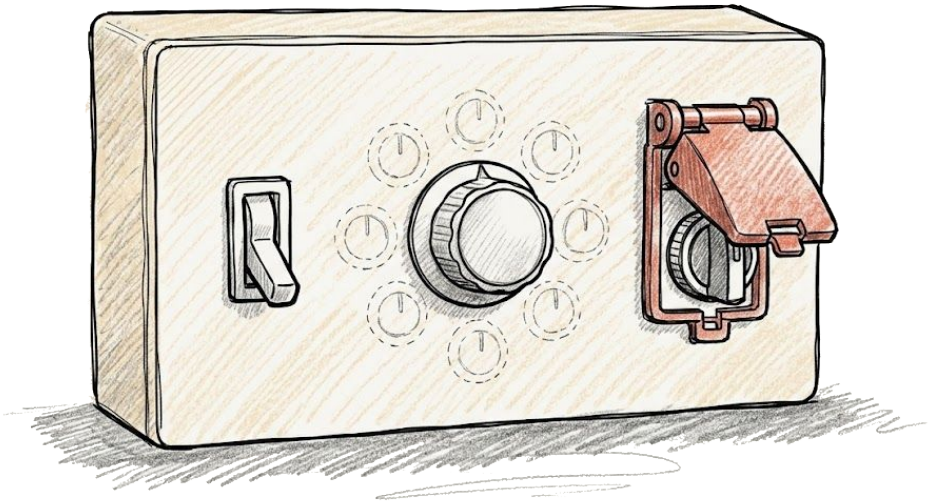


Whichever option you take, you still have to describe it

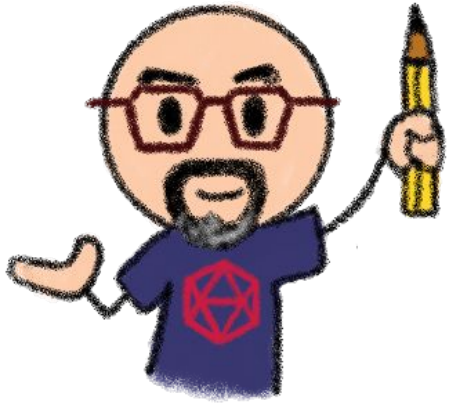
The 3 options ask for **meaningful descriptions**

Classify each capability

- Not an option per API
- One platform has all **three at once**
- That is normal, not a migration



Before I sell it to you



This talk is given in 2026-10-07

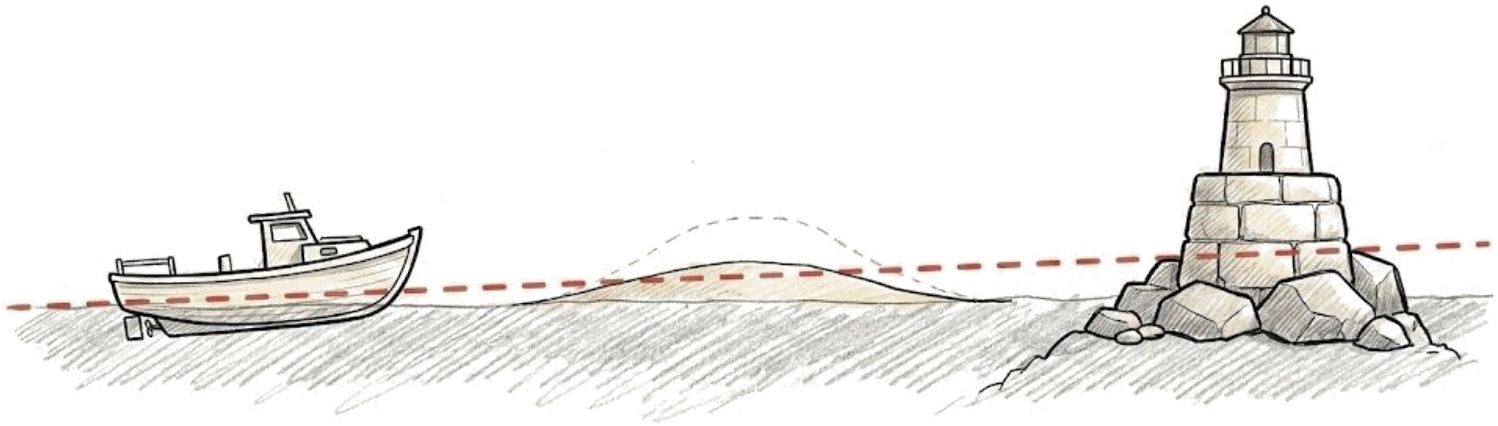
Advises may be outdated in a few months!

Everything I'm about to show you rests on one assumption:

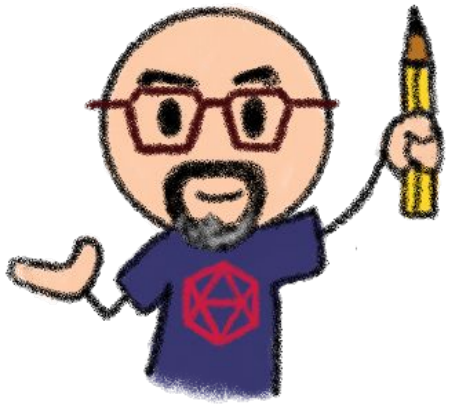
that **today's** models need these abstractions

Shelf life

WRAP	gets better as models get better
RESHAPE	may shrink as models get better
REDESIGN	stays



Why Redesign stays



A better model makes fewer mistakes

It doesn't make them cost less

WRAP

Boring... and boring scales

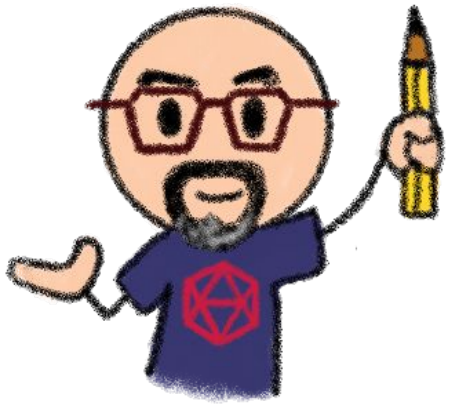


The condition

- Everything the caller needs is explicit
- Typed, bounded, introspectable
- Nothing destructive in reach



The rule



Wrapping isn't bad

Blind wrapping is bad

Not one tool per endpoint

- The reflex: one tool per operation
- **Code mode**: `search` to discover, `execute` to run
- The API stays whole, the agent composes





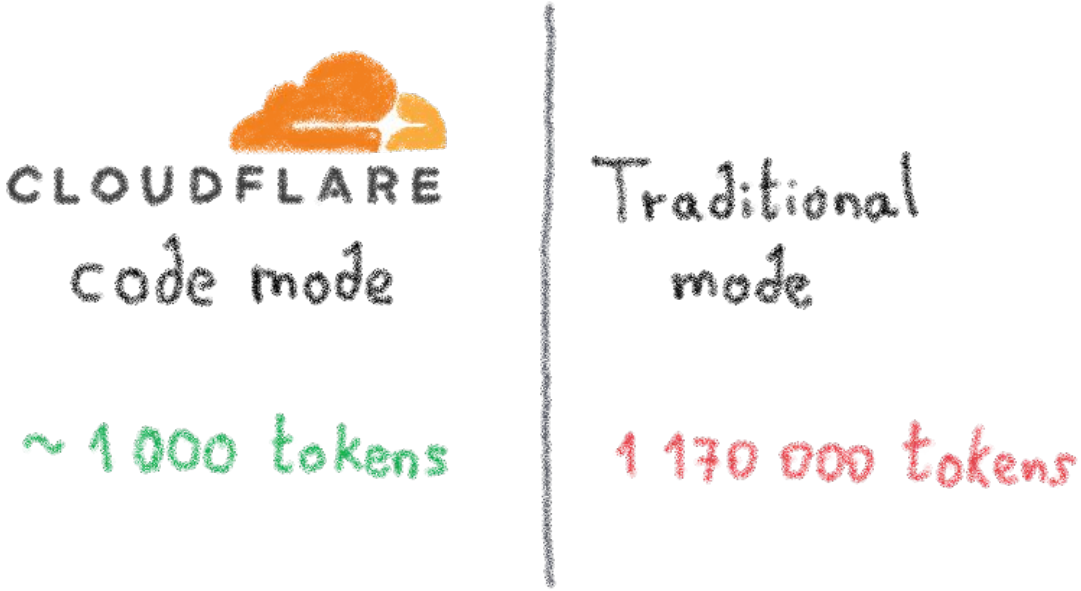


The Cloudflare Case

- **Code Mode MCP**
  - 20 Feb 2026
- **2 tools**
  - `search()` and `execute()`
  - run the generated code inside a Dynamic Worker isolate
- Over **2 500 endpoints**



