One brain, two identities

> **Genericized on purpose.** Real account emails, organisation UUIDs, and seat tiers are replaced by `identity-A` / `identity-B` throughout, so this file is shareable. The shape is the lesson; the identities are nobody else's business.
>
> **Scope.** Part 1 builds the two-account substrate and works for anyone using the `claude` and `codex` CLIs. Part 2 builds a cross-vendor audit pool on top of it; the *pattern* is portable, the exact commands assume a PAI-style harness. Both parts are written so you can stop after Part 1 and still have something useful.

The one-sentence version

There is **one shared brain** and **four credential shells**. The brain holds everything you author - config, skills, agents, hooks, memory. Each shell holds nothing but a login and its own session history. One environment variable decides which shell the CLI wakes up inside, and it is set **per terminal window**, never globally. Once two shells exist per vendor, a second vendor's model can be given a *seat* and used to audit the first vendor's work.

That is the whole idea. Everything below is detail.

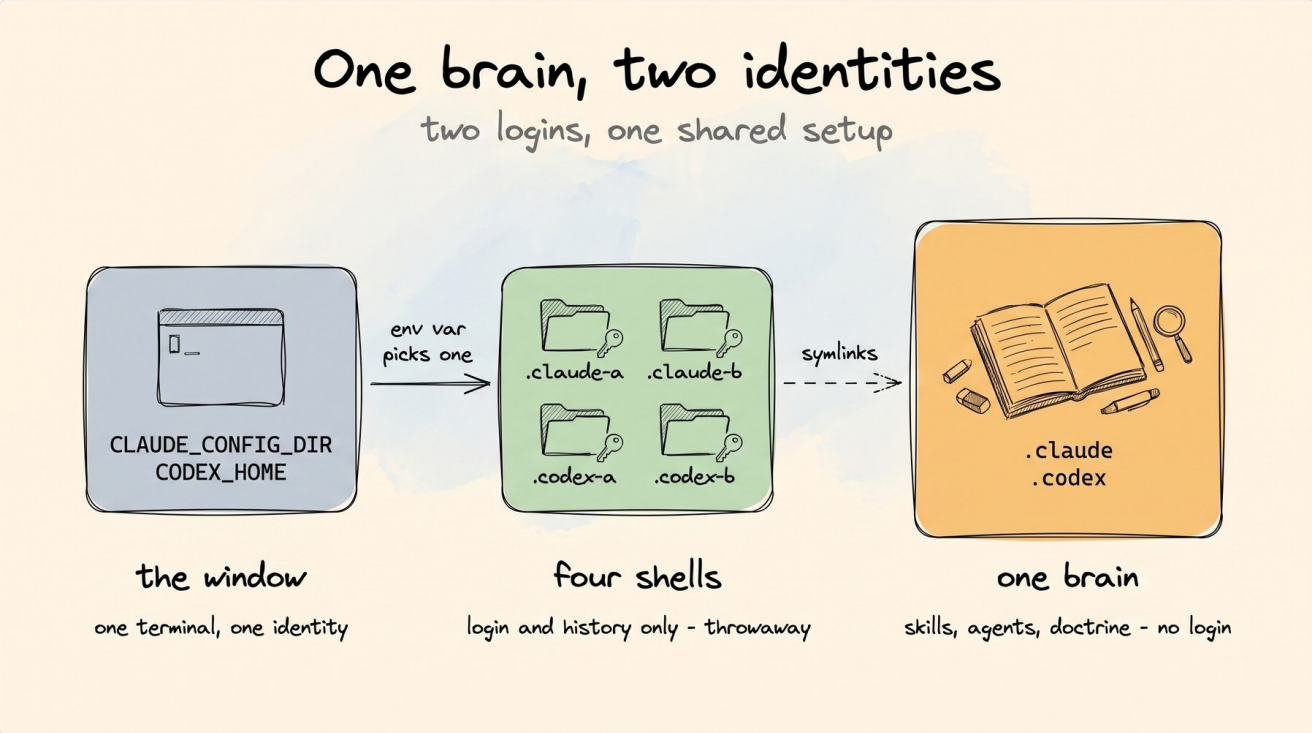


Plate 1 - the account substrate. One terminal picks one identity via an environment variable; four disposable shells hold the logins; one shared brain holds everything you authored. This is the *target* state described by Part 1, not a photograph of any one machine (see Known drift).

