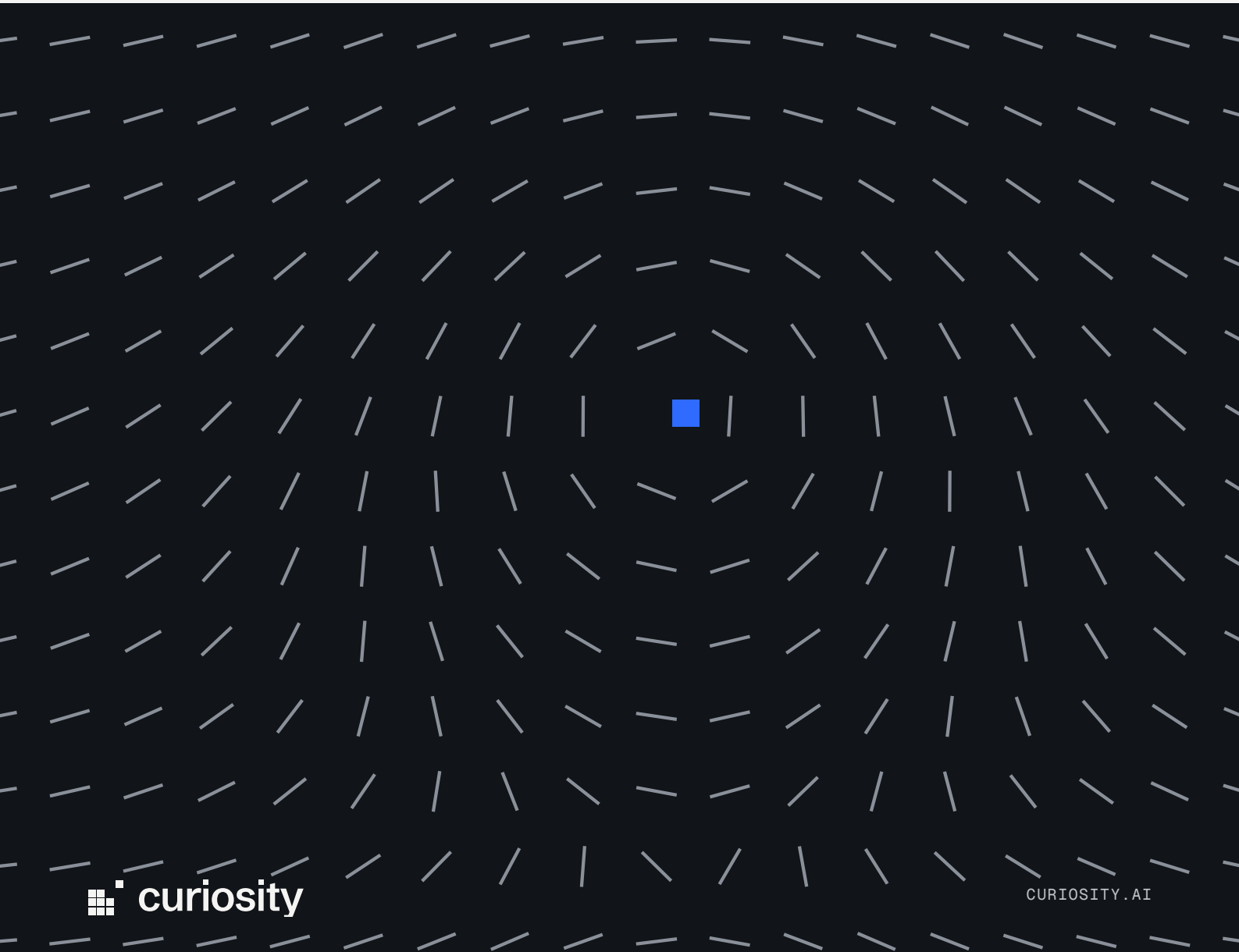


An AI admin you can approve.

Sudo, the admin assistant in Curiosity Studio, changes your configuration in a sandbox, apart from your live deployment. Nothing built there runs until a person approves the diff. Written for the people who decide.

