

Platform
Engineering Day
NORTH AMERICA

Don’t Lose Your Agent’s Mind: Platform Patterns for Stateful Agents

Maia Iyer and Evaline Ju
IBM Research

What we learned scaling agents on an internal platform

Simplified architecture

Isolated sandbox pod

Agents run with no credentials or cluster access; execution state has an owner.

Paired egress gateway

Injects credentials and is the only network exit — enforced by NetworkPolicy.

Persistent workspace

Preserves files, memory, and learned context across restarts and long-running sessions.

What that architecture lets you do

- Agents can idle for days without losing work
- No credential lives in agent code
- A session can span many harnesses
- Support for new ACP-compatible harnesses

What we didn't solve

- **Resource sprawl**
- **Heterogeneous demand** e.g. hardware tiers, extra credentials

Repo: https://github.com/dam-agents

Beyond crash recovery

Crash recovery

- Client disconnects mid-run
- Server still has the result
- Reconnect, reconcile, no loss

State orchestration

- **Resumption**: the laptop quits, the operator picks up any pending run.
- **Isolation**: one user's workload can't see another's.
- **Provenance**: the run that saw file X and skill Y is a lookup, not a reconstruction.
- **Scale**: many sessions, many users, same ownership rules.

Beyond crash recovery - use cases

Platform Engineering Day NORTH AMERICA

Share a workspace

STATE SHARING · FILES, KNOWLEDGE, BENCHMARKS

A benchmarking workload, a shared knowledge base, a dataset too big to pack per-run can be consumed by many agents.

Pick up someone else's work

AGENT → HUMAN → AGENT HANDOFF

An agent produced a half-done result. A second agent or a human needs to detect "done," read what the first saw, and continue.

Scheduled, automated agent runs

CRON, EVENT-DRIVEN

Nightly maintenance workflows, periodic audits, event-driven dispatch

Repetitive, open-ended work

CLAWGENTI · BEYOND DEPENDABOT

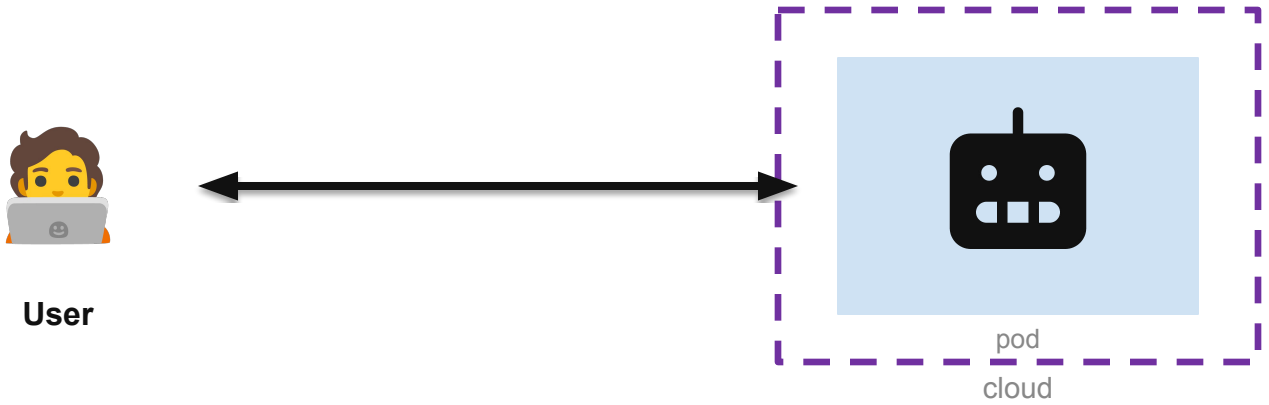
Dependency resolution, lint-fix sweeps, and security triage need judgment and running against a durable workspace.



Platform
Engineering Day
NORTH AMERICA

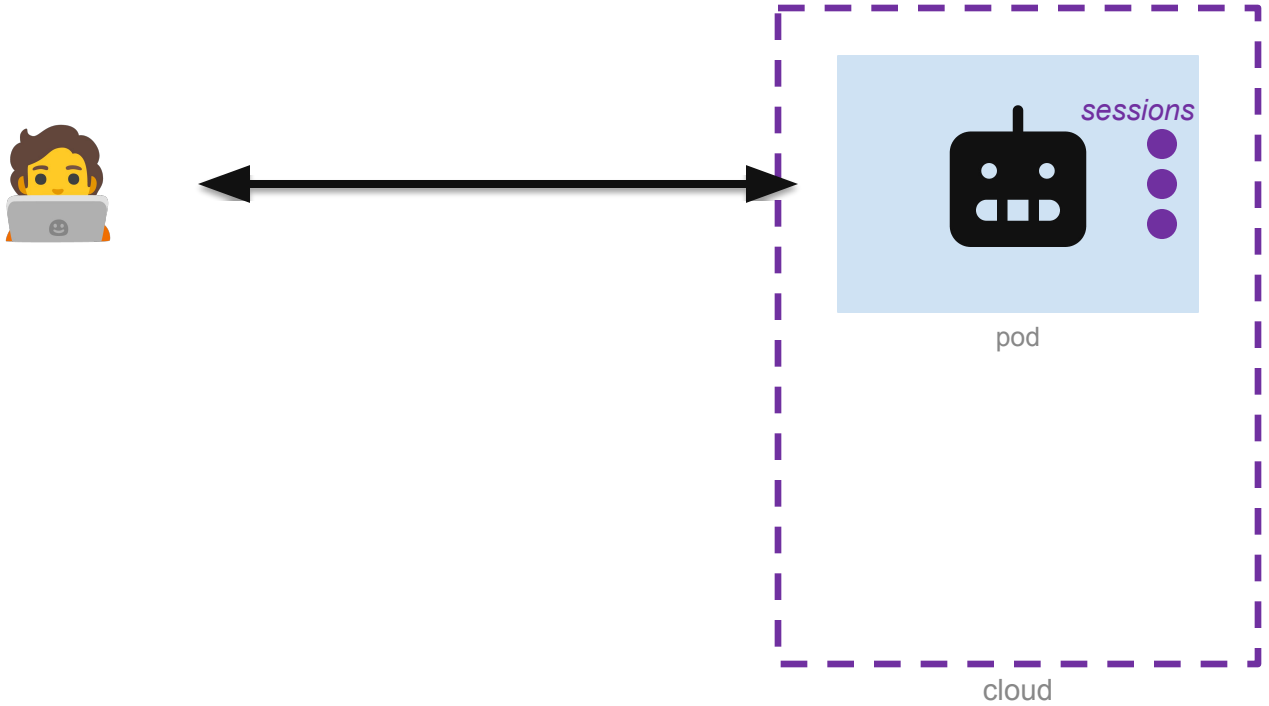
How can we enable the use of cloud resources in these agentic flows?

Where agents live



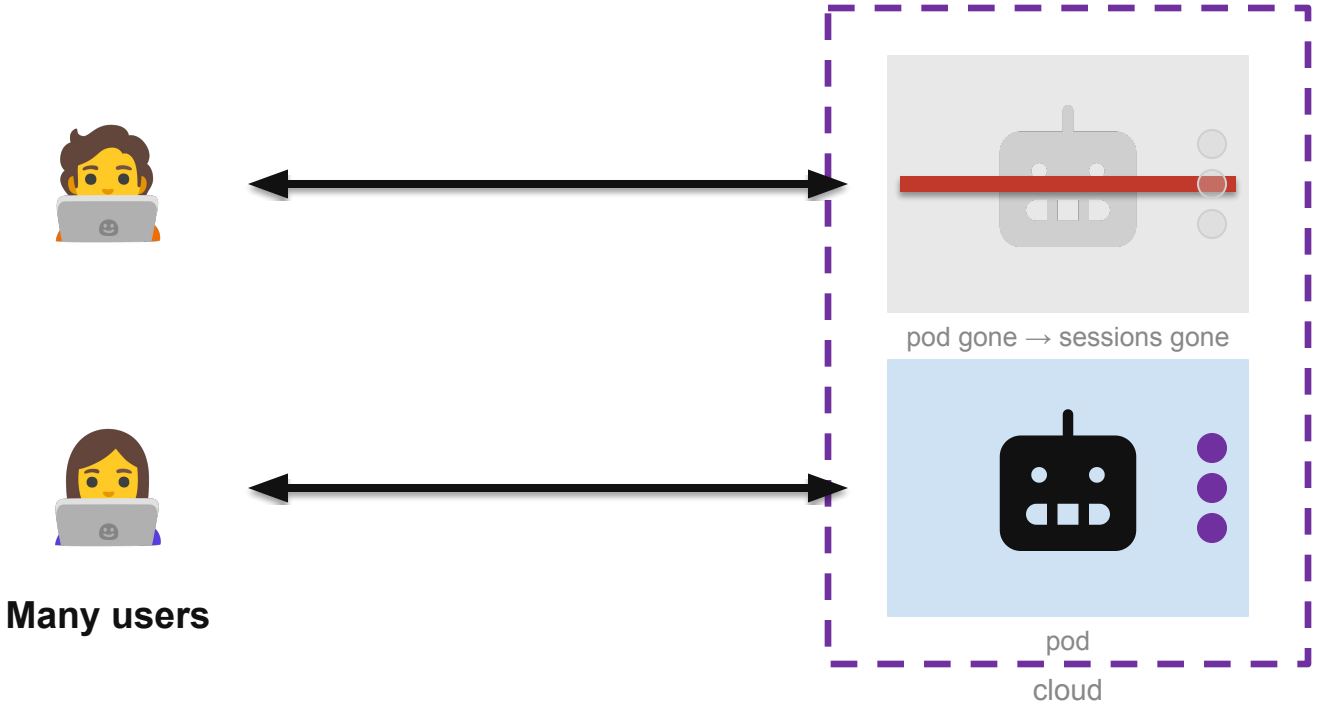
So… just put the agent application in a pod?

