> <

Ada.
An agent-first code editor.

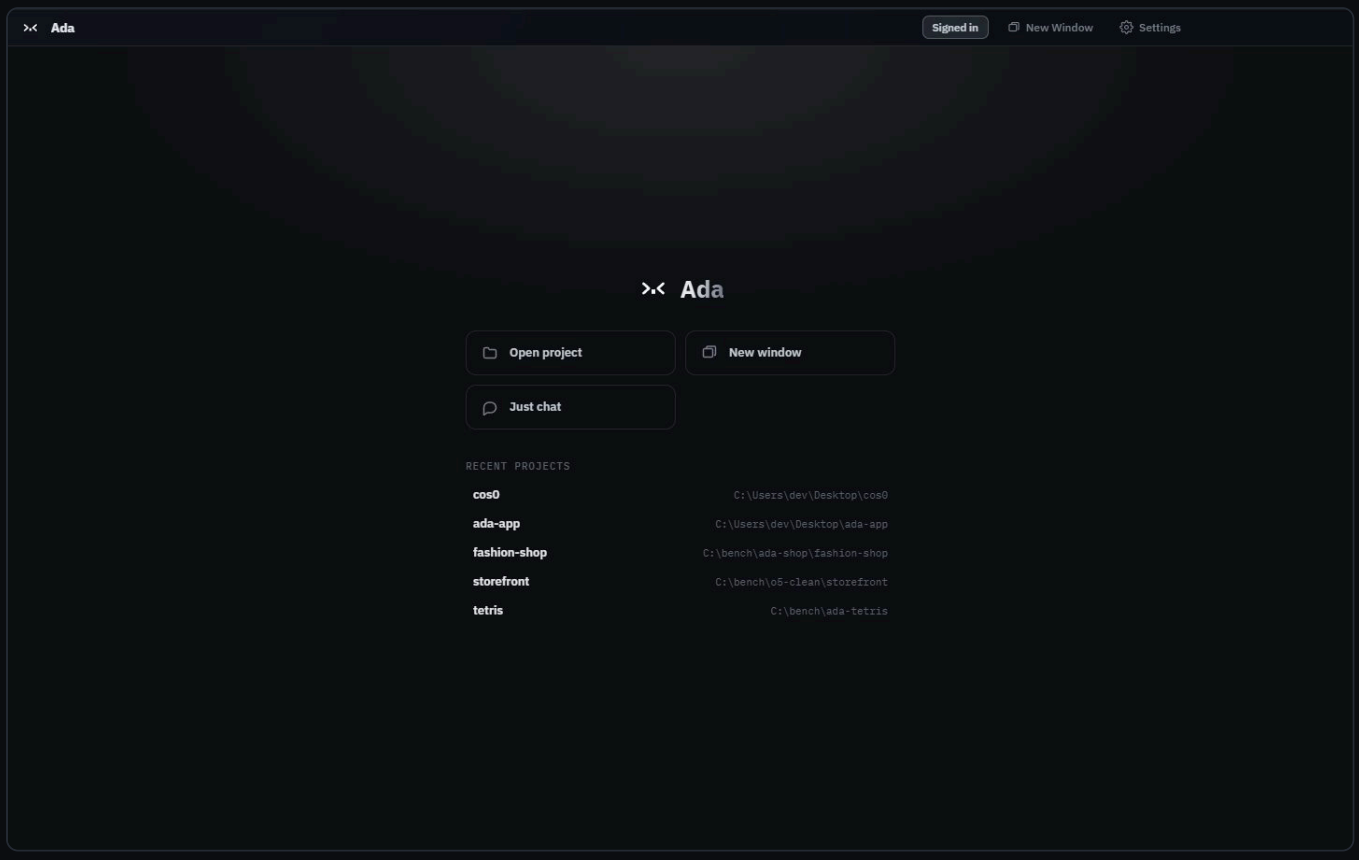
A desktop editor built around a coding agent that reads, writes and runs code — with a context strategy that measurably cuts what every request costs.