The token cost



Same 2 500+ endpoints

Wrapped differently

Why code mode works

Making an LLM perform tasks with tool calling is like putting Shakespeare through a month-long class in Mandarin and then asking him to write a play in it.

It's just not going to be his best work.

Code Mode: the better way to use MCP
Kenton Varda and Sunil Pai

Our Wrap case



clever cloud

mcp-simple-server

`search`, `execute`, and `doc`

https://github.com/CleverCloud/mcp-simple-server



Deliberately dull

Before

- The whole API, undescribed
- The docs, in a browser

After

- The whole API, searchable
- The docs, as a tool

Same surface, nothing redesigned



The AWS case

- **AWS MCP Server**
  - GA 6 May 2026

- **~8 tools** over **15 000+ operations**

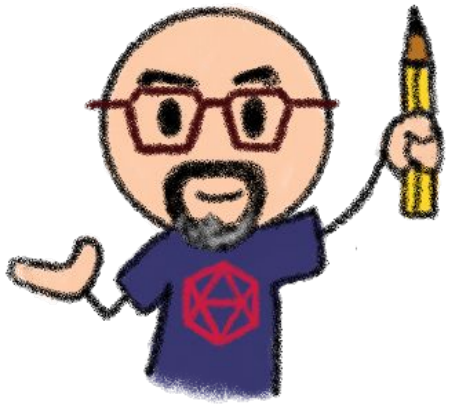


Easy Maintenance



"When we will launch new APIs, they will be supported within days"

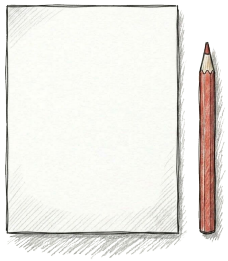
The Wrap rule



Wrap works when the contract is already explicit

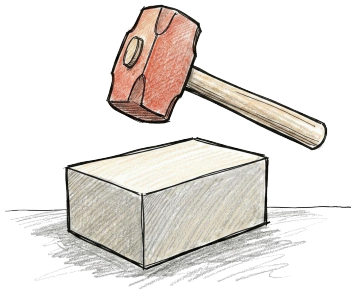
Familiarity is an amplifier, not the condition

MCP tool annotations change the rule?



`readOnlyHint`

I don't change anything



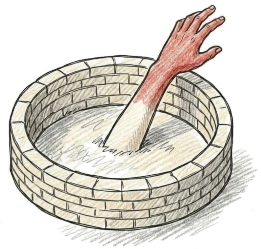
`destructiveHint`

I might destroy, not just add



`idempotentHint`

Calling twice changes nothing more



`openWorldHint`

I reach outside your system

Nobody honours tool annotations

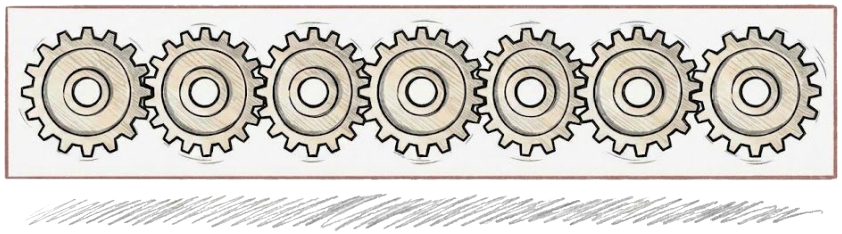
The spec's own words, not a blog post:

"all properties in `ToolAnnotations` are hints"

Clients **must** treat them as untrusted

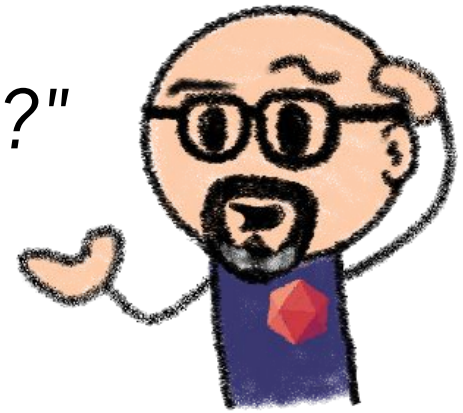
RESHAPE

When the workflow lives in your head



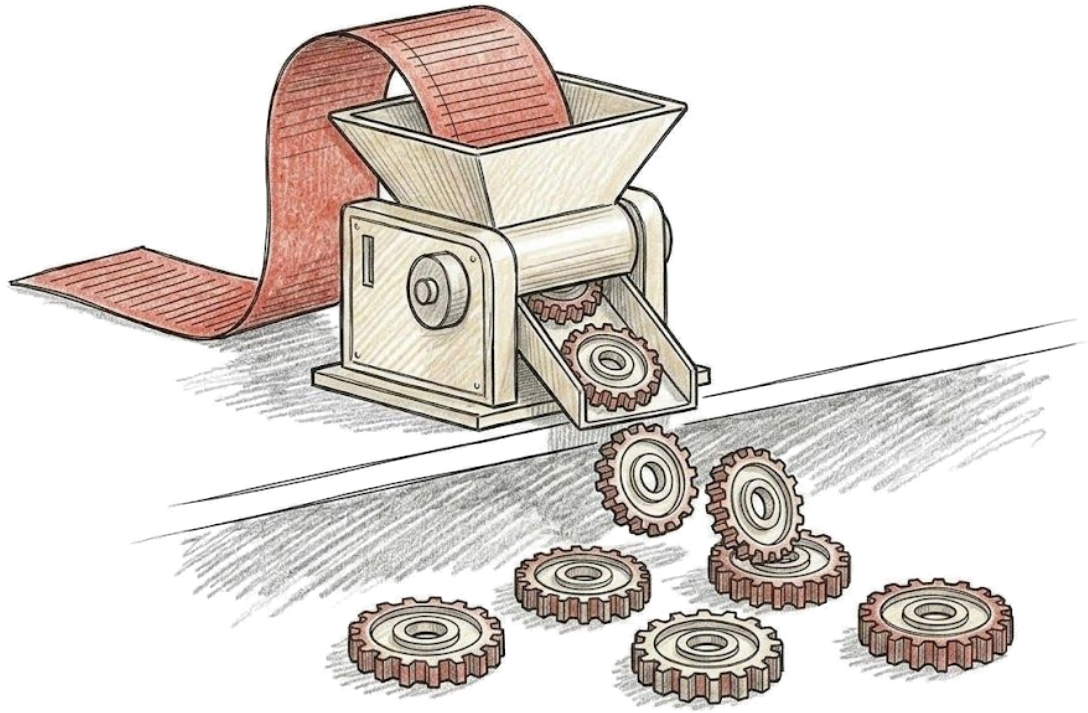
Failure two

"Why isn't my application responding?"



