

// SECTION 01 • FIELD GUIDE

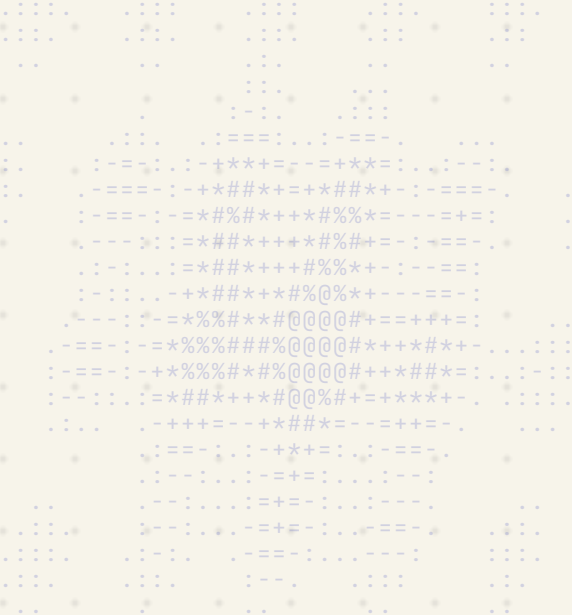
5 CLAUDE CONNECTORS



The five most powerful connectors you can plug into Claude. PostHog, Perplexity, Firecrawl, Playwright, and Composio. What each one unlocks, the exact install, and the prompt to run first.

```
> boot opusjake_os
> resource: five-connectors
> version: v1.0 • 2026
> status: ready_
```

Claude on its own can think. Claude with the right five connectors can see your product data, research the live internet, read any website, drive a real browser, and act inside 1,500+ apps. Here is the stack, the install for each, and the first prompt to run.



THE STACK

#	CONNECTOR	WHAT IT GIVES CLAUDE
01	PostHog	Your product data: analytics, errors, replays, flags
02	Perplexity	Live, cited answers from the web
03	Firecrawl	Any website as clean markdown
04	Playwright	A real browser it can click and type in
05	Composio	1,500+ apps through one connection

// TWO WAYS TO INSTALL

- **Claude app (web + desktop):** composer + → **Connectors** → **Browse connectors** for official ones, or **Add custom connector** and paste a server URL.
- **Claude Code:** one `claude mcp add` line in the terminal, then `/mcp` to sign in.

Every install below gives you the fastest path first.

01 / POSTHOG

The one that changes how you run a product. PostHog holds your analytics, session replays, error tracking, feature flags, experiments, and surveys. Connect it and Claude stops guessing about your users and starts reading what they actually did.

Get it here: posthog.com/mcp

// INSTALL

PostHog is an official connector in Claude's directory. In the Claude app: **+** → **Connectors** → **Browse connectors** → **PostHog** → **Connect**, then sign in. Done.

Claude Code, one line:

```
claude mcp add --transport http posthog \  
https://mcp.posthog.com/mcp -s user
```

Or let the wizard wire it into every AI tool you use at once:

```
npx @posthog/wizard mcp add
```

The server is hosted and free, and the login routes to your US or EU region automatically.

// WHAT IT CAN DO

Over 1,000 tools. The ones you will use first:

- `query-trends`, `query-funnel`, `query-retention` for any analytics question in plain English.
- `query-error-tracking-issues-list` and `query-error-tracking-issue` to find and read crashes.
- `query-session-recordings-list` to pull the replays behind a problem.
- `create-feature-flag` and `feature-flag-roll-out-to-everyone` to ship and roll back.
- `experiment-create` and `experiment-results-get` to run A/B tests from chat.
- `execute-sql` when you want the raw numbers.

// RUN THIS FIRST

Look at my PostHog project for the last 30 days.

1. Where is the biggest drop-off in my signup funnel?
2. What are the top 5 errors this week, and how many users did each one hit?
3. Pull 3 session recordings of users who hit the worst error and tell me what they were trying to do.

Then give me the one fix that would move signups the most.

In Claude Code it goes one step further: show it the stack trace, and it opens the file and proposes the fix.

02 / PERPLEXITY

The answer layer. It searches, reads, and hands Claude a written answer with citations attached, so Claude works from today's internet instead of its training data.

// INSTALL

Claude Code, one line, then `/mcp` to sign in:

```
claude mcp add --transport http perplexity \  
https://api.perplexity.ai/mcp
```

In the Claude app it goes in as a custom connector: **+** → **Connectors** → **Add custom connector**, name it Perplexity, URL `https://api.perplexity.ai/mcp`.

// PICK THE RIGHT TOOL

- `perplexity_search` returns ranked links. Cheapest.
- `perplexity_ask` is a fast answer with live search. The default.
- `perplexity_reason` for problems that need logic more than lookups.
- `perplexity_research` runs deep and returns a cited report. Slow, worth it.

