

■ // SELF-DRIVING WEBSITE · LEVEL-GUIDE

Die **Self-Driving** Website

Fünf Level — vom statischen Deploy bis zur Seite, die aus jedem Besucher lernt. Mit den Befehlen, die du heute kopieren kannst.

TRACKING

A/B-TESTS

CLAUDE CODE

POSTHOG

SELF-DRIVING

Aus dem Reel kennst du die fünf Level. Hier steht, wie du jedes Level konkret erreichst:
Level 2 kostet dich rund 20 Minuten, Level 4 einen einzigen Plugin-Befehl. Die Basis ist PostHog (kostenlos bis 1 Mio. Events im Monat) plus Claude Code.

1 Statisch — der Ist-Zustand

Deine Seite ist deployt, und seitdem passiert nichts mehr. Die ehrliche Standortbestimmung: Wann hast du zuletzt etwas an der Seite geändert? Und woher wusstest du, dass es die richtige Änderung war? Wenn die Antwort „Bauchgefühl“ lautet, bist du auf Level 1. Da starten fast alle, und jedes weitere Level baut direkt auf dem vorherigen auf.

2 Tracking — du siehst, was Besucher wirklich tun

Setup (ca. 20 Minuten):

- 1 PostHog-Account anlegen (`posthog.com` — EU-Cloud verfügbar, kostenlos bis 1 Mio. Events/Monat)
- 2 Im Projektordner: `npx @posthog/wizard@latest` — der Wizard erkennt dein Framework (Next.js, React, Astro ...) und baut das Tracking selbst ein
- 3 In PostHog **Session Replays** und **Web Analytics** aktivieren

TOOL	FRAGE, DIE ES BEANTWORTET
Web Analytics	Woher kommen Besucher, auf welcher Seite steigen sie aus?
Session Replays	Schau dir 10 echte Aufnahmen an: Wo scrollen Leute vorbei, wo hängen sie?
Toolbar-Heatmap	Welche Buttons werden geklickt, und welche nie?

ZUM MITNEHMEN

10 Session Replays sagen dir mehr über deine Seite als 10 Meinungen aus deinem Umfeld.

3 A/B-Test — zwei Varianten, die Daten entscheiden

In PostHog unter **Experiments** legst du einen Test an: ein Feature Flag mit `control` und `test`, dazu ein Ziel-Event (z. B. `cta_click`). Im Code brauchst du nur ein `getFeatureFlag('hero-headline')` für die Variante und ein `posthog.capture('cta_click')` für das Ziel.

Drei Regeln, damit der Test etwas taugt:

- 1 **Eine Änderung pro Test.** Headline oder CTA, sonst weißt du am Ende nicht, was gewirkt hat.
- 2 **Ziel-Event vor dem Start festlegen.** PostHog rechnet die Signifikanz für dich aus.
- 3 **Traffic ehrlich einschätzen.** Unter ~1.000 Besuchern im Monat dauern Tests mehrere Wochen. Teste dann große Hebel wie Headline oder Angebot. Button-Farben lohnen sich erst bei deutlich mehr Traffic.

4 KI-Agent — Claude Code testet für dich

Das ist der eigentliche Sprung. Ein Befehl: `claude plugin install posthog`

Nach dem OAuth-Login hat Claude Code über 27 PostHog-Tools und Slash-Commands wie `/posthog:experiments`. Der Agent liest deine Analytics, baut Varianten im Code, legt Experimente an und wertet sie aus. Du gibst nur noch die Freigabe.

Drei Prompts zum Kopieren (mein Workflow):

- 1 „Schau dir in PostHog die Web Analytics und die letzten 20 Session Replays der Landingpage an. Wo verlieren wir Besucher? Gib mir deine Top-3-Hypothesen.“
- 2 „Nimm Hypothese 1, bau eine Variante B im Code und leg dafür ein PostHog-Experiment mit Ziel-Event `cta_click` an.“
- 3 (7–14 Tage später) „Check das Experiment: Ist das Ergebnis signifikant? Wenn ja, shipp den Gewinner und räum das Feature Flag auf.“

Ohne Claude Code geht derselbe Workflow über den PostHog MCP-Server (`mcp.posthog.com`) in Cursor, Windsurf oder VS Code.

5 Die Seite verbessert sich von selbst

Weg A — PostHog Self-Driving (Open Beta, kostenlos startbar): `npx @posthog/wizard self-driving`

So läuft der Loop: Scouts überwachen deine Analytics, Session Replays und Error Trackings rund um die Uhr. Ähnliche Signale werden zu Reports gebündelt. Ein Claude-Agent prüft jeden Report gegen deine Codebase und öffnet bei lösbaren Problemen automatisch einen Pull Request. Du reviewst und mergst. Danach misst PostHog, ob sich die Ziel-Metrik bewegt hat, und das Ergebnis fließt als neues Signal in die nächste Runde.

Du bleibst dabei das Gate: Ohne dein Review wird nichts gemergt. PostHog nutzt das System selbst — im eigenen Repo laufen bereits so entstandene, gemergte PRs (z. B. `PostHog/posthog#68818`).

Weg B — Personalisierung pro Besucher (Fertigprodukte): Tools wie **Coframe** generieren laufend UI- und Copy-Varianten, testen sie an echten Besuchern und zeigen jedem Visitor die Version, die bei ihm am wahrscheinlichsten funktioniert. Ähnlich arbeiten **Webflow Optimize** (früher Intellimize) und **Mutiny**. Diese Produkte richten sich an Seiten mit ordentlich Traffic und B2B-Budgets. Für den Einstieg ist Weg A der realistischere.

■ // SO FÄNGST DU AN

Deine Checkliste

Sechs Schritte, in der Reihenfolge der Level.

- ✓ PostHog-Account + `npx @posthog/wizard@latest` (ca. 20 Min)
- ✓ **Session Replays** aktivieren und 10 Aufnahmen anschauen
- ✓ Ersten **A/B-Test** mit einem Ziel-Event anlegen
- ✓ `claude plugin install posthog` + die 3 Prompts durchspielen
- ✓ `npx @posthog/wizard self-driving` für die Beta aktivieren
- ✓ Ab jetzt: **jede Woche ein Test**, Gewinner shippen



Links

WAS

WO

Self-Driving Docs

posthog.com/docs/self-driving/self-improving-loop

Self-Driving Setup

posthog.com/docs/self-driving/setup

Claude Plugin

claude.com/plugins/posthog

Experimente per MCP

posthog.com/docs/experiments/create-experiments-mcp

Hintergrund-Story

posthog.com/blog/what-if-your-product-built-itself

■ // BLEIB DRAN

Fang mit Level 2 an — diese Woche.

20 Minuten für das Tracking, und ab dann entscheiden Daten statt Bauchgefühl. Ich teile solche Setups regelmäßig. Folg mir für KI-Tools, die für dich arbeiten.

[@ki.borish](https://twitter.com/ki_borish)