Seven tools

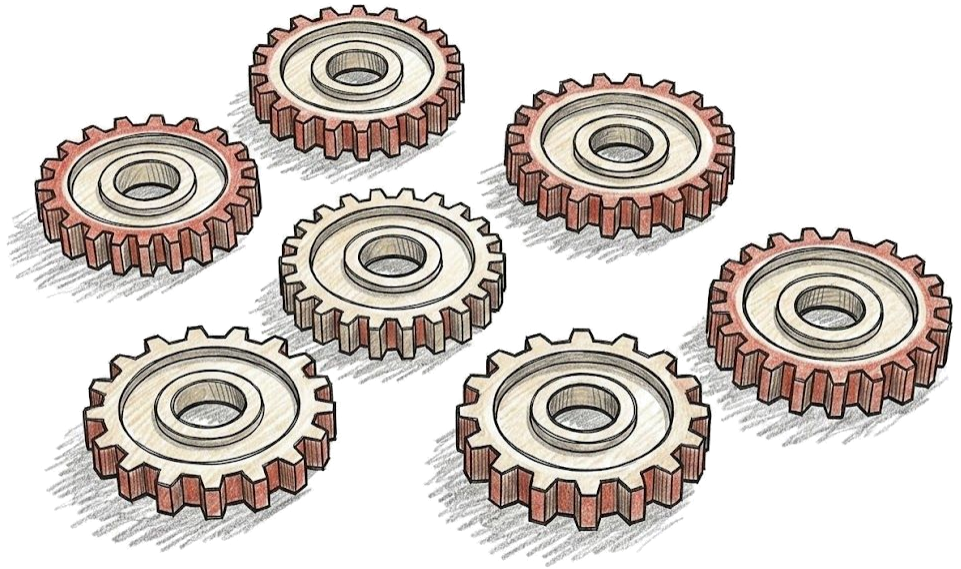
- `get_application`
- `get_deployment`
- `get_logs`
- `get_scaling_configuration`
- `list_instances`
- `list_deployments`
- `list_environment_variables`



Directly exported from the Open API / Swagger

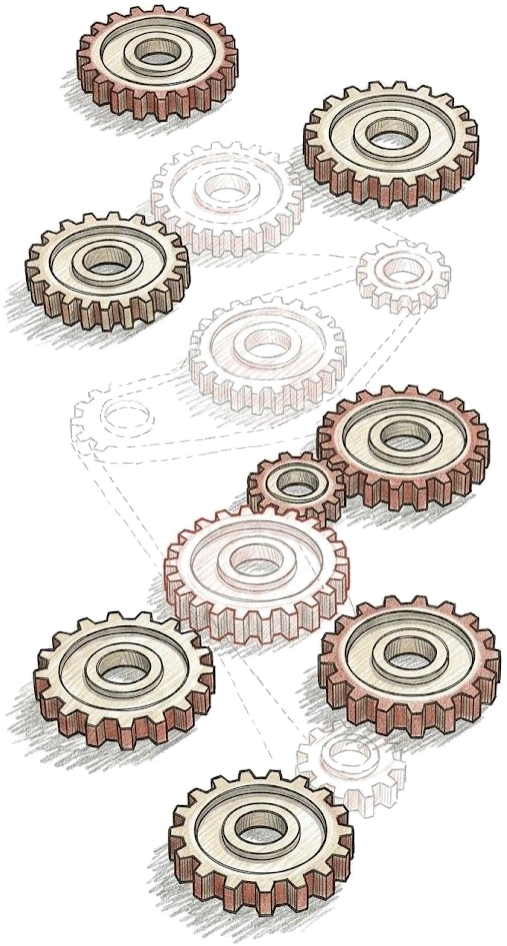
Nothing is wrong

- Each one is correct
- Each one is well-formed
- Each one does what it says



The agent invents the workflow

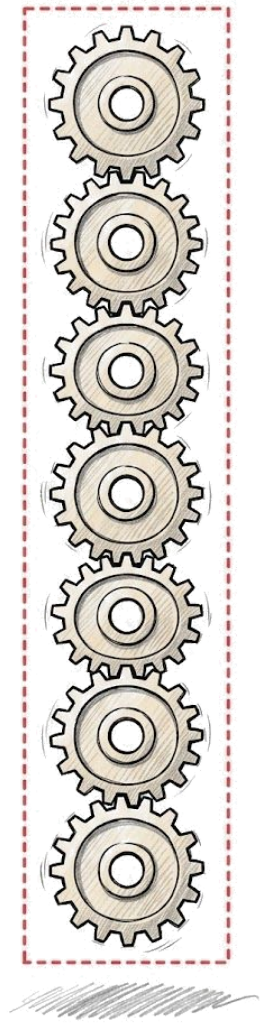
- Which one first?
- Do I need deployments at all?
- How many logs is enough?
- Does `STOPPED` mean broken, or deliberate?
- …



The knowledge exists, it is simply nowhere in the API

A Dependencies gap

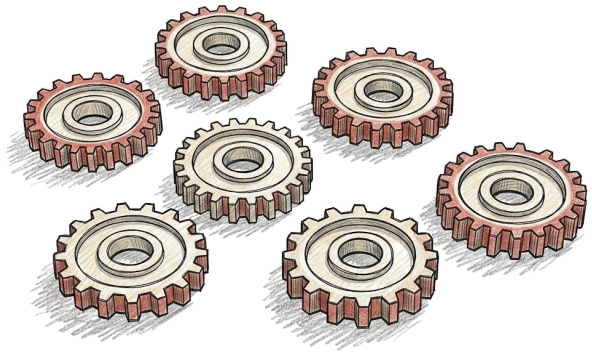
- The calls have an order
- Nothing in the API says what it is
- **That order belongs inside the tool**



The reshape

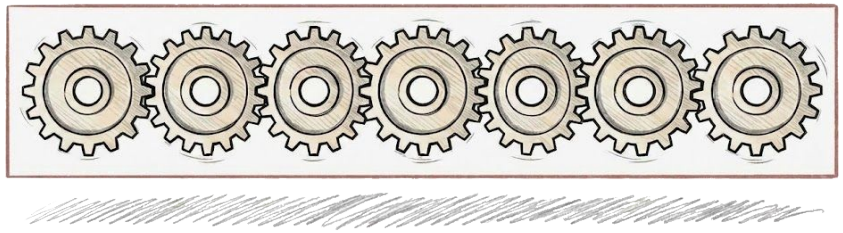
Before

- Seven **endpoint-shaped** tools
- The model orchestrates



After

- `diagnose_application(id)`
- One **outcome-shaped** result



The seven tools can stay

One synthesized result

```
// what state is it actually in
status

// the comparison a human would have made
instances: desired / running / healthy

// "do I need deployments at all?" answered FOR it
latest_deployment

// "how many logs is enough?" answered FOR it
recent_runtime_errors

// the judgement, already made
configuration_warnings
```

The Postmark case

Email delivery platform, transactional email



- One `diagnoseDelivery` function
- Replaces the five calls a human would chain
- Fanned out in parallel
- Tolerant of any single lookup failing

Their words

Postmark

"Collapse the multi-step human workflow into

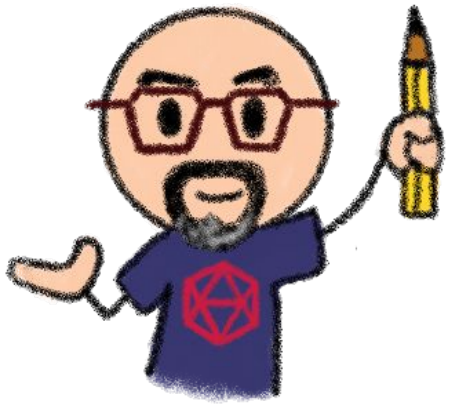
***one outcome-shaped call**,*

rather than exposing the steps and hoping the model

reconstructs the workflow correctly every time."



