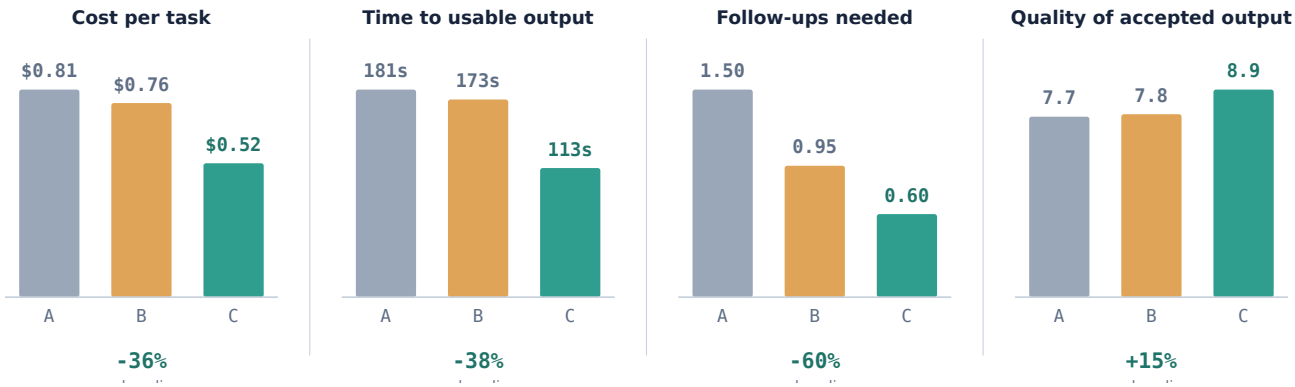
UNABYSS · ENGINEERING NOTES

What context is actually worth

We ran the same twenty tasks sixty times in Claude Code, changing one thing: whether the assistant had our context already structured and waiting, or had to go and assemble it. Same prompts, same connected tools, a fresh session every single run.

Piotr Langier · August 2026 · 20 tasks · 3 setups · 60 runs



Averages across 20 tasks. A = connected tools only. B = tools plus a hand-written context file. C = tools plus Unabyss. Lower is better on cost, time and follow-ups; higher is better on quality.

Connect enough MCP servers and an agent can reach almost everything about your work: mail, calendar, tickets, repositories, documents. What it cannot do is start from an understanding of any of it. Each session opens cold, and the context has to come from somewhere.

Usually it comes from one of three places. You supply it yourself, which costs your time. The agent goes looking for it across ten tools, which costs tokens and minutes. Or it proceeds without it, and returns something generically right and specifically wrong. Unabyss keeps that context structured and ready instead. This test measures what that is worth in ordinary work.

The setup

One variable, three settings. Every run used Claude Code with the same ten MCP servers connected: Apollo, GitHub, Gmail, Google Calendar, Google Drive, Linear, Neon, Slack, Trigify and Vercel. The only thing that changed was what the assistant knew going in.

A Baseline	B Static file	C Unabyss
Tools connected, no context layer. The assistant can reach your data, but knows nothing about you when the session opens.	The same, plus a hand-written CLAUDE.md holding role, company, positioning, team and writing voice.	The same, plus Unabyss, a context layer synced from LinkedIn, the website, X, Slack, GitHub, Linear, Gmail, Drive and Calendar.

Each task was run once per setup, in a fresh session with no memory of anything before it. Prompts were written once and pasted in identically across all three.

What we measured

Cost in dollars, read from the session at the end. **Time** on the clock, from pasting the prompt to holding something actually usable, not to first output. **Follow-ups**, meaning every extra reply needed to get there, including answering questions the assistant asked back. **Quality** of the output we finally

accepted, scored 1 to 10 on a fixed scale. Follow-ups measure the effort to get there; the score measures how good the result was once we stopped.

