CODE PRESENTS

Closing the Development Loop with Containerized Agents

Philipp Bauer

Sr. Software Architect & Developer

About Philipp Bauer

CODE PRESENTS

Sr. Software Architect & Developer, CODE Consulting

20+ Years of Development Experience

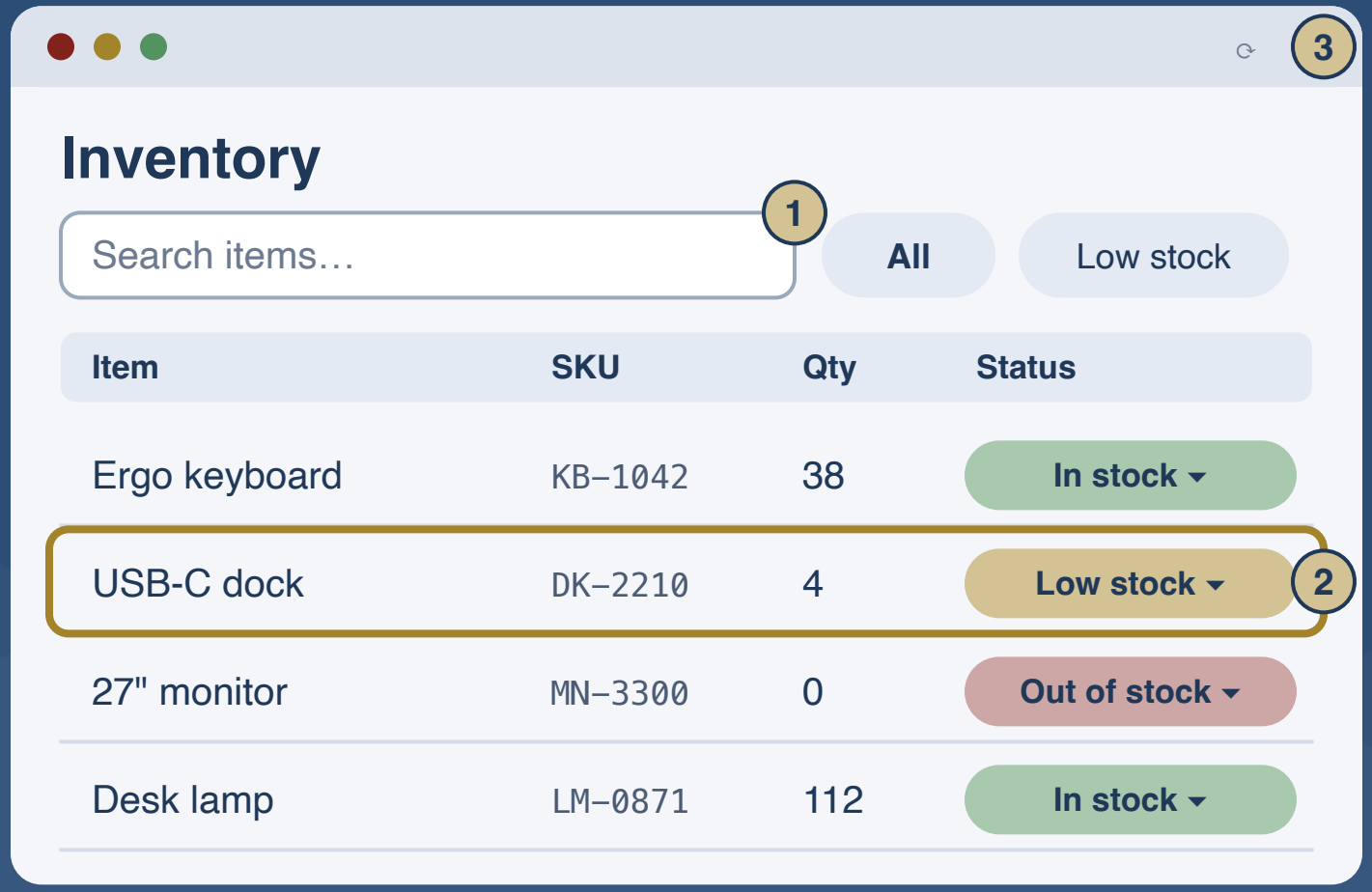
Certified Media Designer



pbauer@codegroup.io

What we're building: an inventory dashboard

CODE PRESENTS

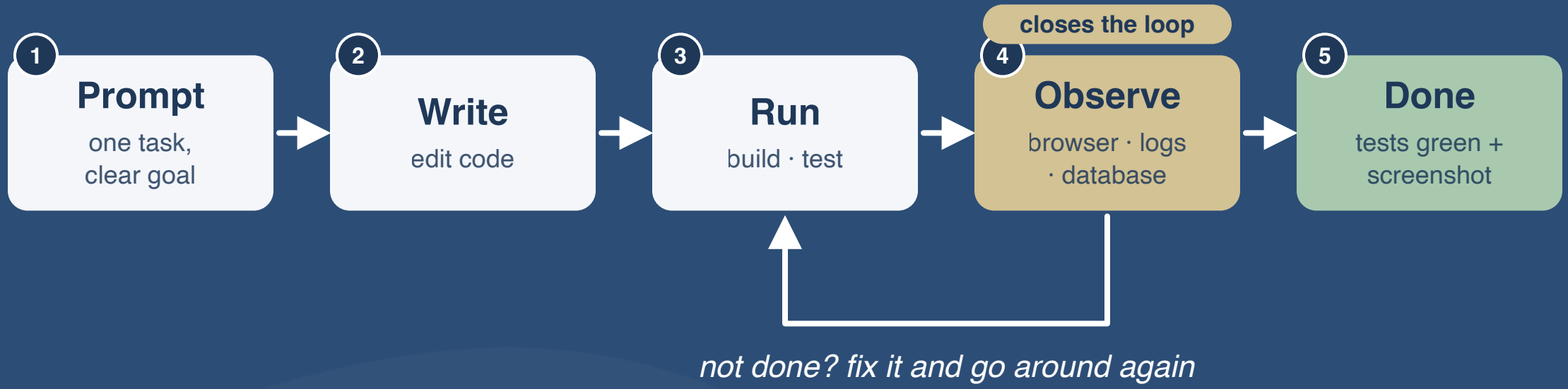


1. **Search filters as you type**
2. **Change a status → saved in SQL Server**
3. **Reload the page → it is still there**

A seeded SQL database, a .NET API and a web UI — written by the agent, checked in a real browser.

