Claude Code

راهنمای عملی Claude Code برای استارتاپ‌ها

۵ قانونی که استارتاپ‌ها برای عرضه محصولاتی فراتر از انتظار استفاده می‌کنند

همراه با بینش‌ها و تجربیاتی از بنیان‌گذاران:

- Artemis Security
- Cainex
- Clay
- ClickHouse
- Cognition
- Commure
- Crosby
- Emergent
- Harvey
- Heidi
- Higgsfield
- Omni
- Parahelp
- Translucent
- Zingage

استارتاپ‌های مبتنی بر هوش مصنوعی (AI Natives) در خط مقدم

اگر می‌خواهید نگاهی به آینده مشاغل بیندازید، از استارتاپ‌ها بپرسید که امروز چگونه کار می‌کنند.

آنتروپیک با بیش از دوازده استارتاپی که به‌سرعت درحال‌رشد هستند گفت‌وگو کرد تا دریابد چگونه از ابزارهای کدنویسی عامل‌محور (Agentic Coding) برای ساخت محصولات و مقیاس‌دهی سازمانی خود استفاده می‌کنند. این استارتاپ‌ها در حال تغییر قواعدی هستند که چه کسی می‌تواند چیزی بسازد، چه چیزهایی کنار گذاشته می‌شوند و چگونه می‌توان میان شیوه ساخت محصول و خود محصول یک چرخه تقویتی ایجاد کرد.

و آن‌ها با سرعتی محصول عرضه می‌کنند که گویی سازمان‌هایی با ۱۰ برابر اندازه واقعی خود هستند.

ClickHouse عرضه قابلیت‌های جدید خود را ۳۰ درصد افزایش داده است.

Omni بهره‌وری تیم مهندسی را ۲ تا ۳ برابر کرده است.

Clay، ۱۰۰ درصد فرایند تریاژ باگ‌ها را خودکار کرده است.

Artemis Security هر هفته بیش از ۶۰۰۰ در خواست ادغام (Pull Request) عرضه می‌کند.

آنتروپیک در این راهنما، به نحوه استقرار منحصربه‌فرد Claude Code در این سازمان‌ها می‌پردازد تا قواعدی که برای عرضه سریع محصولات و حفظ مزیت رقابتی خود دنبال می‌کنند را بررسی کند.

در این مسیر، به‌تدریج به دنبال پاسخ این پرسش نیز خواهیم بود: اگر یک سازمان چرخه عمر توسعه محصول خود را از ابتدا و بر پایه Claude Code طراحی می‌کرد، این فرایند چگونه به نظر می‌رسید؟

نکته: اگر فقط به گام‌های عملی علاقه‌مند هستید، در انتهای این راهنما یک چک‌لیست قرار داده شده که نکات فنی کلیدی مطرح‌شده در هر فصل را یکجا جمع‌بندی می‌کند.

۱

همه درست‌اندرکار هستند

کدنویسی عامل‌محور، موانع ورود را کاهش می‌دهد؛ بنابراین فردی که مسئله را به‌خوبی درک می‌کند، می‌تواند نخستین نسخه راه‌حل را عرضه کند.

همه دست‌اندرکار هستند

کدنویسی عامل‌محور، آستانه و موانع ورود کارکنان غیرفنی به فرایند ساخت محصول را کاهش می‌دهد. با Claude Code می‌توانید فیچرها و قابلیت‌های کاربردی ایجاد کنید، بدون اینکه در یک زبان برنامه‌نویسی یا نحوه کار با یک محیط توسعه یکپارچه (IDE) مهارت داشته باشید.

Mads Lunau Liechti
Parahelp

همان‌طور که «**مدس لونائو لیچتی»، هم‌بنیان‌گذار Parahelp** می‌گوید: «نه‌تنها مهندسان می‌توانستند بسیار بیشتر در جریان کار باشند، بلکه افراد غیرفنی (مثل خود من) نیز ناگهان می‌توانستند در تغییرات رابط کاربری و سایر بهبودهای محصول دست داشته باشند.»

برای بنیان‌گذاران استارتاپ‌ها، این موضوع مزایای آشکاری دارد. نخست اینکه آن‌ها در مقایسه با رقبای بزرگ‌تر خود نیروی انسانی کمتری دارند؛ بنابراین «همه باید وارد میدان شوند و دستی در کار داشته باشند.» اما هدف بنیان‌گذاران صرفاً افزایش ظرفیت کاری نیست؛ اعضای غیرفنی تیم، تخصص حوزه‌ای را نیز با خود به همراه می‌آورند.

Ryan Daniels
Crosby

به گفته **«رایان دنیلز»، هم‌بنیان‌گذار و مدیرعامل Crosby**: «Claude Code معنای وکیل بودن در Crosby را تغییر داد. وکلا بهترین بینش‌ها را درباره محصول دارند، چون خودشان کاربران محصول هستند. تماشای شیوه کارکردن آن‌ها فوق‌العاده است.»

Dr. Thomas Kelly
Heidi

«توماس کلی»، هم‌بنیان‌گذار و مدیرعامل Heidi نیز بر همین نکته تأکید داشتو گفت: «Claude Code برای ما، «مسئله تلفن خراب» را حل کرد (Broken Telephone Problem یا مسئله تلفن خراب یک مسئله کلاسیک در نظریه گراف و بهینه‌سازی شبکه است.) پیش‌تر، یک ایده جدید این‌گونه در تیم حرکت می‌کرد: فرد صاحب ایده،ایده خود را به مدیر محصول می‌گفت، مدیر محصول به طراح می‌گفت، طراح به مهندس می‌گفت... و در نهایت، ذات و جوهره ایده در این زنجیره از بین می‌رفت.

زمانی که چیزی عرضه می‌شد، اغلب دیگر به چیزی که نفر اول و ایده‌پرداز در ذهن داشت، شباهتی نداشت. ضمن اینکه این فرایند هفته‌ها طول می‌کشید. Claude Code این زنجیره را از میان برمی‌دارد. فردی که واقعاً مسئله را درک می‌کند، می‌تواند یک PR عرضه کند و در بخش‌هایی که به تخصص طراحان و مهندسان نیاز دارد، از آن‌ها کمک بگیرد.»