Current problems



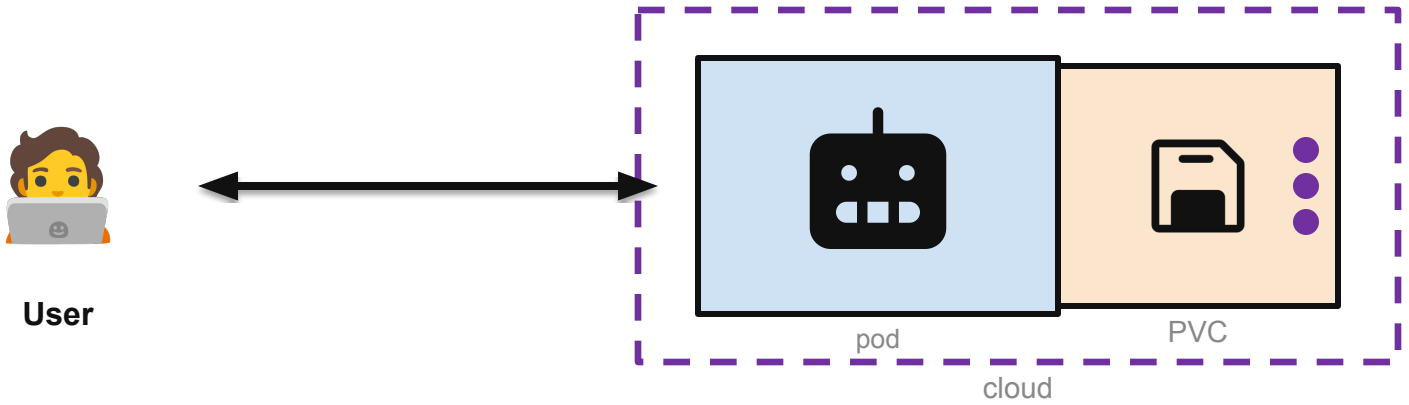
Most popular agents are stateful.

Current problems



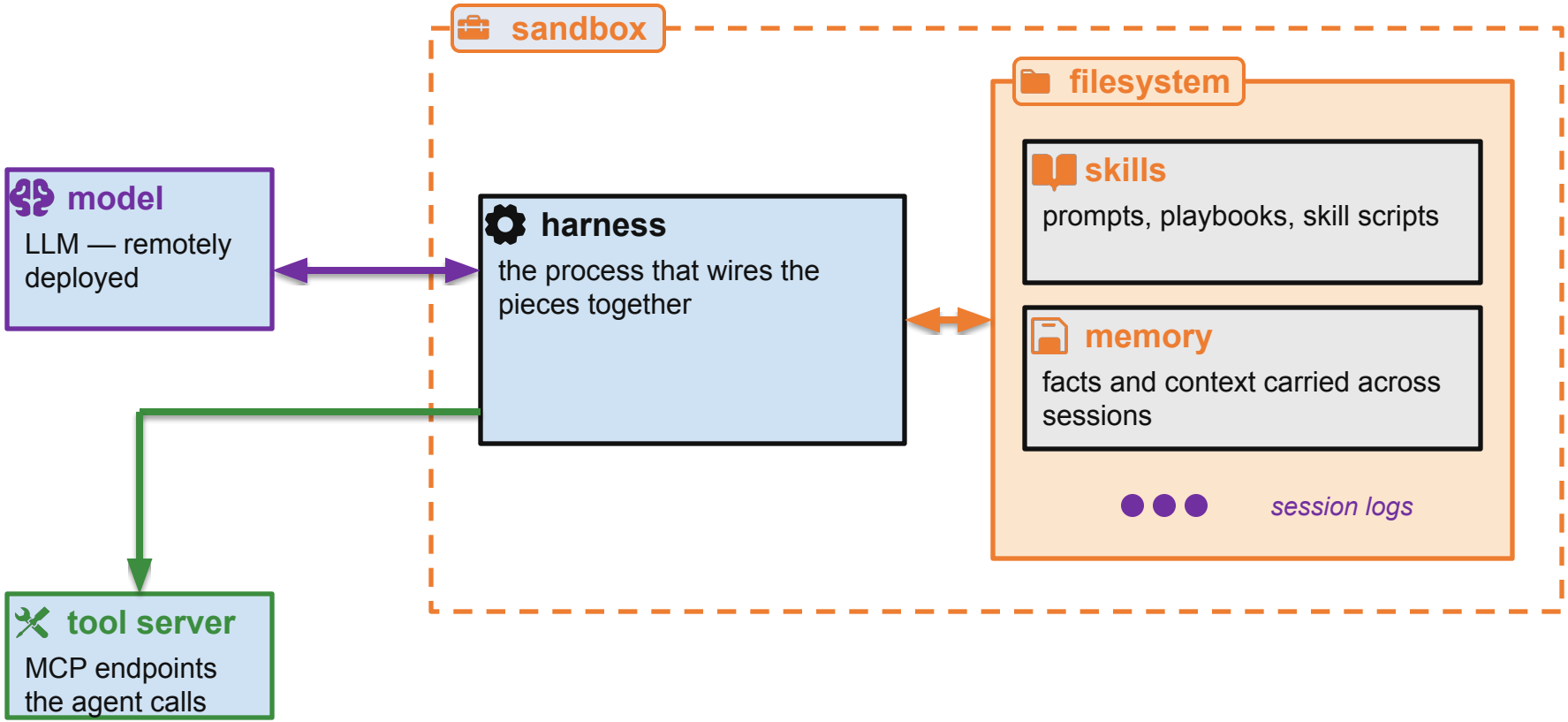
A plain pod gets deleted — and the results vanish.

Current problems



Mount a ***PVC*** — now the state survives.

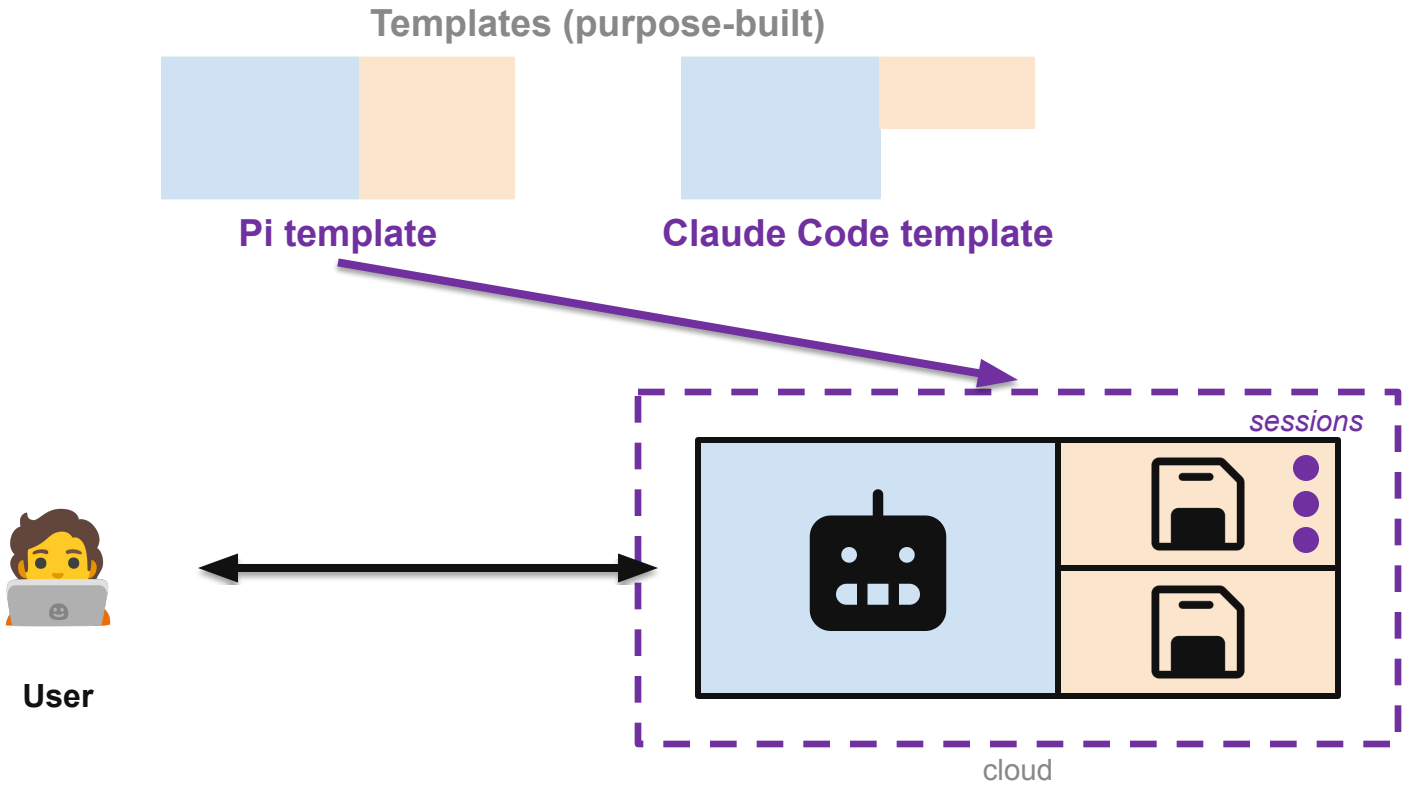
Anatomy of agents



How can we scale usability of cloud resources to many users, many agents, many sessions?