The loop that makes an agent useful

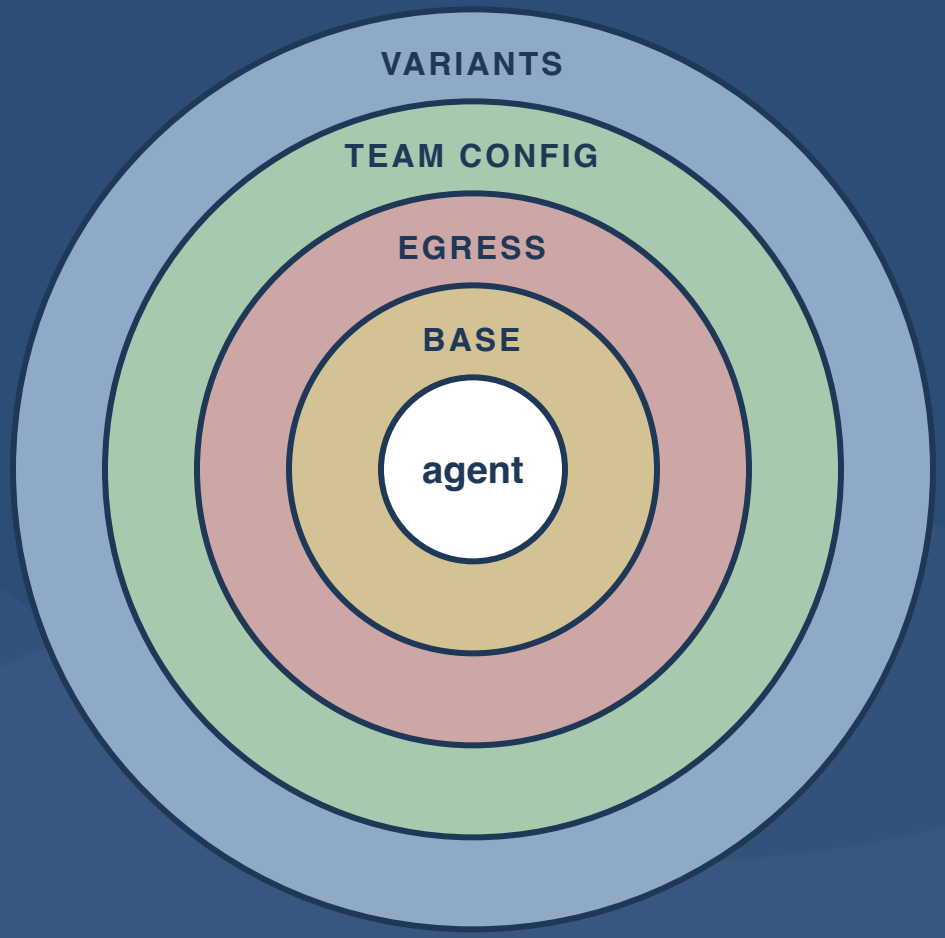
CODE PRESENTS



Everything in this talk exists to make that loop safe and cheap.

Four layers, built from the inside out

CODE PRESENTS



1. **Base image**
   the agent's workshop
2. **Egress wall**
   what may leave the box
3. **Team config**
   who decides what
4. **Variants**
   tailored to the project

CODE PRESENTS



Why a container?

The principles behind every decision

Five principles — we'll meet them in every layer

CODE PRESENTS

1. **Bound the environment, not the human**
   Approval prompts don't scale.

2. **Enforce on the far side of the fence**
   A rule the agent can't reach can't be bypassed.

3. **Test the boundary — don't trust it**
   Including the one an AI helped you write.

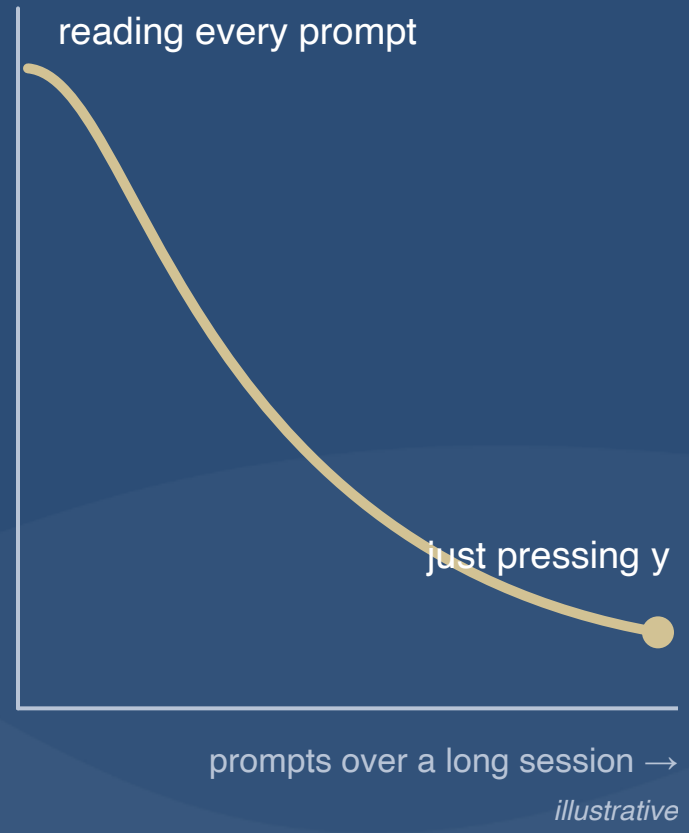
4. **Keep the harness small**
   Tokens spent on setup aren't spent on your task.

5. **Harden the edges — promise no guarantees**
   Detectors help. Enforcers decide.

Approval prompts turn into a reflex

CODE PRESENTS

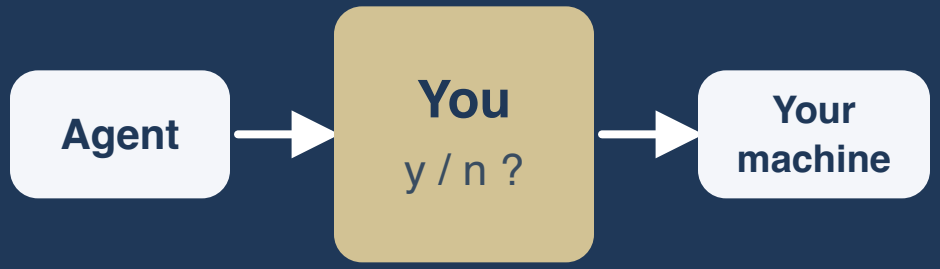
```
#1     allow cat package.json ?                  [y]
#2     allow npm test ?                          [y]
#3     allow git status ?                        [y]
 …
#47    allow rm -rf node_modules ?               [y]
#48    allow npm install ?                       [y]
 …
#212   allow curl https://get.example.sh | sh ?  [y]
```



