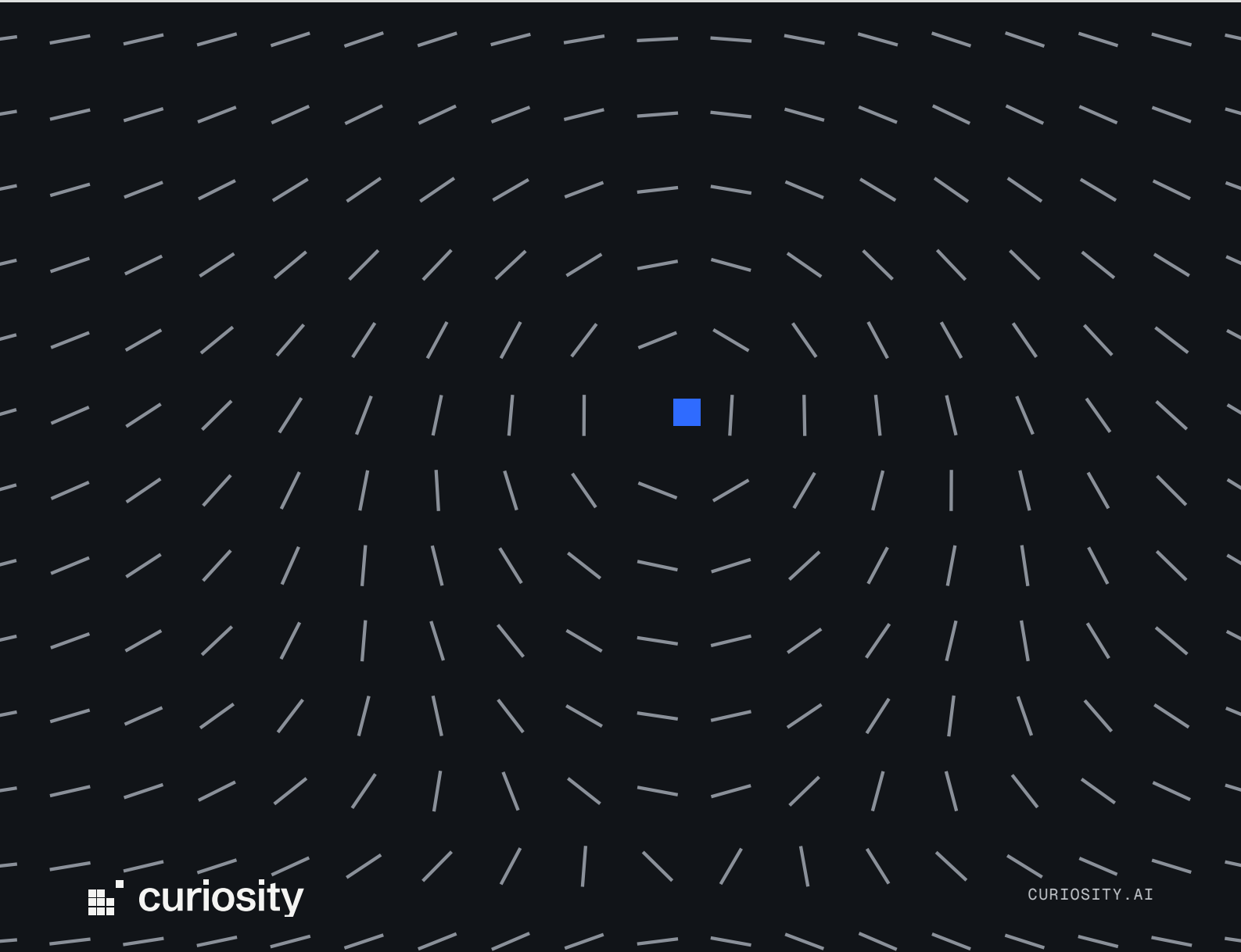


Working with Sudo: say the outcome, read the diff.

Sudo is the admin assistant in Curiosity Studio. This guide is for the people who hand Sudo real changes. It covers what to ask, what happens to a request, how to review the diff and how to teach the assistant.

TOPIC	FOR	DATE
The admin assistant	Studio administrators and developers	October 2026



Ask for the outcome. Approve the diff.

01	Sudo works beside your workspace The admin dock, full screen, and what Sudo sees of your screen.	03
02	Questions change nothing What Sudo reads to answer, and why reading is safe.	06
03	Every change ends in a diff Say the outcome; Sudo picks the surfaces and builds one commit.	09
04	Reviewing is reading the diff What to read, which button to press, what approving does.	12
05	Running things is a request too Tasks, migrations, reindexes and small settings, watched as they run.	15
06	You teach Sudo with a skill Sudo's rules, your conventions, and the habits of a good session.	17
.	Glossary	21

IN ONE PARAGRAPH

Sudo changes your configuration the way a developer would: as files, in a separate sandbox, ending in a diff. Questions cost nothing, so ask freely. For a change, say the outcome you want and let Sudo choose the surfaces. Then read the staged diff and approve one idea at a time. Nothing runs until you do.