The principle



Make the **model decide** what requires **judgement**

Make **software do** what requires **computation**

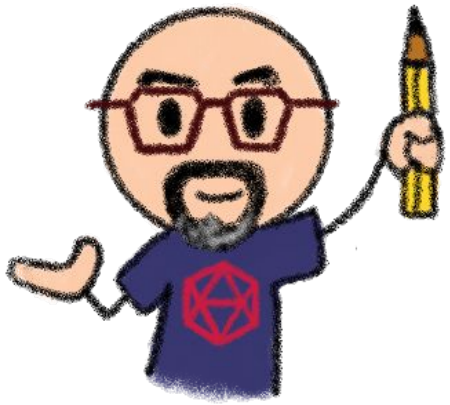
Ordinary code

- If/else, for-loop, while-loop…
- Joining, filtering, ranking…
- Picking the latest, picking the largest…
- …



Deterministic, perfect, free

Making the intent clear to the agents



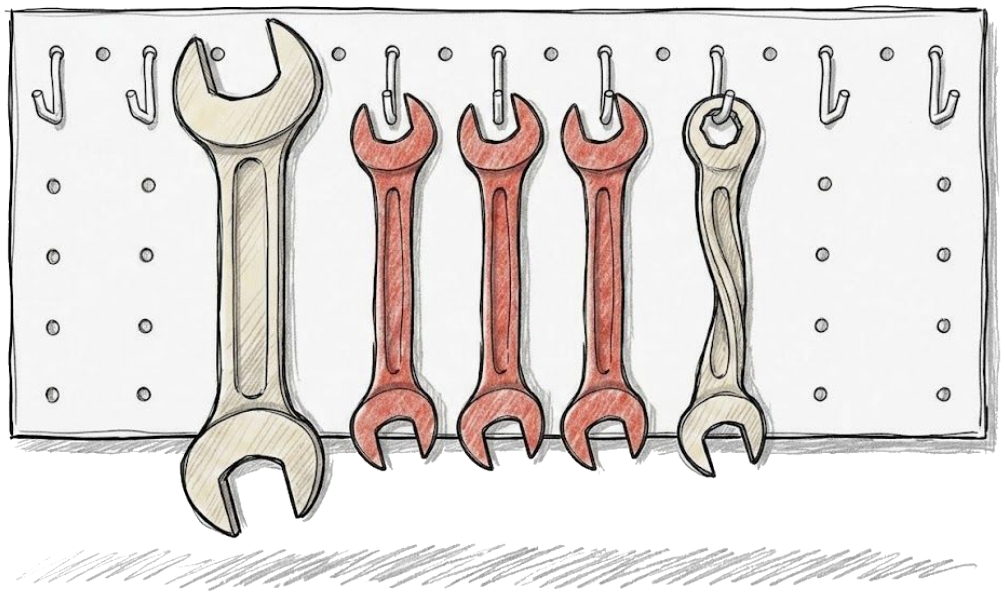
You're not removing capability

You're **changing the unit of affordance**

Affordance: (noum) the quality or property of an object that defines its possible uses or makes clear how it can or should be used

And how many tools are needed?

- *"More tools means worse selection"* is contested
- What degrades performance is **confusable** tools
- Not **numerous** ones



Opposite directions, same principle

GitHub

- 101 → 52 tools



Postmark

- 4 → 24 tools

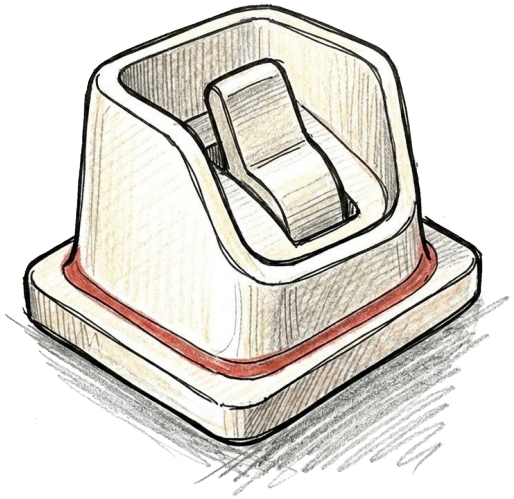


Not a 'many tools' vs 'few tools' problem

A from **endpoint-shaped** to **outcome-shaped** problem

REDESIGN

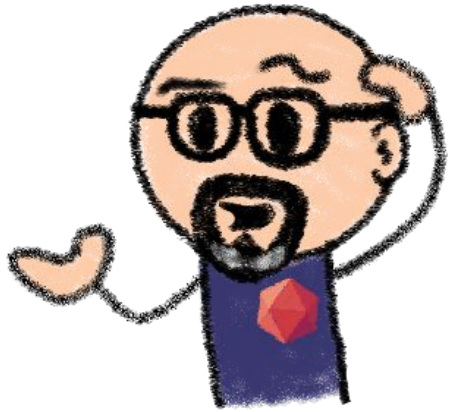
When no tool layer saves you



Two very similar endpoints

```
DELETE /organisations/{id}/applications/{appId}/addons/{addonId}
```

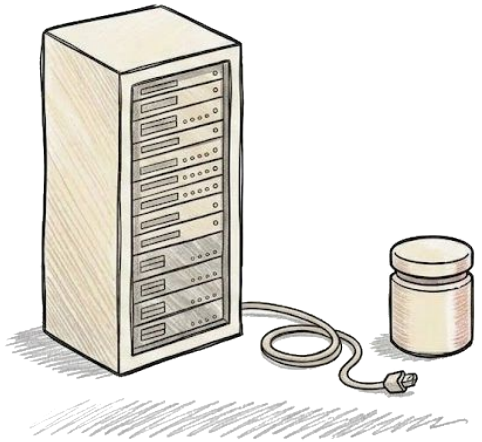
```
DELETE /organisations/{id}/addons/{addonId}
```