TOPIC	FOR	DATE
AI assistants and change control	CIOs and security leads	October 2026



Sudo does the work. A person keeps the say.

01	The risk is the authority, not the model Three fears, and what each one is really about.	03
02	Dangerous things are absent, not blocked A sandbox with nothing in it to misuse.	06
03	Nothing runs until a person approves The loop, and an approval bound to one diff.	09
04	Sudo sees what you see, and no more Your permissions, never elevated ones.	12
05	Every change is on the record One tracked commit per approval, by name.	14
06	Questions to ask any AI admin Seven questions, and three steps to try it.	16
.	Glossary	20

IN ONE PARAGRAPH

An AI assistant with admin rights is a risk when it holds the host's authority, acts unreviewed or sees past its user. Sudo holds none of that. The work happens in a sandbox with nothing dangerous in it, and every read runs as the administrator in the chat. Every change waits as a diff until a person approves it, on the record.

01

The risk is the authority

The three fears behind every no to an AI admin

Why a better model answers none of them

The four mechanisms this paper takes in turn

1.1

The question is what the assistant can reach

Sudo does the work. A person keeps the say, whatever the model decides to try.

Ask a security lead whether an AI assistant may change a production system, and you hear three fears. It does something nobody reviewed. It reaches somewhere it should not. It sees data it should not.

Each fear is about authority, not intelligence. A smarter model answers none of them.

An assistant that shells out to a real shell has the host's authority. Whatever the service account can reach, the assistant can reach. A rule checked at run time is all that stands between them.

An assistant that interprets the script itself has only what its host handed it. Sudo is built the second way.

So do not ask how clever the assistant is. Ask what it can reach, what it can change without you, and what it leaves behind.

AUTHORITY

What an assistant can reach and change, whatever it intends.

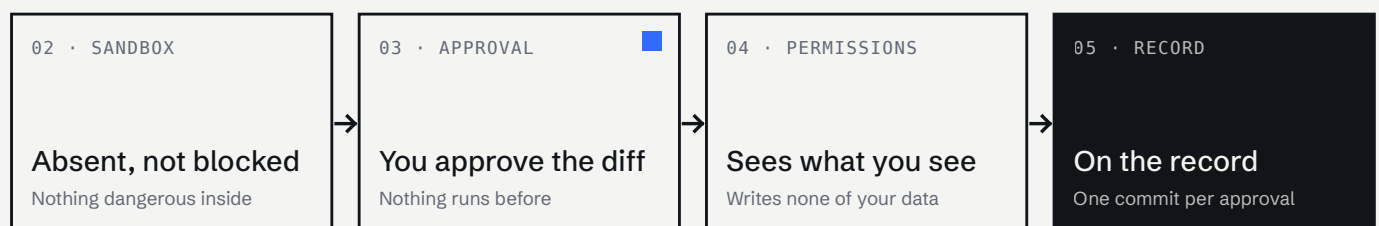


FIG. 1.1 · FOUR MECHANISMS ANSWER THE THREE FEARS. THIS PAPER TAKES THEM IN TURN.

TABLE 1.1

Each fear meets a mechanism

WHAT COULD GO WRONG	WHAT SUDO'S DESIGN DOES	SECTION
A change runs unreviewed	Nothing applies without an approved diff	03
It reaches the host	No host files, no processes	02
It reaches the network	No route out of the sandbox	02
It sees data it should not	Reads with your permissions only	04
It changes your records	Reads your data, writes none	04
It reads a secret	Never sees a secret's value	04
It acts without a trace	One tracked commit per approval	05

READ IT THIS WAY

The left column is the list a security lead brings to the meeting. The middle column is not a policy Sudo follows. It is how the sandbox and the approval are built, so it holds whatever the model decides to try.



02

Absent, not blocked

A shell that never leaves Studio's own process

Two places Sudo may write, and neither is your deployment

One sandbox per chat, with limits that stop cleanly

2.1

Nothing dangerous is there to block

Sudo works the way a developer works: reading files, running commands and handing you a diff. The difference is where those commands run.

