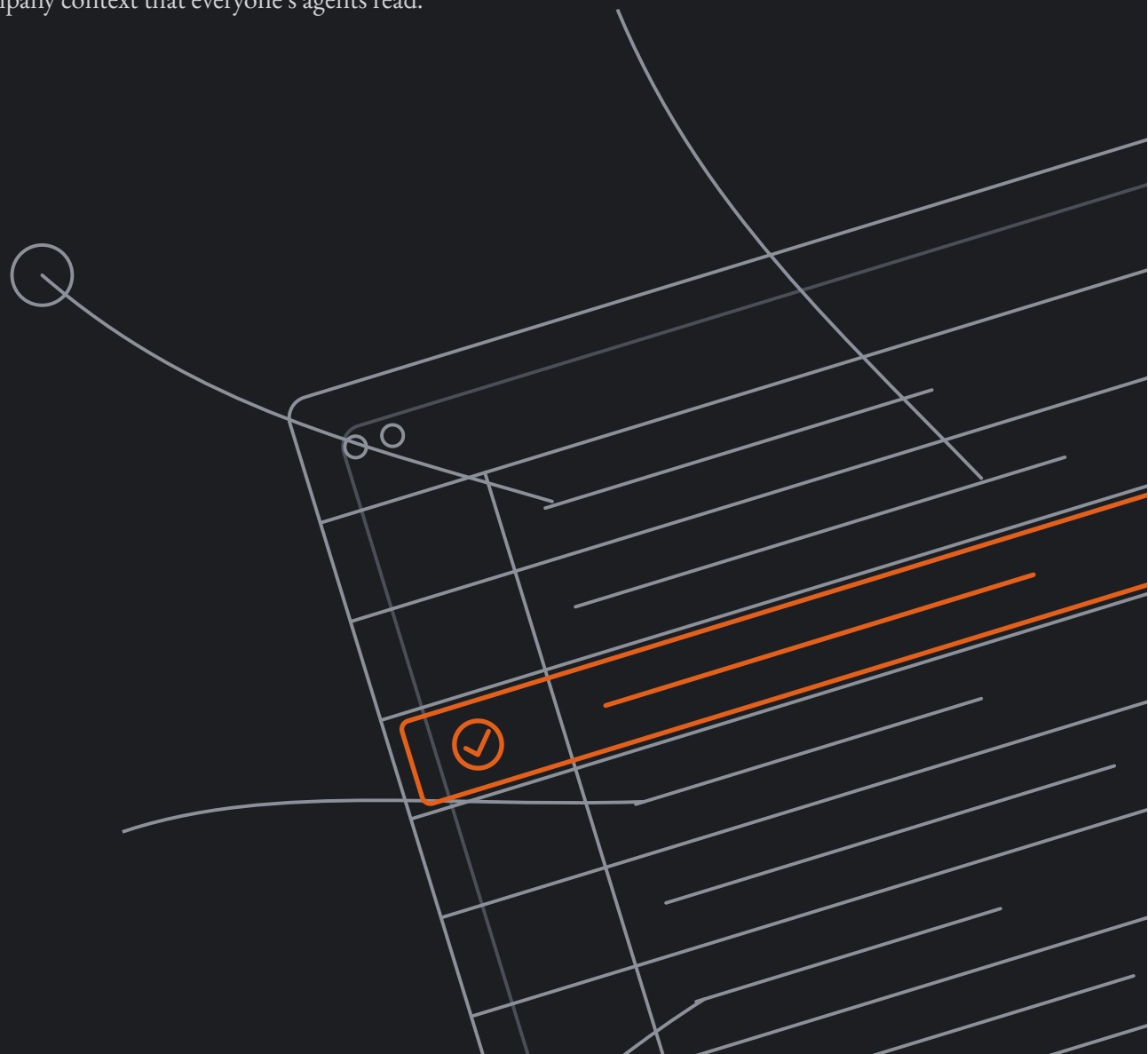
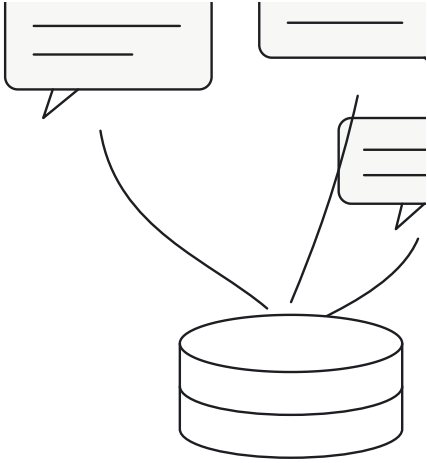
SHAKUDO

A buyer's playbook for data and AI leaders · November 2026

The Multiplayer AI Playbook

Sharing an AI session with a colleague is now easy. This playbook is about the harder part, which is deciding who owns, approves and can change the company context that everyone's agents read.