PRODUCT & BENCHMARK REPORT · v0.1.23

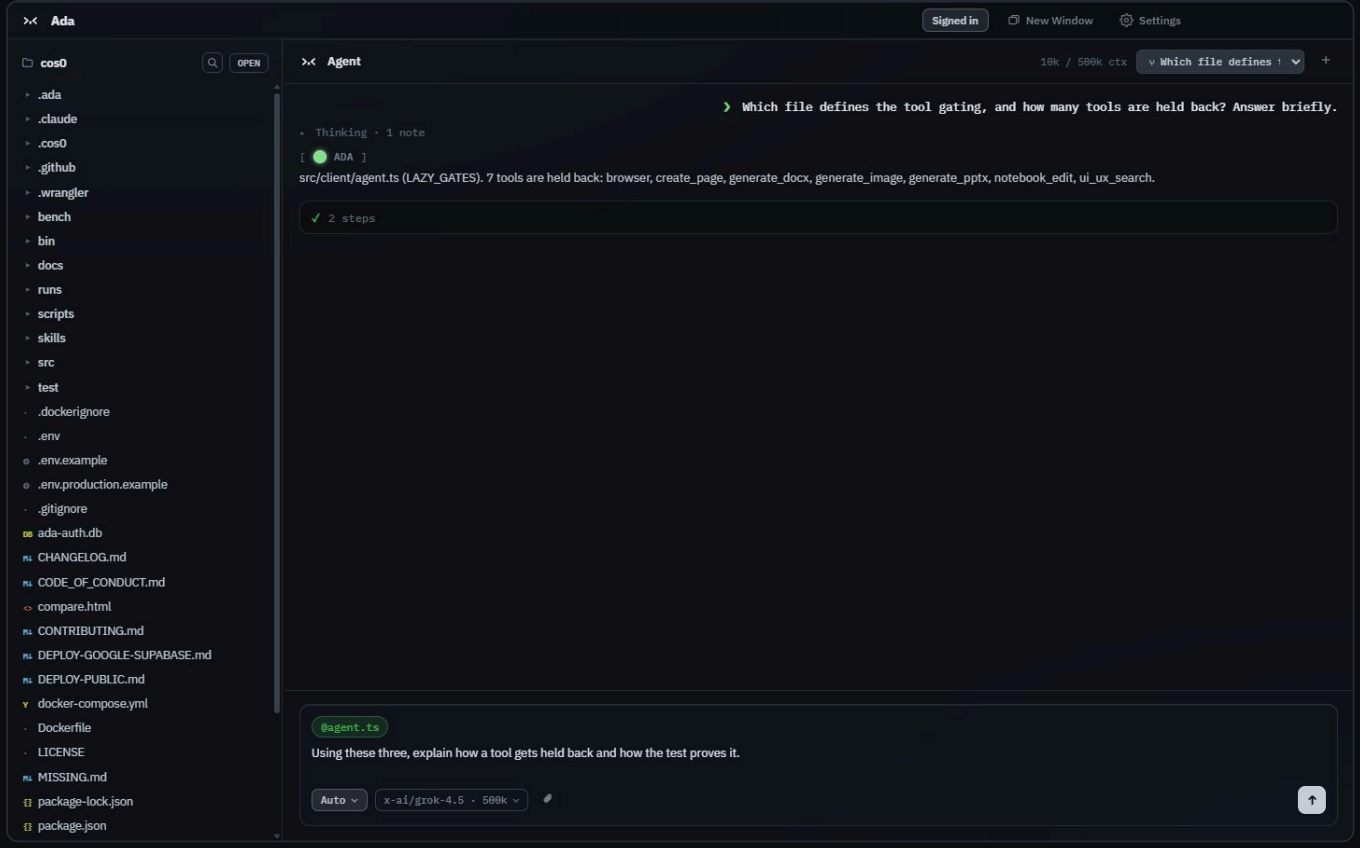
26 July 2026	**Win · macOS · Linux**	**340+ models**	**22 tools · 291 skills**
RELEASED	SIGNED & NOTARIZED	12 PROVIDERS	BUNDLED, NO SETUP

Open a folder. Describe what you want.

The agent works in an isolated git worktree, so nothing touches your working copy until you merge. No folder open? Chat anyway.



