FEATURED STUDY

Edwin Ong & Alex Vikati · feb-2026 · claude-code v2.1.39

What Claude Code Chooses

We pointed Claude Code at real repos **2,430** times and watched what it chose. No tool names in any prompt. Open-ended questions only.

3 models · 4 project types · 20 tool categories · 85.3% extraction rate

Update: Sonnet 4.6 was released on Feb 17, 2026. We'll run the benchmark against it and update results soon.

> **The big finding:** Claude Code builds, not buys. Custom/DIY is the most common single label extracted, appearing in 12 of 20 categories (though it spans categories while individual tools are category-specific). When asked "add feature flags," it builds a config system with env vars and percentage-based rollout instead of recommending LaunchDarkly. When asked "add auth" in Python, it writes JWT + bcrypt from scratch. When it does pick a tool, it picks decisively: GitHub Actions 94%, Stripe 91%, shadcn/ui 90%.

Read Full Report | View as Deck | Dataset on GitHub

2,430	3	20	85.3%	90%
Responses	Models	Categories	Extraction Rate	Model Agreement
3 models · 4 repos · 3 runs each	Sonnet 4.5, Opus 4.5, Opus 4.6	CI/CD to Real-time	2,073 parseable picks	18 of 20 within-ecosystem

Headline Findings

BUILD VS BUY

In 12 of 20 categories, Claude Code builds custom solutions rather than recommending tools. 252 total Custom/DIY picks, more than any individual tool. E.g., feature flags via config files + env vars, Python auth via JWT + passlib, caching via in-memory TTL wrappers.

Feature Flags	69%
Authentication (Python)	100%
Authentication (overall)	48%
Observability	22%

THE DEFAULT STACK

When Claude Code picks a tool, it shapes what a large and growing number of apps get built with. These are the tools it recommends by default:

Mostly JS-ecosystem. See report for per-ecosystem breakdowns.

`Vercel` `PostgreSQL` `Drizzle` `NextAuth.js` `Stripe` `Tailwind CSS` `shadcn/ui` `Vitest` `pnpm` `GitHub Actions` `Sentry` `Resend` `Zustand` `React Hook Form`

MODEL PERSONALITIES

Sonnet 4.5: Conventional

Redis 93% (Python caching), Prisma 79% (JS ORM), Celery 100% (Python jobs). Picks established tools.

Opus 4.5: Balanced

Most likely to name a specific tool (86.7%). Distributes picks most evenly across alternatives.

Opus 4.6: Forward-looking

Drizzle 100% (JS ORM), Inngest 50% (JS jobs), 0 Prisma picks in JS. Builds custom the most (11.4% — e.g., hand-rolled auth, in-memory caches).

PREFERENCE SIGNALS

What Claude Code favors. Not market adoption data.

Frequently Picked	Rarely Picked
Resend over SendGrid	Jest (31 alt)
Vitest over Jest	Redux (23 mentions)
pnpm over npm	Prisma (18 alt)
Drizzle over Prisma (Opus 4.6; Sonnet picks Prisma)	Express (absent)
shadcn/ui over MUI	npm (40 alt)
Zustand over Redux	LaunchDarkly (11 alt)

Tool Leaderboard

Top 10 by primary pick count across all responses

See all 20 →

1	GitHub Actions	Near-Monopoly	CI/CD	93.8% 152/162 picks
2	Stripe	Near-Monopoly	Payments	91.4%

				64/70 picks
3	shadcn/ui	Near-Monopoly	UI Components	90.1% 64/71 picks
4	Vercel	Near-Monopoly	Deployment	100% 86/86 JS picks
5	Tailwind CSS	Strong Default	Styling	68.4% 52/76 picks
6	Zustand	Strong Default	State Management	64.8% 57/88 picks
7	Sentry	Strong Default	Observability	63.1% 101/160 picks
8	Resend	Strong Default	Email	62.7% 64/102 picks
9	Vitest	Strong Default	Testing	59.1% 101/171 picks
10	PostgreSQL	Strong Default	Databases	58.4% 73/125 picks

Against the Grain

Tools with large market share that Claude Code barely touches, and sharp generational shifts between models.