Introduction

Multiplayer AI arrived in October as something teams could share, while the company context underneath it arrived with nobody named to own it.

In the thirty days to 3 November 2026, the phrase "multiplayer AI" picked up six published definitions, one of them from an analyst firm, and a run of product launches from collaboration and platform vendors.[1-8] Most of what shipped lets people share: a session link, a joinable thread, a workspace where agents show up as members.

That part demos well, and it is worth having. The picture behind the term is colleagues and an AI agent working in one persistent place, where anyone can see what the agent and the others are doing and pick up where someone else left off.[1]

Look at what that place has to hold: conversation history, shared documents, permissions, organizational structure, prior decisions and active projects.[1] Every teammate is meant to "start from the same company context",[2] and in one version the agents keep their state on computers of their own.[3]

Shared context needs what production data already has

- ✓ A named owner for every item
- ✓ Approval before any agent reads it
- ✓ A record of where it came from
- ✓ A change history with a way back
- ✓ A location your team controls

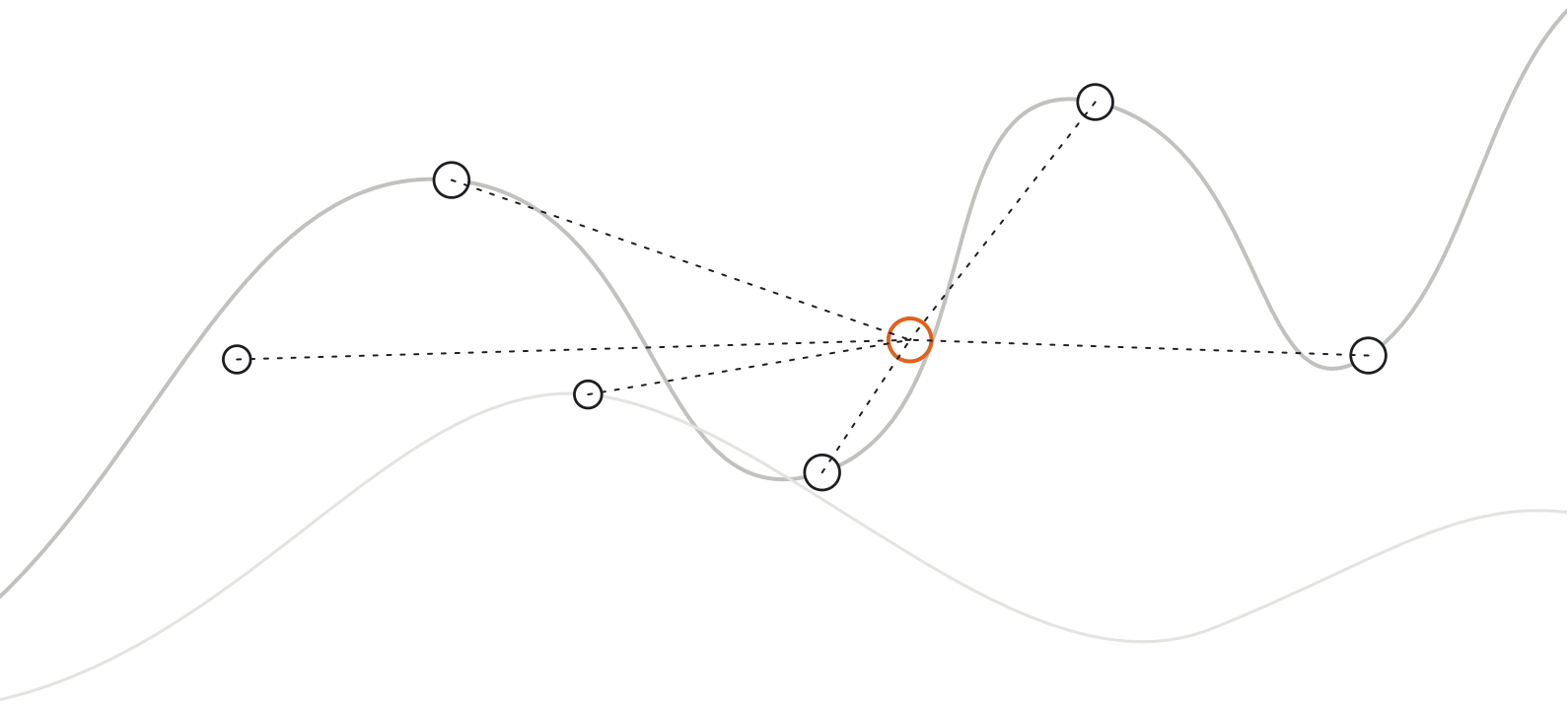
Every one of those definitions assumes a shared, persistent store of company context that many people and many agents read from and write to.