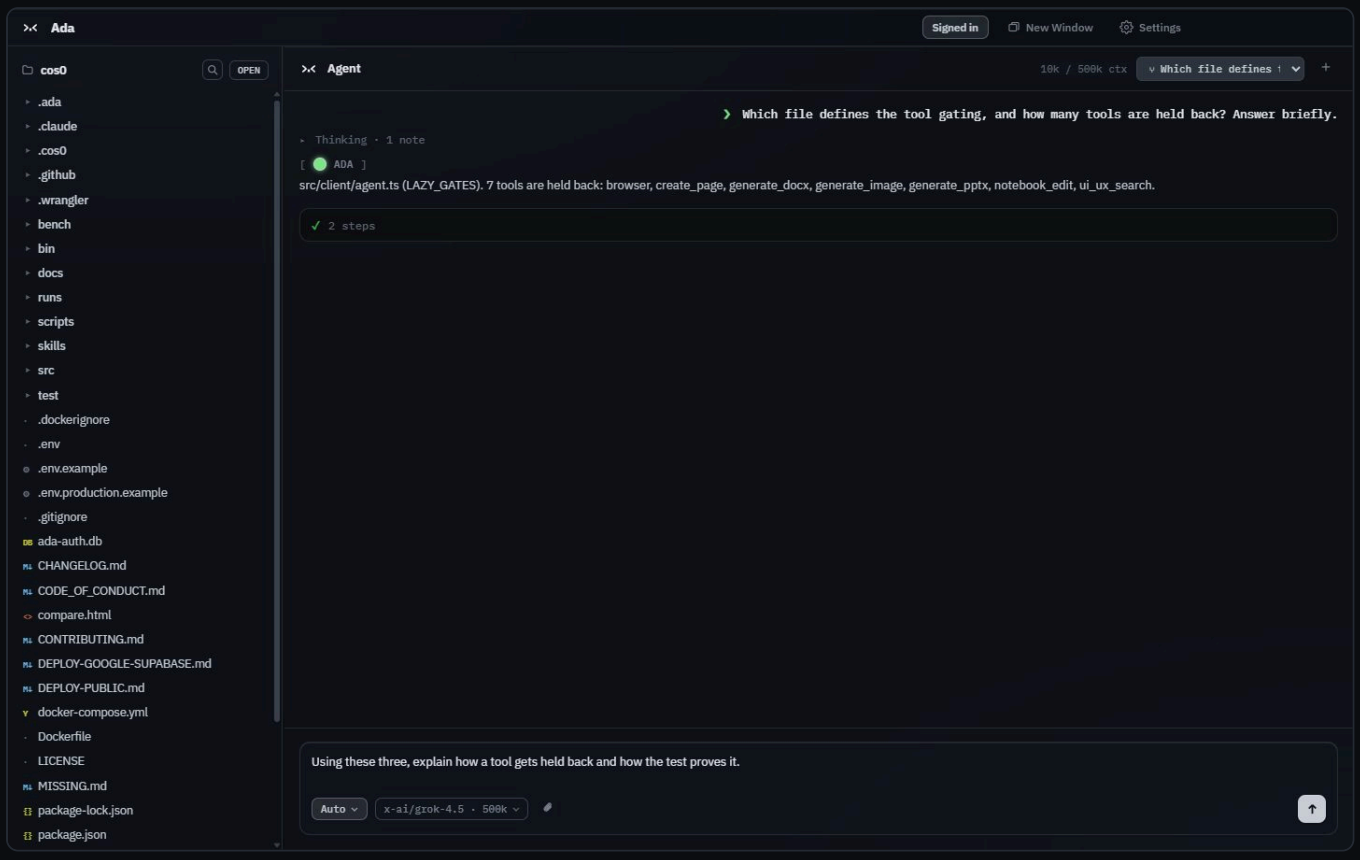
›The welcome screen — open a project, start a chat without one, or pick from the ten most recent.



›A project open: file tree, agent panel, permission modes, model picker. Models switch mid-conversation; the chat carries over.

It shows its work.