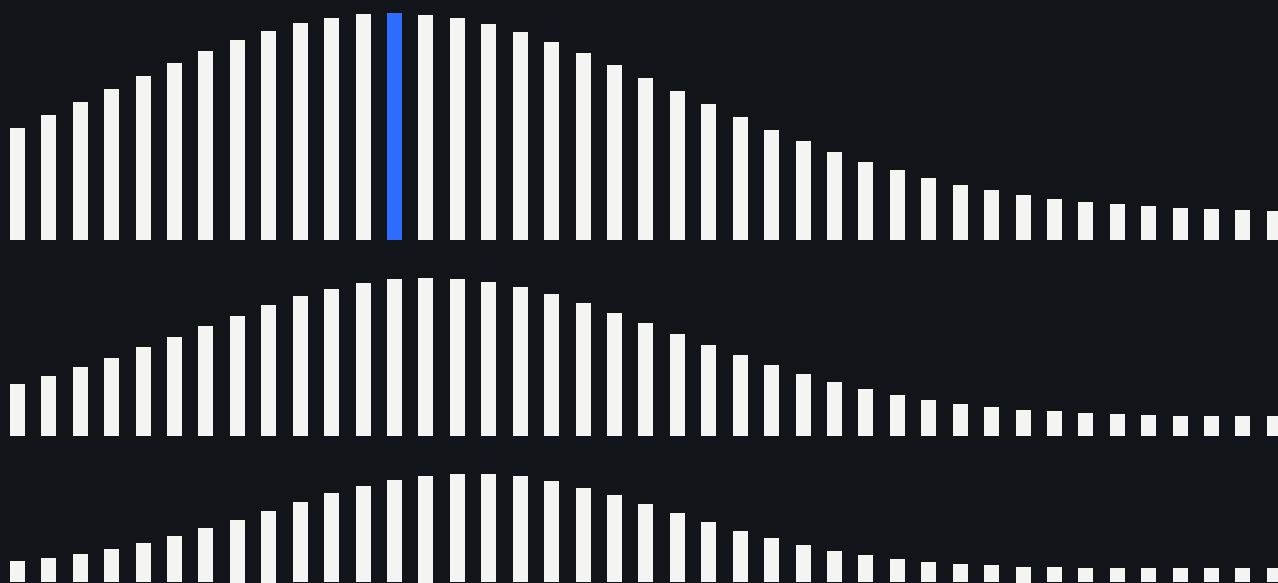
01

Sudo works beside your workspace

The admin dock, and full screen when a diff needs room

What Sudo knows about the screen you are on

The shared terminal, the pending changes and the attach button



1.1

One conversation, in the dock or full screen

The admin chat pane is collapsed by default. It opens from a floating toggle that carries Sudo's avatar. Its expand button opens the same conversation full screen, with room to read a diff.

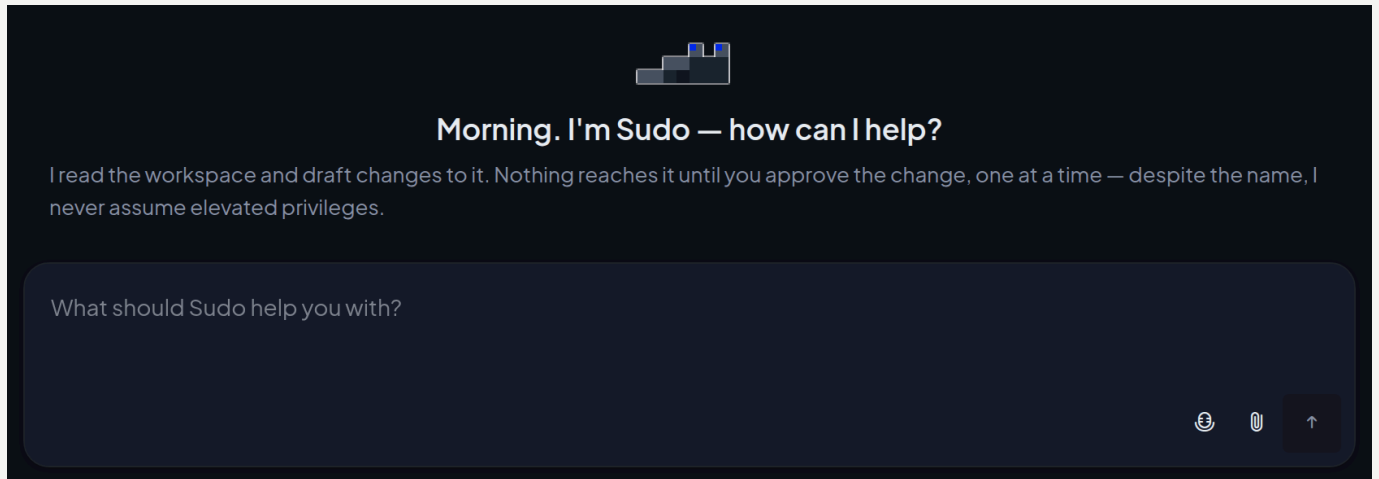


FIG. 1.1 · SUDO FULL SCREEN, BEFORE THE FIRST MESSAGE. THE PAPERCLIP IN THE COMPOSER IS THE ATTACH BUTTON.

1.2

Every chat is a session of its own

A session belongs to the conversation it started in. Two administrators in two chats never share a working directory, an uncommitted edit or a to-do list.

The open session is kept in the address. A refresh, a bookmark or a link sent to a colleague comes back to that session.

NOT A SNAPSHOT

A session keeps only what it changed. Everything else is read from the live workspace on every command, so an endpoint someone edited meanwhile is the one Sudo sees next.