Move the boundary out of your head

CODE PRESENTS

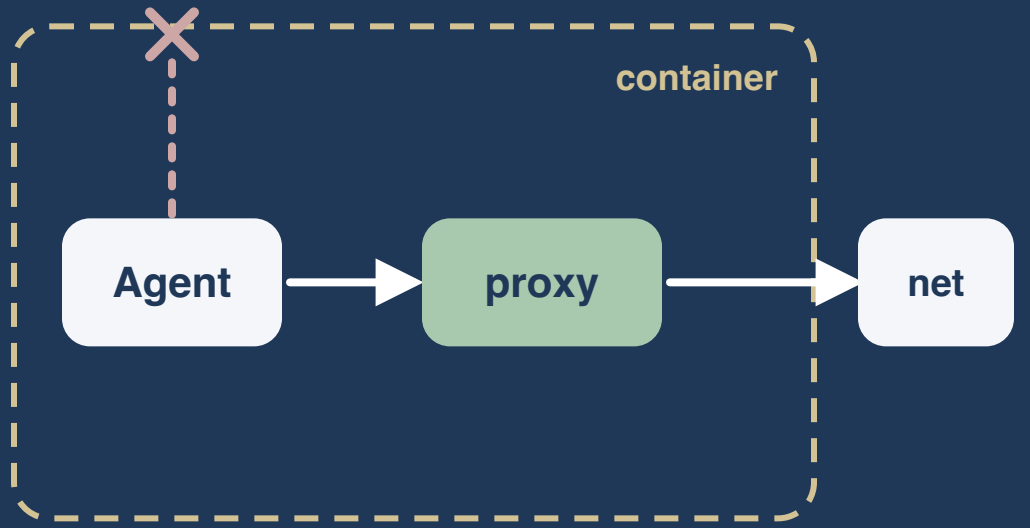
The human as the boundary



every action, all session long

fatigue is the failure mode

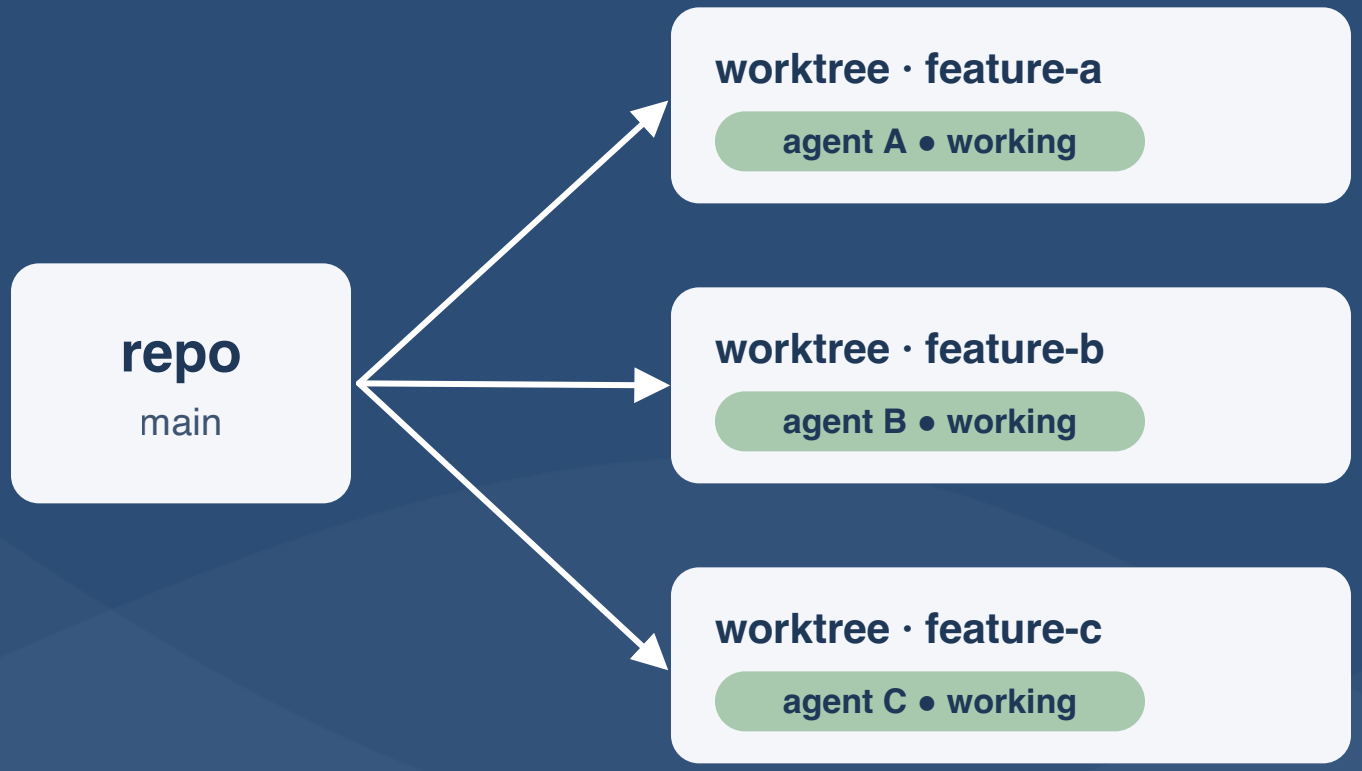
The environment as the boundary



set once · enforced every time

Parallel agents need a boundary, not prompts

CODE PRESENTS



Approve every step?

You can babysit one agent.

Bound the environment

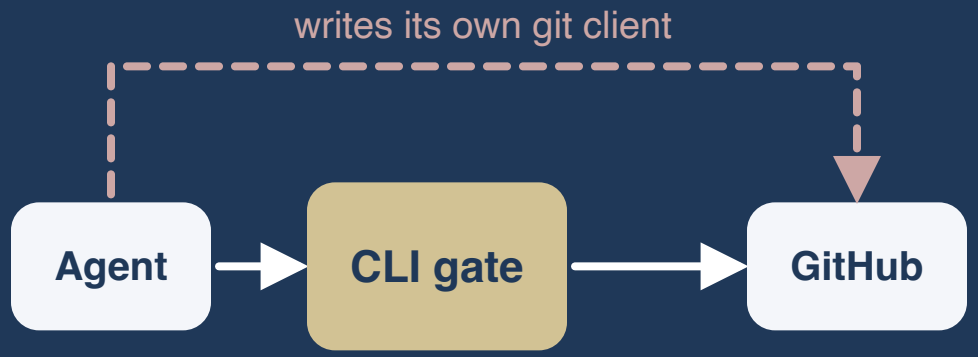
and all three keep running.

Put the rule where the agent can't reach

CODE PRESENTS

A gate on the near side

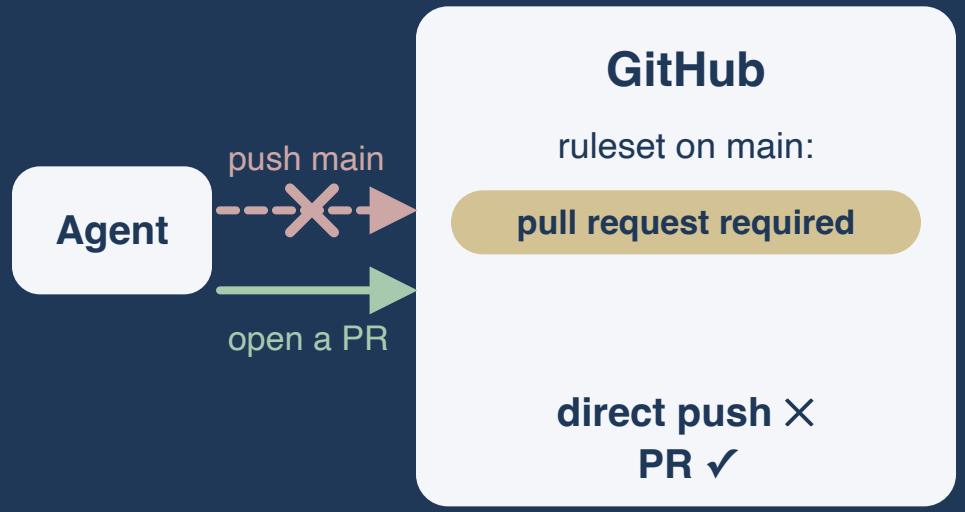
a wrapper around git push



Only works if the agent chooses to use it.

A rule on the far side

enforced by the service itself



Nothing to walk around.

What a container does — and doesn't — do

CODE PRESENTS

DOES

Narrows the blast radius

Runs the agent as a normal user, with no sudo

Keeps your host and home directory out of reach

Lets you control the network

Gives the whole team the same environment

DOESN'T

Make the agent harmless

Protect anything you mount into it

Stop bad instructions from an allowed site

Make a secret safe if the agent can read it

Replace reviewing what it produces

CODE PRESENTS

1

The base image

Everything the agent needs — nothing it shouldn’t have

A workshop for every step of the loop

CODE PRESENTS

WRITE

Pi + Neovim

Language servers and formatters baked in.

RUN

Node · Python

Plus the .NET SDK in the variants.

OBSERVE

Playwright + Chromium

A real browser, driven from the CLI.

REVIEW

hunk · lazygit · glow

Read the agent's diff, not its promises.

RESEARCH

ketch

Search, docs and scraping with a token budget.

ORCHESTRATE

herdr

Many agents in panes, each with its own tools.

Pi: four tools and a tiny system prompt

CODE PRESENTS

`read` `write` `edit` `bash`

4

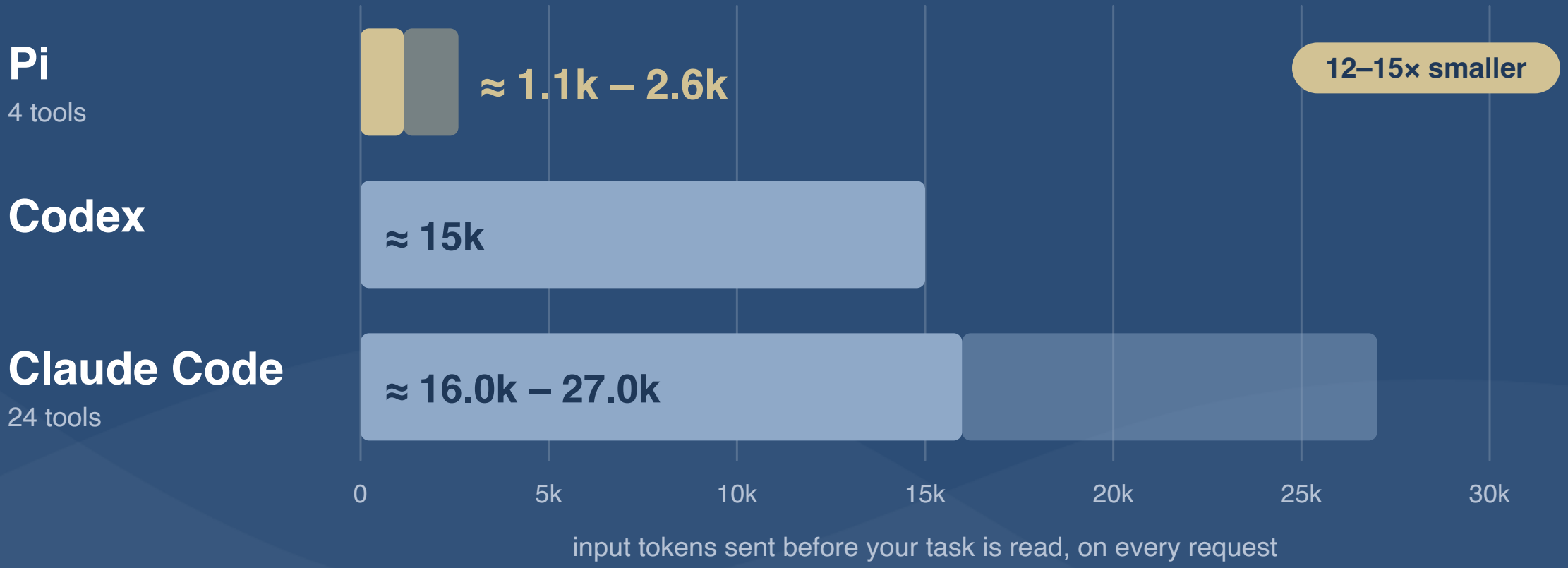
Everything else is a CLI the model already knows — or a skill.