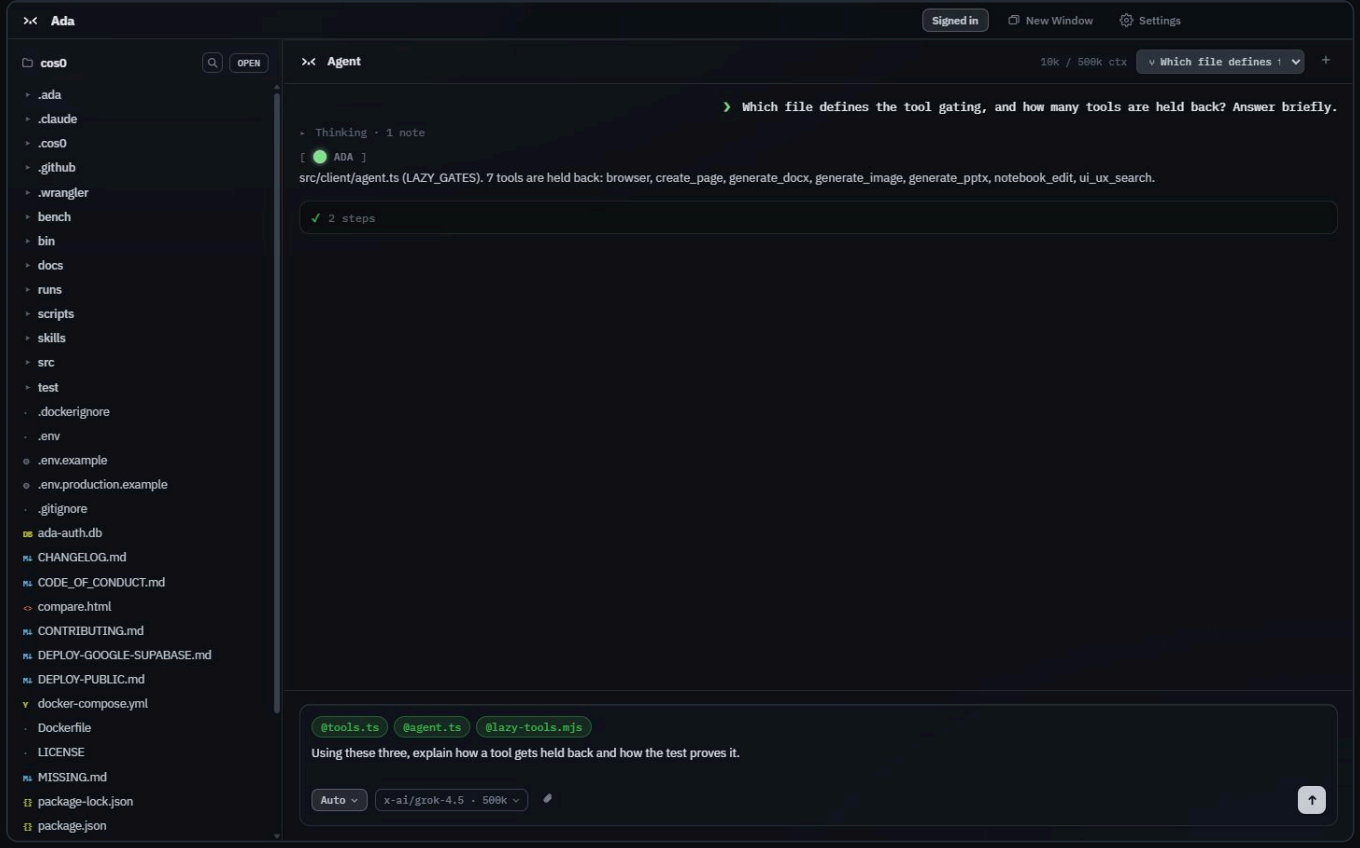
Reasoning, steps taken, and a live context meter on every reply. Below, the agent answers a question about its own source — correctly — in two steps and **10k of a 500k window**.



›A real run. The answer names the file and all seven gated tools; the header reads 10k / 500k context.

Files as context

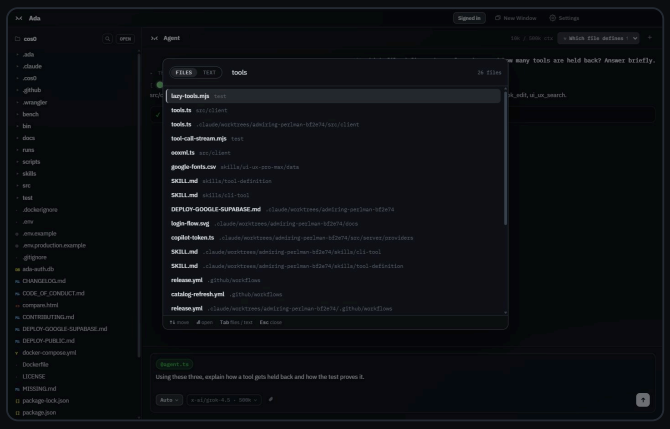
Attach from a picker or drag from the tree. Each file becomes a chip — click to open, hover to remove — so the agent reads what you meant, not what it guessed.



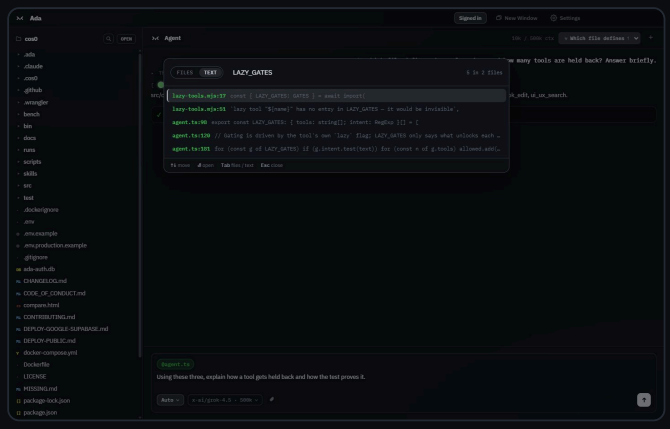
›Three files attached as context before the question is asked.