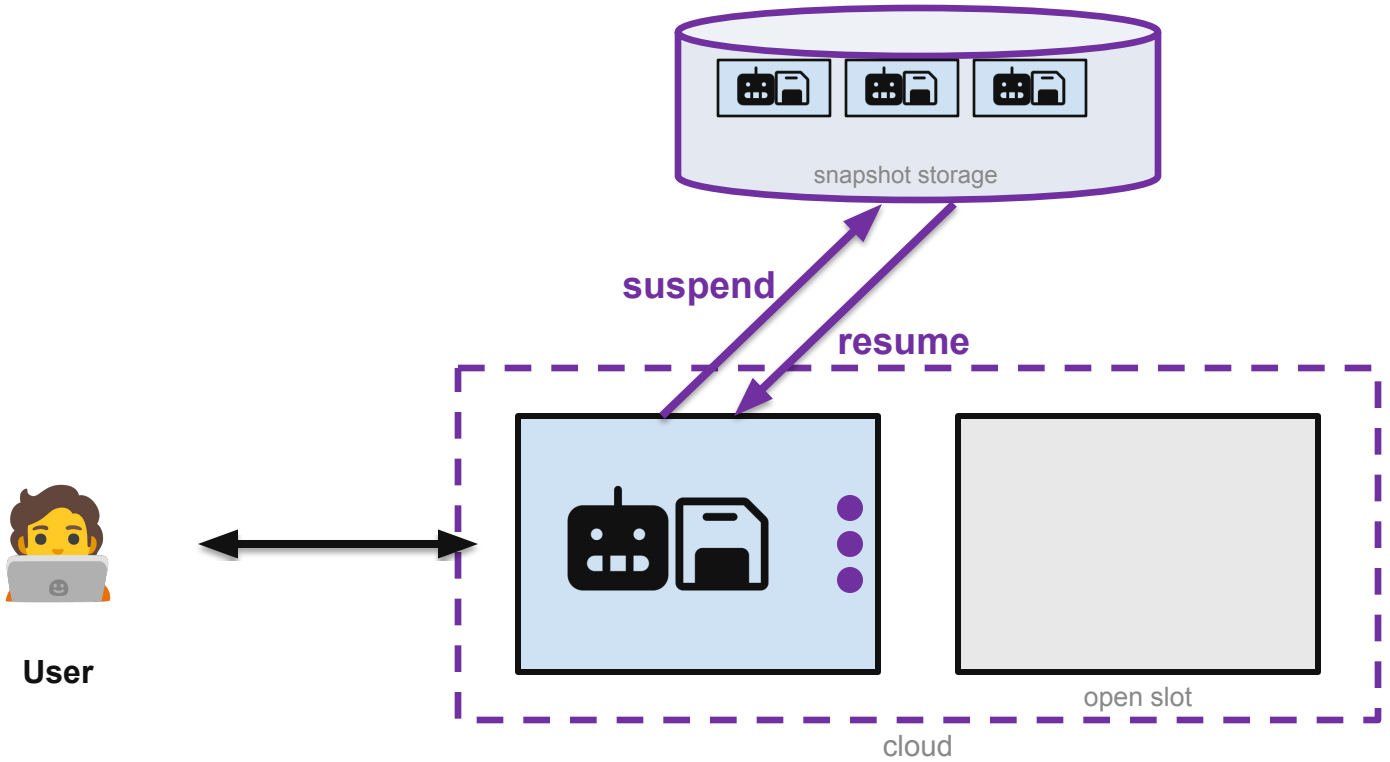
Recent work in the community is starting to answer this →

Emerging solutions: agent-sandbox



A ***template*** defines the sandbox's shape — one per harness.

Emerging solutions: agent-substrate



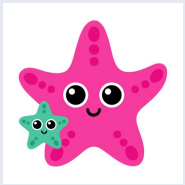
The actor is ***logical***. The worker is ***physical***. Storage keeps them apart.

Harnesses are evolving



Pi

Agent can **overwrite its own code** — maximal configurability.



Omnigent

Meta-harness: **orchestrates and combines** other harnesses.

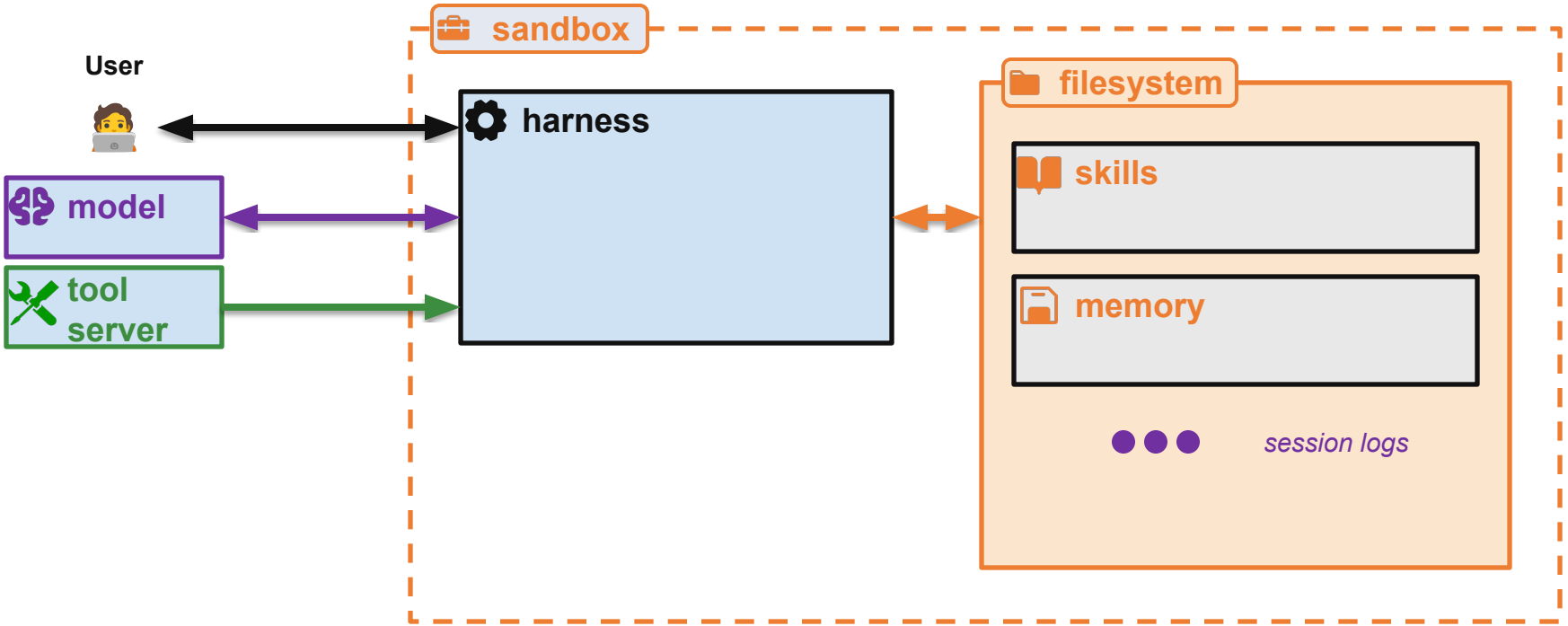


DeepSeek Harness

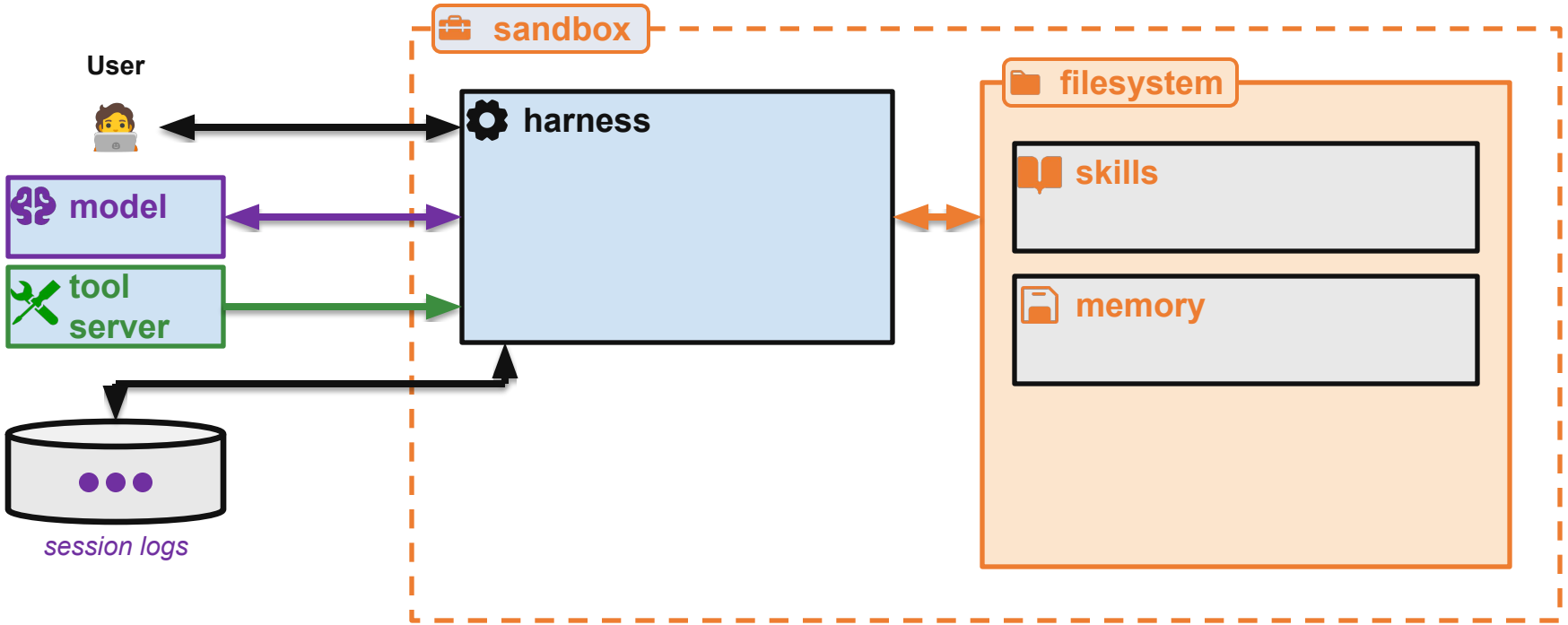
Plugin framework — **swap anything**, even the core loop.

Higher abstractions, but ***fragmentation*** — not consolidation.