Your data team already knows what a store like that needs, because it runs one for tables and pipelines. It needs an owner, an approval step, a record of where each item came from, and a history of changes with a way back. In the October launch material collected for this playbook, those controls barely appear. The people who did write them down were two governance writers and one research group.[9-11]

This playbook argues that the buying decision turns on those controls, and that a platform should be judged on whether your data team can run them on context the way it runs them on data.

Page 3 lays out six layers, from the shared session at the bottom to change control at the top. Pages 4 to 6 take them one at a time. Pages 8 to 12 turn the upper layers into eighteen questions you can score a vendor on in a single meeting, and page 14 lists what to get in writing before you sign.

1 Forrester, Multiplayer AI: Your Impending (And Accidental) Standard, 13 Oct 2026 · **2** Ramp, Multiplayer AI: How Teams Collaborate on AI Work, 30 Oct 2026 · **3** Dock, Multiplayer AI needs agents with their own computers, 1 Nov 2026 · **4** Atlassian, Atlassian Introduces AMP: The Agentic Multiplayer Protocol to Power Human-AI Collaboration, via Business Wire, 7 Oct 2026 · **5** Cisco, Cisco Unveils New Agentic Collaboration Experiences, 7 Oct 2026 · **6** Google Cloud, Gemini at Work 2026: Introducing Gemini agent, 8 Oct 2026 · **7** ThoughtSpot, Introducing AgentSpot: From Chatbots to Multiplayer AI, 22 Oct 2026 · **8** OpenAI, Introducing OpenAI Frontier, 28 Oct 2026 · **9** Trail, How should a memory layer for LLM agents be governed?, 19 Oct 2026 · **10** Lyzr, Best Platforms for Governing Multi-Agent Architectures, 19 Oct 2026 · **11** IBM Research, Transparent, Traceable, Deterministic: Agentic Memory via Knowledge Graphs, ISWC 2026, 25 Oct 2026



Of the six definitions published in October, only one gives a buyer something to test

Six publishers defined multiplayer AI between 12 October and 1 November.[1-3, 12-14] They agree that people belong inside the definition, and the clearest phrasing is that the term "is about who can participate in the work, rather than how many agents are running."[13]

On anything a buyer can check, they disagree. One calls multiplayer AI "the shift ... to a shared one" and offers as proof that it "makes the company smarter".[2] As a definition that goes in a circle: multiplayer means shared, and the evidence is an outcome you see only after you have bought. The analyst version works by recognition ("If my colleagues and I are all working with an AI agent in a shared place ... that's multiplayer AI"), which describes how the thing should feel and leaves open what you should require. Its author calls the name itself "imperfect".[1]

Only one of the six comes with a test. It asks for agents doing the work "on computers of their own", and says you can check any product for it in an afternoon.[3] Its author sells a product built to clear that bar, at "$40 a person a month" for up to five cloud agents, and the test is still worth running, because a definition you can check is worth more to a buyer than one you can only recognize.

Once agents act on company context, that context needs the controls you already give production data

When agents move from answering questions to executing business processes, a wrong fact in their context turns into a wrong action. That makes the context, in one analyst's phrase from later that month, "a production dependency rather than a supporting asset."[15]

A production dependency gets an owner, a gate in front of it, lineage and change control. No data team would accept a shared table that anyone can edit in place, with no record and no way back. Multiplayer AI asks you to accept exactly that for the context your agents act on, unless the platform underneath lets your team apply the controls it already uses.

The governance writing that did appear that month reads like a lineage standard written for agents. One published policy for memory layers asks that every memory "name its source, its owner and the person who approved it", that approval "has to happen before agents read a memory", and that rules change "through the same steps as code".[9] Another newsletter put the open question in its headline: "Multiplayer AI: Who Writes the Rules?"[16]

That makes the buying question narrower than the launch material suggests. Can your data team run the controls it already runs on tables and pipelines on the context under your multiplayer AI, on infrastructure it governs?