گفتن اینکه «همه دست‌اندرکار هستند» برای یک پست لینکدین جمله بسیار خوبی است، اما این رویکرد در عمل چگونه کار می‌کند؟ آیا تیم بازاریابی درخواست‌های ادغام را تأیید می‌کند؟ آیا تیم حقوقی روی پیچیدگی‌های تجزیه‌وتحلیل تست‌های ناپایدار و مشکوک کار می‌کند؟

پاسخی که دریافت شد این است که همچنان تقسیم کار وجود دارد. کارشناسان بازاری همچنان بر بازاریابی تمرکز می‌کنند و توسعه‌دهندگان نیز همچنان بر توسعه تمرکز دارند. اما مرحله بسیار مهم نخست، یعنی تبدیل یک ایده به یک نمونه اولیه قابل‌اجرا و حرکت از صفر به یک (0→1)، برای همه آزاد و امکان‌پذیر است.

همچنین مشاهده شد که مؤثرترین استارتاپ‌ها سازوکارهایی ایجاد کرده‌اند تا این مشارکت‌ها سیستماتیک باشند، نه اینکه به شانس یا جاه‌طلبی فردی وابسته بمانند.

ایجاد ارتباط

انتظار داشتن از کارکنان برای استفاده از هوش مصنوعی یک موضوع است و فراهم‌کردن دسترسی آن‌ها به Claude Code و ابزارهای موردنیازشان موضوعی دیگر.

Kareem Amin
Clay

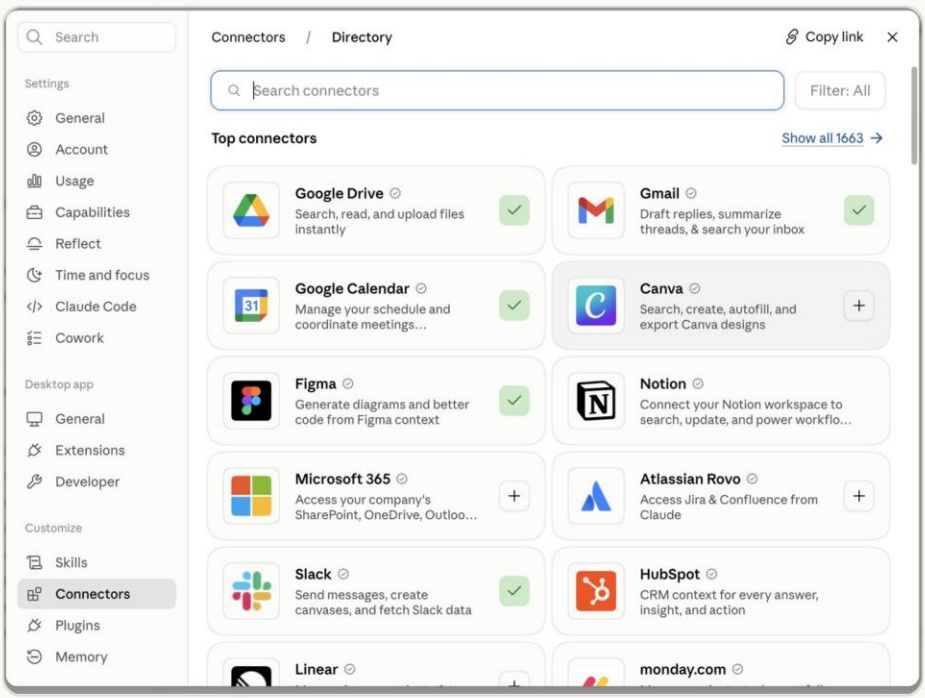
«کریم امین»، هم‌بنیان‌گذار و مدیرعامل Clay می‌گوید: «ما در واقع از مشارکت کارکنان غیرفنی خودداری نمی‌کنیم؛ بلکه به سمت آن حرکت می‌کنیم. دیدگاه ما این است که هر نقشی در حال تبدیل‌شدن به یک نقش مهندسی است، چون می‌توانید برای آن نرم‌افزار بسازید... بنابراین افرادی را استخدام می‌کنیم که اهل دست‌کاری و آزمون‌وخطا هستند و به ساختن علاقه دارند.»

در Crosby، تیم به‌جای اینکه وکلا را به سراغ Claude Code بفرستد، Claude Code را به سراغ وکلا فرستاد؛ به این صورت که آن را به ابزارها و سیستم‌های عملیاتی‌ای متصل کرد که وکلا با آن‌ها آشنا بودند و هر روز با آن‌ها کار می‌کردند.

نکته: Claude نمی‌تواند چیزی را که نمی‌بیند درک کند. یکی از مؤثرترین راه‌ها برای خلق ارزش با Claude، اتصال آن به «منابع حقیقت» (Sources of Truth) و ابزارهایی است که تیم شما هر روز از آن‌ها استفاده می‌کند.

«پروتکل زمینه مدل» یا همان MCP، یک استاندارد متن‌باز برای یکپارچه‌سازی ابزارهای هوش مصنوعی است که به Claude Code امکان دسترسی به ابزارها، پایگاه‌های داده و APIهای شما را می‌دهد. هر زمان که تیم شما متوجه شد در حال کپی پیست کردن اطلاعات از یک ابزار به Claude است، اضافه‌کردن این کانکتورها را بررسی کنید.

وقتی یک ابزار خط فرمان حرفه‌ای (مانند gh، kubectl، bq، psql)، از قبل در سازمان وجود دارد و می‌خواهید Claude بر اساس همان منابعی کار کند که مهندسان شما از آن استفاده می‌کنند، اتصال از طریق واسط خط فرمان یا همان CLI می‌تواند از نظر مصرف توکن کارآمدتر باشد.