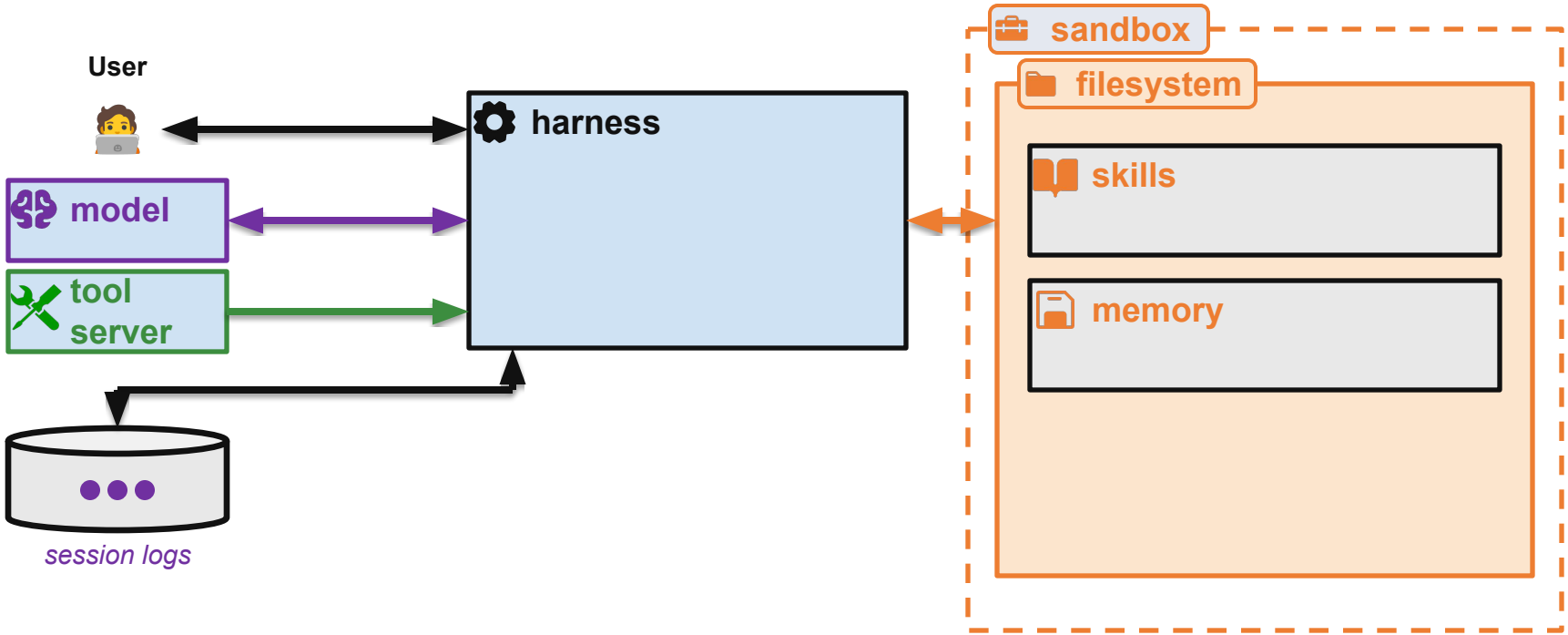
From Monolithic to Managed Agents



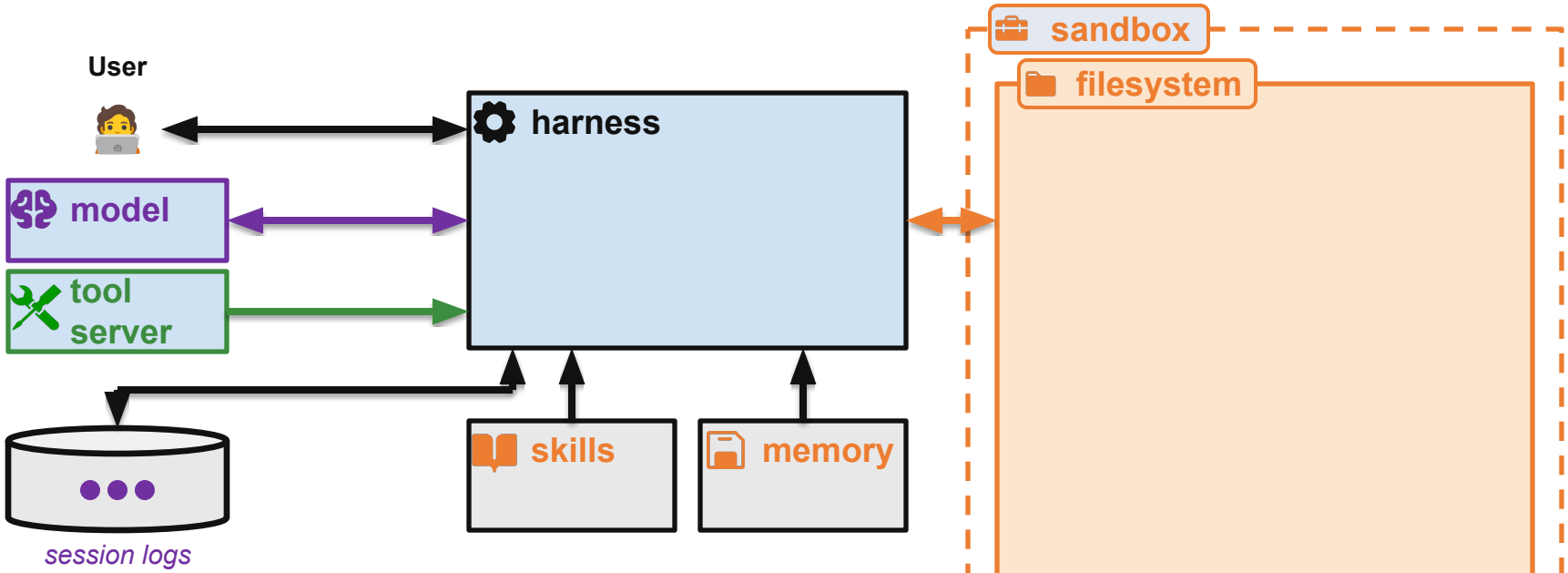
From Monolithic to Managed Agents



From Monolithic to Managed Agents



From Monolithic to Managed Agents



session = (*skills* + *execution sandbox*+ *memory* + *model*)