Name the tool in your prompt, or Claude reaches for `ask` every time.

// RUN THIS FIRST

```
Use perplexity_research on [TOPIC].
```

```
What changed in the last 90 days, who are the three most  
credible sources, and what is still unsettled? Cite every  
claim. If sources disagree, show me the disagreement.
```

03 / FIRECRAWL

The extraction layer. Point it at any URL and Claude gets clean markdown instead of nav bars and cookie banners. Point it at a domain and it gets every page.

// INSTALL

Firecrawl is an official connector. In the Claude app: + → **Connectors** → **Browse connectors** → **Firecrawl** → sign in.

Claude Code:

```
claude mcp add --transport http firecrawl \  
https://mcp.firecrawl.dev/v2/mcp-oauth
```

Want to try it with no account? <https://mcp.firecrawl.dev/v2/mcp> runs scrape, search, and parse keyless.

// THE TOOLS THAT MATTER

- `firecrawl_scrape` turns one URL into markdown. Add `"formats": ["branding"]` and it pulls a site's colors, fonts, and logo.
- `firecrawl_map` lists every URL on a domain. Run it before any crawl.
- `firecrawl_crawl` walks a whole site.
- `firecrawl_parse` reads PDFs, DOCX, and XLSX.
- `firecrawl_monitor_create` watches a page and reports changes.

// RUN THIS FIRST

Use `firecrawl_map` on [COMPETITOR DOMAIN] and show me the URL list before you scrape anything.

Then scrape the homepage, pricing page, and last 5 blog posts. Give me one table: their main claim, their price, and who they say it is for.

04 / PLAYWRIGHT

The hands. It runs a real browser on your machine, so Claude can click, type, log in, and test the pages the other connectors cannot reach.

// INSTALL

It runs locally, so it lives in Claude Code. One line, no key, no account:

```
claude mcp add playwright npx @playwright/mcp@latest
```

Microsoft also ships it as an official Claude Code plugin. Same server, installed from the plugin directory.

// THE TOOLS

- `browser_navigate` opens a page.
- `browser_snapshot` reads it as a structured element tree. This is why it clicks the real button, not a pixel guess.
- `browser_click`, `browser_type`, `browser_fill_form` do the work.
- `browser_take_screenshot` shows you what it sees.

Useful flags: `--isolated` for a throwaway profile, `--headless` to hide the window.

// RUN THIS FIRST

Open [YOUR SITE] in the browser and go through the signup flow like a new user. Snapshot after every step.

Tell me every place you got stuck, every broken link, and every button that did nothing. Stop before any payment or final submit button.

05 / COMPOSIO

The multiplier. One connector, 1,500+ apps behind it: Gmail, Slack, Notion, Sheets, Linear, HubSpot, GitHub, and the long tail nobody else covers.

// INSTALL

In the Claude app: + → **Connectors** → **Add custom connector**. Name it Composio, paste:

```
https://connect.composio.dev/mcp
```

Click **Add** and sign in. Each app asks for its own login the first time Claude uses it.

Claude Code:

```
curl -fsSL https://composio.dev/install | sh
composio login
composio setup --target claude
```

// HOW IT WORKS

Instead of 1,500 tools flooding Claude's context, Composio exposes a handful of meta-tools. `COMPOSIO_SEARCH_TOOLS` finds the right action, `COMPOSIO_MANAGE_CONNECTIONS` handles the login, and `COMPOSIO_MULTI_EXECUTE_TOOL` runs up to 50 actions in parallel.

// RUN THIS FIRST

```
Every morning I want a brief. Check my Gmail for anything from
a customer in the last 24 hours, check Linear for issues
assigned to me, and post a 5-line summary to my #daily Slack
channel. Show me the draft before you post.
```

STACK THEM

Each one alone is useful. Together they close the loop from data to shipped fix.

1. PostHog: find the step in my signup funnel with the worst drop-off this month, and pull 3 replays of users who quit.
2. Perplexity: research how the top 3 products in my category handle that same step. Cite sources.
3. Firecrawl: scrape those 3 signup pages so we can compare copy and fields side by side.
4. Playwright: walk through my own signup and confirm the problem happens.
5. Composio: write it up as a Linear issue with the fix and send the summary to Slack.

// BEFORE YOU CONNECT ANYTHING

- **Start with PostHog.** Your own data beats anything on the internet. [Connect it here.](#)
- **Name the tool in the prompt.** It is faster and cheaper than letting Claude guess.
- **Never put API keys in URLs.** Headers or OAuth only.
- **Stop before submit.** Payments, account creation, and anything irreversible stay on your hands.
- **Only connect servers you trust.** A connector can feed instructions to Claude through what it returns.

Follow [opusjake.ai](#) for the rest of the stack.