Pi documents itself, so it can extend itself.

The harness can be shaped to what each container variant brings.

The harness tax: tokens before your task

CODE PRESENTS



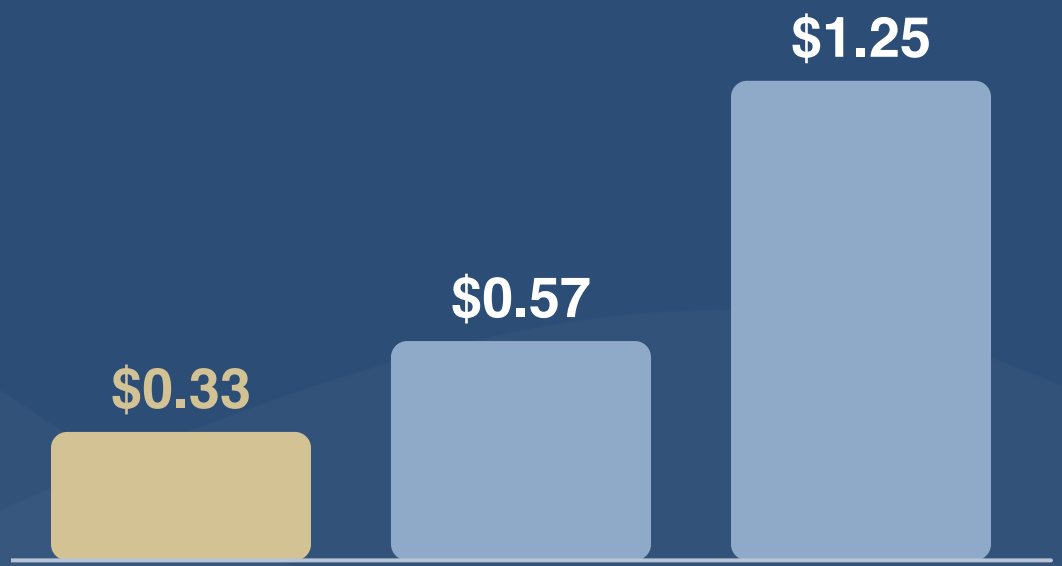
Stock harnesses, short first request. Sources: PointFive, arXiv 2608.01347 (Pi 0.82.1 vs Claude Code 2.1.220) · Portkey "The Harness Tax", Apr 2026 · HarnessTax, UC Berkeley + Arena, Sep 2026. Counts vary with version and tokenizer; bars show the range across sources.

Same model, same tasks — a different bill

CODE PRESENTS

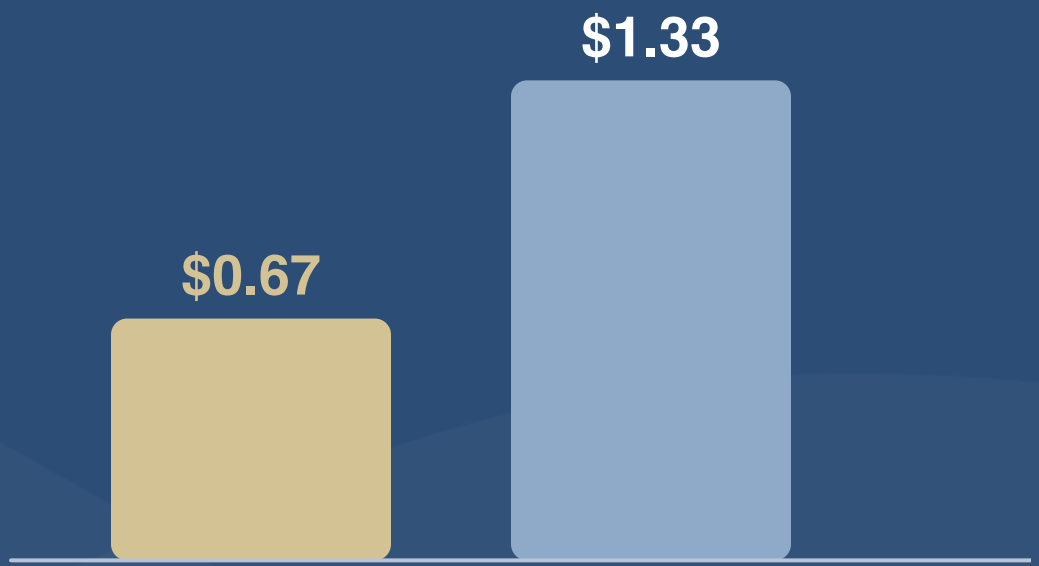
8 coding tasks, same model

all passed 8/8 · one run per task



SWE-bench Lite, one frontier model

Claude Fable 5 · tasks solved per harness



cost per attempt

Sources: rajshah4/harness-benchmark (GLM-5.2, 8 custom tasks, one run each, Pi via ACP) · HarnessTax, UC Berkeley + Arena (SWE-bench Lite, Claude Fable 5). Costs at each study's price list. Stock harnesses — not our container. Caching lowers price, not context size.

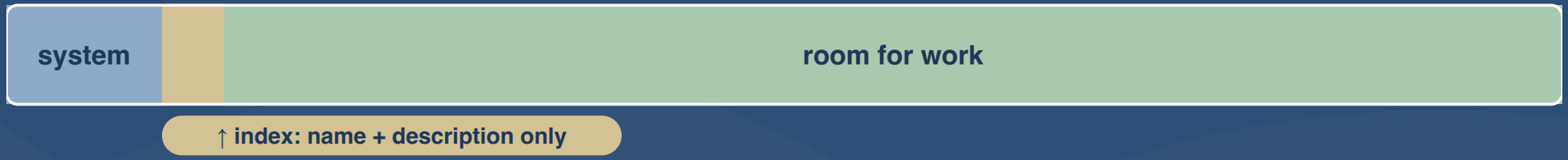
Skills: details load when the task needs them