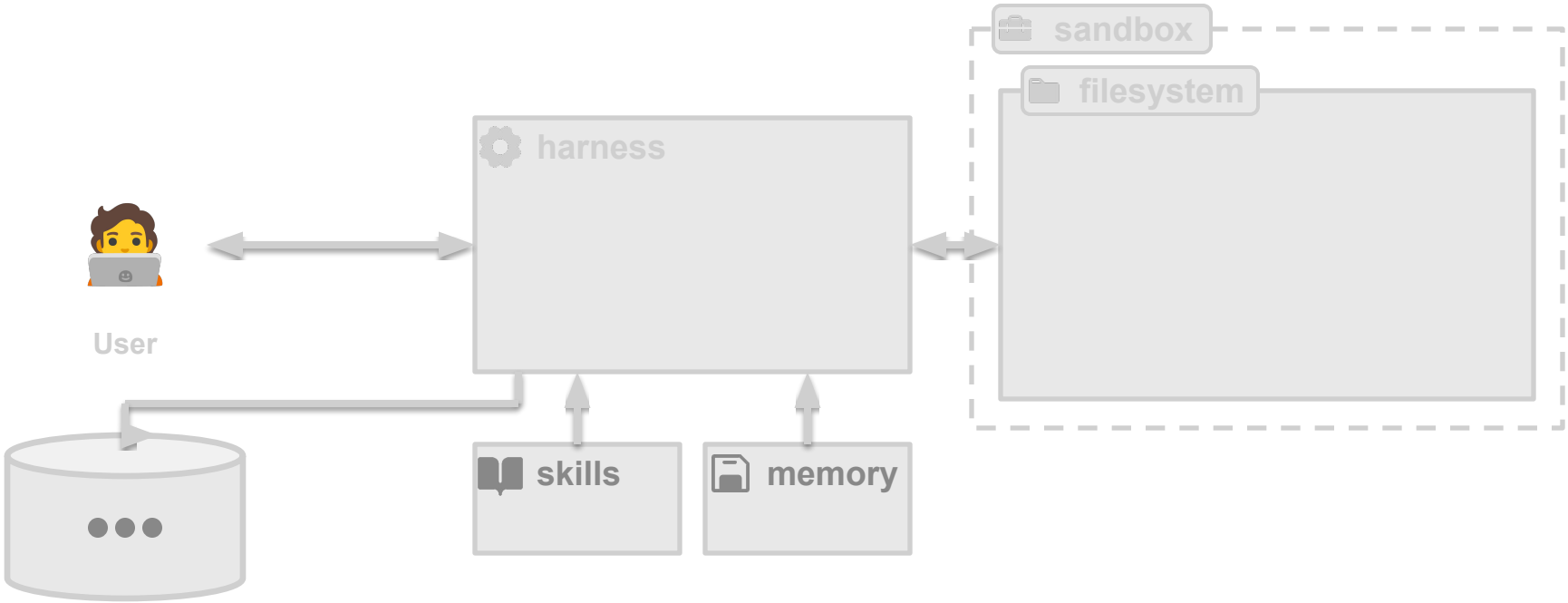


moca

an open-source experiment
Inspired by managed agents

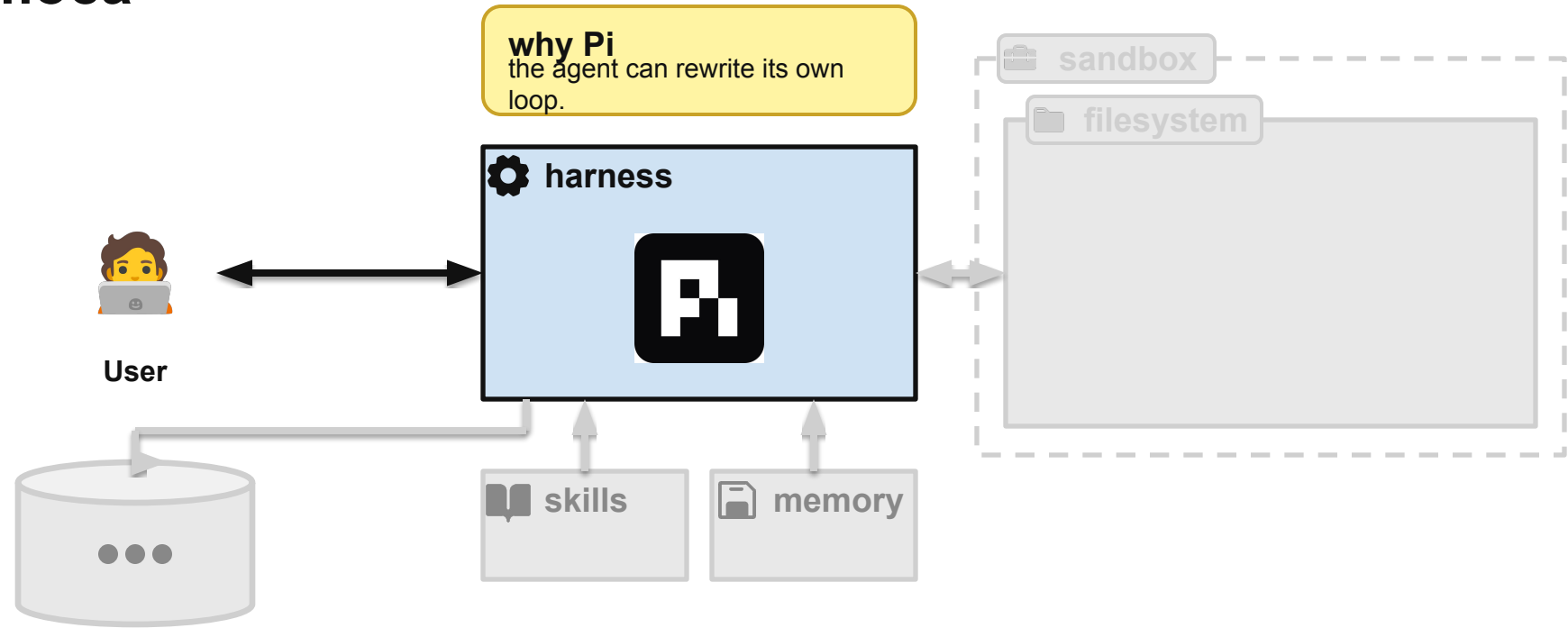
What happens when the harness holds ***nothing***?

moca



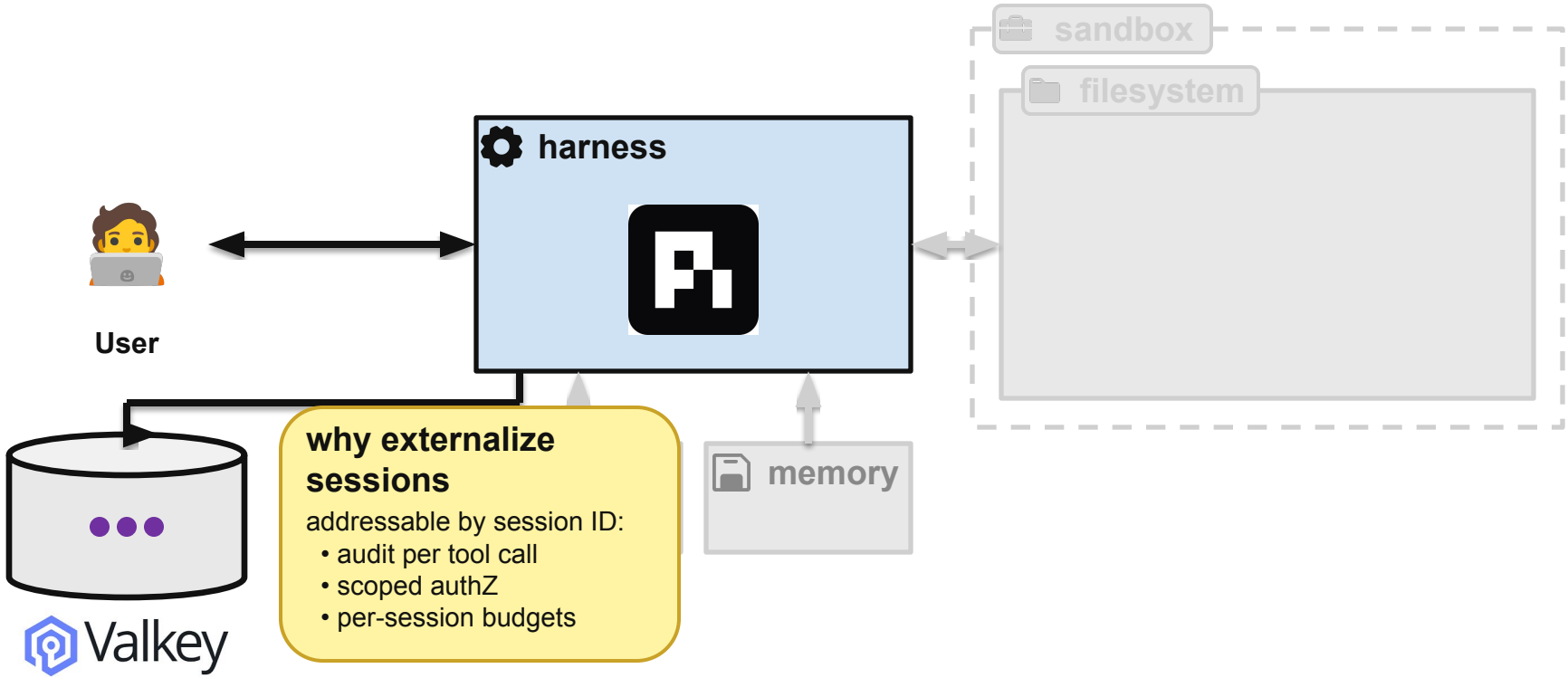
How does ***moca*** implement each piece?

moca



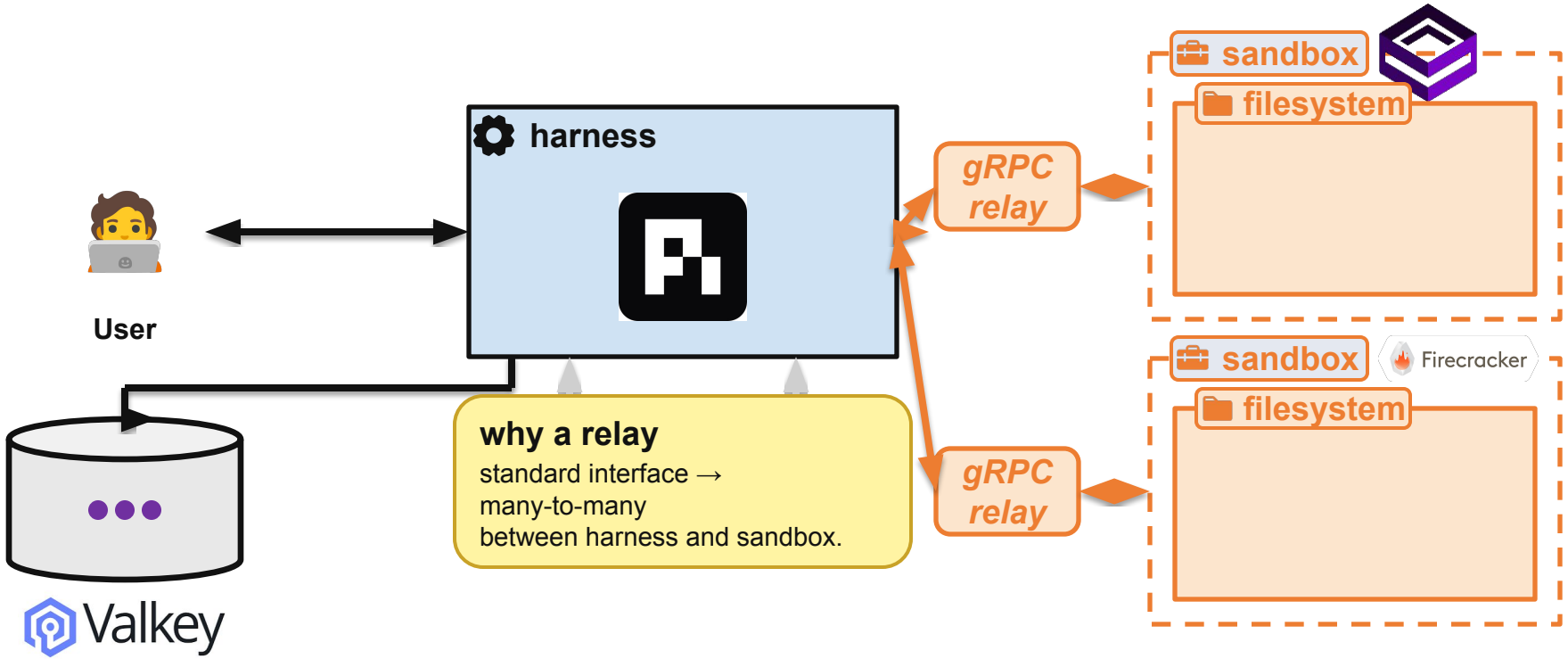
The ***harness*** is ***Pi*** — chosen for its configurability.