The result

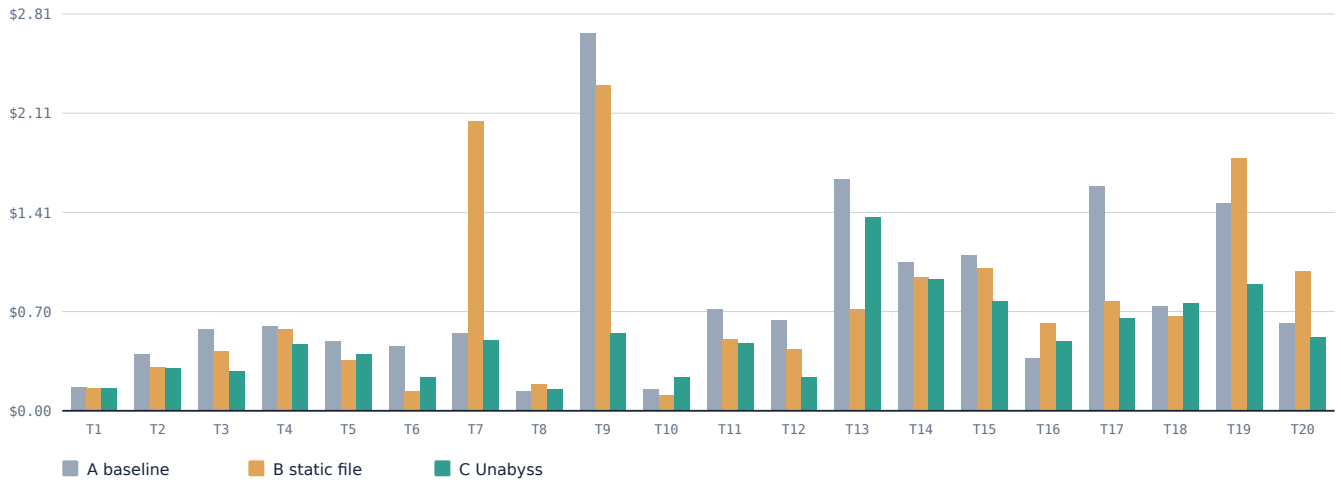
ROUGHLY A THIRD CHEAPER, A THIRD FASTER, AND BETTER ON THE FIRST TRY

Unabyss cut cost by 36% and time by 38% against the baseline. Follow-ups dropped by 60%, from 1.5 rounds of correction per task to 0.6. Quality of the accepted output rose from 7.7 to 8.9 out of 10.

The static file is the more interesting comparison. A well-written CLAUDE.md is the sensible answer if you have never seen a context layer, and it does help: follow-ups fell by 37%. But it barely moved cost or time, and it added almost nothing to quality: 7.84 against the baseline's 7.74. A file tells the assistant what is true in general. It cannot tell it what happened yesterday.

> The static file closed a third of the follow-up gap and almost none of the quality gap. Knowing *about* someone is not the same as knowing what they are working on.

Cost, task by task

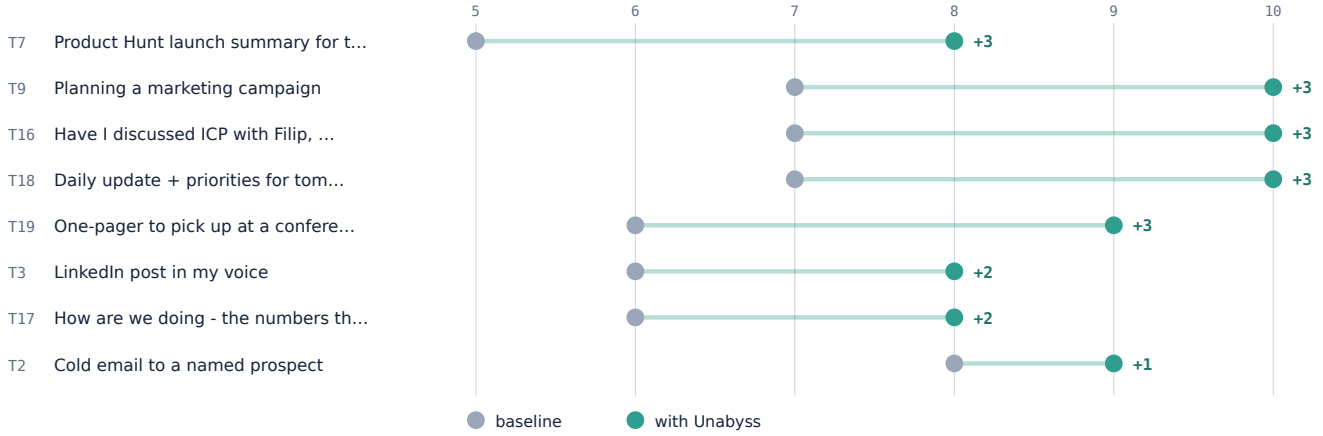


Dollar cost per run. The spread widens with how much the task depends on knowing things: T9, a month-long marketing plan, cost $2.68 at baseline and $0.55 with Unabyss.

Two patterns show up. On short, self-contained tasks the three arms are almost identical, because there is nothing to know, so knowing it saves nothing. The gap opens on work that needs grounding: planning, briefing, anything where the answer depends on facts the assistant does not have.

T1 is worth singling out. It asks for a Python script to deduplicate a CSV, and it needs no context whatsoever. All three arms landed within two cents and three seconds of each other. That is the control working: the differences everywhere else come from context, not from having more tools switched on.

Where quality moved most



Quality of the accepted output, baseline against Unabyss, for the eight tasks with the largest movement.

The biggest jumps share a shape. They are questions about your own situation, where a generically competent answer is worthless.

“Have I discussed our current ICP with Filip, and where?” went from 7 to 10. The baseline searched and surfaced two places where Filip had written something about ICP, which answers the question narrowly and stops there. With Unabyss it located the same material, read across it, and returned what our current thinking on ICP actually is, up to date.

“Give me a daily update and my priorities for tomorrow” went from 7 to 10. Without context this is a productivity lecture. With it, it is a ranked list of the things actually in flight.

“How are we doing? Give me the numbers that matter” went from 6 to 8. The hard part is not fetching numbers, it is knowing which ones matter to this company this month.