Find anything.

One palette, two modes: `Ctrl+P` ranks filenames by fuzzy match; `Ctrl+Shift+F` searches contents and jumps to the line.

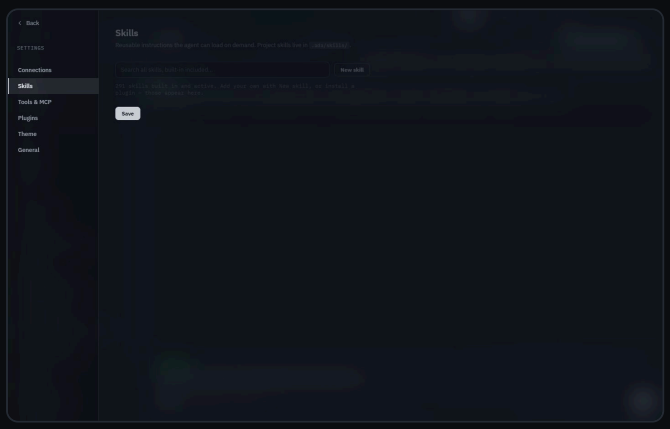


›Open by name — fuzzy-ranked, 26 matches for “tools”.

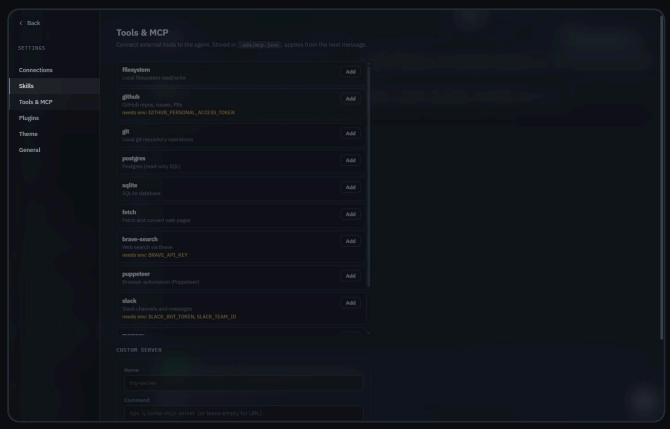


›Find in files — five hits in two files, with the line.

Set up without config files



›291 skills ship built in; the list shows only what you add.



›MCP connectors from a catalogue, stored in .ada/mcp.json.

Why it costs less.

The expensive part of AI coding is not the model — it is how much you send it, on every turn. Four measured choices:

32k
Ada's whole context on a task where the alternative carried ~77k before reading a single file

1,772
tokens of tool schemas on an ordinary turn, against 3,848 ungated

272
tokens for “hi” — no tools, no repo map

2,076
tokens across 7 tools held back until the conversation asks

No standing prompt

Tool schemas are re-sent on every request, so every exposed tool taxes every turn. Decks, documents, images, pages, notebooks, the browser and the design corpus stay out until a turn asks — each behind its own trigger, so asking for a deck does not drag in the notebook schema.

A tool beats a description

TASK	ADA	CLAUDE CODE	WHY
Build a presentation	$0.28	$1.25	Native deck writer vs hand-written markup
“Which file defines the order API routes?”	9,525	18,738	Repo map answers without reading files

And the model is yours to pick. Ada routes to 340+ models across 12 providers. The storefront in the case study cost $2.04 on Grok 4.5 against **$4.12 on Opus 4.7** — same harness, same brief. That lever does not exist in an agent tied to one vendor.

Five tasks, same repository.

Ada and Claude Code: same machine, same model (Opus 4.7), same prompts, matched git worktrees at the same commit.