دایرکتوری کانکتورهای MCP در نسخه دسکتاپ Claude Code

ارائه‌های سرپایی

در مقطعی، ایده‌ها باید فرصتی برای اولویت‌بندی پیدا کنند تا منابع سازمان بتوانند به رساندن آن‌ها به بازار کمک کنند. این مسیر برای مدیران محصول کاملاً روشن است (بالاخره کارشان همین است) اما برای کارکنان غیرفنی به این اندازه روشن نیست.

Clay جلسات بررسی فصلی برگزار می‌کند که در آن نمونه‌های اولیه و پروتوتایپ‌ها بررسی می‌شوند و می‌توانند وارد نقشه راه رسمی محصول شوند.

برای مثال، یکی از اعضای تیم ورود به بازار (Go-to-Market) در Clay به این شیوه یک عامل خودمختار ساخت که از وب‌سایت‌ها بازدید می‌کند، فرم‌های جمع‌آوری سرنخ را پر می‌کند، مدت‌زمان پاسخ‌گویی را اندازه‌گیری می‌کند، تجربه کاربری را امتیازدهی می‌کند و یک گزارش عملکرد تولید می‌کند.

Omni یک کانال اختصاصی در Slack برای نمونه‌های اولیه تولیدشده با Claude دارد که همه افراد، از جمله کارکنان ارشد فنی می‌توانند در آن مشارکت کنند. آن‌ها همچنین اصل مکمل «همه دست‌اندرکار هستند» را اجرا می‌کنند که عبارت است از: «همه با مشتریان صحبت می‌کنند.»

Chris Merrick
Omni

همان‌طور که «کریس مریک»، هم‌بنیان‌گذار و مدیر فناوری Omni عنوان می‌کند اگرچه مهندسان طبعاً تمایلی به تماس با مشتریان ندارند، Omni عمداً آن‌ها را در تعامل مستقیم با مشتریان قرار می‌دهد، زیرا این کار حلقه بازخورد را سریع‌تر می‌کند.

اشتراک‌گذاری مهارت‌ها

مرز میان «همه دست‌اندرکار هستند» و «کارهای تکه‌تکه و پراکنده» می‌تواند بسیار باریک باشد. نمونه‌های اولیه قابلیت‌ها، فارغ از اینکه چه کسی آن‌ها را ساخته، همچنان باید در محصولی یکپارچه شوند که منسجم و یکپارچه به نظر برسد. اینجاست که «مهارت‌ها» یا همان Skills، یعنی فایل‌های دستورالعمل قابل‌استفاده‌مجدد که استانداردها و زمینه (Context) تیم شما را در خود رمزگذاری می‌کنند، می‌توانند کمک کنند تا توسعه محصول، حتی با دموکراتیک‌ترشدن فرایند، همچنان هم‌راستا باقی بماند.

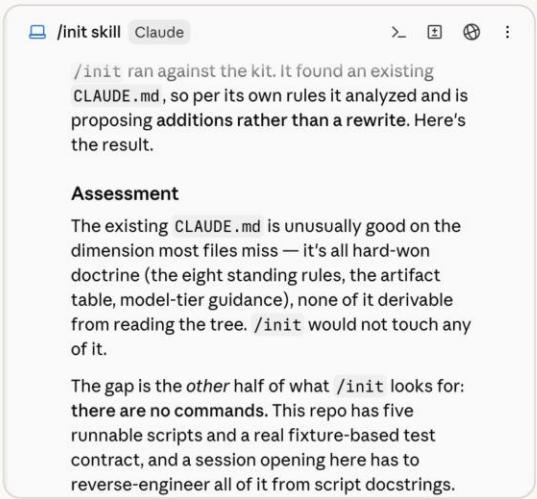
توماس کلی از Heidi می‌گوید: «هر فردی در تیم می‌تواند با استفاده از Claude Code و با ارجاع به سیستم طراحی ما، مؤلفه‌های محصول، محتوای بازاریابی یا مطالب ارائه را تهیه کند. هوش مصنوعی‌ای که با محصول تعامل دارد باید استاندارد بسیار بالاتری را رعایت کند و Claude Code به ما کمک می‌کند این استاندارد را با دقت بیشتری محقق کنیم.»

Mukund Jha
Emergent

این رویکرد همچنین می‌تواند به آماده‌سازی و ورود سریع توسعه‌دهندگان جدید و کارکنان غیرفنی به فرایند کار کمک کند.
«موکوند جا»، هم‌بنیان‌گذار و مدیرعامل Emergent این رویه را به بهترین شکل توضیح می‌دهد: «... ما همچنین یک مخزن GitHub از Skillهای Claude Code داریم که به‌عنوان یک پایگاه دانش مشترک عمل می‌کند و به ما امکان می‌دهد یک جلسه Claude Code را به‌سرعت با اطلاعات شناخته‌شده درباره Emergent راه‌اندازی کنیم؛ اطلاعاتی مانند محل پایگاه‌داده و انبار داده، برخی اطلاعات مربوط به ساختار یا شِمای (Schema) پایگاه‌داده، زمینه کلی شرکت و موارد دیگر... به‌جای اینکه بخواهیم در اینجا همه‌چیز کاملاً بی‌نقص باشد، می‌توان با فایل‌های زمینه‌ای که اندکی قدیمی هستند نیز کنار آمد، مشروط بر اینکه عامل بتواند به‌سرعت آن‌ها را راستی‌آزمایی کند و مسیر خود را اصلاح کند.»

Jack O'Hara
Translucent

«جک اوهارا»، بنیان‌گذار و مدیرعامل Translucent نیز می‌گوید: «مهندسان ما از Claude Code برای راه‌اندازی یک بازار داخلی از عامل‌های تخصصی استفاده می‌کنند که بر اساس نقش سازمانی دسته‌بندی شده‌اند؛ بنابراین تیم‌های مهندسی، تحویل و فروش، هرکدام ابزارهایی دریافت می‌کنند که متناسب با شیوه واقعی کار آن‌ها ساخته شده است.»



دستور /init یک Skill رایج است که یک CLAUDE.md ایجاد می‌کند و به کمک آن، Claude با دستورات ساخت، قراردادها و قالب‌بندی آشنا می‌شود.