1 Forrester, 13 Oct 2026 · **2** Ramp, 30 Oct 2026 · **3** Dock, 1 Nov 2026; price per trydock.ai/pricing, checked 5 Nov 2026 · **9** Trail, 19 Oct 2026 · **12** Brain, What Is Multiplayer AI?, 12 Oct 2026 · **13** Pairon, What is multiplayer AI coding?, 12 Oct 2026 · **14** Koru, What is multiplayer AI? People and agents, together, 22 Oct 2026 · **15** Forrester, Context Engineering And ContextOps: The Missing Foundation For Scaling Agentic AI, 19 Oct 2026 · **16** Atlan, Multiplayer AI: Who Writes the Rules?, 22 Oct 2026

Six layers between a shared chat and a system you can run

Demos show the bottom of the stack, while your auditors and whoever inherits the work will ask about the top.

What an auditor checks



Layer 6

Change control inside your own perimeter

Edits arrive as reviewed diffs with a revert path, kept on infrastructure you govern

Layer 5

Approval and provenance

Every item has a source, an owner and an approver, signed off before an agent reads it

Layer 4

Agent identity

Agents act under their own accounts, with spend limits and a kill switch

Layer 3

Shared context

One body of company context that outlasts the session

What a demo shows

Layer 2

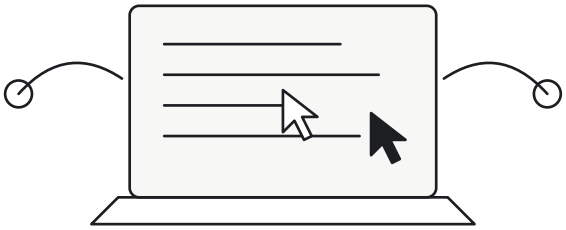
Visibility

Anyone can follow the work while it runs, and step in

Layer 1

Shared sessions

Several people in one live agent session



Layer 1

Shared sessions

Shared sessions are the easiest layer to demo and the cheapest for another vendor to copy, so they should not decide a purchase.

By late October a coding agent session could be shared with a link: one command sends it, and another pulls a colleague into the conversation and the uncommitted worktree.[17] At least four teams, one of them a large data platform vendor, shipped variations on the same idea within a few weeks.[17-20] Engineers get real use from it. A feature that four teams can ship inside one month is still a poor anchor for a platform decision.

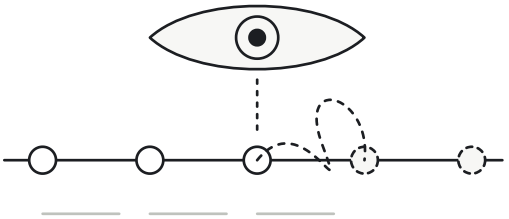
What it looks like

- A link that drops a colleague into a live agent session
- Co-driving or forking a session[18]
- Several coding agents side by side, with approvals from any browser

Risks

- One product's own documentation warns that "Edit is a code-execution grant on the host"[18]
- Approval rests with whoever is hosting: "the host approves what guests ask for"
- In one published comparison, one coding agent's share link gave guests a read-only view, and another agent's session could be driven by only one person[21]

Focus: treat sessions as table stakes. Ask who approves a guest's action when the host is offline, and where that approval is recorded.



Layer 2

Visibility

Most of October's complaints were about not seeing the work, but being able to step in, and leave a record when you do, matters more.

The month's problem statements describe work only one person can see. "The good prompts sit in a chat window no one else can see," one launch post said.[22] Another says the work falls apart because "nobody can see the whole exchange and nothing records who decided what."[23] A developer described an agent left running overnight that got stuck in a test loop, while a teammate could see only a branch name and had no way to nudge it.[24]

That second complaint has two halves, and the second half is the requirement. Seeing the exchange gives you a viewer. Recording who decided what gives you an audit trail.

What it looks like

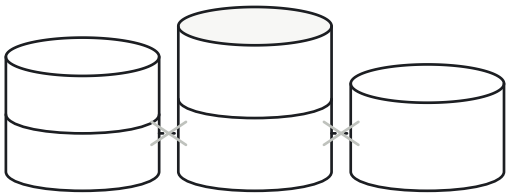
- A live view of what the agent and colleagues are doing
- Approval requests that reach a person on a phone or in a browser

Risks

- A view with no way to intervene, as in the overnight test loop
- Watching without a record, so nobody can reconstruct who decided what

Focus: require that every human intervention on an agent leaves a record with a name and a time.

17 Delta, Claude Code is now multiplayer, 27 Oct 2026 · **18** Databricks, What tools let multiple people collaborate in one AI coding agent session? (Omnigent), 9 Oct 2026 · **19** shared-session: multiplayer Claude Code sessions by link, 16 Oct 2026 · **20** Bellows, Live-share multiple coding agents side-by-side and approve changes from any browser, 2 Nov 2026 · **21** Eigen, Shared Workspaces for Claude Code and Codex: What Your Team Can Actually Share in 2026, 22 Oct 2026 · **22** Agor, Announcing Agor Cloud Open Beta, 13 Oct 2026 · **23** Wavelave, How to Run Multiple AIs Together, 10 Oct 2026 · **24** DEV Community (dev.to) post, 28 Oct 2026



Layer 3

Shared context

The suite vendors that shipped shared context in October each kept it inside their own products.