2.0×
less context — 444k tokens against 893k

$1.63
cheaper in total — $3.05 against $4.68

4.5×
cheaper on the presentation task

4 / 5
tasks cheaper; Tetris was the exception

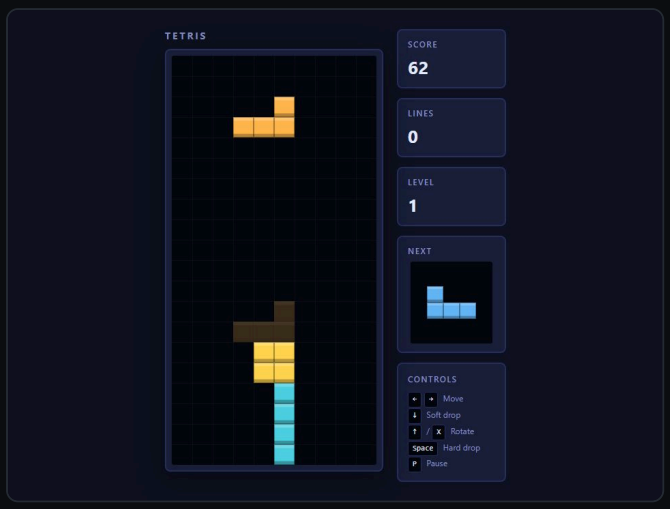
TASK	ADA IN	ADA OUT	ADA COST	CC IN	CC OUT	CC COST
Summarise the project (avg of 3)	31,856	1,811	$0.2046	95,941	758	$0.5291
Create an API to get pricings	206,954	7,970	$1.2340	424,706	3,688	$1.5339
Tetris game	109,041	8,726	$0.7634	95,708	2,919	$0.6885
Portfolio website	57,075	11,282	$0.5674	128,389	2,065	$0.6738
Project presentation (.pptx vs HTML)	38,726	3,484	$0.2807	148,150	6,557	$1.2517
TOTAL	443,652	33,273	$3.0501	892,894	15,987	$4.6770

Ada sent half the input and produced twice the output. 444k in against 893k, while writing 33,273 tokens of answer against 15,987 — the saving is not the agent saying less. It read less to say more.

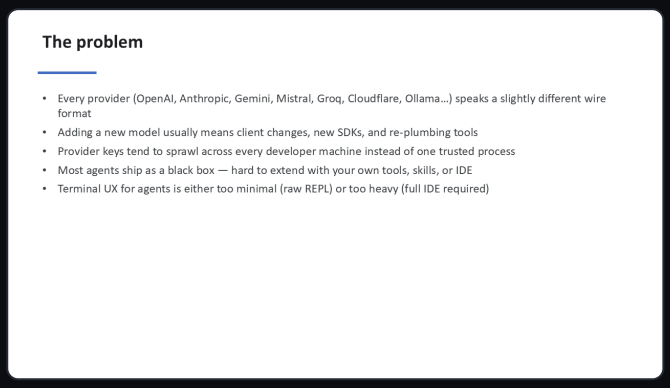
> **Where it does not help.** Tetris is greenfield — it needs nothing from the repository, so a map of the repository buys nothing and Ada's extra reading is pure overhead. That is the shape of the whole result: **the further a task sits from your existing code, the smaller the advantage.** Ada was also slower overall (655s against 273s).

What the runs produced.

Not transcripts — deliverables. The game runs, the deck opens in PowerPoint.

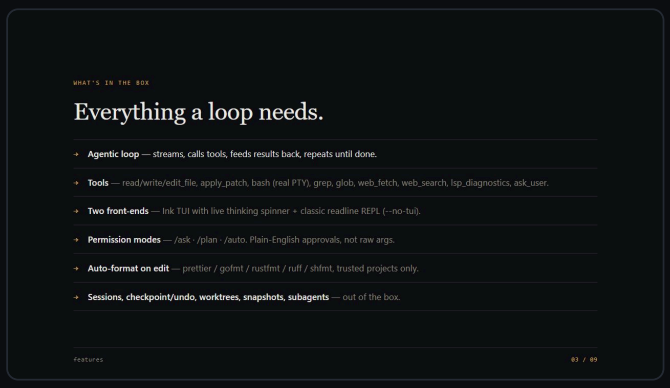


›Tetris — Ada · $0.76. Playable on arrival: score, levels, next-piece preview, hard drop. The one task Ada lost on price ($0.76 vs $0.69).



›Project presentation — Ada · $0.28. A real 14-slide .pptx from the native deck writer; opens in PowerPoint, not a webpage pretending.

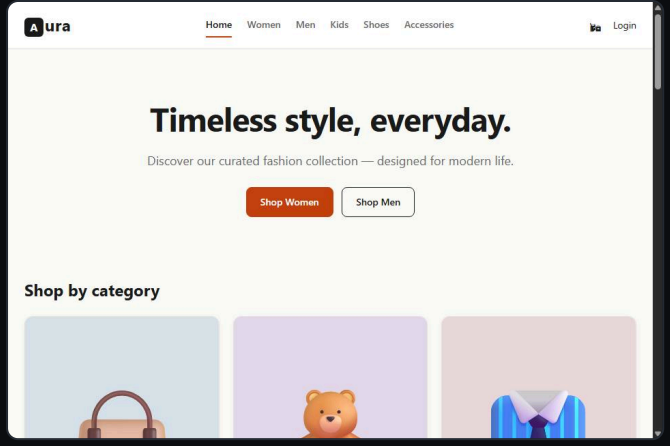
›The same brief on Claude Code · $1.25 — a hand-built HTML page styled as slides.