Sudo's shell never leaves Studio's own process. It is Computerwelt, an open source .NET library: a bash and a Python written in managed C#, over one virtual filesystem. There is no bash binary and no Python runtime underneath.

So the dangerous parts are absent, not refused at run time. No path resolves to a file on the server. Nothing is forked or executed, and no container is started. Network commands are withheld rather than present and refusing. If the library ever grows one, a session does not silently acquire it.

Writes go to two places only: Sudo's own copy of your configuration, and scratch space discarded with the session. Everything else is read only.

Code you approve can still call out once it runs. That is the code's authority, not the assistant's, and it exists only after you approved it.

SANDBOX

A space where the assistant works with only what its host hands it: no host files, no processes, no network.

WITHHELD

Left out of the sandbox entirely, rather than present and denied by a rule.

KEY POINT

What is missing is missing by construction, not refused at run time.

2.2

Each chat gets a sandbox of its own

A session belongs to one conversation. Two administrators in two chats never share a working directory, an unfinished edit or a plan.

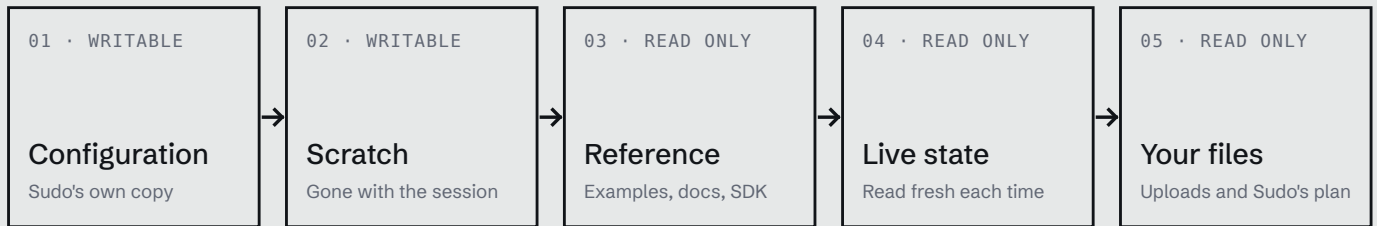


FIG. 2.1 · WHAT A SESSION CAN REACH: NINE MOUNTS IN FIVE GROUPS. TWO ARE WRITABLE, AND NEITHER IS YOUR DEPLOYMENT.

Only what the session changed persists: the files it wrote, the paths it deleted, its plan and its history. Everything else is read again from your live deployment on every command. If someone edits an endpoint in the admin pages meanwhile, Sudo sees that edit next time.

Every script runs inside fixed limits. A limit that is reached raises an error instead of returning part of an answer. A search that stopped halfway is never reported as the whole.

Resuming a session finds Sudo's files as they were left, in a deployment that has moved on. It is not a snapshot.

KEY POINT

Only a write to Sudo's copy of your configuration can become a change, and only once you approve it.

LIMITS

120 s per script, 100,000 commands, 4 MB of output.
An idle session ends after 30 min.