One unlinks, one destroys

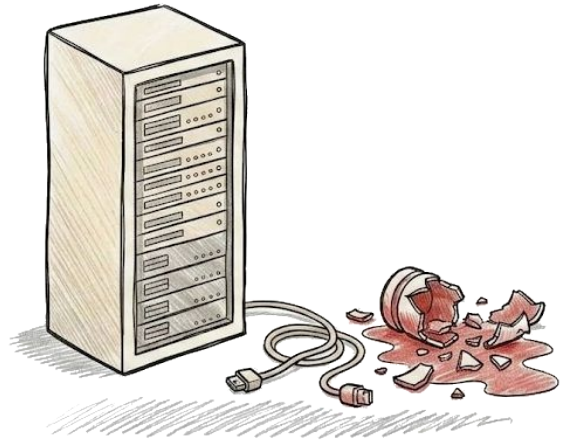
Scoped to the app

- Unlinks the add-on



Scoped to the org

- Destroys it, and its data



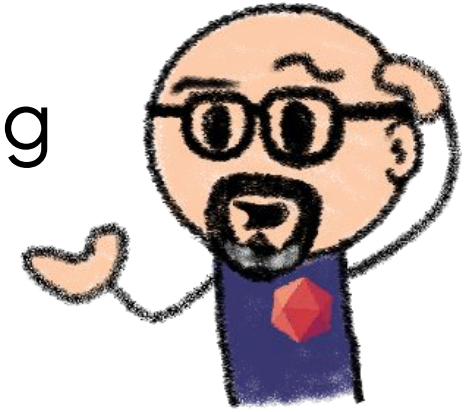
Same verb, same resource

One path segment apart

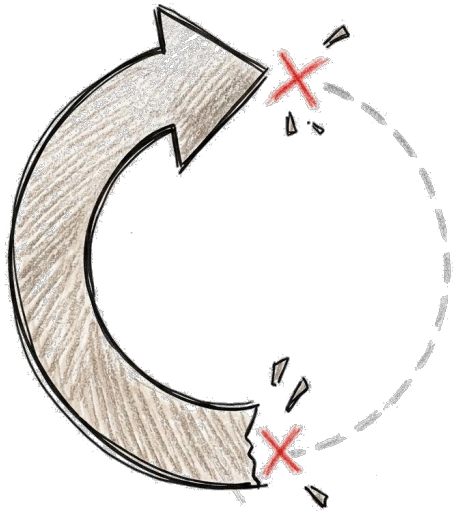
And even worse

The safe one has a summary

The destructive one has nothing

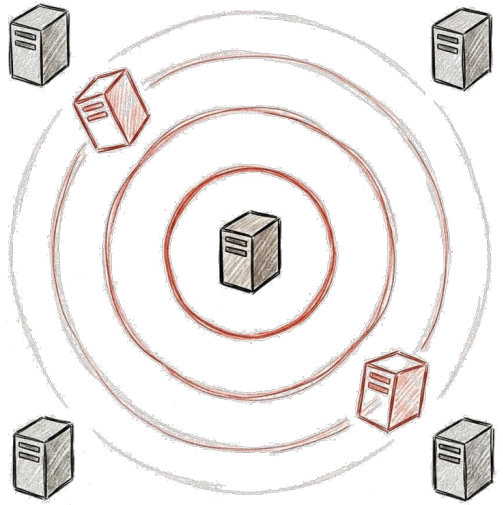


Two questions



Reversibility

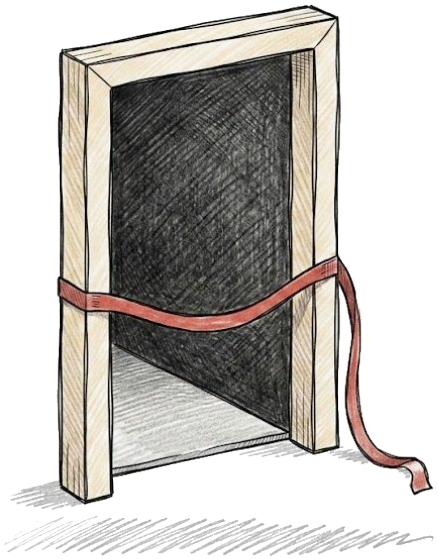
Can I undo this?



Blast radius

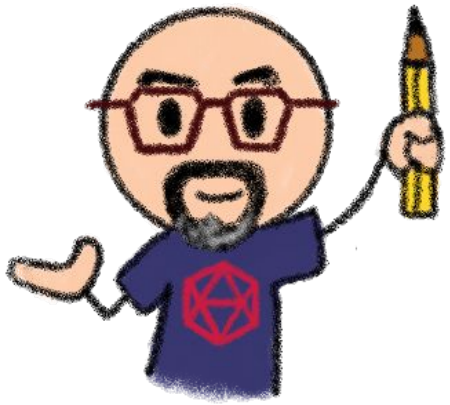
What else does this touch?

What to do? Just describe it?



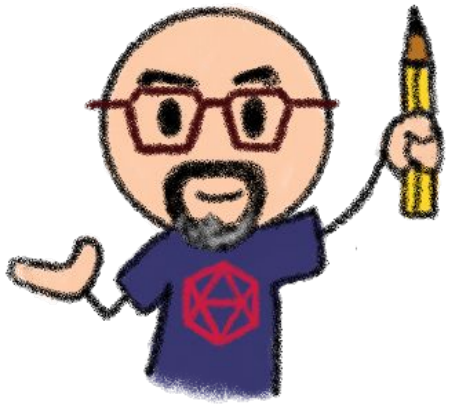
WARNING! THIS DELETES THE DATABASE!!!

No



That is **not an architecture**

The distinction

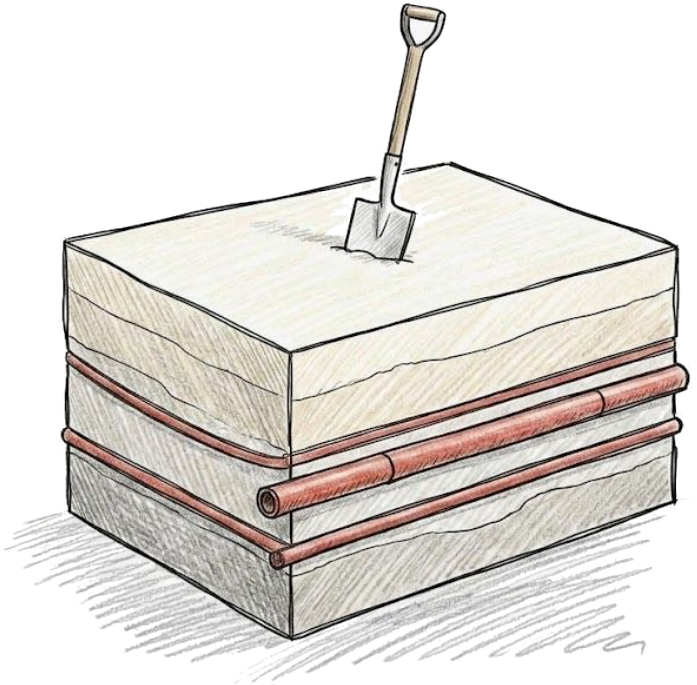


Reshape changes the **abstraction**

Redesign changes the **invariant**

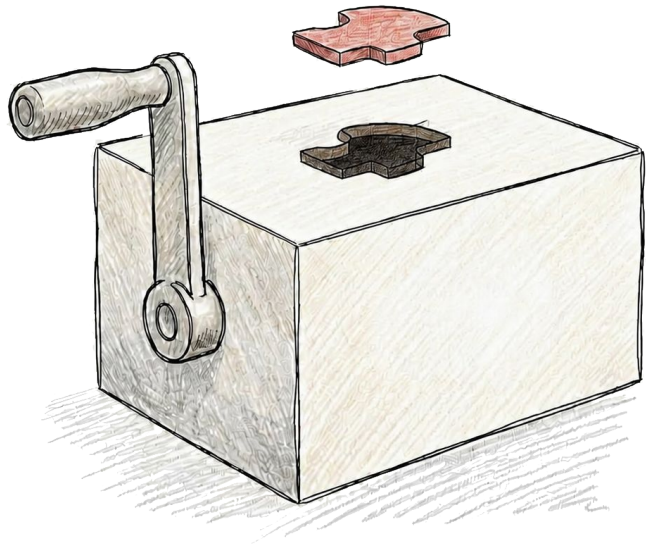
Prepare

```
prepare_addon_deletion(addon_id)
   addon
   applications_using_it
   latest_backup
   recoverable_until
   consequences[]
```



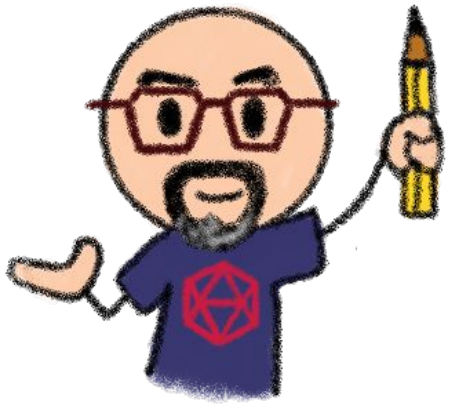
Then execute

```
execute_addon_deletion(plan_id)
```



prepare → inspect → execute

Who owns safety



The system owns the safety boundary

Not the model

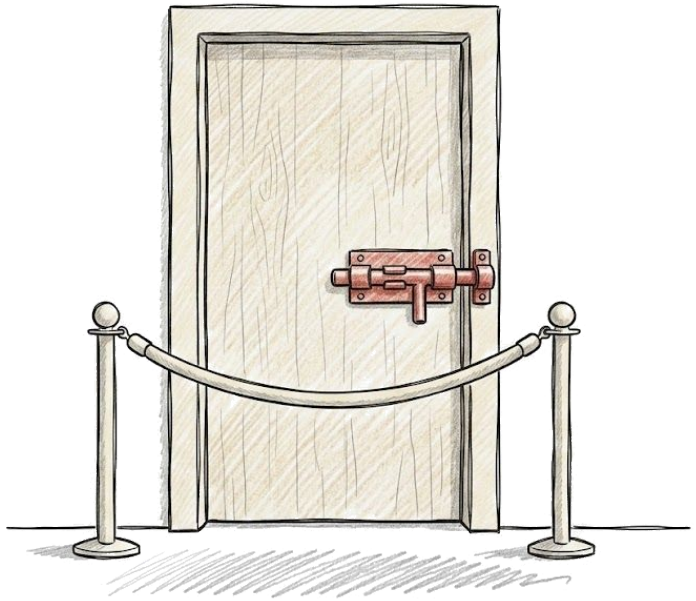
Stop trying to build a model that cannot be fooled.
Build the system around it, so that when the model is fooled,
and it will be, nothing important breaks.