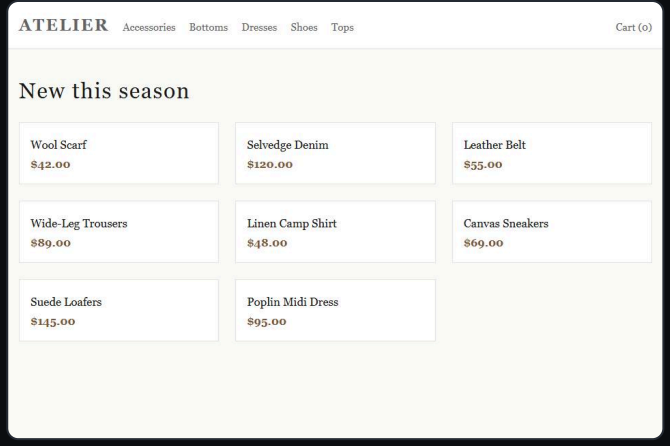
›Claude Code's deck, content slide. 4.5× the cost for markup that PowerPoint cannot open.

One brief, six builds.

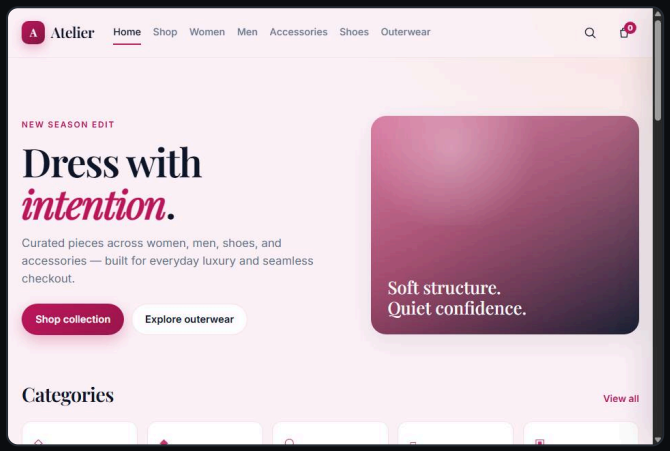
“Fashion management site with different categories and checkout system with backend and db.” Identical wording to every agent. Each build was installed, run, and driven through a real checkout — not just read.



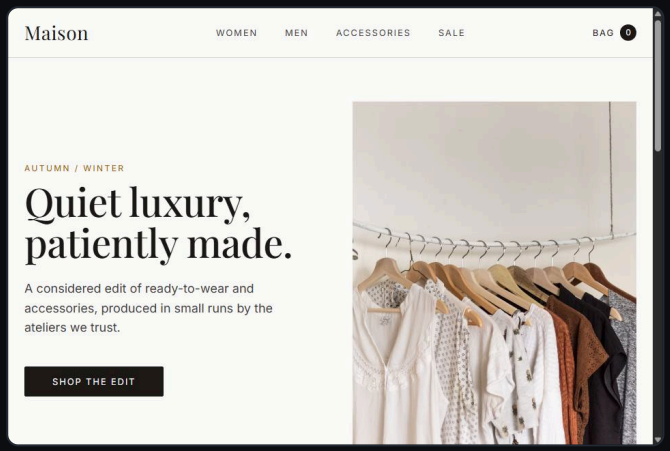
›Aura — Ada · Opus 4.7 · $4.12. The only build with real user accounts.



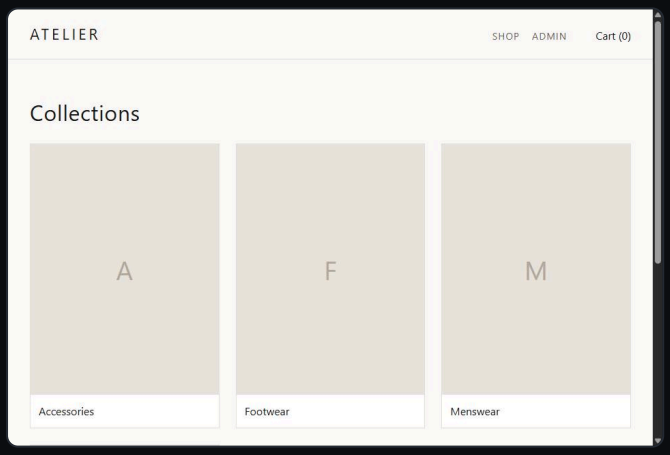
›Atelier — Claude Code · Opus 4.7 · $1.91. Minimal, ships a test.