03

Nothing runs until you approve

The loop from a request to an applied change

An approval bound to the exact diff you read

Why silence is never taken as a yes

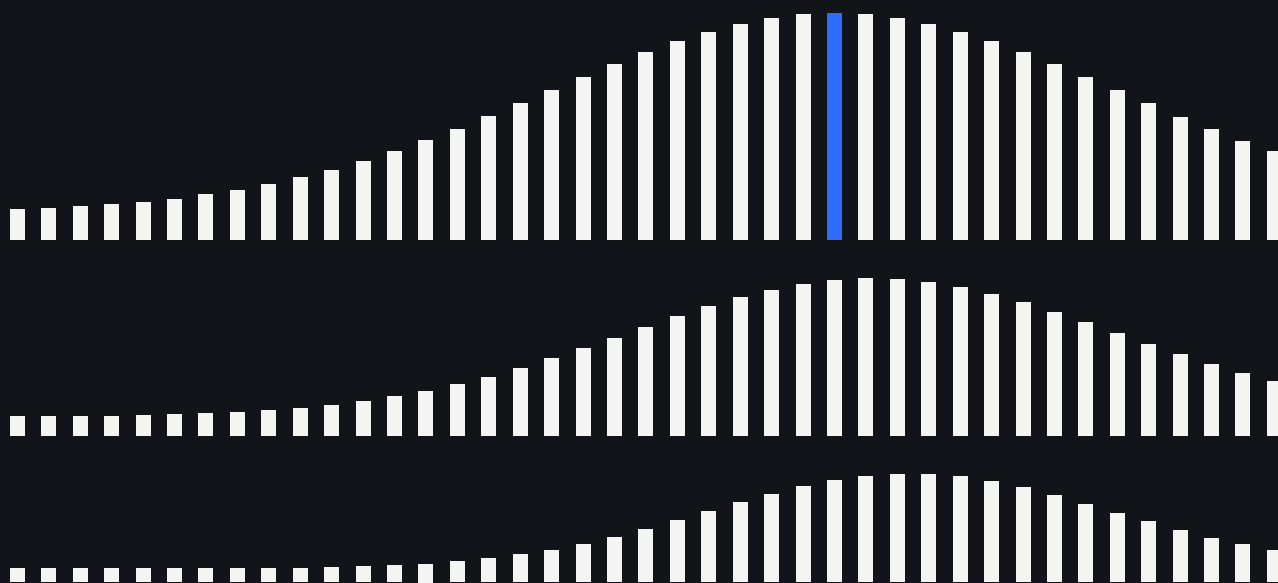
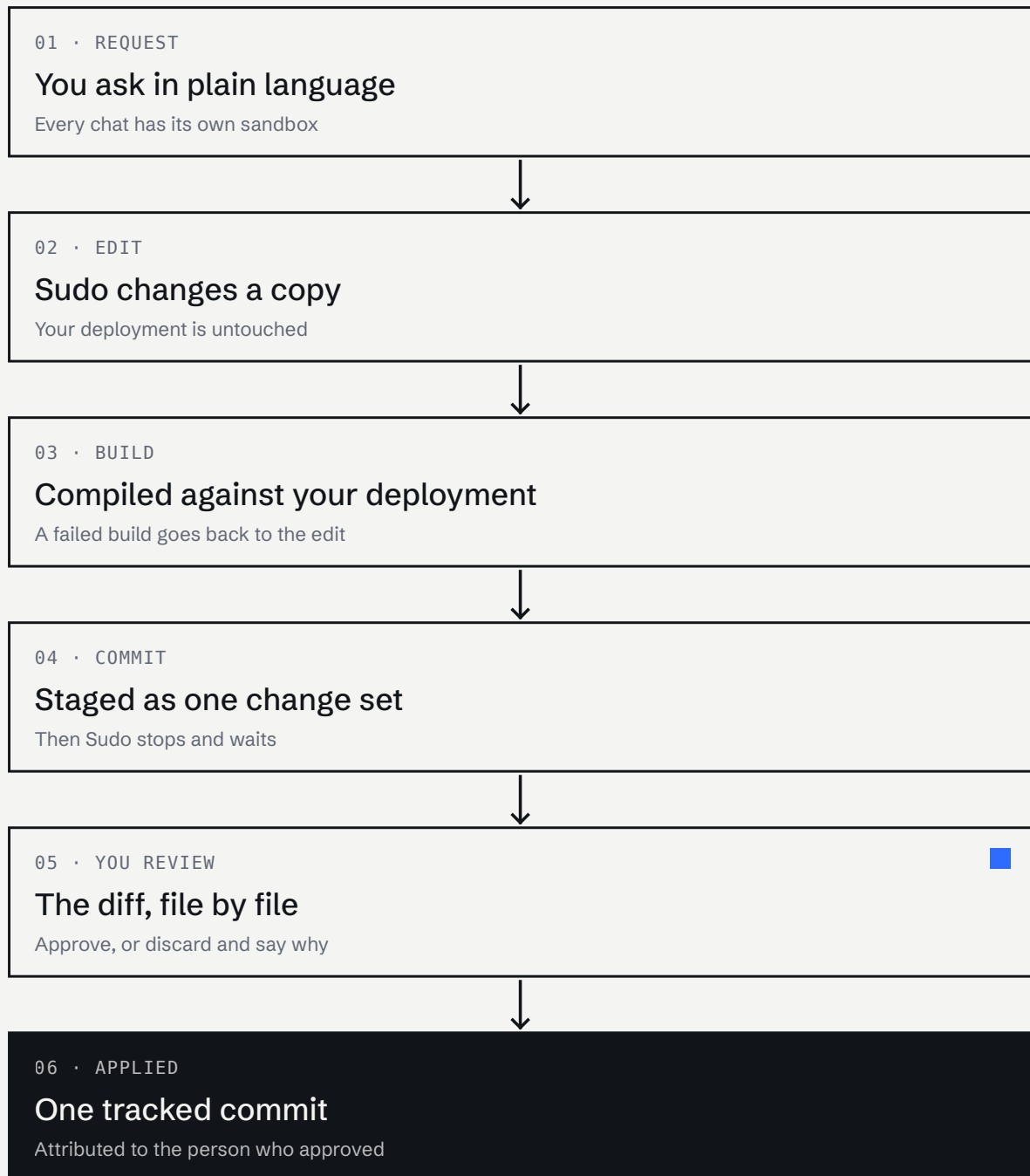