نکته: Skillها را می‌توان از طریق یک دایرکتوری در سراسر شرکت به اشتراک گذاشت تا بهترین رویه‌های یک کارمند، بلافاصله در اختیار سایر کارمندان نیز قرار گیرد.
از فایل‌های CLAUDE.md در هر زیرشاخه از مخزن خود استفاده کنید تا قراردادها و استانداردهای کدنویسی مخصوص آن زیرشاخه را تعریف کنید؛ یعنی همان استانداردهایی که هر بار اعمال می‌شوند.
برای گردش‌کارهای رویه‌ای درخواستی از Skills استفاده کنید.
برای اطلاعات بیشتر، مطلب Steering Claude Code: when to use CLAUDE.md, skills, hooks, and subagents را مطالعه کنید.

۲

کارهای خسته‌کننده را خودکار کنید

عامل‌ها ۸۰ درصد کارهای دستی چرخه عمر توسعه را بر عهده می‌گیرند تا مهندسان وقت خود را صرف مواردی کنند که واقعاً به قضاوت انسانی نیاز دارند.

کارهای خسته‌کننده را خودکار کنید

از آغاز انقلاب صنعتی، همه شرکت‌ها به دنبال افزایش بهره‌وری از طریق فناوری بوده‌اند؛ اما این استارتاپ‌ها بودند با سرعت و عمق پذیرش فناوری خود را از دیگران متمایز کرده‌اند.

بنیان‌گذاران این استارتاپ‌ها معتقدند هوش مصنوعی جزء ضروری مأموریت شرکتشان است. بسیاری از آن‌ها صراحتاً بر این باورند که عامل‌ها ۸۰ درصد کارهای دستی را بر عهده می‌گیرند تا مهندسان بتوانند وقت خود را صرف مواردی کنند که واقعاً به قضاوت انسانی نیاز دارند.

Shachar Hirshberg
Artemis Security

به گفته **«شاچار هیرشبرگ»، هم‌بنیان‌گذار و مدیرعامل Artemis Security**: «همه در حال رقابت برای ساخت محصولات هوش مصنوعی هستند، اما تعداد بسیار کم از شرکت‌ها در حال بازطراحی شیوه واقعی اداره شرکت خود هستند؛ هر چند که مورد دومی موفقیت‌های بزرگ‌تری به همراه دارد. Artemis Security به‌عنوان یک شرکت مبتنی بر هوش مصنوعی یا اصطلاحاً AI-Native فعالیت می‌کند، نه شرکتی که صرفاً از هوش مصنوعی استفاده می‌کند. این رویکرد سرعت ما را به‌شدت افزایش می‌دهد و به ما اجازه می‌دهد به مشتریان کمک کنیم حملات سایبری سریع را متوقف کنند.»

مشخصاً مشاهده شد که هوش مصنوعی در این شرکت‌ها نسبت به سایر سازمان‌ها، یکپارچگی بسیار عمیق‌تری با مراحل چرخه عمر توسعه نرم‌افزار (SDLC) دارد و همچنین عامل‌های اختصاصی بیشتری برای انجام وظایف تکرارشونده از ابتدا تا انتها طراحی شده‌اند. بیایید چند نمونه از هر دو مورد را بررسی کنیم.

چرخه عمر توسعه نرم‌افزار مبتنی بر هوش مصنوعی (AI-Native SDLC)

بسیاری از استارتاپ‌های حاضر در این بررسی، روش‌هایی را برای تسریع ورود و آماده‌سازی تیم‌های خود برای فرایندهای کدنویسی عامل‌محور پیاده‌سازی کرده‌اند.

برای مثال، «موکوند جا» می‌گوید: «از همان روز اول، نیروی جدید با هدایت Claude به سمت فایل Markdown مناسب، کل محیط توسعه خود را راه‌اندازی می‌کند. اگر Claude در طول فرایند ورود به سازمان با چیزی خراب یا منسوخ مواجه شود، آن فایل را به‌روزرسانی می‌کند.»

نکته: قابلیت پیش‌نمایش کد یا همان Code Review که در زمان انتشار این گزارش به‌صورت پیش‌نمایش پژوهشی در دسترس است، یک سرویس مدیریت‌شده چندعاملی در Claude Code است. این قابلیت یک مرحله بررسی خودکار را روی درخواست‌های ادغام در مخازنی که فعال کرده‌اید اجرا می‌کند. می‌توانید یافته‌های آن را به‌صورت دستی اصلاح کرده و تغییرات را ادغام یا اصطلاحاً Push کنید یا با کامنت کردن عبارت «@Claude» روی همان یافته، این چرخه را به‌صورت کامل ببندید. (البته درصورتی‌که GitHub Actions را راه‌اندازی و پیکربندی کرده باشید.)

نشان‌گر	اهمیت	مفهوم
🔴	مهم	باگی که باید پیش از فرایند ادغام و مرجیگ رفع شود
🟡	اصلاح جزئی (Nit)	مشکل کوچک، ارزش حل‌کردن دارد اما توقف نه
🟣	ازپیش‌موجود	باگی که در کدبیس وجود دارد اما از طرف این درخواست ادغام معرفی نشده است

این مهندسان باید به‌سرعت وارد فرایند کار شوند، زیرا این تیم‌ها با سرعت بالایی محصول می‌سازند.

Tanay Tandon
Commure

به گفته **«تاناى تاندون»، مدیرعامل و بنیان‌گذار Commure**: «مهندسان در اینجا در حال هماهنگ‌کردن ناوگان عامل‌ها هستند، مشکلات داده‌های محیط تولید را در همان روزی که شناسایی می‌شوند برطرف می‌کنند و به طور هم‌زمان چندین PR را در دست اجرا دارند. یکی از مهندسان، یک طرح ابتکار با حدوداً ۱۳ تیکت را با استفاده از زیرعامل‌های Claude به‌صورت موازی اجرا کرد؛ به‌طوری که هرکدام مسئول یک تیکت و PR مربوط به آن بودند.»