Not every task moved, and that is the point. Five came out level or slightly behind. Where a task is self-contained, context is dead weight: a small overhead for tool definitions and nothing to show for it. The value is concentrated exactly where you would expect it to be.

Every task

TASK		COST			TIME			QUALITY		
		A	B	C	A	B	C	A	B	C
T1	CSV dedupe script	$0.17	**$0.16**	**$0.16**	17s	**14s**	16s	n/a	n/a	n/a
T2	Cold email to a named prospect	$0.40	$0.31	**$0.30**	106s	68s	**62s**	8	**9**	**9**
T3	LinkedIn post in my voice	$0.58	$0.42	**$0.28**	171s	235s	**40s**	6	6	**8**
T4	Catch me up on the week	$0.60	$0.58	**$0.47**	65s	92s	**39s**	**7**	6	**7**
T5	Pre-call brief	$0.49	**$0.36**	$0.40	74s	**56s**	61s	10	10	10
T6	Positioning framing recall	$0.46	**$0.14**	$0.24	124s	**13s**	50s	9	9	9
T7	Product Hunt launch summary for the team	$0.55	$2.05	**$0.50**	250s	438s	**211s**	5	7	**8**
T8	Email in Polish to a client	**$0.14**	$0.19	$0.15	35s	44s	**25s**	9	9	**10**
T9	Planning a marketing campaign	$2.68	$2.31	**$0.55**	618s	431s	**271s**	7	9	**10**
T10	Explaining what we do to my mum	$0.15	**$0.11**	$0.24	16s	**14s**	31s	8	**9**	**9**
T11	Write me a CV	$0.72	$0.51	**$0.48**	235s	346s	**177s**	**10**	7	7
T12	Most repetitive tasks I could automate	$0.64	$0.44	**$0.24**	101s	85s	**51s**	8	**10**	8
T13	Review our Trigify workflows	$1.64	**$0.72**	$1.37	336s	**102s**	259s	9	**10**	**10**
T14	Deck: 10 tech updates from last month	$1.05	$0.95	**$0.93**	262s	251s	**244s**	9	9	**10**
T15	What is the team currently working on	$1.10	$1.01	**$0.78**	240s	210s	**144s**	**9**	8	**9**
T16	Have I discussed ICP with Filip, and where?	**$0.37**	$0.62	$0.49	**39s**	79s	83s	7	7	**10**
T17	How are we doing - the numbers that matter	$1.59	$0.78	**$0.66**	262s	99s	**92s**	6	5	**8**
T18	Daily update + priorities for tomorrow	$0.74	**$0.67**	$0.76	101s	114s	**99s**	7	6	**10**
T19	One-pager to pick up at a conference	$1.47	$1.79	**$0.90**	382s	565s	**195s**	6	7	**9**
T20	Press pitch to a journalist	$0.62	$0.99	**$0.52**	195s	201s	**106s**	7	6	**8**

Best result in each row shown in teal. T1 is the control task and was not scored for quality.

How quality was scored

10	Flawless. No mistakes, no placeholders, personalised, in my tone.
9	No mistakes, up to one placeholder, mostly personalised.
8	No mistakes, placeholders present, personalised to some extent.
7	Up to one small mistake, placeholders, not fully generic.
6	Small mistakes or one large one, rather generic.
5	Incomplete, many small mistakes or up to two large ones, generic.
0–4	Not good enough to use. Re-run.

Method

ENVIRONMENT	Claude Code, one fresh session per run. No resuming, no carry-over between tasks.
RUNS	20 tasks × 3 setups × 1 run = 60 runs, plus 9 further runs repeating three tasks to test repeatability.
PROMPTS	Written once, pasted identically into all three arms.
CONNECTED TOOLS	Apollo, GitHub, Gmail, Google Calendar, Google Drive, Linear, Neon, Slack, Trigify, Vercel, in all three arms.
UNABYSS SOURCES	LinkedIn, website, X, Slack, GitHub, Linear, Gmail, Google Drive, Google Calendar.
PREPARATION	The CLAUDE.md file for arm B was written immediately before the test, covering the same ground as the context layer.
SCORING	The output we accepted and stopped on, judged against the scale above.
REPEATABILITY	Three tasks (T9, T16, T17) were run a second time in all three setups. Individual figures moved, in one case by half, but the ordering did not: Unabyss produced the best output in all six task-runs and the lowest cost in five of six, and eight of the nine quality scores landed within a point of the first round. Treat the percentages as approximate and the direction as settled.

What we take from it

The headline numbers are good, but the shape underneath them matters more. Context is worth nothing on work that does not need it, and worth a great deal on work that does. Most real work is the second kind. The tasks that improved were not exotic. They were catching up on the week, briefing for a call, drafting an email, working out what to do tomorrow.

We also ran three of the tasks a second time to see how much a single run wobbles. Costs moved, but the ranking did not: Unabyss came out ahead on quality in every repeated task, in both rounds.

The other finding is the one we did not expect to be so clean. A static context file, written carefully and kept current, is a real improvement over nothing. It is also a snapshot taken the moment you save it. Context is worth something when it is written down. It is worth a great deal when it is structured, and keeps itself current.