Steve Wilson and Rock Lambros
OWASP Project Leads



Instructions vs enforcement

- Never make the LLM your authorization layer
- Prompt **instructions are UX**, **enforcement** belongs in **software**



How about the restart endpoint?

Before

- `POST /deployments`

  `202 Accepted`

- The restart takes time
- How does the LLM know it's done?

After

- `deploy_application(...)`

  → `task_id`

Same failure as the env var



Now 'done' has a definition

Still working

- `queued` · `building` · `deploying`

Done

- `running` · `failed`

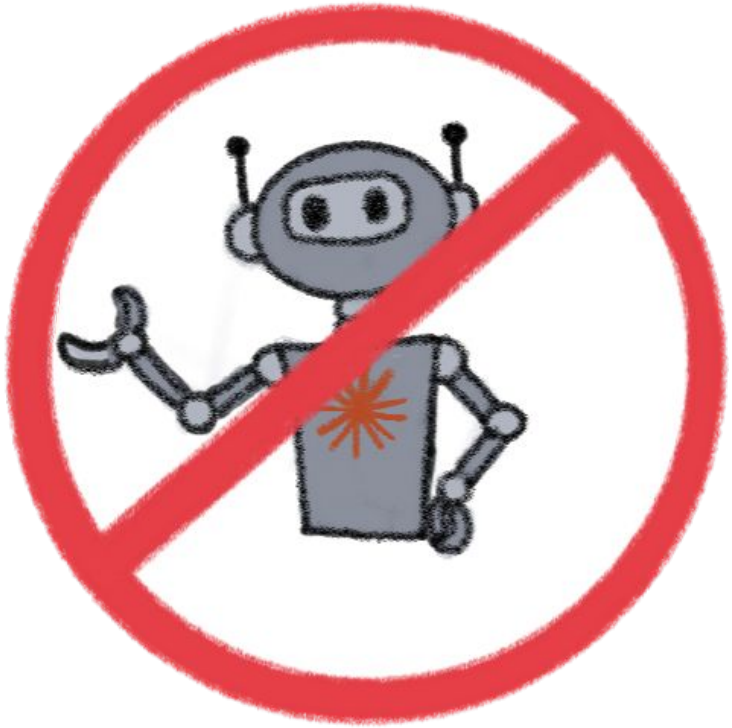
`failed` is done, not "not yet"



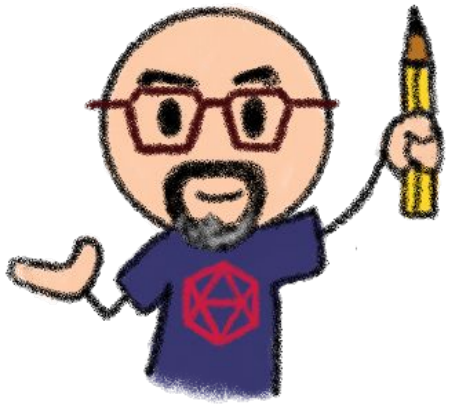
Sometimes there should be no tool

`POST /billing/credits`

- Maybe there is no autonomous tool at all
- Human approval, or nothing



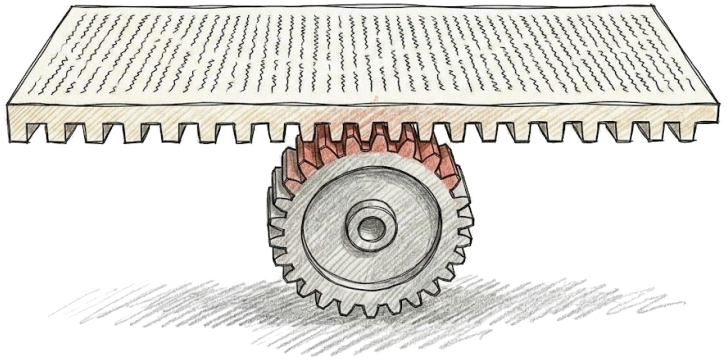
The limit



Agent-friendly does not mean agent-autonomous

The description

When prose becomes behaviour



Documentation, or interface?