CODE PRESENTS

Load every instruction up front



Pi: an index first, details on demand

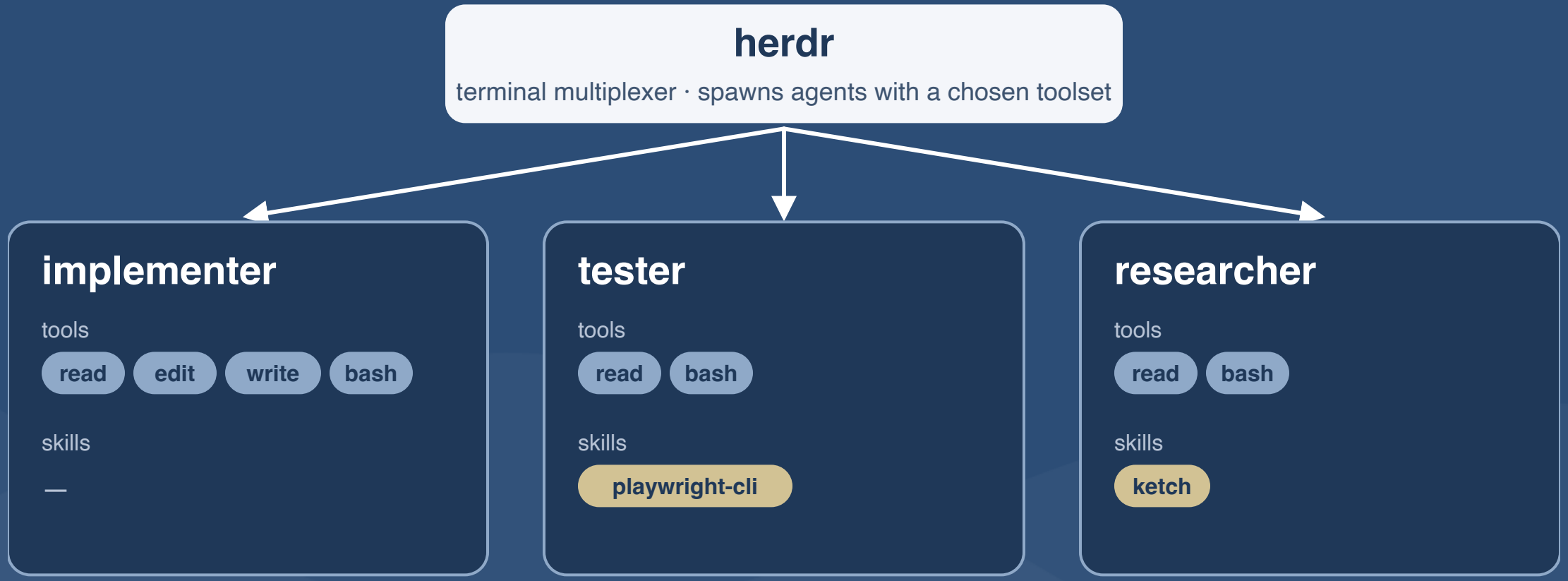


A task matches → just that one SKILL.md is read



illustrative — proportions not measured

Scoped agents: only the tools they need



The toolbox is root-owned. The work isn't.

CODE PRESENTS

container

Root-owned
the toolchain

Pi · herdr · hunk · Neovim

apt packages

sshd configuration

agent: read and run only

pi-owned
disposable home

$HOME · shell config

nvm / node · conda

project dependencies*

agent: read and write

Volumes
persistent

/workspace — your code

~/.pi/agent — provider config

/etc/ssh — host keys

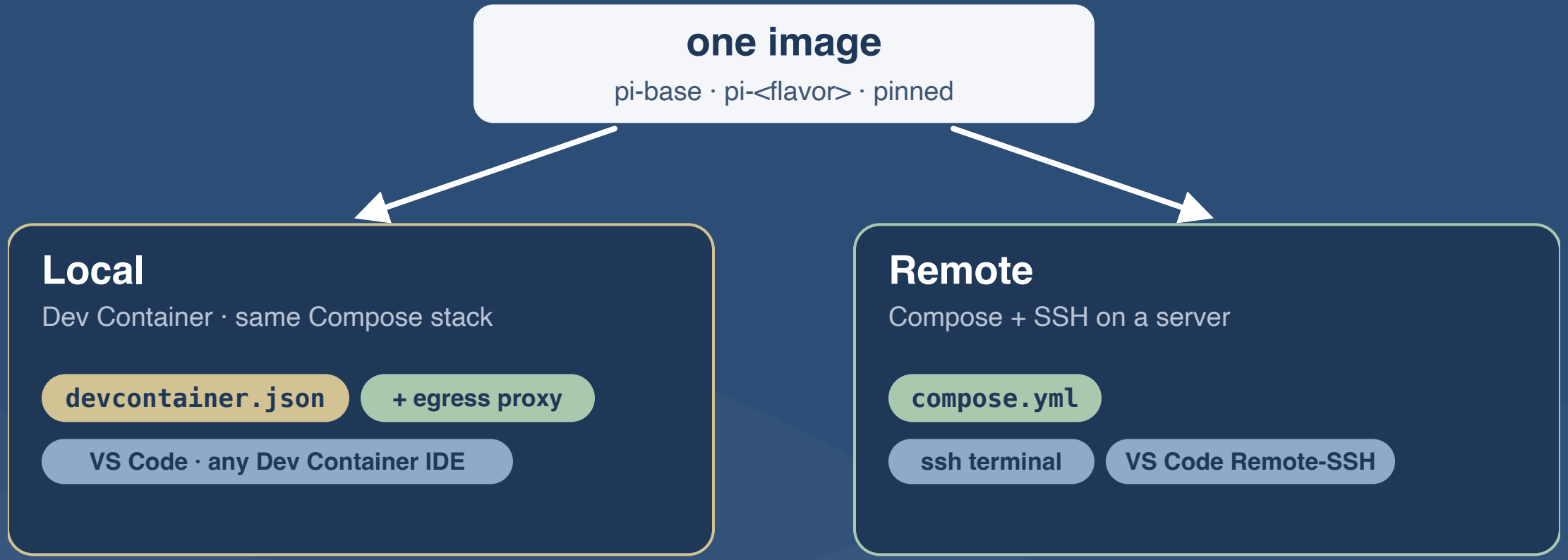
survives a recreate

user pi · uid 1000 · no sudo · key-only SSH

* npm / pip / NuGet restores go through the egress proxy

One image, two ways to use it

CODE PRESENTS



same tools · same pins · same wall

The home directory is disposable — on purpose

CODE PRESENTS

Container

rebuilt from the image on every recreate

$HOME resets

- shell config
- editor cache
- anything the agent “installed”

recreate

/workspace

your code

~/.pi/agent

provider config · + sessions

~/.vscode-server

IDE server

/etc/ssh

host keys

named volumes survive

Every build step proves itself

CODE PRESENTS

guard 1 · the browser must really launch

```
RUN pip install -q playwright && playwright install chromium \
&& python -c "… pg.goto('data:text/html,<title>ok</title>');
assert pg.title()=='ok'"
```

guard 2 · nothing root-owned under $HOME

```
RUN test -z "$(find /home/pi -user root -print -quit)"
```

guard 3 · lockfiles are the review

```
nvim/lazy-lock.json mason-lock.txt
# committed · the build asserts they match · a diff is the change review
```

RECORDING 1

CODE PRESENTS

Tour the base image

1. ssh in — tools on the path, no sudo
2. spawn a scoped agent in herdr
3. Playwright: open a page, take a screenshot