در این سازمان‌ها، Claude Code نه‌تنها به کدنویسی کمک می‌کند، بلکه کد را نیز بررسی می‌کند. «توماس کلی» نیز دراین‌خصوص می‌گوید: «ما بررسی‌های خودکار کد را بر اساس چارچوب‌های فنی و انطباقی تأییدشده خود اجرا می‌کنیم؛ مشکلات حیاتی را علامت‌گذاری می‌کنیم و تغییرات پیشنهادی را پیش از عرضه هر چیزی، برای بررسی به افراد مناسب ارجاع می‌دهیم.»

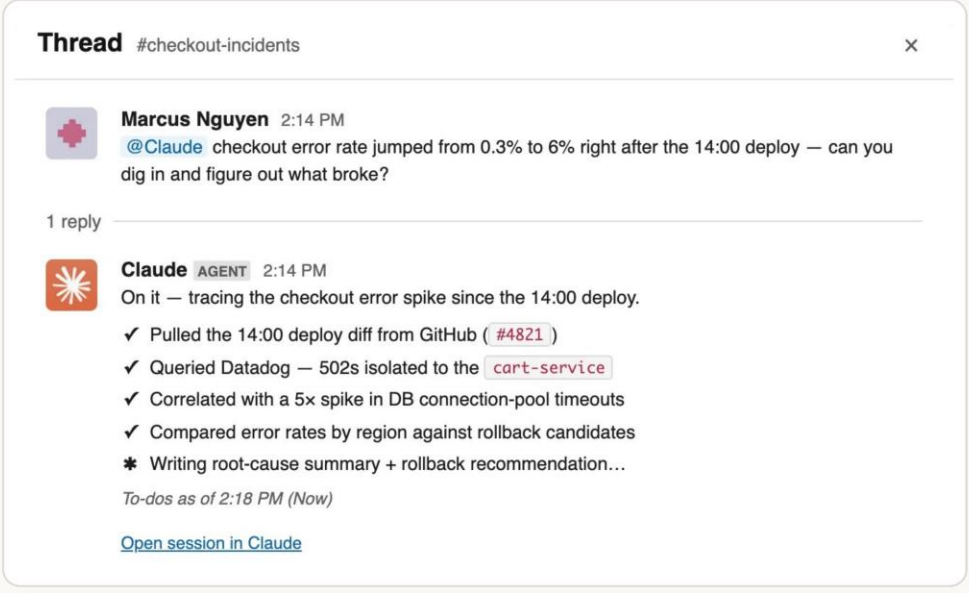
برخی از این سازمان‌ها همچنین عامل‌های سفارشی برای بازبینی کد، تست و یکپارچه‌سازی مستمر (CI) ساخته‌اند. این استارتاپ‌ها توجه زیادی به ساخت حلقه‌های بازخورد و اجرای تکرارشونده داشته‌اند، نه صرفاً استقرار کد.

«جک اوهارا» بنیان‌گذار Translucent نیز می‌گوید: «عامل موردعلاقه من، عاملی به نام «Translucent code reviewer» است که تغییرات را در چند مسیر بررسی می‌کند، آن‌ها را از زوایای مختلف ارزیابی و نتایج را به شیوه‌ای ترکیب می‌کند که گویی یکی از مهندسان ارشد ما این کار را انجام داده است؛ اما سریع‌تر از چیزی که هر فردی به‌تنهایی بتواند انجام دهد.»

«کریم امین» هم دراین‌رابطه می‌گوید: «... Clay عاملی ساخته است که تریاژ باگ‌ها را از مرحله بررسی اولیه تا پیشنهاد تغییرات کد برای رفع آن‌ها بر عهده می‌گیرد.»

نکته: در چند ماه گذشته، Claude Tag به‌عنوان نخستین پاسخ‌دهنده در حالت آماده‌باش برای خرابی‌های CI/CD در آنتروپیک فعالیت کرده است. Claude در هر رخداد جدیدی که گزارش وضعیت داشته، نخستین گزارش وضعیت را تهیه و معمولاً اولین تحلیل خود را ظرف ۱۵ دقیقه منتشر کرده است.

Claude Tag حساب کاربری سرویس اختصاصی خود را دارد و به ابزارهایی که یک مهندس CI در آنتروپیک به آن‌ها نیاز دارد، مانند Datadog یا Grafana دسترسی دارد. دستورالعمل‌های ثابت در فایل‌های Markdown و در قالب Skills قرار گرفته‌اند و در یک مخزن GitHub نگهداری می‌شوند. به‌این‌ترتیب، چند عضو تیم می‌توانند این دستورالعمل‌ها را به‌صورت تکرارشونده بهبود دهند و تغییرات را درست مانند خود کد مدیریت کنند.



Claude Tag یک بحث جاری در Slack را دنبال و پیشرفت کار را در کانال گزارش می‌کند.

Alexey Milovidov
ClickHouse

این رویکرد در ClickHouse بیش از همه مشهود بود؛ جایی که **«الکسی میلوویدف»، هم‌بنیان‌گذار و مدیر فناوری** آن عنوان کرد که این شرکت فعال در حوزه پایگاه‌داده تقریباً هر مرحله از SDLC را به یک حلقه خودمختار تبدیل کرده است. دو عامل اختصاصی که برای رفع تست‌های ناپایدار (Flaky Tests) و شناسایی پوشش تست‌های ناقص طراحی شده‌اند، اکنون به ترتیب دومین و سومین مشارکت‌کننده بزرگ در مخزن ClickHouse هستند. خانواده جداگانه‌ای از عامل‌ها نیز عملیات را بر عهده دارند و این تیم از Claude Code برای ساخت و بهبود تکرارشونده همین عامل‌ها استفاده می‌کند.

تسریع فرایندها با عامل‌ها

الگوی ثابت دیگری که مشاهده شد این بود که این استارتاپ‌ها نه‌تنها از حلقه‌های عامل‌محور در Claude Code برای تسریع اقدامات توسعه‌محور خود استفاده می‌کردند، بلکه عامل‌هایی نیز ایجاد می‌کردند تا فرایندهای تکرارشونده و اغلب خسته‌کننده را تسریع کنند.