1.3

Sudo sees the tab you are looking at

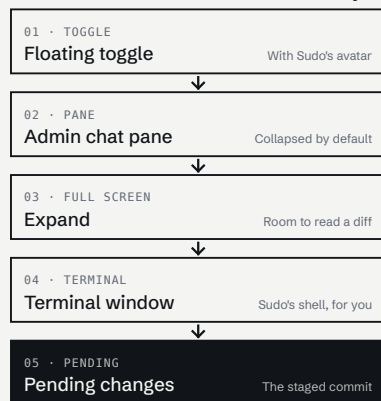
The admin interface hands Sudo a UI context block. The surface in your foreground tab is the active context: it is what "this endpoint" means when you say it. Tabs beside it are background context, used to tell similar names apart.

The pending changes panel holds the staged commit, if there is one, and the button that opens it for review.

The chat toolbar opens a terminal window on the same shell Sudo is working in. Use it when you would rather look yourself. Tab completion offers the uncommitted files.

When you open the editor of a built-in AI tool, agent, skill or prompt template, Sudo perches on it. That is how you tell a built-in from something your workspace authored.

The attach button uploads a file into the conversation, and Sudo reads it at `/home/uploads/<name>`. A PDF or a spreadsheet reads as extracted text, and a `.zip` as a directory to walk.



Five places to know before your first request. The last one is where every change waits for you.

FIG. 1.2 · WHERE SUDO LIVES IN THE ADMIN INTERFACE.

SUDO'S EARS

The state shows only in two blue half cells on the ears.

Both for Ready, one for Working, an ear and the tail for Found, none for Idle.

ACTIVE CONTEXT

The surface in your foreground tab. It is what "this" refers to in your request.

KEY POINT

Open the endpoint, then say "this endpoint". Sudo knows which one you mean.

02

Questions change nothing

Six questions worth asking on your first day

What Sudo reads to answer each one

Why reading leaves nothing behind

TABLE 2.1

What Sudo reads for each question

YOU ASK	SUDO READS	CHANGES ANYTHING
Which endpoints are public?	grep over code/endpoints/	☐
What can the triage agent call?	One file in code/agents/	☐
Which node types have no index?	/proc/nodetypes, /proc/indexes	☐
How many Ticket nodes are there?	graph stats, graph sample	☐
Are scheduled tasks failing?	/proc/scheduled-tasks, /proc/exceptions	☐
What broke after the deploy?	/proc/exceptions, then /proc/audit	☐

☐ NO

READ IT THIS WAY

Every row is a read. Despite the name, Sudo never assumes elevated privileges. graph and query run under your own permissions and skip node types that are never returned to clients.

2.2

Sudo reads the live system, every time

/proc is regenerated on every read. The same cat a minute later shows how far a running task has got, and jq works on it directly.

So ask about the workspace as it is, not as you remember it.

NOTHING TO APPROVE

A session that only ever read leaves nothing behind. Ask as many questions as you like.

2.3

Nine mounts, and only two take writes

Sudo's sandbox is one filesystem. Your configuration sits in `/workspace`, as the definitions export writes it. Only a write there is a change to your workspace.

```

/
├─ workspace/      your configuration      [writable]
├─ reference/      an example of every file [read-only]
├─ docs/           one page per code scope [read-only]
├─ sdk/            the C# surface         [read-only]
├─ proc/           live runtime state    [read-only]
├─ session/        Sudo's to-do list     [read-only]
├─ stage/          your Stage uploads    [read-only]
├─ home/uploads/   files attached to the chat [read-only]
└─ tmp/            scratch, gone with session [writable]

```

LISTING 2.1 · THE SANDBOX'S MOUNTS, AS THE DOCUMENTATION LISTS THEM, COMMENTS SHORTENED.

LIMIT	VALUE
Wall clock per script	120 s
Commands per script	100,000
Output captured	4 MB
Idle session lifetime	30 min

2.4

Absent, not blocked

A cap that is reached raises an error rather than returning part of an answer. There is no host filesystem, no process spawning and no network: `curl`, `ssh` and `git` are withheld, not refused.

APPROVED CODE IS DIFFERENT

Code in your workspace can call out once you approved it and it runs. That is the code's authority, not Sudo's.