CODE PRESENTS

2

The egress wall

What may leave the box — and how we know

Two problems — the wall solves one

CODE PRESENTS

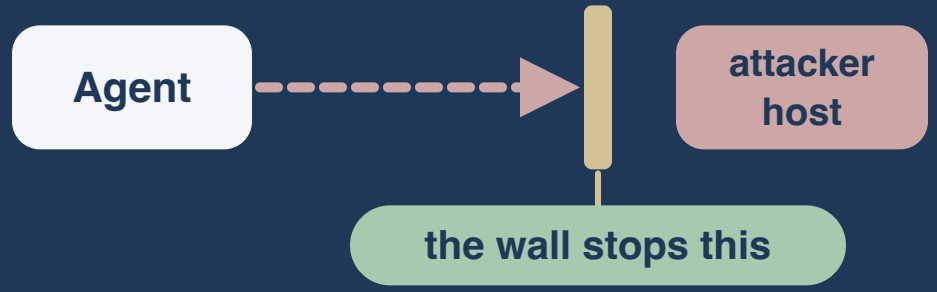
A Exfiltration

the agent sends data out,
or fetches a payload

without the wall

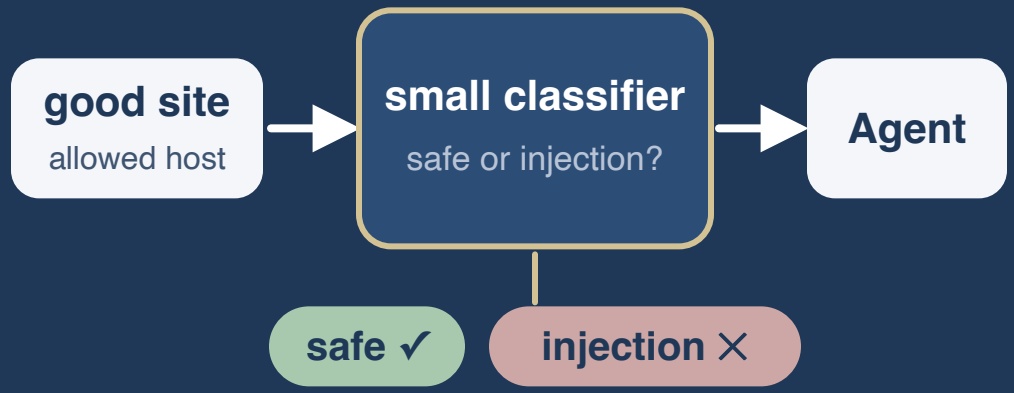


with the wall



B Injected instructions

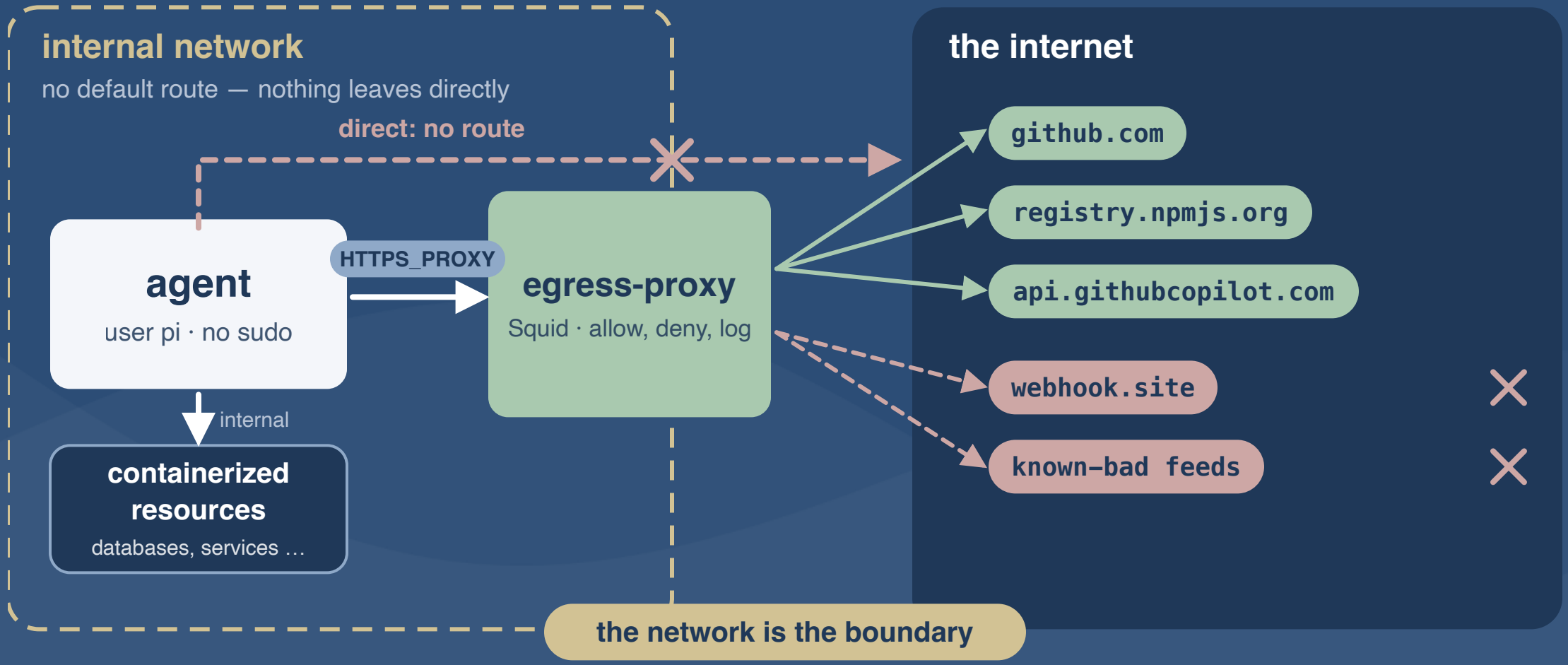
malicious text arrives
inside normal content



checked before it reaches the agent

Instructions suggest. A network boundary enforces.

CODE PRESENTS



Don’t trust it. Test it.

CODE PRESENTS

inside the agent container

```
$ curl --noproxy '*' -w '%{http_connect}' https://example.com
000                                          ignoring the proxy: no route
$ curl -w '%{http_connect}' https://github.com
200                                          through the proxy
$ curl -w '%{http_connect}' https://webhook.site
403                                          blocked, and logged

proxy log: CONNECT webhook.site:443 TCP_DENIED/403
```

Output from the live wall: `%{http_connect}` prints the proxy’s answer to the tunnel request; 000 means no connection was made.