Why anyone would want this

Two agent CLIs, two accounts on each, and one problem: both CLIs assume a single home directory that mixes **who you are** with **what you've taught it**.

Mix those, and you get three failures:

1. **Two logins fight over one config file.** Log into the second account and the first is gone.
2. **Duplicate the whole directory per account** and your skills, agents, and doctrine fork. Fix a bug twice. Then three times.
3. **Switch accounts globally** and a long-running session in the other window silently starts billing the wrong account. This one is the dangerous failure, because nothing announces it.

The fix separates the two things that got mixed:

Concern	Wants to be	Lives in
Doctrine, skills, agents, hooks, memory	**shared, single-copy**	the brain
Credentials, session history, telemetry	**isolated, per-account**	the shells

And the payoff beyond tidiness: **two credentialed seats per vendor is what makes concurrency and cross-vendor auditing possible at all.** One account cannot audit itself in parallel.

The shape on disk

```
~/.claude                ← BRAIN (Claude). No usable login.
│                           Holds: CLAUDE.md, PAI/, skills/, agents/,
│                           hooks/, commands/, plugins/, projects/, settings.json
├── ~/.claude-a          ← SHELL, identity-A.  9 symlinks → brain
└── ~/.claude-b          ← SHELL, identity-B.  9 symlinks → brain

~/.codex                 ← BRAIN (Codex). No login at all.
│                           Holds: AGENTS.md, prompts/, memories/, rules/, skills/
├── ~/.codex-a           ← SEAT, identity-A.  auth.json + own state
└── ~/.codex-b           ← SEAT, identity-B.  auth.json + own state
```

Identity alignment is the design rule worth copying: the letter `a` means *the same human identity on both vendors*. `-a` is identity-A on Claude **and** on Codex. Without this rule you end up with crossed directories, and a cross-vendor tool launched from an `a` window silently audits and bills the `b` account. That exact bug happened here and was caught only by a cross-vendor audit stepping in its own footgun.

What each Claude shell symlinks back to the brain

Nine entries, verified identical in both shells:

```
CLAUDE.md   settings.json   PAI   skills   agents
commands    hooks           plugins        projects
```

What each Claude shell keeps as its own real files

```
.credentials.json   ← the login. Never shared, never backed up, never copied.
.claude.json        ← account identity + per-project state
sessions/  history.jsonl  telemetry/  tasks/  shell-snapshots/
paste-cache/  file-history/  policy-limits.json  projects-cache
```

The split is exactly right: **authored content is symlinked, earned state is local.**

The switch: one env var per window

```
# Claude
alias claude-a='CLAUDE_CONFIG_DIR="$HOME/.claude-a" claude'
alias claude-b='CLAUDE_CONFIG_DIR="$HOME/.claude-b" claude'

# Codex
alias codex-a='CODEX_HOME="$HOME/.codex-a" codex'
alias codex-b='CODEX_HOME="$HOME/.codex-b" codex'
```

Two variables carry the entire mechanism:

- `CLAUDE_CONFIG_DIR` → which Claude shell
- `CODEX_HOME` → which Codex seat

Because the aliases set the variable **inline**, the setting dies with the command. Nothing is global. Four `-a` windows and four `-b` windows run side by side against two independent rate budgets, and neither can disturb the other.

A small resolver (`ctx use a|b`) exports **both** variables at once, so one command puts a whole terminal on one identity across both vendors. Its account map is a plain JSON file:

```
{
  "accounts": {
    "a": { "claude": "~/.claude-a", "codex": "~/.codex-a" },
    "b": { "claude": "~/.claude-b", "codex": "~/.codex-b" }
  }
}
```

Adding a third account is one more row plus one more login. That is the payoff of the design.

Part 1 - build the account substrate

Six steps. Everything is reversible before the logins; step 5 is the point of no easy return.

Step 1 - keep your existing directory as the brain

Do not move it, do not rename it. It already holds everything you've authored.