One collaboration vendor grounded its new agent protocol in its own work graph.[4] A productivity suite gave its agent the same memory and skills across the suite's apps.[6] A communications vendor built its update around people, “digital teammates” and shared enterprise context.[5] Each choice makes sense for the vendor making it. A company that runs all three ends up with company context in three places, governed three ways, with nothing joining them.

A protocol reaches further than a single product, but this one is still grounded in its owner's graph. Ask any vendor offering one where the context sits when the agent on the other end belongs to someone else.

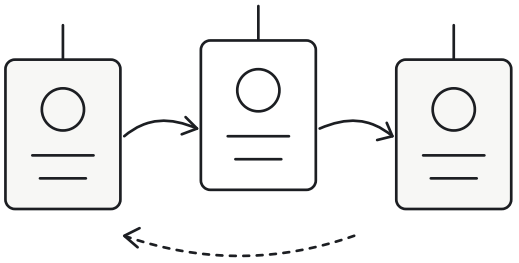
What it looks like

- One agent memory carried across a vendor's suite
- Context assembled from a vendor's own work graph
- Every teammate starting “from the same company context”[2]

Risks

- Context splits by vendor, each copy under its own retention terms
- Nobody owns the joins between the copies

Focus: get the lifecycle in writing. What gets written to shared context, who can read it, and when it expires.



Layer 4

Agent identity

An agent that acts on its own needs a name you can revoke and a budget you can cap.

Two early-October launches made agents members of a workspace, with their own identities, accounts, mailboxes and shared files,[25, 26] and another product puts each agent on a persistent machine of its own.[3] An analytics vendor launched its agent product with the line that “this next wave gives the whole company an AI workforce”.[7] If that is the plan, the workforce needs badges and budgets.

Once agents call other agents, identity is how you find out which one did what, and how you stop it. One governance write-up gives the example of a research agent that calls a pricing agent, which calls a supplier agent, which times out and retries 40 times, with every retry billed.[10]

What it looks like

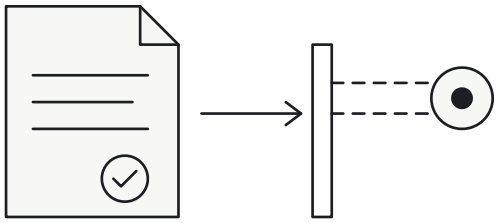
- Agents as named members of a team workspace
- Agents with their own accounts, mailboxes and files
- Agents running on persistent machines of their own

Risks

- Chains of agent calls with no spend ceiling
- Agents acting under a person's credentials, so the log names the wrong actor

Focus: treat agent identity, authorization on agent-to-agent calls, spend limits and kill switches as the minimum control set.[10]

2 Ramp, 30 Oct 2026 · **3** Dock, 1 Nov 2026 · **4** Atlassian via Business Wire, 7 Oct 2026; see also Atlassian, The Atlassian platform is for multiplayer human-agent work, 6 Oct 2026 · **5** Cisco via BusinessPressWire, 7 Oct 2026 · **6** Google Cloud, 8 Oct 2026 · **7** ThoughtSpot, 22 Oct 2026 · **10** Lyzr, 19 Oct 2026 · **25** Kylon, AI Agents Were Built for One Person. We Built Kylon for Teams., 4 Oct 2026 · **26** Agent Workplace, Agent Workplace is live, 5 Oct 2026



Layer 5

Approval and provenance

Review after the fact comes too late, because the agent has already read the memory.

The rule most products skip is approval before the read. Approval "has to happen before agents read a memory; review after the fact catches errors only once an agent has already used them." Every memory an agent can act on should name its source, its owner and its approver, "so a wrong answer can be traced to a line and a decision."[9] A research paper on graph-based agent memory made the same case for memory that is "transparent, traceable, deterministic".[11]

The selling point that "what one person figures out becomes everyone's starting point" is also the risk statement for this layer, because a wrong fact spreads the same way.[2] Memory poisoning was a live security topic through the month.[28] A security lab write-up on 8 October showed instructions planted in an agent's memory surviving beyond the initial compromise.[29]

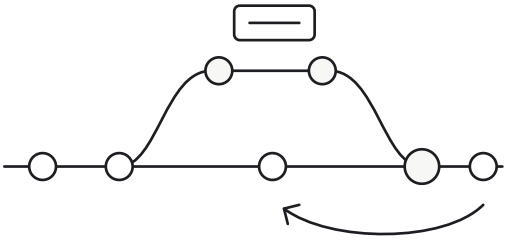
What it looks like

- A write gate in front of shared memory[27]
- Source, owner and approver attached to every memory
- Unapproved items held back from every agent

Risks

- One session's unreviewed output becomes everyone's starting point
- A poisoned or stale fact survives a restart

Focus: make approval before read a contract term, and ask to see it enforced in the demo.