moca



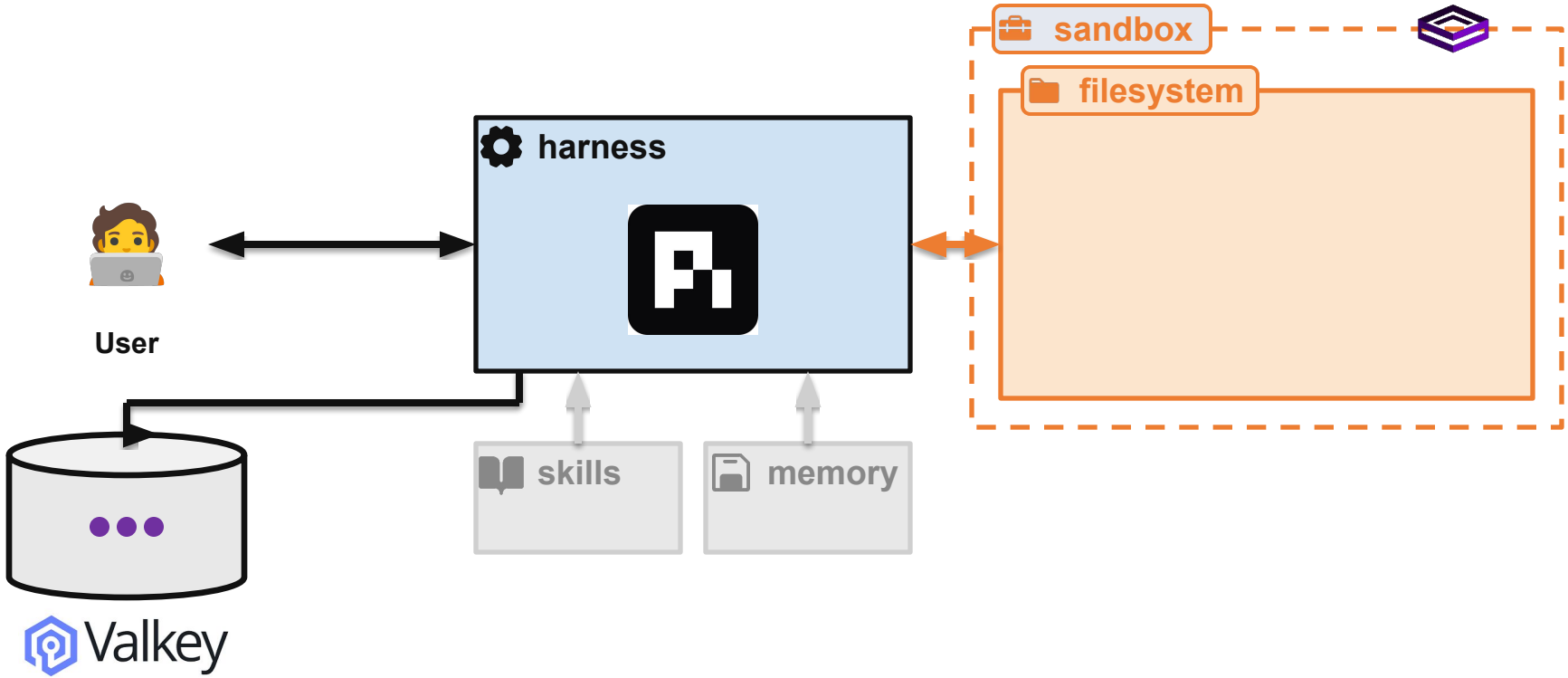
Sessions live in ***Valkey*** — durable, addressable, and tied to a ***session ID***.

moca



A *relay* standardizes the agent ↔ sandbox interface — backends are pluggable: *agent-sandbox* or *Firecracker*.

moca



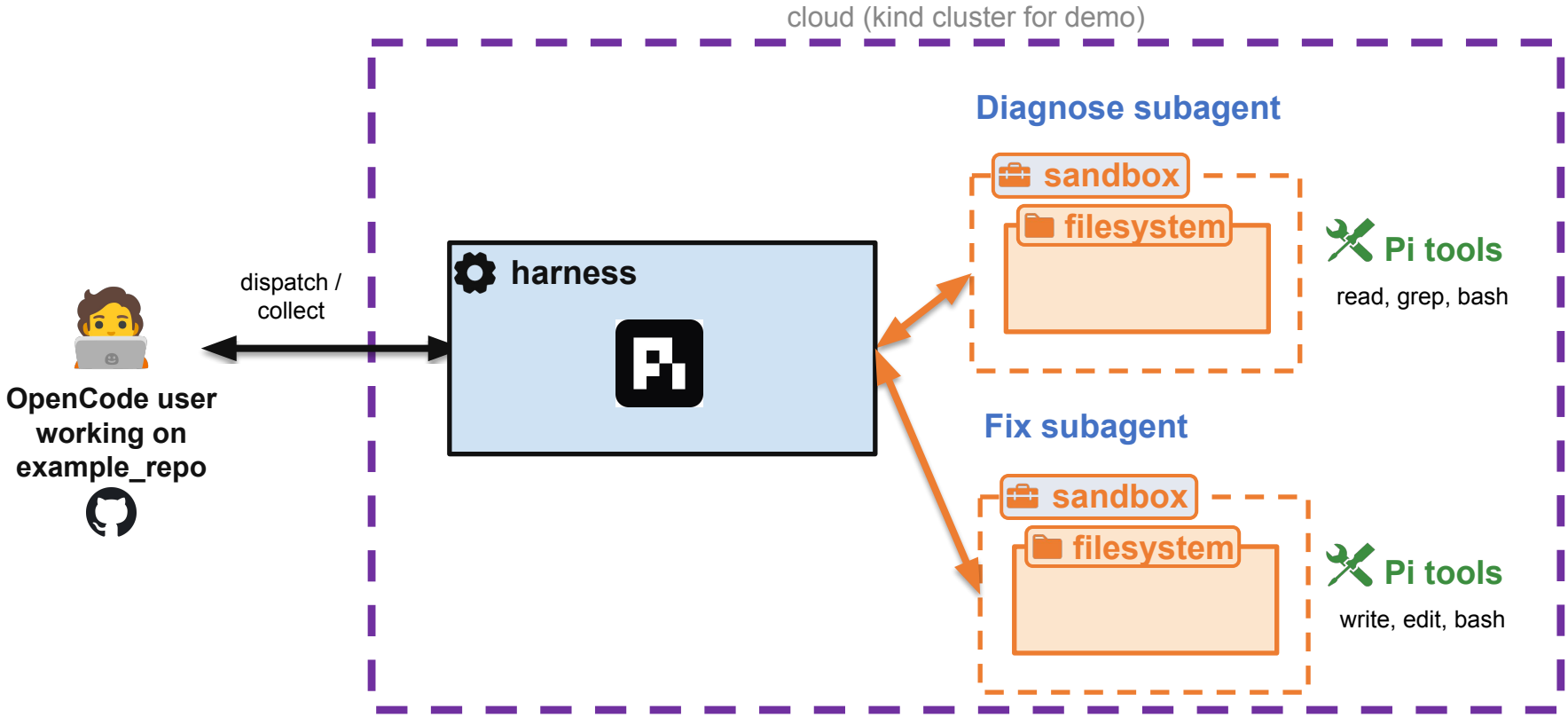
moca — managed agents, end to end.

Delegating subagent tasks to moca

a working prototype

From ***architecture*** to ***practice***.