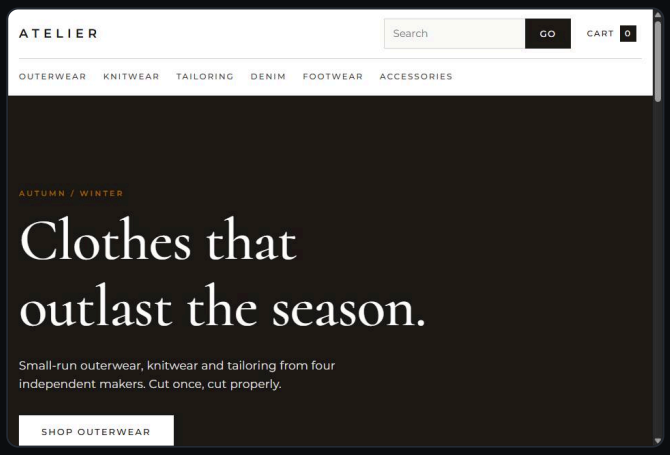
›Atelier — Ada · Grok 4.5 · $2.04. Strongest front end; shipped broken — one missing bracket.



›Maison — CC + design corpus · Opus 4.7 · $6.22. Best art direction.



›Atelier — Claude Code · Opus 5 · $1.04. Two files, zero dependencies.



›Atelier — CC + corpus · Opus 5 · $2.39. Corpus-chosen palette and type.

What each build actually shipped.

Every build got the categories, cart, checkout and SQLite the brief asked for. The difference is what they added without being told to.

CAPABILITY	AURA ada · opus 4.7	ATELIER cc · opus 4.7	ATELIER ada · grok	MAISON cc + corpus	ATELIER cc · opus 5
SQLite database	yes	yes	yes	yes	yes
ACCOUNTS AND ADMINISTRATION — ASKED FOR BY NOBODY					
Users table	yes	—	—	—	—
Password hashing	bcryptjs	—	—	—	—
Admin, role-gated	is_admin + requireAdmin	—	—	shared key	env password
Sessions	JWT	cookie	—	express-session	basic auth
CORRECTNESS OF THE MONEY PATH					
Checkout decrements stock	yes	yes	yes	yes	yes
Inside a transaction	yes	yes	—	yes	yes
Ships a runnable test	—	yes	—	yes	yes
DELIVERY					
Ran without a fix	yes	yes	no	yes	yes
Works offline	yes	yes	webfonts	webfonts	yes

Only one build could be logged into. Aura was the single build of six that created a `users` table, hashed passwords with bcryptjs, issued JWTs and gated administration behind a role — verified by a 403 without it. Nothing in the brief asked for accounts. A capability gap survives scrutiny in a way a percentage does not.

Model economics changed mid-benchmark.

Re-running the same brief on Opus 5 reconciles exactly against list pricing on three independent runs:

PER MILLION TOKENS	INPUT	OUTPUT	CACHE READ	CACHE WRITE
Opus 4.7	$15.00	$75.00	$1.50	$18.75
Opus 5	$5.00	$25.00	$0.50	$6.25

Same brief, same harness, same machine — only the model differs:

RUN	OPUS 4.7	OPUS 5	CHANGE
Storefront, no design corpus	$3.37	$1.04	3.2× cheaper
Storefront, with design corpus	$6.22	$2.39	2.6× cheaper

Read this honestly. Opus 5 read *more* cached tokens than 4.7 on the identical task (2.28M against 2.24M) and produced more output. The saving is the price cut, not efficiency — and it applies to any harness, including this one, since Ada routes to the same models.

Method, and what this does not show.

How it was run

- Matched git worktrees of the same repository at the same commit for every arm.
- Identical prompt wording; costs read from each tool's own reported usage.
- Every storefront was installed, started, and driven through a real checkout over HTTP — stock decrements and order rows verified in the database, not assumed.
- Screenshots are of the running builds and the shipped application.

Known limits

- **Ada is slower.** 655s against 273s across the five tasks.
- **It lost the storefront on price.** Plain Claude Code shipped a working build for $1.91 against Ada's $4.12 — a far smaller build, but cheaper is cheaper.
- **Ada pipes `npm install` output through the model** — most of the 764k input tokens on that task. Fixable, not yet fixed.
- **Ada's own cost display runs ~14% high** against provider billing.
- Single runs except where noted; small samples, one machine, one network.

ADA v0.1.23 · BENCHMARKS 24–26 JUL 2026 · WINDOWS 11, NODE 22 · COSTS IN USD FROM EACH TOOL'S REPORTED USAGE, CROSS-CHECKED AGAINST PROVIDER BILLING WHERE AVAILABLE