این کارها اغلب فعالیت‌های روزمره و روتین بودند تا توجه بیشتری بتواند معطوف به مزیت رقابتی، روابط با مشتریان و رشد درآمدی شرکت شود. یکی از رایج‌ترین فرایندهایی که Claude آن را تسریع کرد، تحلیل خودکار داده‌ها (Self-Service Data Analytics) بود.

تقریباً همه این شرکت‌ها فرایندی در اختیار داشتند که به آن‌ها اجازه می‌داد با استفاده از داده‌های تازه، از جمله داده‌های غیرساختاریافته، تصمیم‌های سریع بگیرند؛ داده‌های غیرساختاریافته داده‌هایی هستند که سوخت لازم برای چرخش‌ها و تغییر مسیرهای ضروری در چرخه زندگی یک استارتاپ را فراهم می‌کنند.

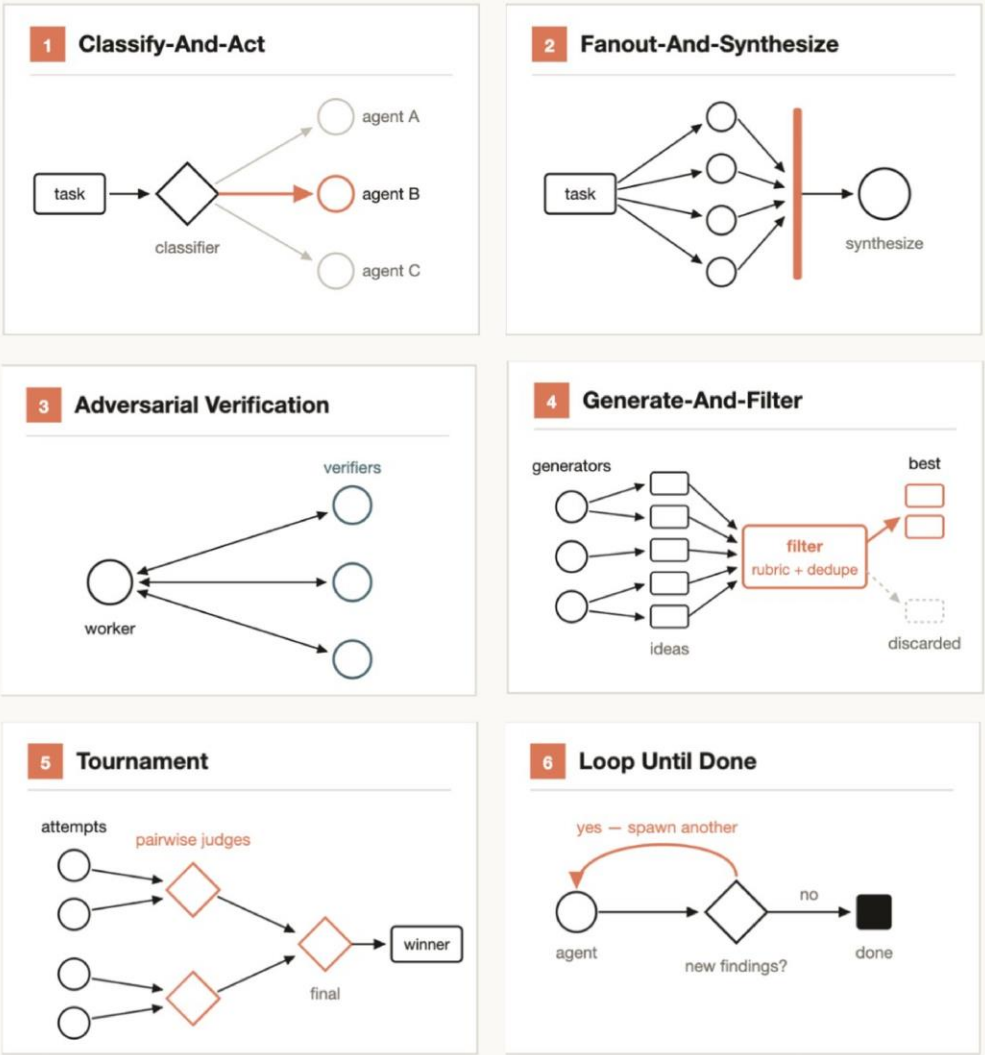
برای مثال، Clay یک عامل تحلیل داده داخلی ساخته و Heidi از Claude Code برای دسته‌بندی بازخورد مشتریان و پزشکان در کنار داده‌های مربوط به میزان استفاده از محصول استفاده می‌کند تا سیگنال‌های مهم برای استخراج بینش‌های محصول را شناسایی کند.

هم ClickHouse و هم Omni محصولاتی عرضه می‌کنند که این نوع تحلیل داده مبتنی بر هوش مصنوعی را درون خود محصول ارائه می‌دهند و همه این قابلیت‌ها متکی به Claude هستند.

نمونه‌های دیگر شامل خلاصه‌سازی هزاران سند حقوقی با استفاده از زیرعامل‌ها در Crosby، بررسی گسترده داده‌های خسارت برای شناسایی ناهنجاری‌ها در سایت‌های مختلف در Commure و استخراج مستمر داده‌های مالی بیمارستان‌ها برای یافتن نشانه‌های هشداردهنده‌ای است که هیچ تیم تحلیلگری نمی‌توانست به‌موقع شناسایی کند؛ رویکردی که Translucent از آن استفاده می‌کند.

نکته: می‌توان از گردش‌کارهای پویا برای ایجاد چندین زیرعامل و اجرای تحلیل موازی روی حجم زیادی از داده‌ها استفاده کرد یا از آن‌ها برای انجام یک «بازبینی خصمانه» (Adversarial Review) روی کار یک عامل دیگر بهره گرفت. هنگام استفاده از مدلی مانند Claude Opus یا Claude Fable می‌توانید بگویید: «چند زیرعامل را به‌صورت موازی به کار بگیر» یا «از یک گردش‌کار استفاده کن.»