Demo: laptop handoff to 2 cloud subagents



moca architecture - state focus



OpenCode user

cloud

harness

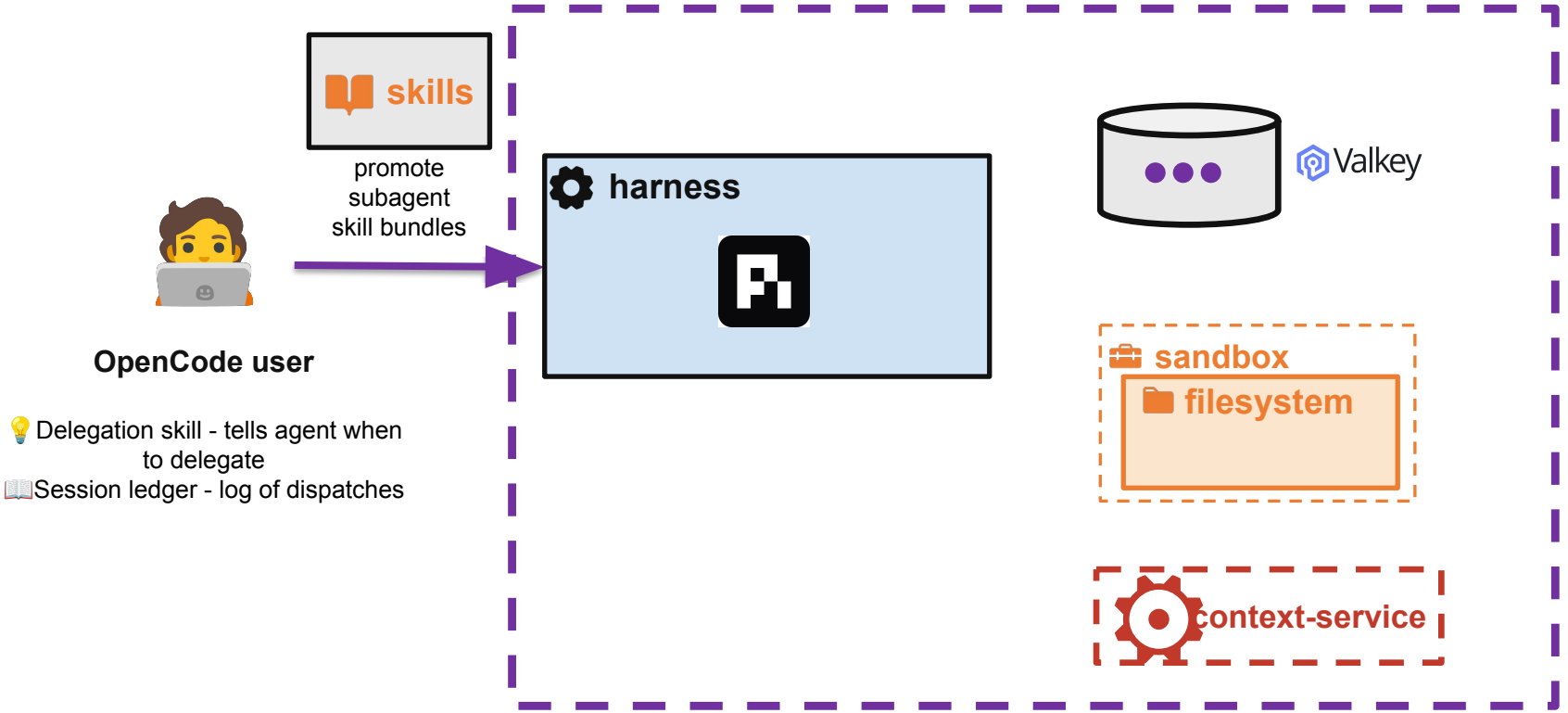
Valkey

sandbox

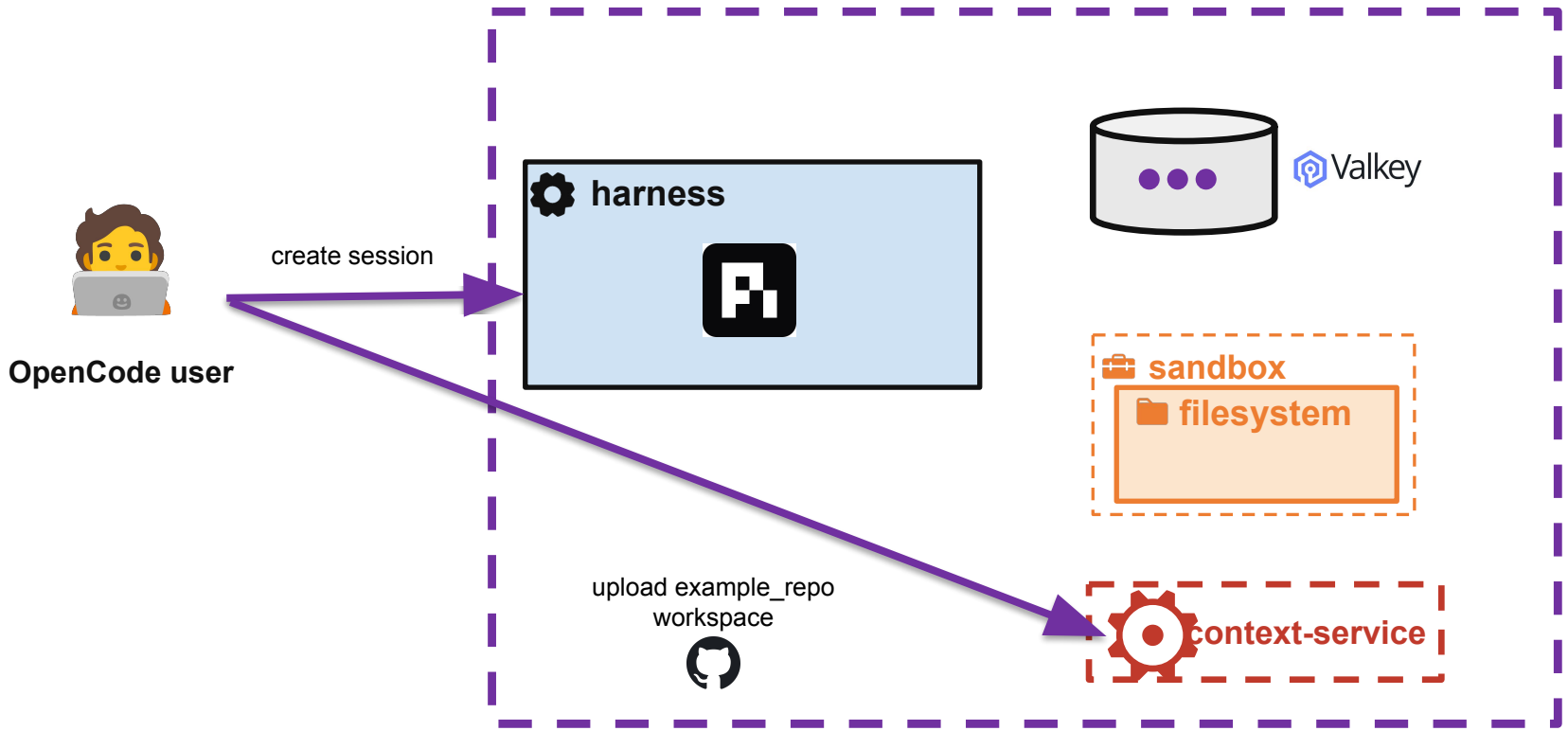
filesystem

context-service

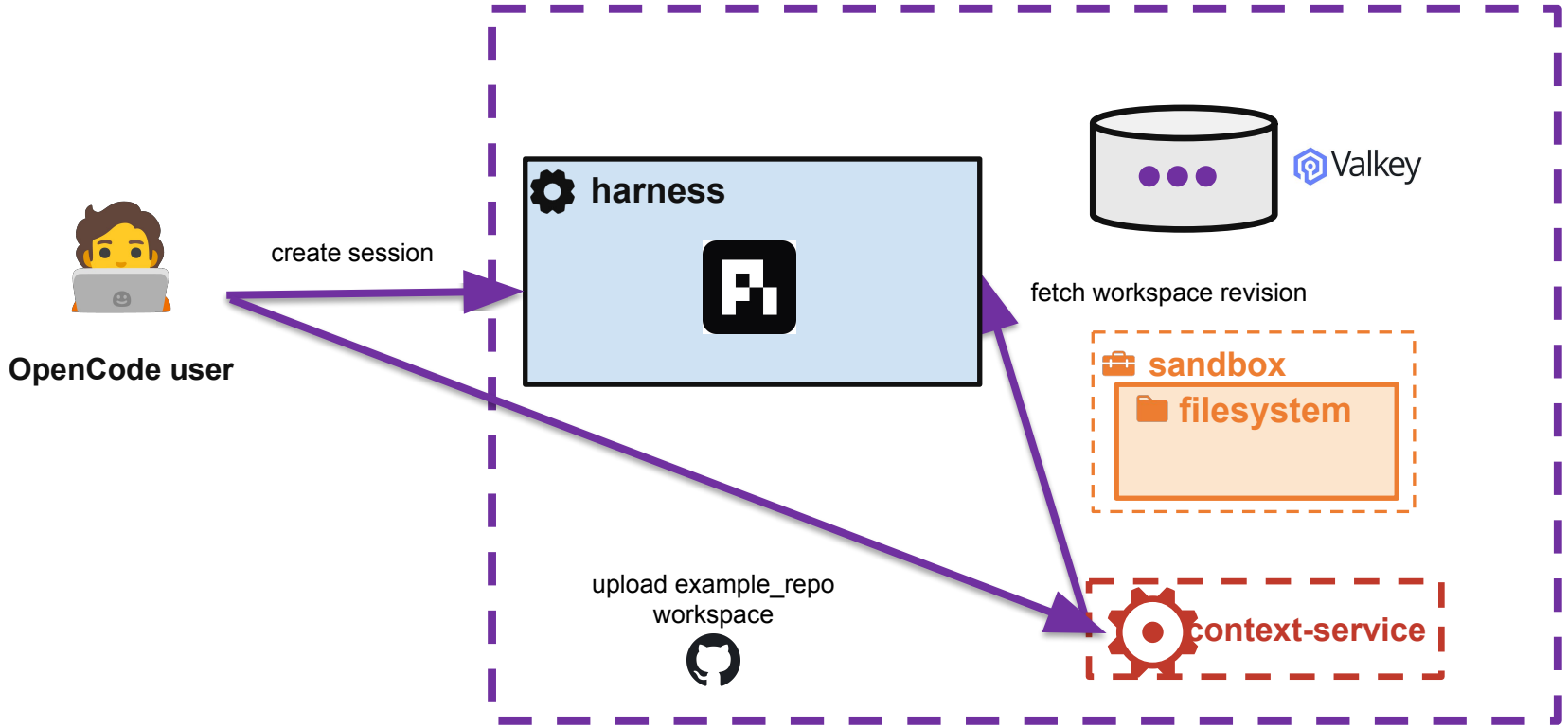
moca architecture - state focus



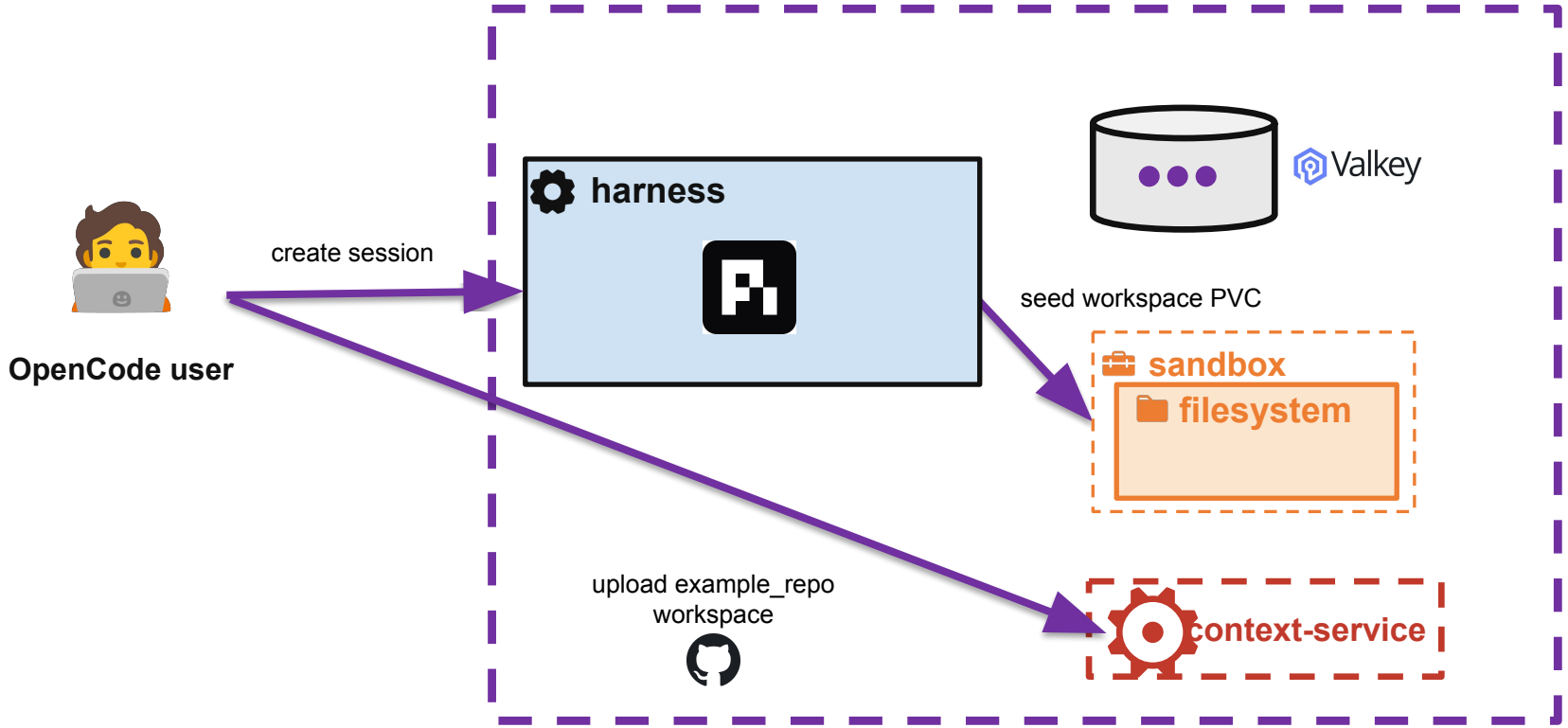
moca architecture - state focus



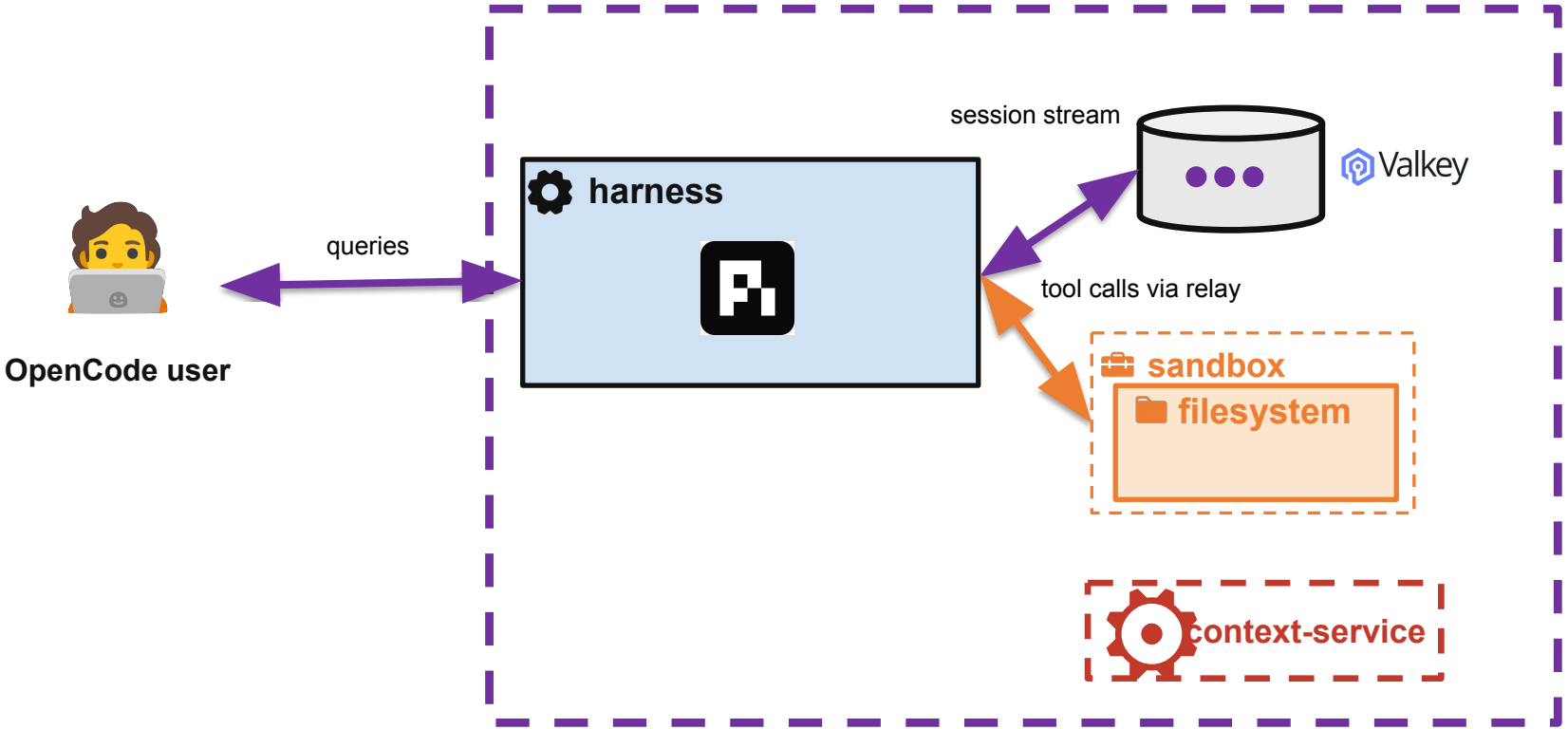
moca architecture - state focus



moca architecture - state focus



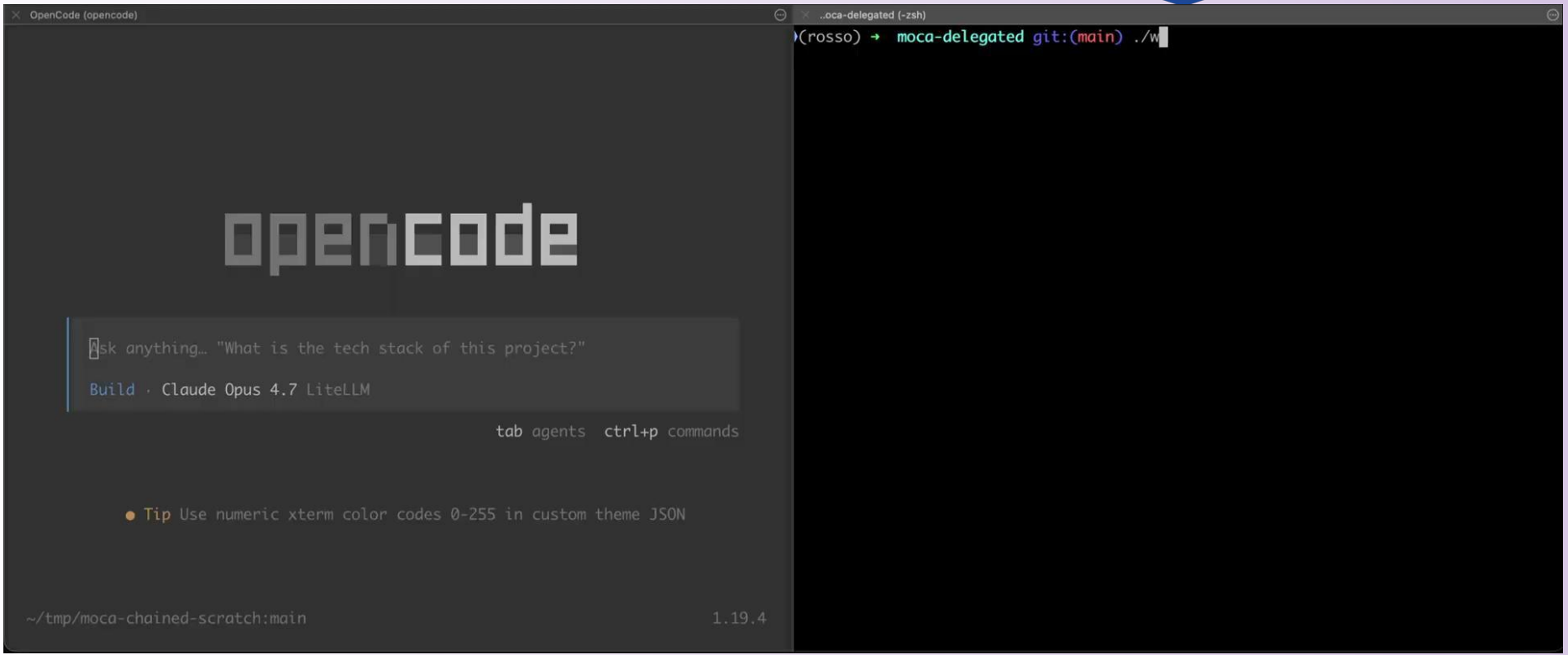
moca architecture - state focus



Demo



Platform Engineering Day
NORTH AMERICA



Every piece of state has ownership

Skills

ON THE SERVER, PROMOTED ONCE

The working directory

FROZEN, COPIED TO EACH SUBAGENT'S DISK

The session's turns

ON THE SERVER, WRITTEN EVERY TURN

The delegation ledger

ON THE LAPTOP, APPEND-ONLY

Nothing durable lives in the agent process.

Platform patterns

1. Make the session first-class and server-held
2. Treat agent state as harness-neutral data