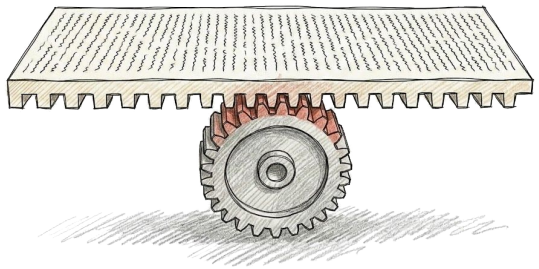
For a human

- **Interface** – enforced
- **Documentation** – inert, read once

For an agent

- The **description is the interface**
- Read at inference time,

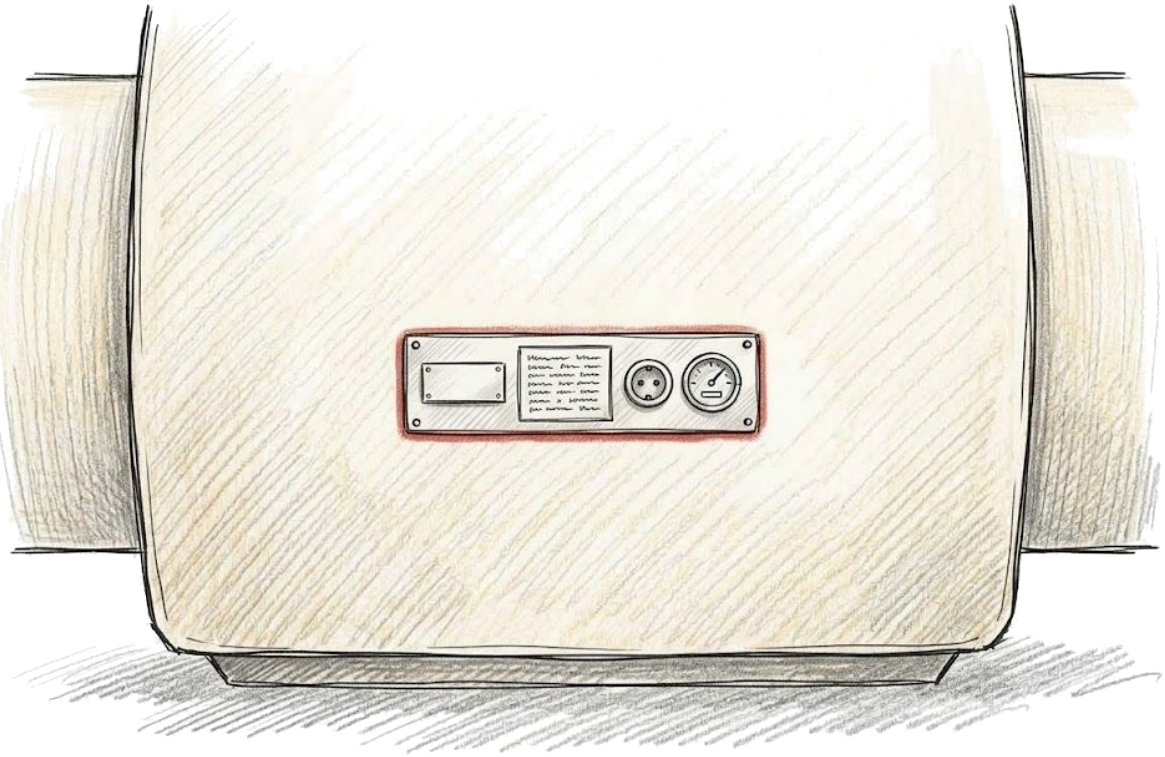
  on every call



The prose is in the execution path

The whole UI

- Tool name
- Description
- Parameters
- Result, and errors

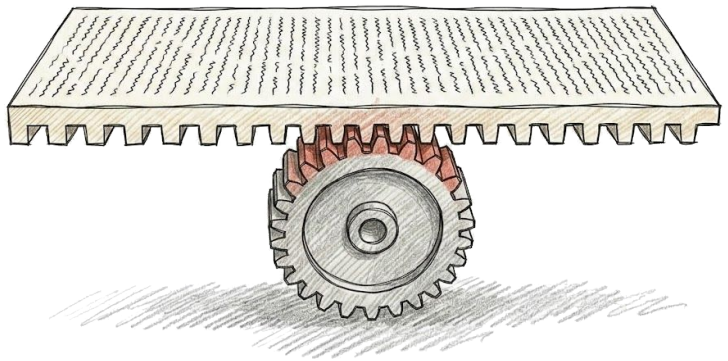


Nothing more

Two descriptions

Before

- `restart(id)`



After

- Purpose
- When to use it
- Side effects
- **When not to use it**

What rewriting the prose buys

- Same models, same tools, same tasks
  - Only the descriptions changed
- Multi-step tasks completed: **33.5% → 44.6%**

StableToolBench – Learning to Rewrite Tool Descriptions

arXiv 2602.20426

The honest ceiling

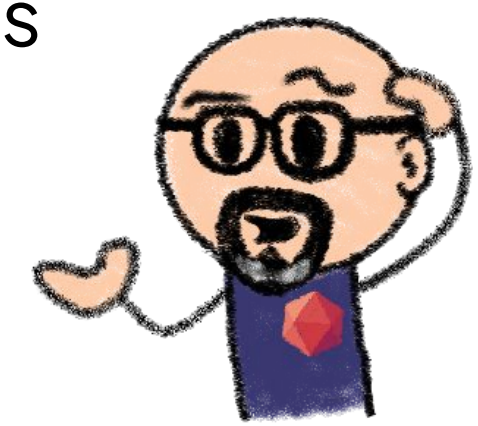
- On an already-good baseline: **+1.4%**
  - The worse your descriptions are today
  - The bigger your win

BFCLv2 Live, from a baseline of 86.4%

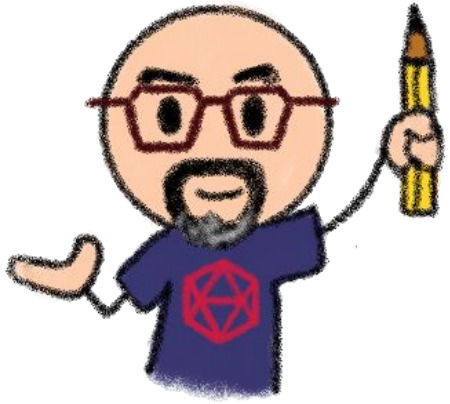
Same study

Remember

310 operations, 0 descriptions



Are my descriptions good?



How do you know you wrote it well?

You stop judging the text and start **measuring the behaviour**

Behavioural tests

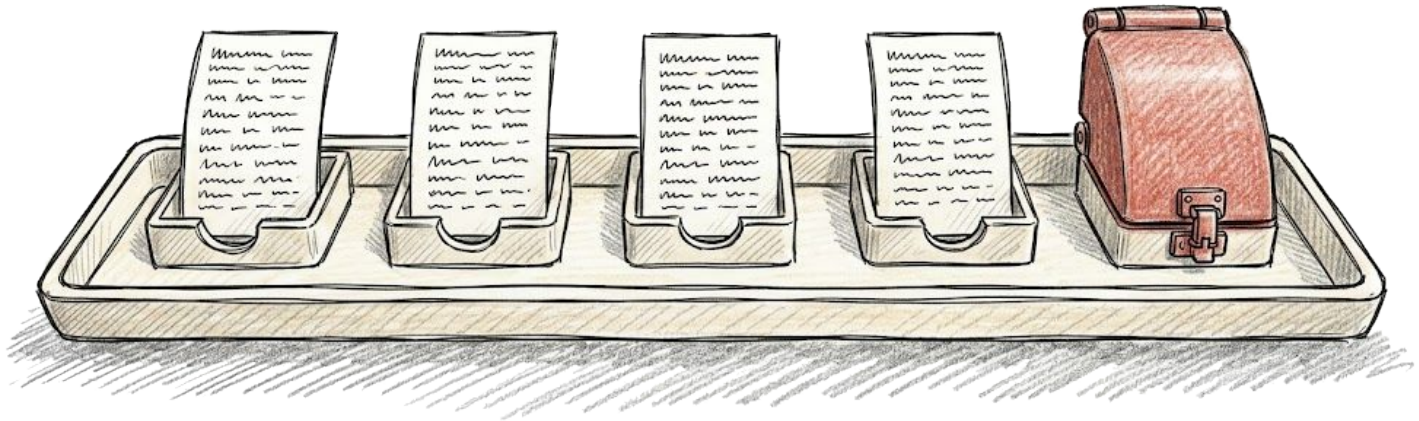
"Is production healthy?"

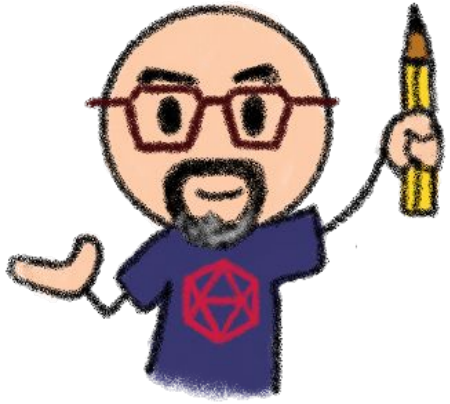
"Why did yesterday's deployment fail?"

"Scale this app for tomorrow's traffic."

"Reduce our bill without causing downtime."

"Delete the test database."