Step 2 - create the shells

```
for a in a b; do
  d=~/.claude-$a
  mkdir -p "$d" && chmod 700 "$d"
  for i in settings.json CLAUDE.md skills agents commands hooks plugins PAI projects; do
    ln -sfn ~/.claude/$i "$d/$i"
  done
done

for a in a b; do
  d=~/.codex-$a
  mkdir -p "$d" && chmod 700 "$d"
  for i in AGENTS.md prompts memories rules skills; do
    ln -sfn ~/.codex/$i "$d/$i"
  done
done
```

`chmod 700` matters: these directories will hold OAuth tokens. Apply it to the brains too - `chmod 700 ~/.claude ~/.codex` - since the brain carries env files.

Step 3 - install the shell layer as a file, not as `.bashrc` edits

Put the aliases and the resolver function in **one script inside the brain**, and have `.bashrc` contain a single `source` line, marker-guarded so it stays idempotent:

```
grep -q 'ctx multi-account' ~/.bashrc || printf '\n# >>> ctx multi-account >>>\nsource "$HOME/.claude/PAI/TOOLS/ctx.sh"\n# <<< ctx multi-account <<<\n' >> ~/.bashrc
```

The brain is backed up; `.bashrc` on a fresh machine is not. This is the difference between a five-minute restore and an afternoon of archaeology. Put the restore procedure in a comment header **inside that script**, so the instructions travel with the file that needs them.

Step 4 - log in, once per shell

```
claude-a    # then /login  → identity-A
claude-b    # then /login  → identity-B
codex-a login --device-auth
codex-b login --device-auth
```

`--device-auth` **on the Codex logins is load-bearing.** Plain browser OAuth reuses whatever session your default browser already holds, so **both seats authenticate to the same account** and you discover it weeks later when concurrency mysteriously never happens. Device auth lets each home authenticate a different account - sign into the intended account when the device code prompts.

Never copy an `auth.json` **between homes.** The OAuth is single-use; copying it breaks both seats. If a seat refuses to switch accounts, delete its `auth.json` and log in again - that is the documented reset.

Step 5 - optionally empty the brain of credentials

Move the login out of the brain into a shell, so the brain is purely shared content:

```
pgrep -af claude || echo "no claude running - safe to move"
mv ~/.claude.json              ~/.claude-a/.claude.json
mv ~/.claude/.credentials.json ~/.claude-a/.credentials.json
```

⚠️ **Only with every agent process exited.** A running CLI rewrites its config file, so it will happily recreate a login in the brain seconds after you move it - at the worst possible moment.

Step 6 - verify isolation properly

Note shell B's credential hash, do real work in shell A, then re-check that B's hash is unchanged:

```
sha256sum ~/.claude-b/.credentials.json    # before
# ... run work in a claude-a window ...
sha256sum ~/.claude-b/.credentials.json    # must be identical
```

Three distinct hashes across brain and both shells is the proof that matters.

Pin the credential file, not the live config. An isolation check that hashes `~/.claude.json` reports failures that aren't real - the running agent rewrites that file constantly.