Blocklist by default — your team's dial

CODE PRESENTS

your team picks the point



Blocklist

allow everything except
known-bad destinations

low friction

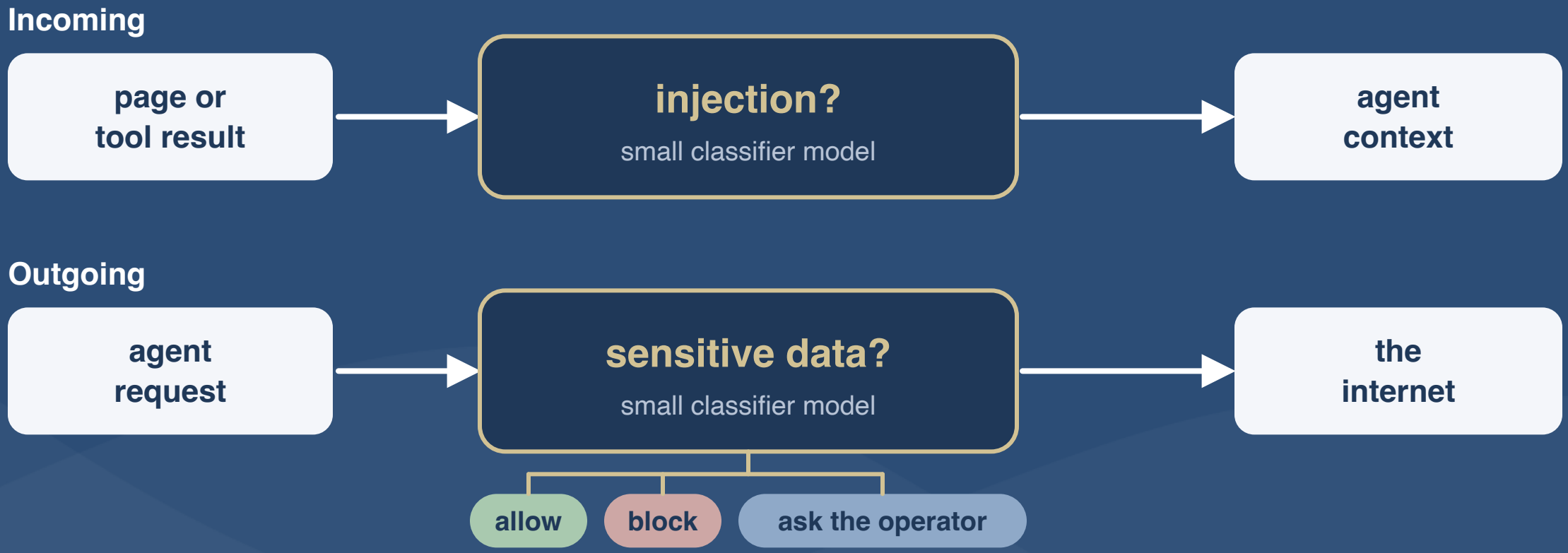
Allowlist

allow only approved hosts —
someone has to maintain the list

high assurance

Reachable is not the same as trusted

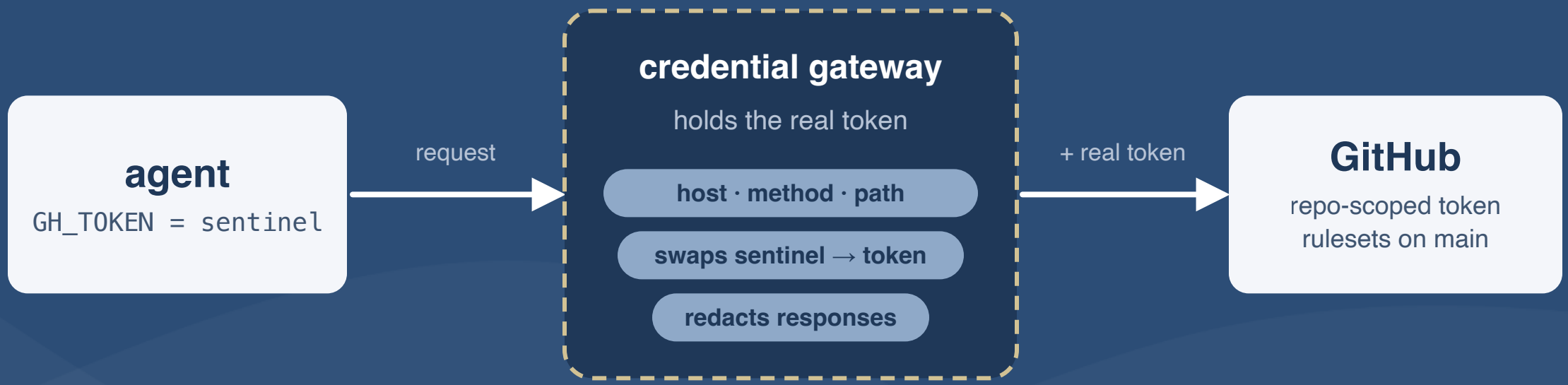
CODE PRESENTS



unsure or unavailable → fail closed

a detector, not the boundary — the harness has to enforce the verdict

Secrets: the agent holds a placeholder



The gateway protects custody.

Token scope and rulesets bound what the credential can do.

RECORDING 2

CODE PRESENTS

Watch the wall

1. tail the proxy log while the agent works
2. try a blocked host — see the 403 and the log line
3. the model call itself travels through the proxy

CODE PRESENTS

3

Your team's configuration

Same image. Your rules.

Who decides what

CODE PRESENTS

THE AGENT CAN

Do the work

Edit code, run tests

Build and run the app

Drive a browser

Restore project dependencies — through the proxy

THE TEAM DECIDES

Set the rules

Which tools ship in the image

What the wall allows

Which feeds and repositories are reachable

Which model to use

OUT OF REACH

Not the agent's call

System packages — no sudo

The egress policy

Credentials and their scopes

The image itself

Six files, six decisions

CODE PRESENTS

`compose.yml`

Networks, ports, volumes

What the container can reach and keep.

`egress/squid.conf`

Blocklist or allowlist

Plus the feed and local lists.

`authorized_keys`

Who may connect

Public keys only — no passwords.

`git-identity.env`

Who commits as

Agent commits are labelled as the agent.

`models.json`

Which model

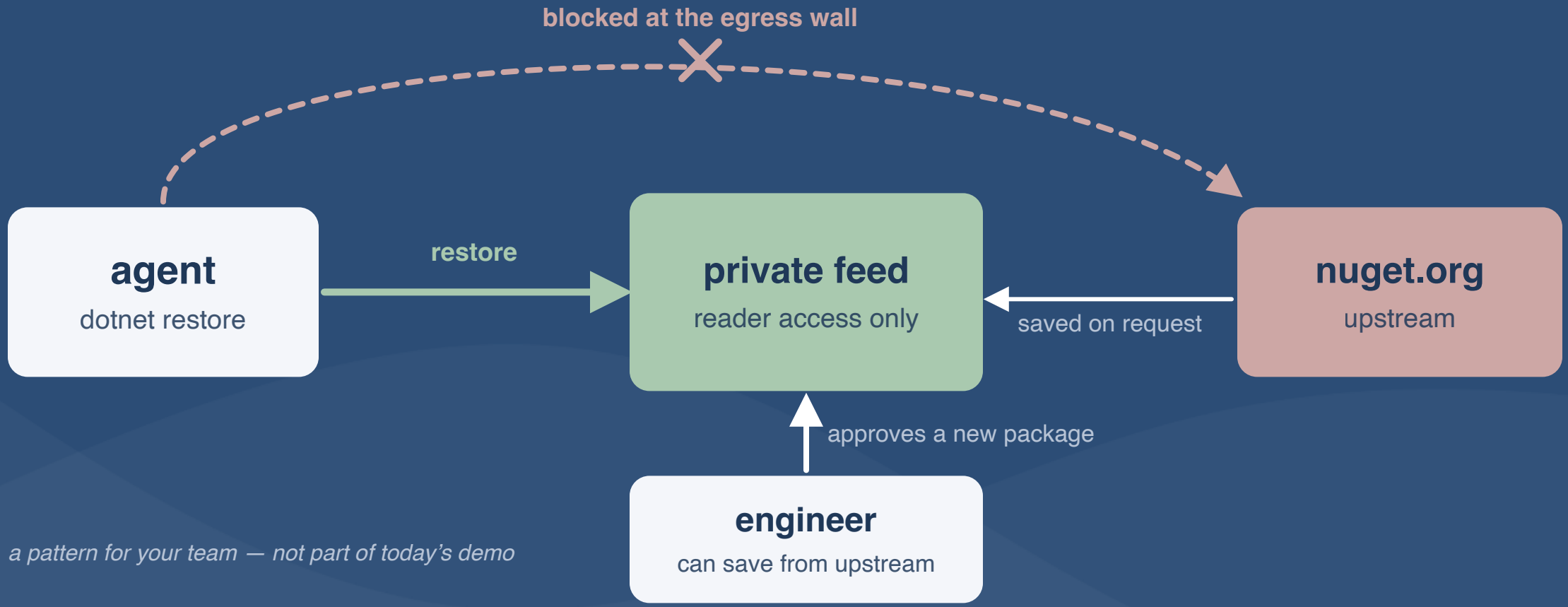
Local, Copilot, or any provider.

`devcontainer.json`

Local use

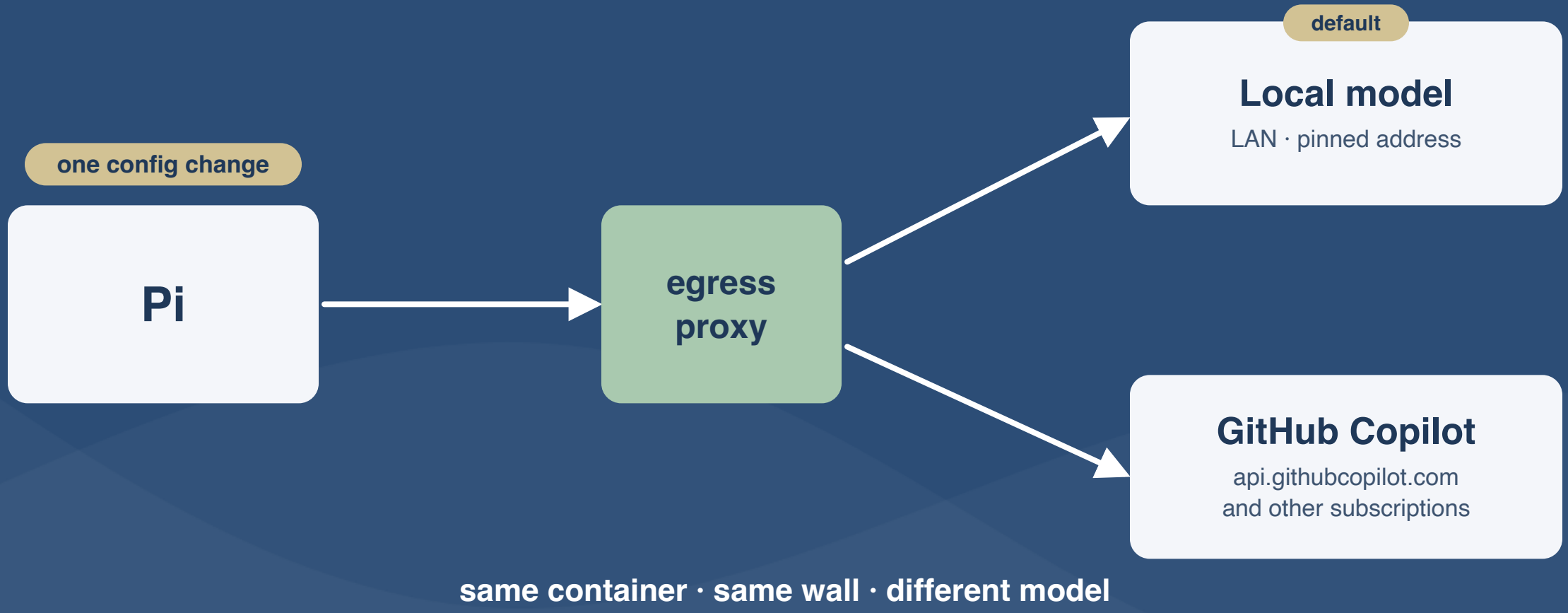
The same image on a laptop.