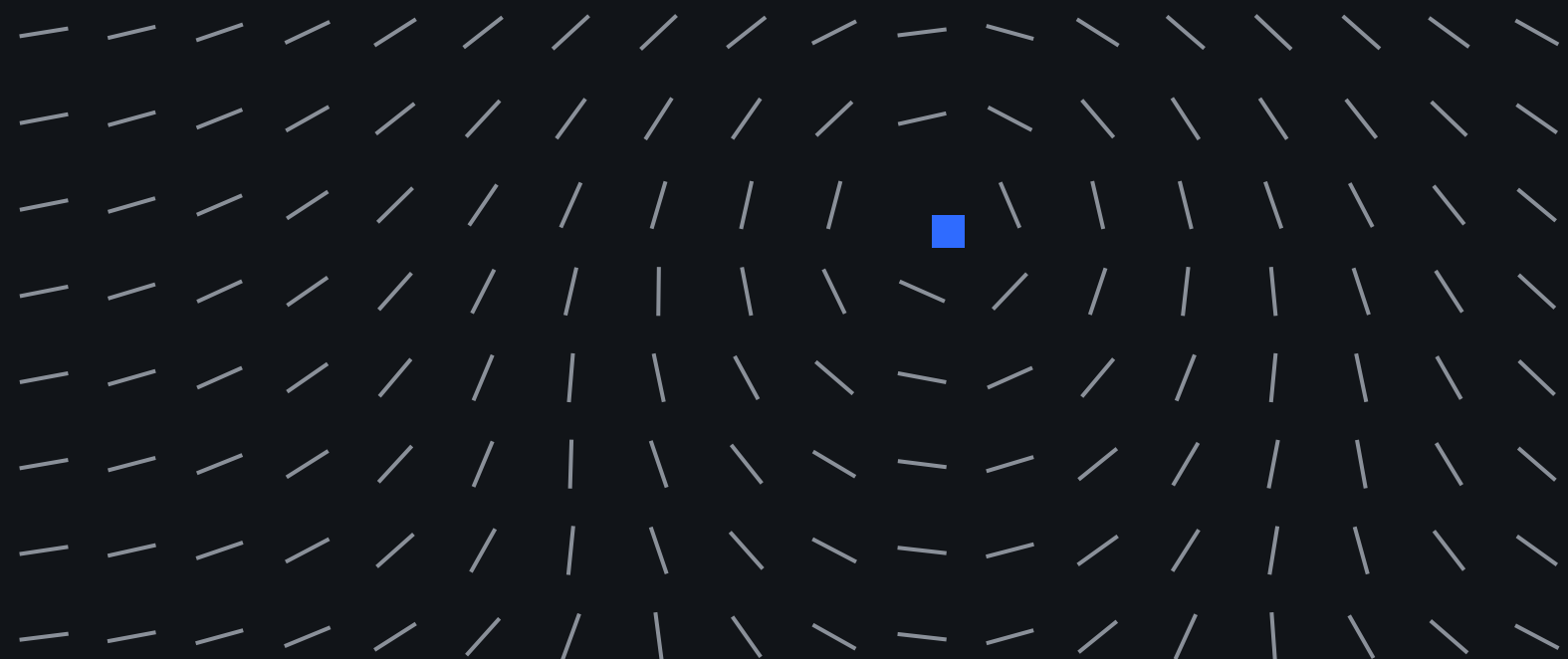
03

Every change ends in a diff

Say the outcome, and let Sudo choose the surfaces

Read, edit, build, commit, then stop

One request across several surfaces, in one commit



3.1

Say the outcome, not the surface

"Users should be able to filter tickets by product" beats "add a facet". The answer might be a facet, an index setting or a schema field. Sudo can look before deciding.

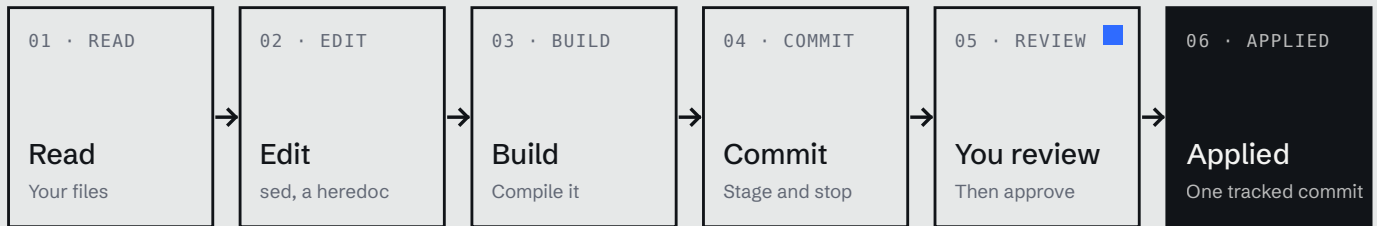


FIG. 3.1 · THE LOOP. EVERYTHING BEFORE YOUR REVIEW HAPPENS IN SUDO'S OWN COPY.

Ask for an endpoint that returns open tickets grouped by product. Sudo reads `/reference` for the file shape and `/docs` for the scope. Then the assistant mints a UID with `uid` endpoint, builds until it compiles, and commits.

A chat AI tool that looks up a customer's open cases is one file under `code/chat-ai-tools/`. It retrieves as the user, not as the system.

A triage agent is one file under `code/agents/`, prompt and tools together. Hand Sudo a CSV, and a schema and a connector come back in one commit, because they only make sense together.

BUILD BEFORE COMMIT

`commit` compiles every changed file against the scope it will run in. If anything fails, it refuses the whole set.