FIGURE 3.1

Ask, review, apply

Everything before your review happens in Sudo's own copy. Your deployment is untouched until you approve.



03
Every changed file is compiled first.
One failure refuses the whole set.

05
The marked step. A person reads
the diff and decides.

06
Your configuration history records
it under the approver's name.

3.2

Approval is bound to the diff you read

An approval is not a confirmation box. It is bound to one staged commit, the one on your screen. If Sudo stages a newer commit while your review is open, that review refuses to apply. Approving would apply something nobody read.

You can change the draft on the review screen. Saving writes your edit into Sudo's copy too, so that copy and what you apply never disagree. Or leave a comment and request changes.

There is one way in. An approved commit is an ordinary import of the files that differ, through the same code that imports an uploaded configuration export. No second set of writers can drift from it.

A failed build stops the commit. A forced commit exists, says so loudly, and still needs your approval.

Approving says so in the conversation, in your voice. Discarding keeps Sudo's edits and says nothing there, so silence is never read as approval.

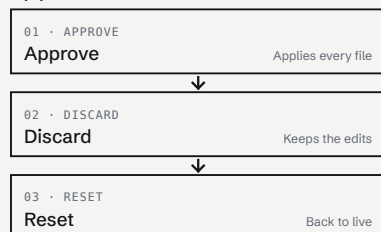


FIG. 3.2 · THREE CONTROLS. ONLY THE FIRST REACHES YOUR DEPLOYMENT.

Three controls decide what happens to Sudo's work. Only Approve changes your deployment. Discard and Reset change nothing but the assistant's own copy.

Closing the tab during an approval costs you the progress log, not the change.

DIFF

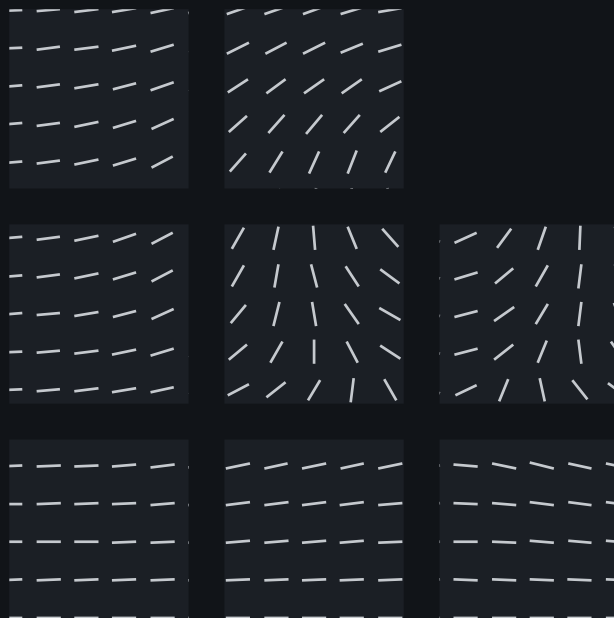
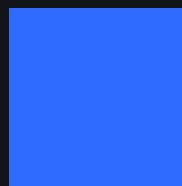
The exact lines a change adds, removes or alters, file by file.