Part 2 - build the cross-vendor audit pool

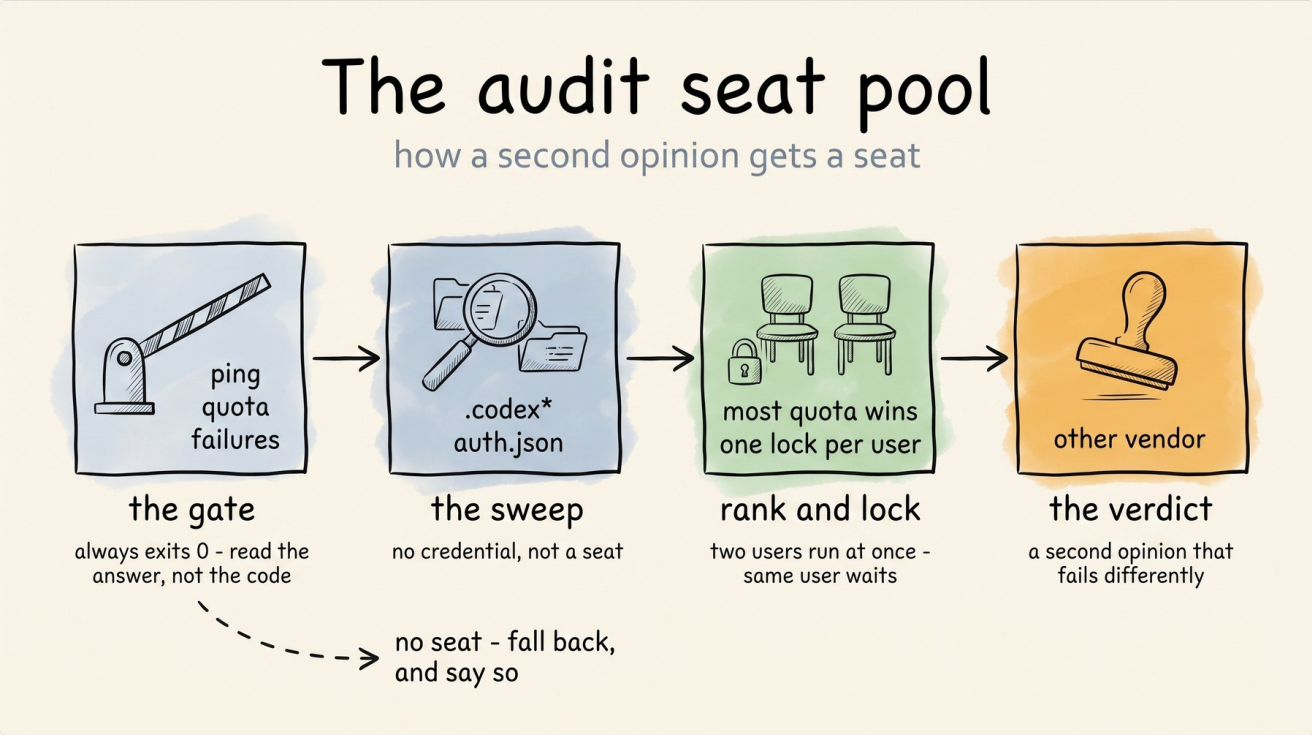


Plate 2 - the cross-vendor audit seat pool. Four questions in order: is the other vendor reachable, which directories actually hold a credential, which seat has the most budget left and is it free - and when there is no seat, fall back *and say so*.

The two-account split is not decoration. It exists so a **cross-vendor auditor** can run: a second opinion produced by a *different vendor's* model, so its mistakes are not correlated with the model that wrote the work.

The auditor runs OpenAI's model through the `codex` CLI, billed against a ChatGPT subscription rather than an API key. Each Codex shell is therefore a **seat**: one login, one quota budget, one lane of concurrency. Two seats mean two audits can run at once.

Step 7 - the pool needs no configuration

This is the part people over-engineer. **There is no registry to populate.** The pool reads the disk at call time:

1. List every `~/.codex*` directory.
2. Keep the ones holding a **readable** `auth.json`.
3. Sort by name, for a deterministic first-free sweep.

If you completed step 4, the pool already exists. Two consequences worth internalising:

- **The shared Codex brain is globbed, then rejected** - it has no `auth.json`, so it can never become a seat. Absence of a credential is what makes it a brain rather than an account. The on-disk scan, not any config file, is the source of truth. (A config file listing accounts may exist for humans to read; keep it in sync, but know the scan wins.)

- **A naive shell glob would silently miss every seat.** `~/.codex*` entries are **dotfiles**, which shell globbing excludes by default. Read the directory listing directly. Anyone reimplementing this must not use a bare glob and assume it worked.

Step 8 - verify the pool

```
bun ~/.claude/PAI/TOOLS/CodexAuditPool.ts --probe-pool
```

Live output from this machine, reformatted for reading:

```
{"pool":[
   {"account":".codex-a","home":"/home/pauli/.codex-a","state":"free",
    "user_fp":"b56465e1fd8f","account_fp":"c1a1cce73469","weekly_used_percent":5},
   {"account":".codex-b","home":"/home/pauli/.codex-b","state":"free",
    "user_fp":"a12d5a6f5b17","account_fp":"c1a1cce73469"}],
 "free_count":2,"busy_count":0,
 "distinct_users":2,"distinct_accounts":1,
 "shared_account_homes":`".codex-a",".codex-b"`,
 "shared_user_homes":[],"unidentified_user_homes":[]}
```

Read it like this - and this is the acceptance test for the whole build:

- `distinct_users: 2` → **the thing you actually want.** Two independent quota budgets, so two audits can run at once.
- `distinct_accounts: 1` with both seats in `shared_account_homes` → both seats belong to one ChatGPT Team account. **This is fine and expected.** Two seats of one Team account are two distinct *users*.
- `~/.codex` is **absent from the pool** → correct. It was globbed and rejected for having no credential.
- `unidentified_user_homes: []` → no seat had an unreadable token owner.