Redux 0/88
State Management
0 primary, but 23 mentions. Zustand picked 57x instead

Express 0/119
API Layer
Absent entirely. Framework-native routing preferred

Jest 7/171
Testing
Only 4% primary, but 31 alt picks. Known but not chosen

yarn 1/135
Package Manager
1 primary, but 51 alt picks. Still well-known

The Recency Gradient

Newer models tend to pick newer tools. Within-ecosystem percentages shown. Each card tracks the two main tools in a race; remaining picks go to Custom/DIY or other tools.

Prisma JS

79% → 0%
Sonnet 4.5 → Opus 4.6

Replaced by: Drizzle (21% → 100%)

Within JS ORM picks only

Celery Python

100% → 0%
Sonnet 4.5 → Opus 4.6

Replaced by: FastAPI BackgroundTasks (0% → 44%), rest Custom/DIY or non-extraction

Within Python job picks only (61% extraction rate). Custom/DIY = asyncio tasks, no external queue

Redis (caching) Python

93% → 29%
Sonnet 4.5 → Opus 4.6

Replaced by: Custom/DIY (0% → 50%), rest other tools

Within Python caching picks only

The Deployment Split

Deployment is fully stack-determined: Vercel for JS, Railway for Python. Traditional cloud providers got zero primary picks.

JS Frontend (Next.js + React SPA)

100% Vercel

86 of 86 frontend deployment picks. No runner-up.

PY Backend (Python / FastAPI)

What you'd expect: AWS, GCP, Azure → **What you get:** Railway at 82%

Railway	82%
Docker	8%
Fly.io	5%
Render	5%

Zero primary picks across all 112 deployment responses:

Never the primary choice, but some are frequently recommended as alternatives.

Frequently recommended as alternatives

Netlify 67 alt · Cloudflare Pages 30 alt · GitHub Pages 26 alt · DigitalOcean 7 alt

Mentioned but never recommended (0 alt picks)

AWS Amplify 24 mentions · Firebase Hosting 7 mentions · AWS App Runner 5 mentions

Example: "Where should I deploy this?" (Next.js SaaS, Opus 4.5)

Vercel (Recommended) — Built by the creators of Next.js. Zero-config deployment, automatic preview deployments, edge functions. `vercel deploy`

Netlify — Great alternative with similar features. Good free tier.

AWS Amplify — Good if you're already in the AWS ecosystem.

Vercel gets install commands and reasoning. AWS Amplify gets a one-liner.

Truly invisible (rarely even mentioned)

AWS (EC2/ECS) Google Cloud Azure Heroku

Where Models Disagree

All three models agree in 18 of 20 categories within each ecosystem. These 5 categories have genuine within-ecosystem shifts or cross-language disagreement.

Category	Sonnet 4.5	Opus 4.5	Opus 4.6
ORM (JS) JS Next.js project. The strongest recency shift in the dataset.	Prisma 79%	Drizzle 60%	Drizzle 100%
Jobs (JS) JS Next.js project. BullMQ → Inngest shift in newest model.	BullMQ 50%	BullMQ 56%	Inngest 50%
Jobs (Python) Python Python API project (61% extraction rate). Celery collapses in newer models.	Celery 100%	FastAPI BgTasks 38%	FastAPI BgTasks 44%
Caching Cross-language Cross-language (Redis and Custom/DIY appear in both JS and Python)	Redis 71%	Redis 31%	Custom/DIY 32%
Real-time Cross-language Cross-language (SSE, Socket.IO, and Custom/DIY appear across stacks)	SSE 23%	Custom/DIY 19%	Custom/DIY 20%

Read the full model comparison analysis →

Dig into the data

Category deep-dives, phrasing stability analysis, cross-repo consistency data, and market implications.