۶ الگوی گردش‌کار



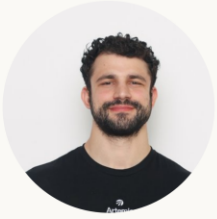
۳

اعتماد کنید، اما راستی‌آزمایی کنید

نمی‌توانید یک فرایند را خودکار کنید، مگر اینکه ابزاری قابل‌اعتماد برای پایش و راستی‌آزمایی نتیجه آن داشته باشید.

اعتماد کنید، اما راستی‌آزمایی کنید

این قانون، مکمل ضروری قانون دوم است: کارهای خسته‌کننده را خودکار کنید. نمی‌توانید یک فرایند را خودکار کنید، مگر اینکه ابزاری قابل‌اعتماد برای پایش و راستی‌آزمایی نتیجه آن داشته باشید.



Dan Shiebler
Artemis Security

«دن شیبلر»، هم‌بنیان‌گذار Artemis Security درباره افزایش سرعت استقرار محصولات در این شرکت گفت این افزایش سرعت تنها به این دلیل امکان‌پذیر است که: «ما سرمایه‌گذاری عمیقی روی زیرساخت تست، سازمان‌دهی کدبیس و سیستم‌های دانش تیم انجام داده‌ایم؛ سیستم‌هایی که به عامل‌ها اجازه می‌دهند کار را از ابتدا تا انتها عرضه کنند. این همان چرخه تقویتی‌ای است که با Claude ساخته‌ایم: کدبیس، پایگاه دانش و تیم خود را به شیوه درست ساختاربندی کنید تا هر مشارکت جدید، اثر مشارکت‌های قبلی را تقویت کند.»

Victor Hunt
Zingage

«ویکتور هانت»، هم‌بنیان‌گذار و مدیرعامل Zingage نیز عنوان کرد: «اوایل کار، ما به Claude استقلال دادیم و این سیستم همان کاری را کرد که هوش مصنوعی انجام می‌دهد و کدهای قابل‌قبول را با سرعت زیادی نوشت. مشکل این بود که به طرقی از معماری ما فاصله گرفت که در ظاهر درست به نظر می‌رسیدند، اما در واقع درست نبودند. بنابراین ما... همه موارد تغییرناپذیر یا اصطلاحاً ناوردا (Invariant) را مکتوب کردیم. اینکه چگونه مسائل را صورت‌بندی می‌کنیم. چه چیزهایی تحت هر شرایطی باید درست باشند. چگونه به‌جای اعتماد به یک پاسخ مطمئن و قاطع، اثبات کنیم که چیزی واقعاً کار می‌کند. ۵۶۷ خط درباره شیوه تفکر این تیم.»

نکته: آنچه نباید تغییر کند را در فایل CLAUDE.md در هسته اصلی مخزن خود قرار دهید. Claude این فایل را در ابتدای هر جلسه می‌خواند؛ لذا قواعد معماری، مرزهای امنیتی و الزامات غیرقابل‌تغییر شما در هر جلسه همراه عامل خواهند بود.

برای روشن‌شدن موضوع، هیچ‌یک از این استارتاپ‌ها عامل‌ها را مستقیماً برای ادغام در شاخه اصلی (Main) رها نمی‌کنند و بعد امیدوار نمی‌مانند که همه چیز خوب پیش برود. بسیاری از آن‌ها در صنایع به‌شدت رگولاتوری و قانون‌گذاری‌شده فعالیت می‌کنند و به چارچوب‌های حاکمیتی قدرتمند نیاز دارند. Cainex نمونه‌ای بسیار گویا از ترکیب عامل‌ها با کنترل‌های قطعی است؛ این شرکت از این ترکیب برای خواندن پرونده‌های پزشکی و تولید کدهایی استفاده می‌کند که فرایند صدور صورت‌حساب بیمارستان را هدایت می‌کنند.

Uriah Israel
Cainex

«اوریا اسرائیل»، هم‌بنیان‌گذار و مدیر فناوری Cainex می‌گوید: «در کدنویسی حوزه پزشکی، یک کد اشتباه صرفاً یک غلط تایپی نیست؛ بلکه یک رخداد مرتبط با صورت‌حساب و انطباق مقرراتی است. همین یک واقعیت تعیین می‌کند که ما چگونه سیستم خود را می‌سازیم.»

او ادامه می‌دهد: «این همان حلقه‌ای است که Claude Code برای ما اجرا می‌کند. ما یک دسته از داده‌ها را با یک عامل پردازش می‌کنیم و حسابرسان ما خروجی را در یک اپلیکیشن داخلی بررسی می‌کنند. آن‌ها فقط کدها را نمی‌بینند؛ بلکه استدلال مدل را نیز می‌بینند و درباره هر دو مورد نظر می‌دهند... همه چیز نسخه‌بندی‌شده و قابل حسابرسی است.»

«سپس Claude Code وارد عمل می‌شود. این سیستم پیش‌بینی‌های اولیه را همراه با تمام اصلاحات و نظرات، مستقیماً از پایگاه‌داده می‌خواند. هر اصلاح بر اساس نوع کد مربوطه برچسب‌گذاری می‌شود؛ بنابراین Claude Code می‌داند با یک مسئله مربوط به تشخیص، یک مسئله مربوط به اقدام پزشکی یا دسته دیگری مواجه است و می‌تواند مستقیماً به دستورالعملی مراجعه کند که بر همان نوع خاص از کدگذاری حاکم است.»

«سپس از آنجا، بخش مربوط به دستورالعمل‌های عامل که باعث ایجاد اشتباه شده را پیدا و آن را اصلاح می‌کند یا اگر مورد واقعاً جدید باشد، دستورالعمل جدیدی می‌نویسد. هر تغییر بر اساس مجموعه‌ای نسخه‌بندی‌شده از دستورالعمل‌ها اعمال می‌شود و در برابر سوابقی که در آن‌ها خطا رخ داده بود، تست می‌شود. قانونی که ما اعمال می‌کنیم این است: اصل را اصلاح کن، نه مثال.»