If you see `distinct_users: 1`, both seats authenticated to the same account - the browser-OAuth trap from step 4. Delete one seat's `auth.json` and redo it with `--device-auth`.

Step 9 - understand the lock before you trust the concurrency

This is the subtle part, and getting it wrong wastes real money.

A single ChatGPT **Team account** can hold two seats. Those two seats meter independently, so they genuinely parallelize. The lock is therefore keyed on the **user** fingerprint, never on the account and never on the task being audited:

- Two different users → two different locks → true parallel audits.
- The *same* user twice → one lock → the second attempt refuses rather than corrupting a shared token.

So `account_fp` is **informational**; `user_fp` is **load-bearing**. Collapsing two seats because they share an account would destroy the concurrency the whole design exists to provide. Do not "fix" this.

A seat whose token owner cannot be read is **dropped fail-closed** - never pooled. Unknown ownership is disqualifying, not "probably fine".

Step 10 - how a seat gets chosen

Seats are `free`, `busy`, or `degraded`. Free seats are ranked by **remaining quota** and the best one wins. The 5-hour window is used when the plan reports one; **Team plans don't emit it**, so ranking falls back to the weekly window. If you build this on a Team plan and wonder why 5-hour numbers are missing, that's why.

Step 11 - the honesty gate

Before an audit runs, a gate answers one question: *is the cross-vendor path actually usable right now?*

```
bun ~/.claude/PAI/TOOLS/CodexAuditPool.ts --gate
```

Three checks feed it - a cheap liveness ping, a quota breaker, and a **failure-typed** breaker that opens when real audits keep dying even though the ping passes.

Two design details carry the whole lesson:

- **The gate always exits 0.** You read the `backend` field, never the exit code. A gate that signalled through exit status would be indistinguishable from a gate that crashed.
- An audit returning a clean verdict of **"fail" is not a failure.** Confusing *the auditor disliked the work* with *the auditor is broken* would trip the breaker every single time the tool did its job correctly.

When the gate says the cross-vendor path is unusable, the audit falls back to the same-vendor model and **marks itself as not cross-vendor** in its output. The fallback is allowed; pretending it was cross-vendor is not. A guard blocks launching the auditor directly while the pool is healthy, because a direct launch would skip the pool and produce a same-vendor audit wearing a cross-vendor label.

The transferable rule: a degraded mode must announce itself in its output. Silent degradation is worse than failure, because failure gets investigated and silence gets trusted.

Step 12 - one caution before your first real audit

Note that `--gate` and `--health` acquire the *same* per-user lock a real audit uses. A health probe and a live audit on one seat therefore genuinely serialize - that is correct behaviour, not a bug, but it means a probe loop can block real work.

Traps, each one paid for

A symlinked `settings.json` propagates writes back into the brain. An overlay session that edits shared settings changes them for every account. Intended for shared doctrine - but for per-account settings, drop a real `settings.local.json` in the shell rather than editing the symlinked file. Heavy directories (`skills/`, `PAI/`, `hooks/`) are unaffected, because writes go *into* them rather than replacing them.

A shared brain means shared secrets. Both shells symlink the brain, so identity B can read env files belonging to identity A. Fine for one human with two accounts. Not fine if the two accounts are two people.

`claude doctor` **hangs under a fresh, loginless shell** - it waits on stdin and exits 124. You cannot verify a shell non-interactively before logging into it. Verify after.

Cross-vendor tools may hardcode the brain path. Anything reading `~/.codex` directly instead of honouring `CODEX_HOME` will audit and bill the wrong seat no matter which window launched it. Grep your tooling for hardcoded home paths before trusting the switch.

Env files want `chmod 600`. Easy to miss, and a shared brain multiplies the exposure.