Layer 6

Change control inside your own perimeter

If a vendor cannot show you who changed a memory and how to undo it, there is no change control to buy.

Rules for agents should change "through the same steps as code: a proposed diff, a named reviewer, a merge, and a revert path."[9] Your data team already works this way for pipelines and would refuse anything less for a production table.

The second half of this layer is location. A change history is only as trustworthy as the system that keeps it. A history held on a vendor's infrastructure, under the vendor's retention terms, is the vendor's record of your decisions, and you read it on the vendor's terms.

What it looks like

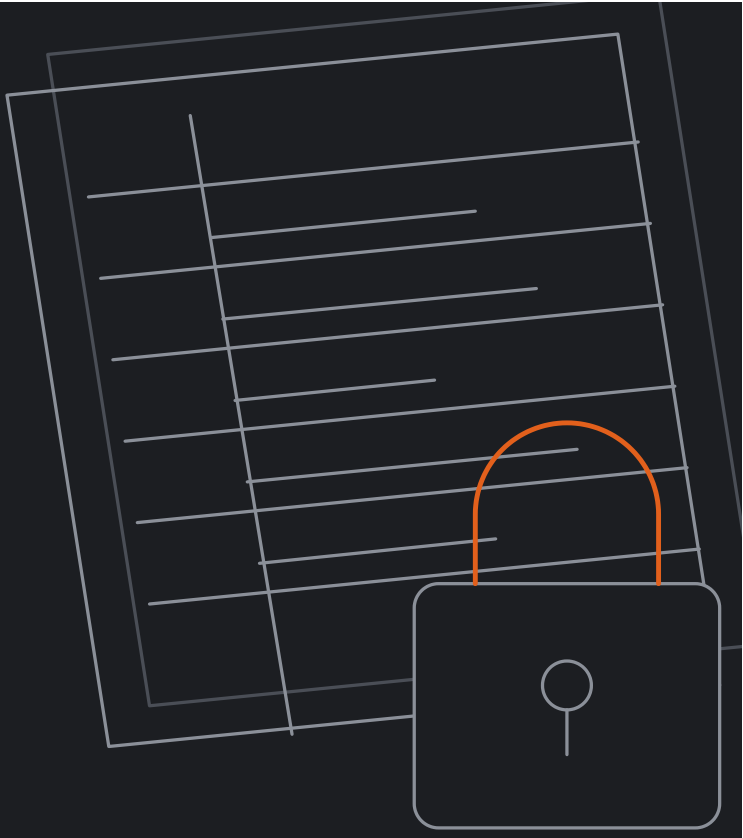
- Edits to shared context arrive as reviewed changes with a revert path
- A history your auditors can query without opening a ticket
- Context stored on infrastructure you govern, under your retention rules

Risks

- Edits in place, with the chat log as the only record
- An exit clause that returns your files but not your context or its history

Focus: ask to see one edited memory, its edit record and the approver's name. If the vendor cannot show it in the meeting, score this layer as missing.

2 Ramp, 30 Oct 2026 · **9** Trail, 19 Oct 2026 · **11** IBM Research, ISWC 2026, 25 Oct 2026 · **27** Richard Hogan, themicrosoftcloudblog.com, Agent memory needs a write gate, not a bigger store, 28 Oct 2026 · **28** SC Media, After memory poisoning, an AI agent needs more than a restart, 23 Oct 2026 · **29** Zenity Labs, AgentCorruption: Weaponizing Agent Memory for Persistent Hijacking, 8 Oct 2026



The scorecard

Eighteen questions for the vendor meeting

These questions cover layers 3 to 6, where October's launch material had least to say. Each one can be answered with a demo or a document. Use them on the platform you are evaluating, or on the one you already run.

How to score a vendor on the eighteen questions

Score each question from 1 to 10, using the anchors at 1, 5 and 10. Score only what you saw working. A roadmap slide scores 1.

Add the eighteen scores. The total falls between 18 and 180, and page 13 tells you what it means.

Section A

Where shared context lives and how long it is kept

Every other control depends on this one, because your team cannot enforce a rule on a store it does not run.