IF AN IMPORT FAILS

The errors are reported and the edits are kept. Nothing is lost.

KEY POINT

What you approve is the diff you were shown.



04

Sudo sees what you see

Reads under your permissions, never elevated ones

What stays out of reach, even for a reader

Why your own account narrows Sudo further

4.1

Despite the name, no elevated privileges

Sudo reads your deployment as the administrator in the chat, and writes none of your data.

Sudo never assumes elevated privileges. Every read runs under the permissions of the administrator in the chat. Record types that are never returned to clients are skipped.

Your data is read as you can read it, and none of it is written. A migration that fills a new field runs only after you approve it.

Some things stay out of reach even for a reader. Sudo can declare a secret and write code that reads it, but never see its value.

Access groups can be read, but membership is yours to manage.

The assistant installs no packages, runs no backups, restarts nothing and touches no licenses.

Your own account narrows Sudo further. For an administrator who may only read, nothing can be staged or started. The same holds for every administrator while write protection is on.

Reading still works, and the draft is kept for someone who may approve it. The shell stays, so Sudo can still read.

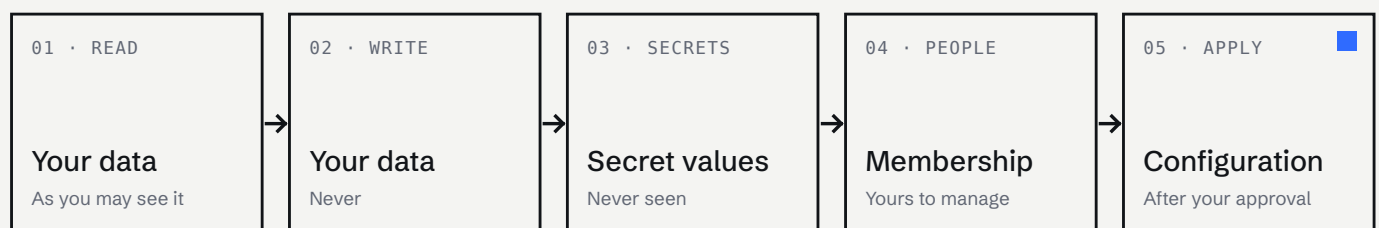


FIG. 4.1 · WHAT A SESSION ALLOWS SUDO. ONLY THE LAST STEP CHANGES ANYTHING, AND ONLY THROUGH YOU.

05

Every change is on the record

One tracked commit per approval, by name

The same history as a change made by hand

Sudo's manner written down where you can read it

5.1

One trail, the same as a manual change

Every change Sudo applies lands in the same tracked history as one made by hand. You need no second audit process for the assistant.

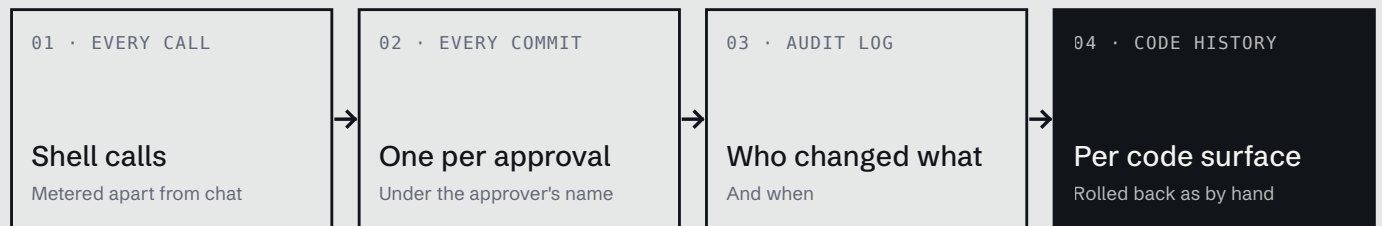


FIG. 5.1 · WHERE SUDO'S WORK IS RECORDED, FROM THE FIRST COMMAND TO THE APPLIED CHANGE.