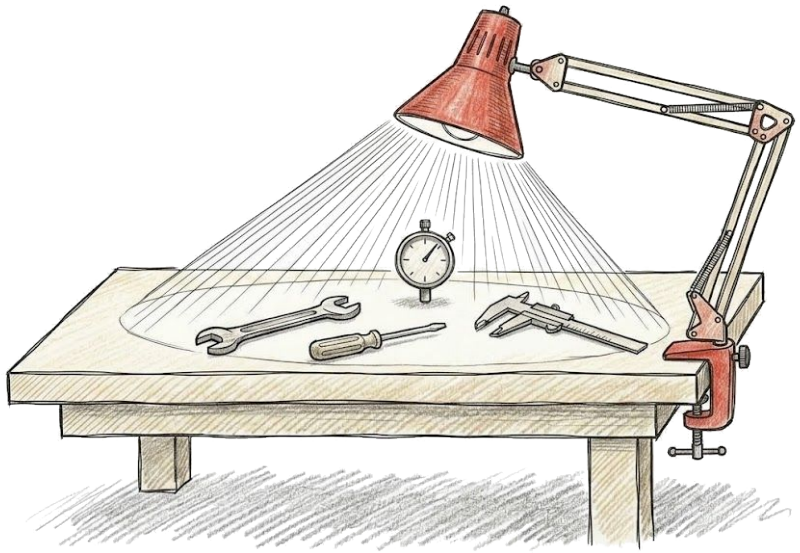


Don't guess

Evaluate

Better for humans

The part I promised in the abstract

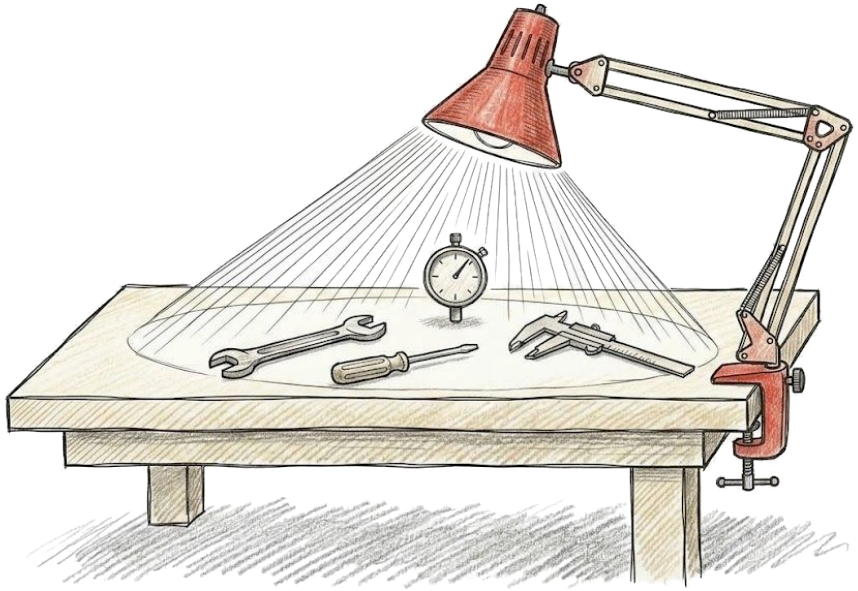


The walk-back

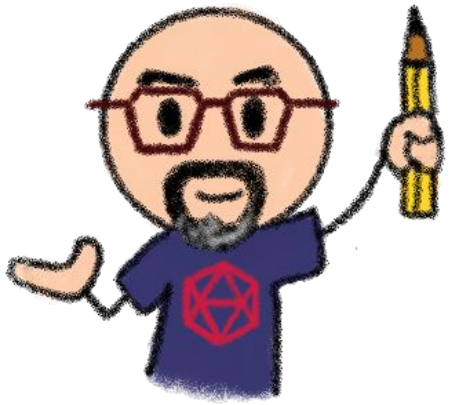
- env var → **Completion**
- diagnosis → **Dependencies**
- delete → **Reversibility · Blast radius**

What they share

- Explicit completion
- Explicit blast radius
- Explicit dependencies
- Scoped authorization
- Predictable, reversible operations



Notice



None of those are AI features

They're the things we always said we'd get round to.

Two real cases

Cloudflare

- Dynamic Workers, built for agents
- Now a primitive for every paid user

Slack

- One blanket `search:read` scope
- Became four granular ones

Agent pressure, shipped for everyone

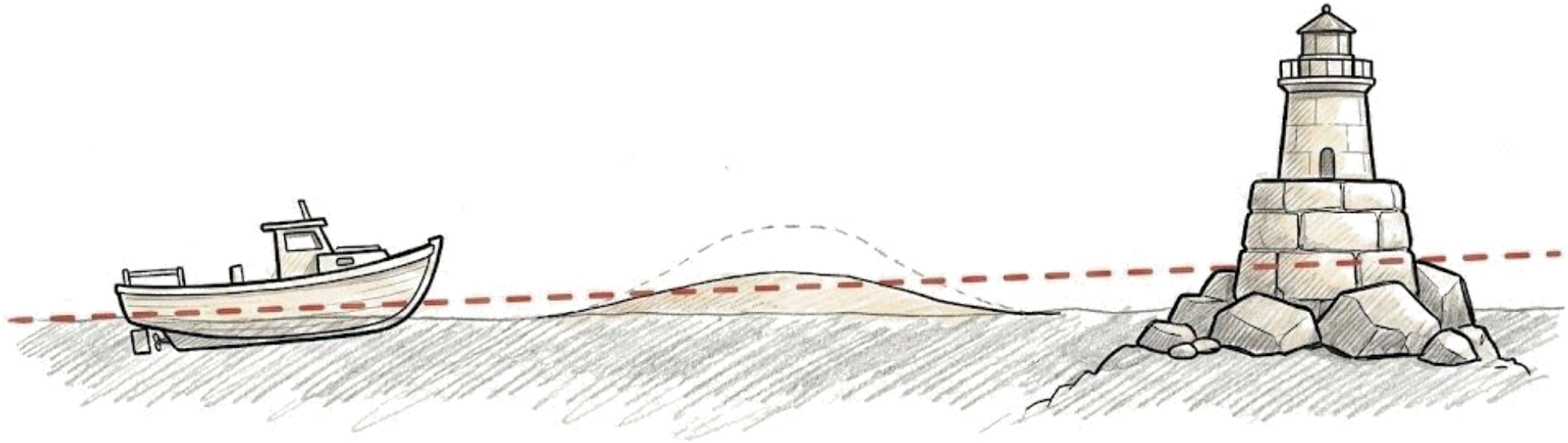
This is early

- Very little of this is published
- What is published stops at the tool layer
- The API underneath usually did not change

So the interesting work is still unclaimed



How big is the gap?



WRAP

Already explicit

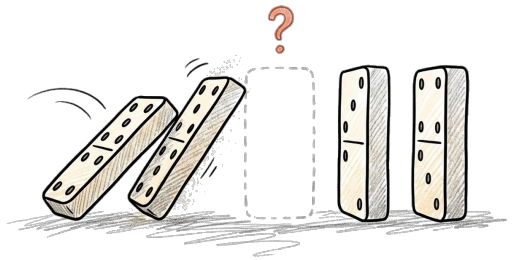
RESHAPE

Scattered knowledge

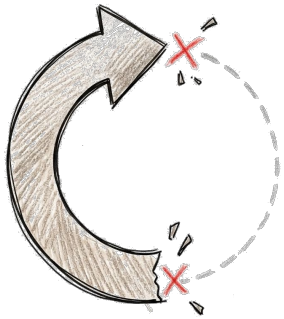
REDESIGN

Unstated invariant

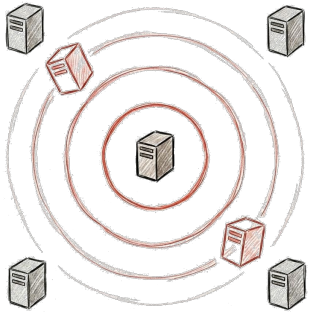
The four questions



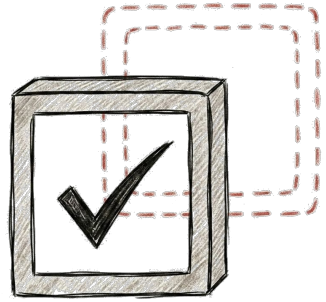
Dependencies



Reversibility



Blast radius



Completion

Agents are the users your API was never designed for

The good news is that designing for them means designing for the developer reading your docs at 2am...

and you already owed them that

Reshapr

A tool sharing the same mindset



reShapr

Turn artifacts into MCP Server



Enterprise-grade MCP servers

🛡️ No code gen!

Reshaping API to MCP



RENAME
INCLUDE
EXCLUDE
ENRICH



SIMPLIFY
CONDENSE
FILTER
FORMAT



SECURE MCP
ENDPOINT &
API BACKEND
AUDIT

That's all, folks!

Thank you all!



Please leave your feedback!