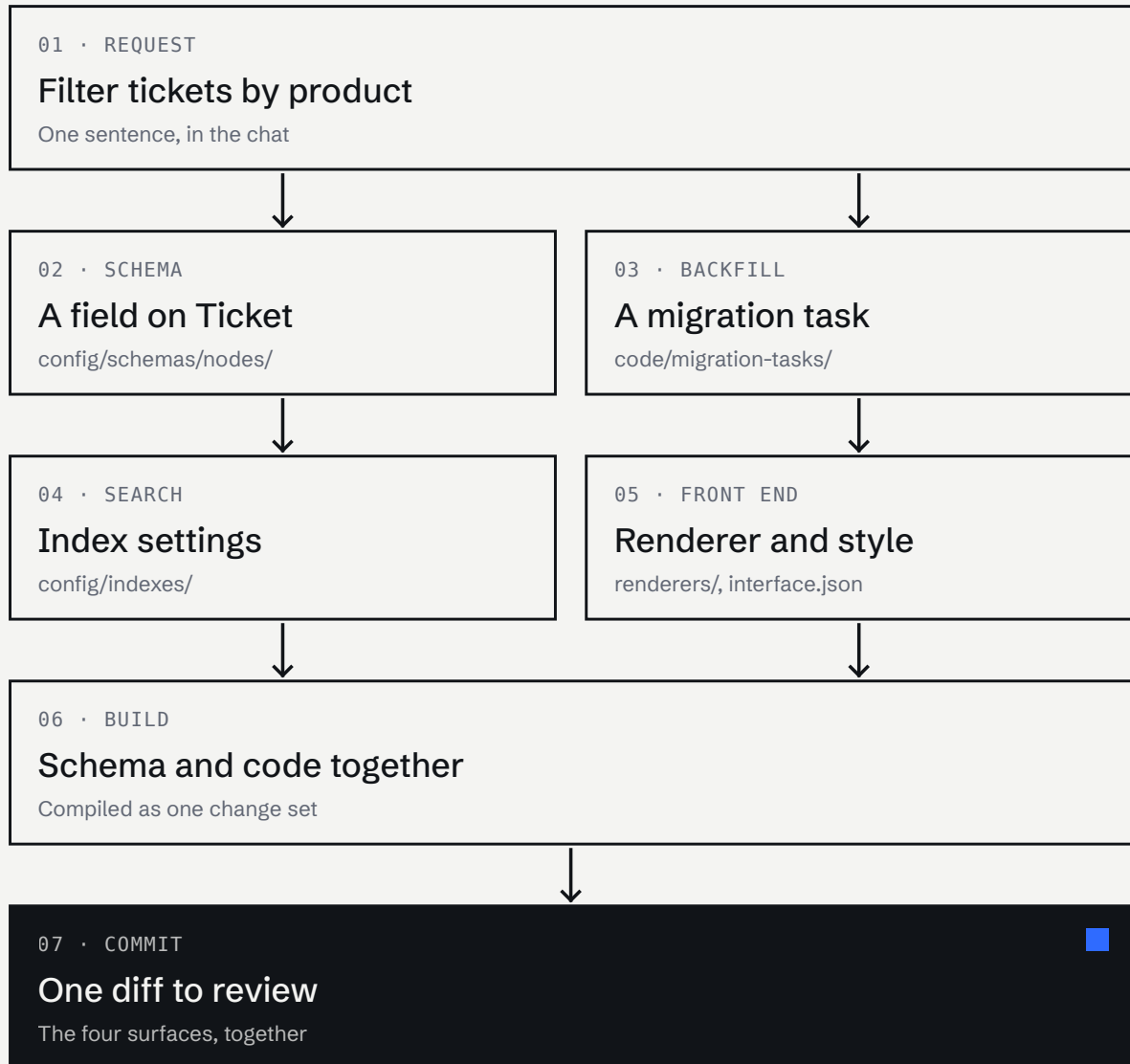
COMMIT STOPS

`commit` stages the change set and stops. Nothing reaches the workspace until a person approves it.

FIGURE 3.2

One ask, one commit

"Users should be able to filter tickets by product, and see the product on the ticket card." One sentence, and Sudo finds every surface it touches.



02

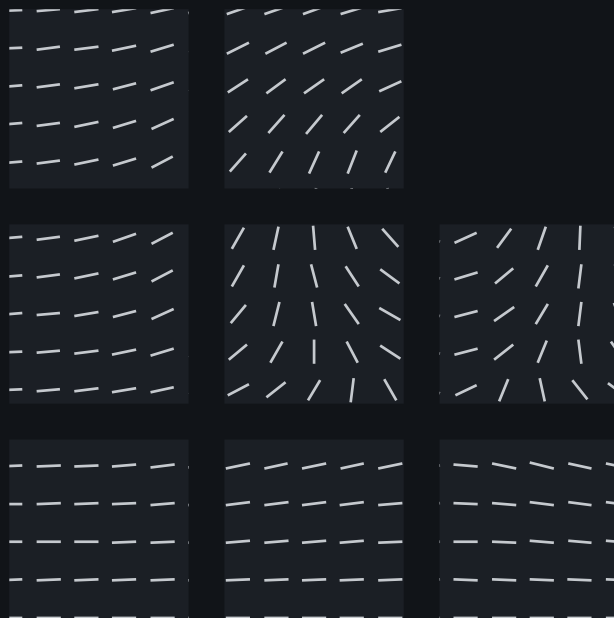
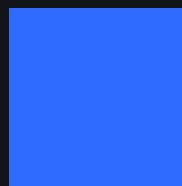
The field is a property on a C# class. Its access control sits as attributes on the type and on each property.

06

build compiles the schema and the code that uses it together, so both go in one commit.

07

The marked step. After you approve, run `migration` starts the backfill. Until then, nothing has run.