Every shell call is metered into your deployment's own metrics, apart from ordinary chat tools. Every applied commit is one entry in the configuration history, attributed to the administrator who approved it. Admins read it in the audit log.

Every code surface keeps its own history too. An endpoint, task or index changed through Sudo is reviewed and rolled back like one changed by hand.

Sudo's voice and boundaries are a skill in your deployment that you can read and edit. The skill says to check intent, target and impact before acting, and to ask first on anything destructive or security sensitive.

The guarantees do not rest on that skill. They rest on the sandbox and the approval.

KEY POINT

Everything Sudo does is something you can do by hand, in the same history.

SKILL

Written instructions in your deployment that shape how Sudo works. Visible and editable, not hard wired.

06

Questions to ask any AI admin

Seven questions to ask any vendor, us included

The answers to worry about

Three steps to try it on a machine you control

TABLE 6.1

Seven questions for any AI admin assistant

ASK THE VENDOR	THE ANSWER TO WORRY ABOUT	SUDO'S ANSWER
What can it reach on the host?	What the service account can	Nothing outside its sandbox
Can it reach the network?	Blocked by a rule	No route out
What runs without review?	Small changes, automatically	Nothing at all
What exactly do we approve?	A summary of the change	The diff, bound to its commit
Whose permissions does it use?	Its own service account	The admin in the chat
Can it read our secrets?	If it has access	Never a secret's value
Where is the record?	In the vendor's logs	Your audit log and history

READ IT THIS WAY

Ask these before the pilot, not after the first incident. An answer from the middle column means the safety rests on the model behaving. Ask us too.

6.2

Ask to see it, not to hear it

Every answer in the right column can be shown, not only stated. Ask Sudo to stage a change, then discard it and check that your deployment did not move.

Then sign in as an administrator who may only read, and ask for the same change. Sudo drafts it but cannot stage it.

WRITE PROTECTION

A deployment setting. While it is on, Sudo can stage nothing and start nothing, for any administrator.

Try it in three steps

01

Install the free developer version of Studio on a machine you control.

02

Ask Sudo for one small change, then read the staged diff.

03

Discard it once, approve the next one, then find it in the audit log.

BEFORE YOU START

- ☐ Which administrators should be able to approve a change?
- ☐ Which change would you hand Sudo first, and which never?
- ☐ Should write protection be on for some administrators?
- ☐ Who reads the audit log, and how often?

What to take away

01

Sudo does the work. A person keeps the say.

No change runs until someone approves the diff.

02

Judge the authority, not the model.

Ask what an assistant can reach, change and leave behind.

03

Look for the record.

One tracked commit per approval, under the approver's name.

Terms used in this paper

APPROVAL

A person's decision to apply one staged commit, bound to the exact diff they were shown.

COMMIT

A staged set of changed files, applied as one tracked change once approved.

PERMISSION-AWARE

Reading only what the person asking may see in the source system, checked at query time.

SECRET

A stored credential that code may use and the assistant may never read.

AUDIT LOG

The record of who changed what in your deployment, and when.

DIFF

The exact lines a change adds, removes or alters, file by file.

SANDBOX

A space where the assistant works with only what its host hands it: no host files, no processes, no network.

SESSION

One chat with Sudo and its own sandbox, never shared with another chat.

Ask Sudo. Approve what gets built.

Book a demo and watch Sudo stage a change on data shaped like yours, then approve or discard it yourself. Or install the free developer version and try it on a machine you control.

curiosity.ai/request-demo

DATA RESIDENCY

EU, on premises or private cloud

COMPLIANCE

GDPR

MEMBER

KI Bundesverband