1 Where does the shared context physically sit: conversation history, documents, decisions and the memory agents write?

1 The vendor could not say exactly where
5 In the vendor's cloud, in a region you choose, under the vendor's retention terms
10 Inside your VPC, on-prem or air-gapped, on compute you own

2 Is there a written rule for what gets written to shared context, who can read it, and when it expires?

1 Whatever the agents produce is kept indefinitely
5 Retention settings exist, but nobody owns the policy
10 A written lifecycle, owned by a named team and enforced by the platform

3 Does one body of context serve every tool your teams use, or does each tool keep its own?

1 Each product keeps its own memory, with no joins
5 Connectors copy context between tools on a schedule
10 One governed store that every agent and tool reads from

Section B

Ownership and approval

Each item needs a named person who will fix it when it is wrong, and someone has to approve it before any agent reads it.

4 Does every item in shared context carry a named owner?

1 The owner is whoever wrote it, if anyone can tell
5 Collections have owners; individual items do not
10 Every item has an owner, and ownership moves to a named person when someone leaves

5 Can a new memory be held until a named person approves it, before any agent reads it?

1 Agents read anything the moment it is written
5 Review happens after the fact, through reports
10 Unapproved items are invisible to agents, and the platform enforces it

6 Can you decide what is shared and what stays private before anything is rolled out?[2]

1 Sharing is on by default for everything
5 Per-workspace settings, chosen by whoever set up the workspace
10 Scope set by policy for each class of data, before rollout

7 When a colleague joins a live session, who approves what they can run?

1 The host clicks yes, and nothing is recorded
5 The host approves, and the approval is logged
10 Policy decides what guests may run, and every approval is logged with a name

2 Ramp, Multiplayer AI: How Teams Collaborate on AI Work, 30 Oct 2026

Section C

Tracing an agent's answer back to its source

When an agent gets something wrong, you will want the line it read and the person who let it in.

8 Can you trace any answer an agent gave to the memory items it used, and each item to its source?

1 No trace beyond the chat transcript
5 Retrieved documents are listed; memory items are not
10 Every answer links to the items it used, and each item to its source, owner and approver

9 Is there a controlled path for writing into shared memory, or can any session write to it?

1 Any session can write anything
5 Writes are filtered by type or size, with no reviewer
10 A write gate with rules, a reviewer and a log

10 If a wrong or poisoned fact gets into shared memory, can you find every agent that read it and every answer it touched?

1 Restart the agent and hope
5 You can delete the fact, but not find what it touched
10 Quarantine the item, list its readers and answers, and roll back

Section D

Change control for rules and memory

Your pipelines already change through review, and the context your agents act on should change the same way.

11 Does a change to a rule or a memory arrive as a proposed diff with a named reviewer?

1 Edited in place
5 A version history, with no review step
10 A proposed diff, a named reviewer, a merge and a revert path

12 Can your team return shared context to its state on a given date?

1 Your team has no way to do it
5 For some stores, by asking the vendor
10 Yes, for any store, without the vendor, and the revert itself is recorded

13 Can an auditor see who changed what, when, and who approved it, without asking the vendor?

1 Only through a support ticket
5 Exports on request, in the vendor's format
10 Directly, in your own systems, next to the lineage you keep for data

14 When someone leaves, does their work stay with the company?

1 Their history goes with their account
5 An admin can recover it by hand
10 The context stays, ownership moves to a named person, and the next person starts where they stopped

Section E

Agent identity, spending limits and the terms of exit

Agents that act need identities you can audit and limits you can enforce, and your context should still be yours after the contract ends.

15 Does each agent act under its own identity, separate from the person who launched it?

1 Agents use a person's credentials
5 Agents share a service account
10 One identity per agent, with its own permissions and its own log

16 When one agent calls another, is the call authorized and recorded?

1 Any agent can call any agent
5 Calls are logged but not authorized
10 Policy authorizes each call, and the record names both agents

17 Can you cap what a chain of agents spends, and stop it while it runs?

1 You find out from the invoice
5 Budget alerts at the account level
10 Spend limits for each agent and a kill switch your team holds

18 If you leave the vendor, what comes with you?

1 An export of documents, with no memory and no history
5 Memory exports in a proprietary format
10 Context, history and approvals stay on your infrastructure, because they never left it

What your total score says about the platform

Add the eighteen scores. The lowest possible total is 18 and the highest is 180.

18 to 54

A shared chat window

You have sessions and perhaps a live view, which is enough for a pilot inside one team as long as customer and regulated context stays out of it.

55 to 90

A shared workspace with a memory

Context persists, but nobody owns it and anything can write to it. Expect a launch demo scored on its own to land here.

91 to 126

Shared context with an owner

Approvals and provenance exist for some stores. Close the gaps in section D before agents act on it unattended.