Rollback and restore

Rollback, stage A - pre-login (zero impact). Only symlinks and empty state exist:

```
rm -rf ~/.claude-a ~/.claude-b ~/.codex-a ~/.codex-b
# then remove the marker-guarded block from ~/.bashrc
```

Rollback, stage B - post-login (destructive). The shells now hold real credentials. Archive, never delete:

```
# if you did step 5, put the login back into the brain first:
mv ~/.claude-a/.claude.json      ~/.claude.json
mv ~/.claude-a/.credentials.json ~/.claude/.credentials.json
# preserve the rest instead of destroying it:
mkdir -p ~/.agent-rollback-bak && mv ~/.claude-b ~/.codex-a ~/.codex-b ~/.agent-rollback-bak/
```

The brains are never destructively touched in either stage.

Restore on a fresh machine. The brain is in version control; the logins are not, and never should be. Restore the brain, re-add the one `.bashrc` source line, rebuild the shells with step 2, then re-login with step 4. Tokens never leave the machine, so **every restore ends with four interactive logins** - plan for that rather than being surprised by it.

Known drift in this installation

Recorded honestly, because a teaching document that hides its own gaps teaches the wrong lesson. **The build steps above describe the correct target state; these four items are where this machine currently differs.**

1. **The brain is identity-free in config but still held a token file.** `~/.claude/.claude.json` has `oauthAccount: null`, yet `.credentials.json` was still present - an OAuth token that **expired 2026-07-30 06:04 EEST**, so it grants nothing. Step 5's credential move was designed as optional and left undone. Net: the brain is *functionally* loginless, not *literally* credential-free. Deleting the expired file completes the intended end state.
2. **The Codex seats are asymmetric.** `~/.codex-b` symlinks five shared entries (`AGENTS.md`, `prompts`, `memories`, `rules`, `skills`); `~/.codex-a` symlinks only `AGENTS.md`. So identity-A's Codex silently does **not** see shared prompts, memories, rules, or skills. Four missing links. Step 2 above is written symmetrically and is the correct target.
3. **The Codex brain is documented elsewhere as "blank/unused" but isn't.** It holds `AGENTS.md`, `prompts/`, `memories/`, `rules/`, `skills/` plus its own databases and sessions - a genuine shared brain that the seats symlink into. The related claim that it "is not discovered by the pool" is imprecise: discovery does glob `~/.codex*` and *does* see it, then rejects it for holding no readable `auth.json`. Correct phrasing: **globbed, then rejected for having no credential** - not invisible.

4. **Permissions are uneven.** All four shells and the Codex brain are `700`; the Claude brain is `755`. Since the brain carries env files, `700` is the consistent choice.

The transferable principle

Separate **what you are** from **what you know**. Let identity be a cheap, disposable shell - one login, one history, thrown away and rebuilt in a minute. Let knowledge be a single expensive artifact, authored once, backed up, symlinked everywhere.

Then make switching between identities a property of **the window you're typing in**, never a property of the machine. A switch that lives in one terminal cannot silently reach into another.

And once you have two cheap identities, you have something better than tidiness: you have a **second, differently-wrong opinion** on your own work - and a gate that admits, out loud, when it couldn't get one.

Colophon

This document was created by **Saul** (Paulius's AI assistant, running on Claude Code) on **2026-07-31 00:34 EEST (Europe/Vilnius)**.

The prose was written from structure read directly off disk on 2026-07-31, not from memory: directory listings, symlink targets, file permissions, credential expiry, and a live `--probe-pool` capture. Account emails, organisation identifiers and seat tiers are deliberately genericized to *identity-A* / *identity-B* so the document is shareable.

The two plates were generated with the StoryArt pipeline (Visualize workflow, `technical-diagram` composition, PAT Narratives hand-drawn aesthetic v3.1.0) via Gemini 3 Pro Image at 2K, 16:9. Two earlier attempts on the cheaper Flash tier were discarded - the first misread the metaphor, the second misspelled the directory labels - and both are logged as calibration data in the paired narrative.

Sources in the vault:

`MIND/References/2026-07-31__two-account-agent-cli-substrate.md`

`BODY/narratives/_user/2026-07-31__one-brain-two-identities__narrative.md`

`BODY/narratives/_user/2026-07-31__one-brain-two-identities__plate1-substrate.jpg`

`BODY/narratives/_user/2026-07-31__one-brain-two-identities__plate2-pool.jpg`