04

Reviewing is reading the diff

What the review screen shows you

Approve, Discard and Reset, and the card that refuses

What approving does, in order

4.1

The review screen shows the change itself

The pending changes panel opens Sudo's staged commit. Nothing has reached the workspace yet. What you approve is the diff you were shown.

```
code/endpoints/open-tickets-by-product.cs · code endpoint
After this change
1+ [endpoint: Curiosity.Endpoints.Path("open-tickets-by-product")]
2+ [endpoint: Curiosity.Endpoints.CacheFor(120)]
3+ [endpoint: Curiosity.Endpoints.EnableMonitoring]
4+ [endpoint: Curiosity.Endpoints.ReadOnly]
5+ [endpoint: Curiosity.Endpoints.AccessMode("AllUsers")]
6+
7+ //Runs as the calling user, so a product whose
8+ //tickets they cannot see reports zero.
9+ var rows = new List<object>();
10+
11+ foreach (var product in Q().StartAt("Product").AsEnumerable())
12+ {
13+     var open = Q().StartAt(product.UID)
14+         .Out("Ticket", "_HasTicket")
15+         .Count();
16+
17+     if (open == 0) continue;
18+
19+     rows.Add(new
20+     {
21+         product = product.UID.ToString(),
22+         open     = open,
23+     });
24+ }
25+
26+ return rows;
27-
```

FIG. 4.1 · A COMMIT ADDING ONE CODE ENDPOINT. READ EVERY LINE HERE BEFORE YOU APPROVE.

FILE LIST

The changed files, with a side by side diff of the one you select. Best for checking one file closely.

PULL REQUEST

Every file's diff down one page, with a comment box under each. Best for a change across several files.

EDIT OR ASK

Edit the right-hand side and save it into Sudo's copy, or comment and request changes. Either way, the shell and what you approve stay one file.

TABLE 4.1

Three buttons, and a card that refuses

BUTTON	WHAT IT DOES	SUDO'S EDITS
Approve	Applies every file, as one commit	Applied, attributed to you
Discard	Throws the commit away	Kept, for another try
Reset	Returns Sudo's shell to live	Thrown away
A stale card	Refuses to apply	Unchanged; open the newer one

READ IT THIS WAY

A card left open while Sudo staged a newer commit refuses, because approving it would apply something nobody read. Discard says nothing in the chat, so silence is not approval.

4.2 Approving applies deletions first

A change set that removes an endpoint and adds one on the same route removes first. Then each file goes through the definitions importer, with its warnings reported as they happen.

On success it is one tracked commit, attributed to you. Sudo's session reloads, so the next command reads what was applied.

IN YOUR VOICE

Approving says so in the chat, in your voice, with any notes you typed while reviewing. If part of it fails, the errors are reported and the edits are kept.