3. Make the harness/sandbox boundary an open protocol

Open questions

How does token spend get attributed through an agent call chain?

Per-call cost is measurable. Attributing it to a user, a workload, or a budget holder isn't yet a shared primitive.

What's the shared audit trail across dispatch and model?

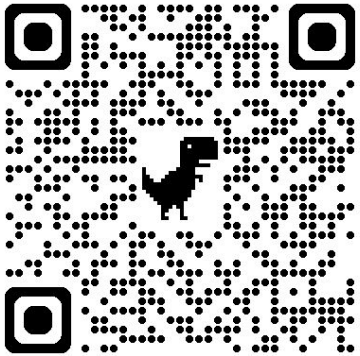
The dispatch side knows who ran what; the proxy side knows what was asked of the model. We want to join them into a provenance record.

How should agents get and share context?

We currently version and store skills and workspaces separately, but expanding for multiple users and agents may shape how we want to share revisions.

Resources

- Project: https://www.rossoctl.dev/
- Demo: https://github.com/rossoctl/state-lab
- moca: https://github.com/rossoctl/moca
- Eventing talk: “The Seams Are the Trap: Agent Fleets from CNCF Parts You Already Have”
  - Thursday 1:50PM, Salt Palace Level 2 | 255 BC



https://www.rossoctl.dev/



moca repo