Dependencies: a feed, not the internet



a pattern for your team — not part of today's demo

Local first — any provider on demand



same container · same wall · different model

Ship environment changes like code

CODE PRESENTS

1 Edit
Containerfile or lockfile

2 Build
guards fail loudly

3 Tag
keep the previous image

4 Recreate
verify — or roll back

--force-recreate on one service also recreates its dependents.

$HOME resets — volumes survive.

RECORDING 3

CODE PRESENTS

Switch the model

1. local model: one prompt, answer streams back
2. change the provider setting
3. same prompt via GitHub Copilot — same container, same wall

CODE PRESENTS

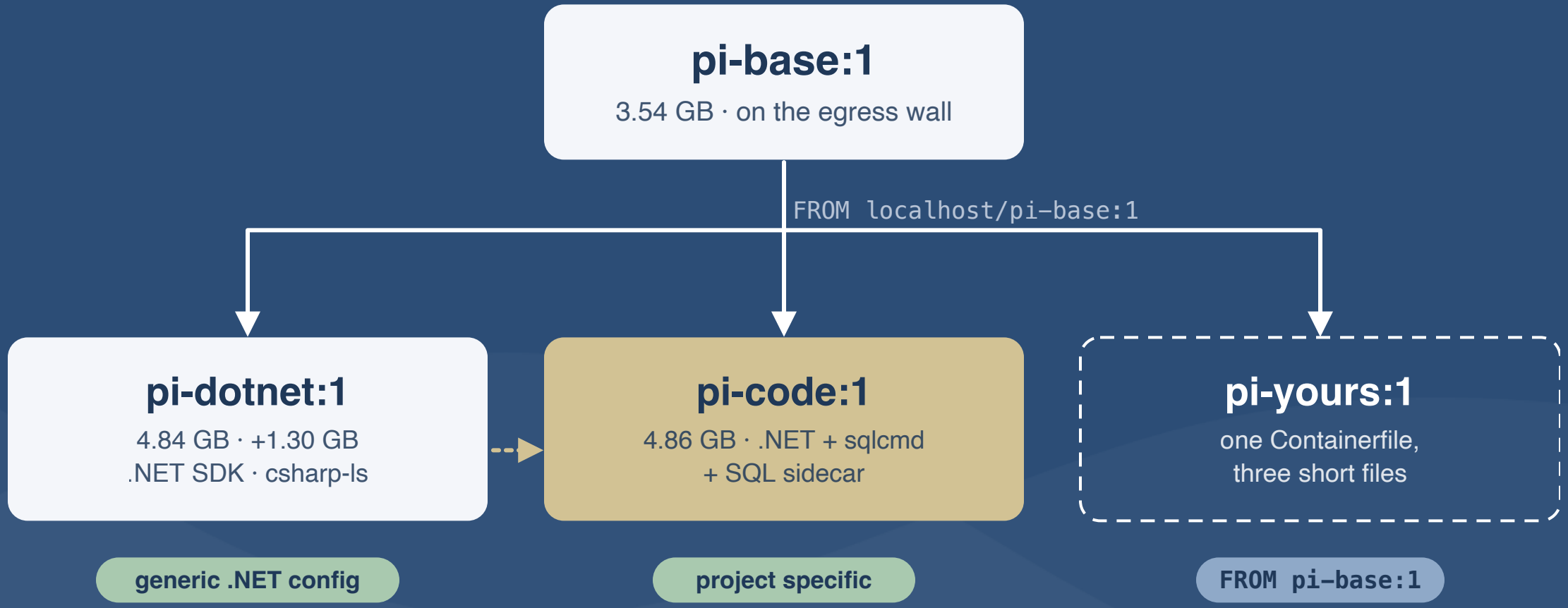
4

Variants

Tailor the base to one project

A family of images, one base

CODE PRESENTS



pi-code's recipe was copied from pi-dotnet

pi-code: add what the project needs

CODE PRESENTS

pi-code:1 · 4.86 GB

data layer
sqlcmd 1.10.0 · seed utility (Bogus)

.NET layer +1.3 GB
SDK 10.0.403 · csharp-ls · csharpier

pi-base:1 3.54 GB
Pi · herdr · ketch · hunk · glow
Playwright + Chromium · Neovim IDE layer
nvm / Node · conda / Python · git

`sql:1433` →

SQL Server 2022
Developer · separate container
pinned by image digest

100 seeded rows

the image gets tools
the sidecar gets state

A scoped login, enforced by the database

CODE PRESENTS



SA password:

never in the agent container

The agent logs in as pi_demo and may:

✓ SELECT ✓ INSERT ✓ UPDATE ✕ DELETE ✕ CREATE TABLE

scoped to one table — enforced by the database, not by the agent's good behaviour

Three traps we already fell into

CODE PRESENTS

TRAP 1

Rebuild the children

After any base rebuild, rebuild every variant — they bake in the old one.

TRAP 2

SSH has its own environment

sshd builds a clean one, so proxy settings must be set server-side.

TRAP 3

Test the wiring, not the install

A tool can be present and still unusable — assert that it works.

The task: it's done when all five are true

CODE PRESENTS

- [ ] The API lists, searches and updates items
- [ ] The UI filters as you type
- [ ] A status change survives a reload
- [ ] The agent's own tests pass
- [ ] A Playwright run proves it — with a screenshot

RECORDING 4

CODE PRESENTS

pi-code builds the dashboard

1. one prompt, local model
2. the agent writes the API and the UI
3. it drives the browser, finds a problem, fixes it

RECORDING 5

CODE PRESENTS

Review the result

1. tests green, Playwright screenshot
2. read the diff in hunk
3. what the proxy log saw during the run

Hardening, not guarantees

CODE PRESENTS

LAYER	STOPS	DOESN'T STOP
Base image	The agent changing its own system toolchain	Damage inside what you mount
Egress wall	Unmediated traffic and known-bad hosts	Bad content on an allowed host
Team config	The agent widening its own access	A team that sets the rules too loosely
Variants	Every project carrying every tool	The upkeep of one more image

Take-aways

CODE PRESENTS

1. Bound the environment, not the human.
2. Enforce where the agent can’t reach — then test it.
3. Keep the harness small; add what the project needs.

AI Consulting Services

CODE PRESENTS

Check out our Executive Briefing offer!

We can help with your AI needs

What does AI mean for you?

“Skunk Works” Projects

codemag.com/AI
codemag.com/ExecutiveBriefing



Free Subscription

CODE PRESENTS

The leading software development magazine, written by expert developers for developers.

Free subscription for all of you (and your friends)

Subscribers get CODE Focus issues free of charge!

Share this link to our free subscription:

bit.ly/cp-2026-11-subscribe



CODE Staffing

CODE PRESENTS

Disrupting the world of staffing!

Giving our customers the ability to have staff on par with Silicon Valley companies ...

... and our employees a work environment in a bleeding-edge tech company with the industry leading benefits!

codestaffing.com



CODE PRESENTS

Q&A

Contact us with your questions!

CODE Contact

codemag.com
info@codemag.com
facebook.com/codemag
x.com/codemagazine

Presenter Contact

pbauer@codegroup.io
linkedin.com/in/philippjbauer
blog.philippbauer.org