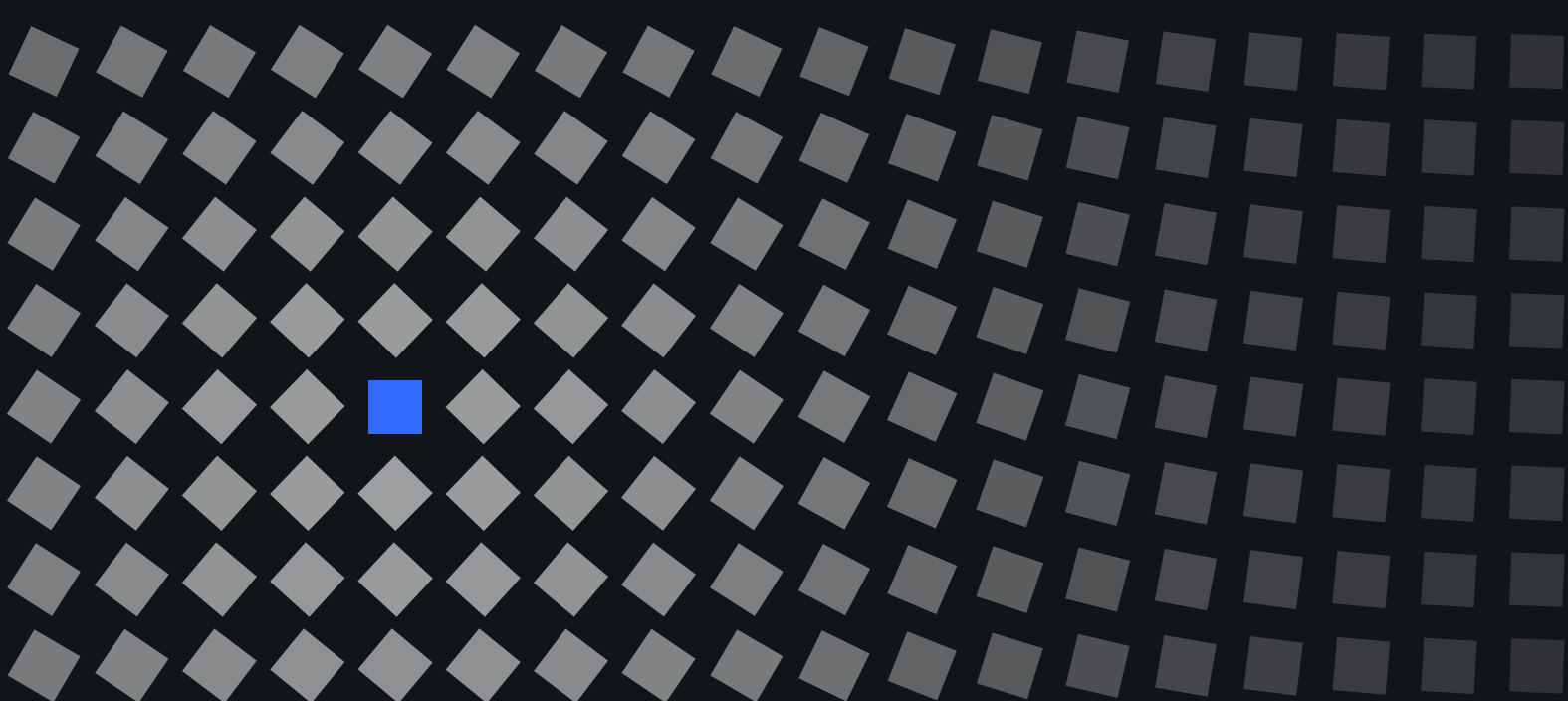
05

Running things is a request too

Tasks, migrations and reindexes, started from the chat

Watching a run until it finishes

Small settings and deletions, each a small diff



5.1

Ask Sudo to run it, then to watch it

Starting a task, a migration or a reindex is a command in Sudo's shell. Each run gets a folder under `/proc/runs/<id>/` with its status, log and output, read as the run progresses.

```
run task "Nightly refresh"
cat /proc/runs/<id>/status
cat /proc/runs/<id>/log
run reindex Ticket
run migration
```

LISTING 5.1 · THE RUN COMMANDS AS THE DOCUMENTATION GIVES THEM. <ID> IS THE RUN SUDO STARTED.

YOU ASK	SUDO DOES
Run the nightly refresh and tell me if it worked.	<code>run task</code> , then reads status and log
Reindex Ticket and tell me when it is done.	<code>run reindex Ticket</code> , then polls the run folder
Turn on the copy disclaimer, keep chats 30 days.	Two values in one config/ <code>*.json</code> file
Delete the old ingest-from-ftp task.	Commits the deletion; a built-in is refused

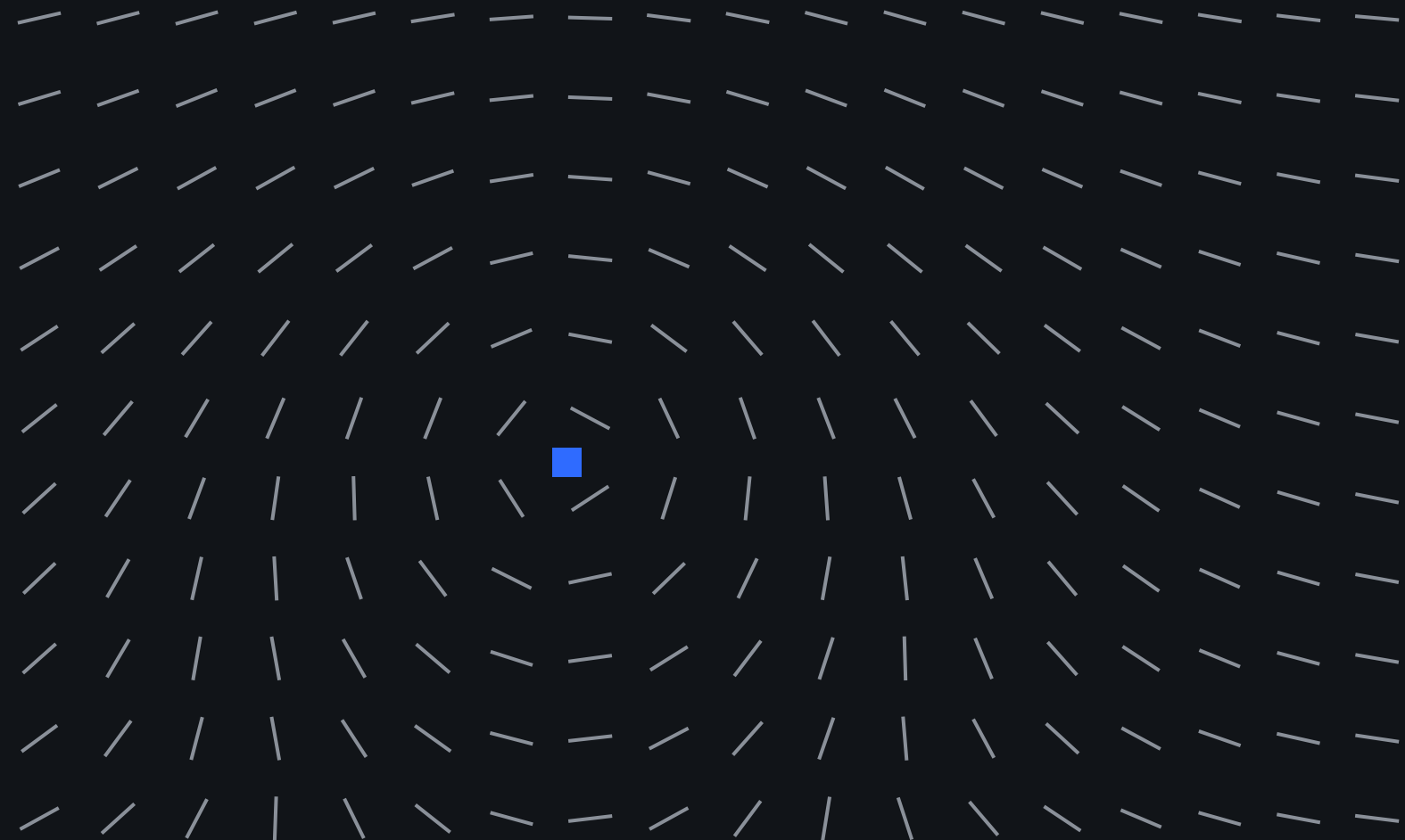
5.2

Every run and every change leaves a trail

Every shell call is metered into the workspace's own metrics, under an Admin Agent bucket. Every applied commit is one entry in the configuration history, attributed to whoever approved it.