127 to 158

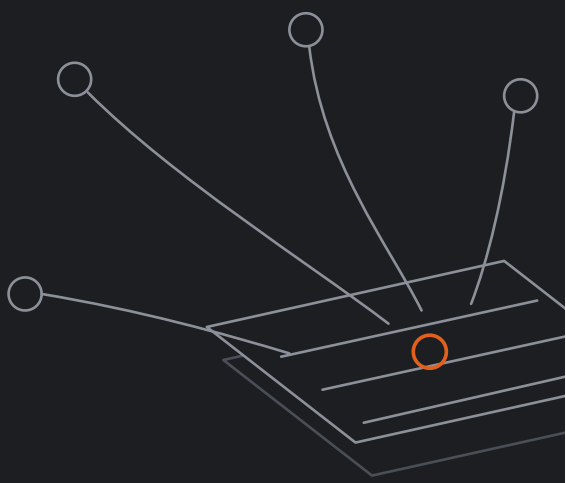
Context you could defend in an audit

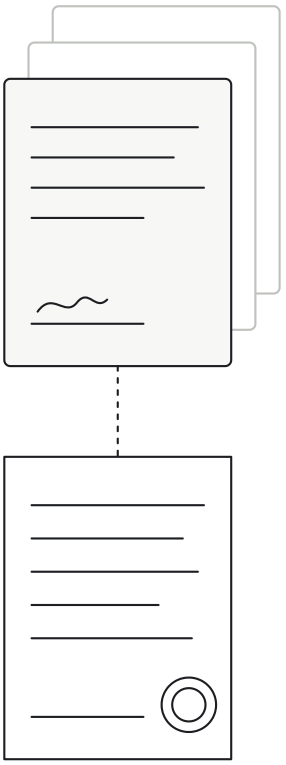
Changes are reviewed and reversible. Check where it lives and what leaves with you at exit.

159 to 180

Context run like production data

The context is owned, approved, traced and versioned on infrastructure you govern, so you can widen the rollout.





Next steps

Six steps that take you from a scorecard to a signed contract

The scorecard shows where a platform stands. These six steps turn the low scores into requirements that a vendor has to answer in writing, before anyone signs.

1 **Score what you already run.** Put the eighteen questions to your current stack before any vendor meeting. The total is your baseline, and your weakest section is where the next purchase has to help.

2 **Decide where shared context will live before you pick a tool.** Settle what is shared and what stays private before you roll anything out.[2] Write the decision down and give it an owner.

3 **Run the edited-memory test in every demo.** Ask to see one memory that someone changed, the record of the change and the name of the person who approved it.

4 **Write the controls into the contract you sign.** A named owner per item, approval before any agent reads, provenance on every answer, reviewed changes with a revert path, the physical location of context and its retention terms, and what you take with you at exit.

5 **Give agents identities and budgets before they call each other.** One identity per agent, authorization on agent-to-agent calls, spend limits and a kill switch your team holds.

6 **Run the pilot on a single handoff between two people.** Handoffs are where multiplayer AI is meant to pay off first,[2] so measure whether the next person starts where the last one stopped.

2 Ramp, Multiplayer AI: How Teams Collaborate on AI Work, 30 Oct 2026

Why Shakudo argues from your own compute

Every control in this playbook depends on where the shared context sits. An owner, an approval gate, a provenance record and a change history are only as strong as the system that enforces them. If the context lives inside a vendor's product, your controls are whatever that product offers, under its retention terms.

Shakudo is a sovereign AI platform. It runs inside your own VPC, on-prem or air-gapped, and deploys wherever Kubernetes runs. Agents, RAG pipelines and open-weights models run on compute you own, with data, models and context staying inside the perimeter.

That does not write your governance for you. It puts the context your multiplayer AI depends on next to the data your team already governs, on infrastructure where it already sets owners, approvals, lineage and change control. Your rules apply there, without waiting for a vendor's settings page to catch up.

Shakudo holds SOC 2 Type II. J.T. Magen uses Shakudo to run open-source AI on infrastructure the company controls.

Read more: shakudo.io/product/sovereign-ai · shakudo.io/platform

Shakudo deploys on the infrastructure you already use

AWS Azure GCP Oracle Private cloud On-prem Air-gapped

About Shakudo

Shakudo is a sovereign AI platform for running agents, RAG pipelines and open-weights models on compute you own, inside your VPC, on-prem or air-gapped, across AWS, Azure, GCP, Oracle or private cloud.

See Shakudo running on infrastructure you control

For more information: www.shakudo.io · shakudo.io/contact-us



For more information

www.shakudo.io

shakudo.io/contact-us