READ IT BACK

Under Manage / Access / Audit log, or from Sudo's sandbox at `/proc/audit`. Code surfaces keep their own history, so a change made through the assistant rolls back like one made by hand.



06

You teach Sudo with a skill

The rules Sudo follows, written where you can read them

Your conventions, so you stop repeating them

Habits that make a session go well, and Sudo's limits

6.1

Sudo's rules are a skill you can edit

The assistant's voice and boundaries are a skill in your workspace, sudo-persona. It is visible and editable, not hard-wired. Read it before your first real change.

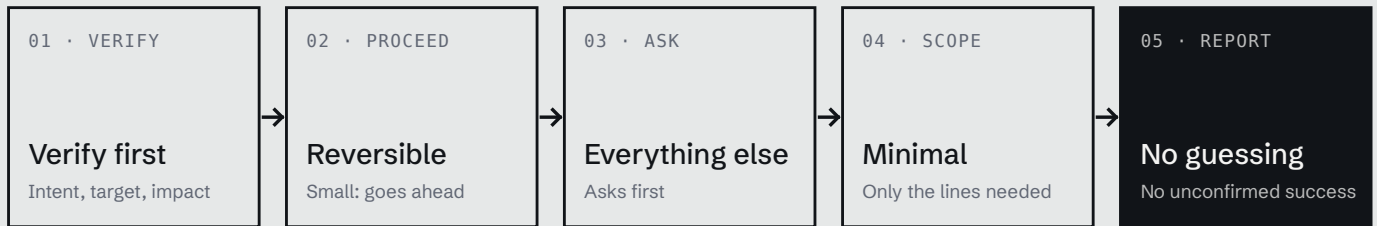


FIG. 6.1 · HOW SUDO-PERSONA SHAPES THE ASSISTANT'S BEHAVIOR.

Asking first covers anything destructive, security sensitive, externally visible or materially ambiguous. Small, reversible work goes ahead, and a change still ends in a diff you approve.

Anything you would repeat at the start of every conversation belongs in a skill too. Think of how your team names endpoints, where agents live, who signs off on a schema. Write it down once, and every session starts with it.

Skills are part of the configuration Sudo reads, beside endpoints and agents. Ask for a draft, and review it like any other change.

KEY POINT

If you say it at the start of every session, it belongs in a skill.

SKILL

Written instructions in your workspace that shape how an assistant works. sudo-persona holds Sudo's.

6.2

Five habits for a good session

Let Sudo plan first on anything with several steps. The to-do list is rendered to `/session/todo.md`, so you can see what is done and what is left.

Hand over the file rather than describing it. Attach the spreadsheet, the PDF or the zip. A CSV becomes columns and null counts before it becomes a schema.

Ask for the diff early. "Show me what that would change" catches a misunderstanding while it is still one file. A wrong guess caught there costs a sentence, not a review.

Correct Sudo in the chat for anything more than a small fix. The assistant learns what you meant, not just the line. Discard keeps the edits, so say what to change and ask for a new commit.

One commit per idea. A diff that does one thing is one you can read in a sitting. The exception is work that only makes sense together, like a schema and the migration that fills it.

YOUR ACCOUNT SETS THE LIMIT

For a read-only administrator, `commit` and `run` refuse, as for anyone while administrator write protection is on. Reading and drafting still work.

WHAT SUDO WILL NOT DO

Write without your approval. Change your data. Reach the network. Manage people or read a secret value. Install packages, take backups, restart the workspace or touch licensing.

KEY POINT

One commit per idea.
A small diff is a diff you will actually read.

Three things to keep

01

Say the outcome.

Name what users should be able to do. Sudo picks the surfaces.

02

Read the diff.

Nothing reaches the workspace except a commit you approved.

03

Approve one idea at a time.

A small commit is easy to read, approve or discard.

Terms used in this guide

ACTIVE CONTEXT

The surface in your foreground tab, which "this" refers to in a request.

COMMIT

A built change set, staged in Sudo's sandbox, waiting for a person to approve or discard it.

/PROC

Live runtime state as files, regenerated on every read: types, exceptions, audit, runs.

SUDO-PERSONA

The skill in your workspace that holds Sudo's voice and boundaries. You can read and edit it.

ADMIN DOCK

The chat beside the admin pages, opened from the floating toggle with Sudo's avatar.

DEFINITIONS EXPORT

Your configuration as plain files: the format Sudo edits and an approved commit imports.

SANDBOX

Sudo's own filesystem and shell for one chat, with no network and no processes.

/WORKSPACE

Your configuration as files. The one mount where a write is a change to your workspace.

Ask Sudo. Approve what gets built.

Open the admin dock and hand Sudo a first, small change: a setting, a short endpoint, a question about what is running. Read the staged diff, then approve it. The detail is in the documentation.

docs.curiosity.ai

DATA RESIDENCY

EU, on premises or private cloud

COMPLIANCE

GDPR

MEMBER

KI Bundesverband