«سپس نوبت به پس‌آزمایی (Back-test) می‌رسد. یک سابقه می‌تواند بیش از یک کدگذاری قابل‌قبول داشته باشد؛ بنابراین موضوع صرفاً تطبیق رشته‌ای نیست. این بررسی، تطبیق معنایی را با مجموعه کدهای موردقبول ما ترکیب می‌کند و در کنار آن از یک داور استفاده می‌کند که می‌پرسد: آیا این یک خطای واقعی است یا صرفاً یک مسیر معتبر دیگر؟ Claude Code نیز مقایسه‌های خودش را به این فرایند اضافه می‌کند.»

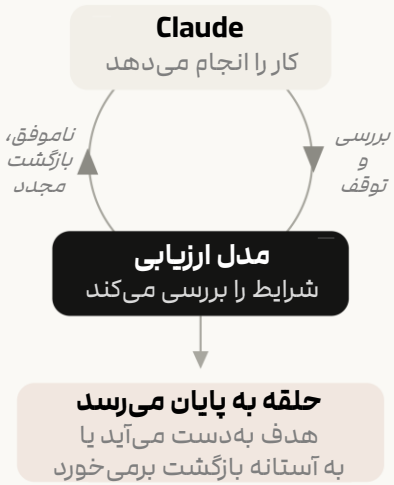
«این سیستم تغییر پیشنهادی را روی یک اصطلاحاً «مجموعه طلایی» (Golden Set) به‌علاوه نمونه‌های تصادفی اجرا می‌کند و پیش از عرضه هر چیزی، هرگونه پس‌گرایی یا به اصطلاح فنی همان رگرسیون (Regression) را آشکار می‌کند. خروجی نهایی یک فهرست کوتاه است: ویرایش‌های پیشنهادی، سوابقی که نتوانستهِ حل‌وفصل کند و پرسش‌هایی که نیاز به پاسخ دارند. مهندسان وقت خود را صرف موارد واقعاً دشوار می‌کنند، نه ۸۰ درصد کارهای دستی.»

از این گردش‌کار اختصاصی حوزه صورت‌حساب خدمات درمانی می‌توان نکات عمومی متعددی برای بنیان‌گذاران استخراج کرد.

به‌عنوان‌مثال، Cainex از متخصصان هر حوزه استفاده می‌کند تا به طور مستمر استدلال Claude را بررسی و هدایت کنند و اطمینان یابند که این راهنمایی‌ها به بخشی از یک حلقه خودبهبوددهنده تبدیل می‌شوند. بااین‌حال، این متخصصان قرار نیست خطاها موردبه‌مورد را اصلاح کنند؛ راهنمایی آن‌ها به‌عنوان بخشی از یک حلقه خودبهبوددهنده مورداستفاده قرار می‌گیرد.

نکته: حلقه‌ها عامل‌هایی هستند که چرخه‌های کاری را تا رسیدن به یک شرط توقف تکرار می‌کنند. آن‌ها می‌توانند روش مؤثری برای استفاده از Claude Code در کارهای مستقل‌تر یا با مدت‌زمان طولانی باشند. می‌توانید از Skills برای تعریف معیارهایی استفاده کنید که عامل باید به آن‌ها دست پیدا کند. هرچه این معیارها شفاف‌تر و دقیق‌تر تعریف شوند، بهتر است و از عامل بخواهید تا رسیدن به هدف، فرایند را تکرار کند.

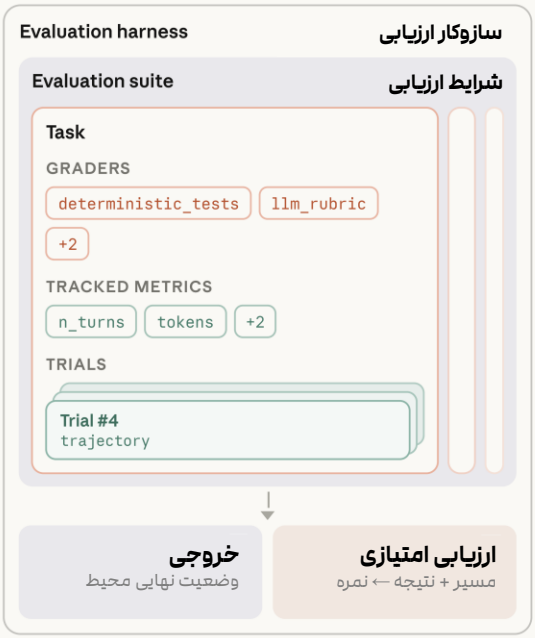
برای مثال، بسیاری از سازمان‌ها عامل‌ها یا لوپ‌هایی برای تست‌های ناپایدار ایجاد می‌کنند، زیرا شرط توقف در اینجا روشن و مستقل است؛ عامل می‌تواند با اجرای مجدد تست، اصلاح خود را راستی‌آزمایی کند و این کار را تا زمانی ادامه دهد که تست با موفقیت اجرا شود.



نکته مهم دیگر، دقت و وسواس در حفظ یک مجموعه طلایی قدرتمند برای ارزیابی است؛ یعنی مجموعه‌ای از جفت‌های پرسش‌وپاسخ تأییدشده که تیم از آن‌ها برای بررسی دقت عامل استفاده می‌کند. هر استارتاپ باید برای موارد کاربرد کلیدی خود، چندین مجموعه ارزیابی (Eval) داشته باشد و آن‌ها را به طور منظم به‌روزرسانی کند تا بتواند از انحراف و رانش عملکرد (Drift) جلوگیری کند و همچنین مدل‌های آینده نیز را ارزیابی کند.

Alex Mashrabov
Higgsfield

«**الکس ماشرابوف»، هم‌بنیان‌گذار و مدیرعامل Higgsfield** نیز دراین‌خصوص می‌گوید: «Claude Code شیوه مدیریت سرعت تغییر مدل‌ها را در شرکت ما متحول کرده است. مدل‌های جدید ویدئو و تصویر دائماً وارد بازار می‌شوند. هر مدل جدید پیش از استقرار، به Skillها، ارزیابی‌ها، منطق مسیریابی و تست‌های تولیدی جدید نیاز دارد. Claude Code این چرخه را از چند روز به چند ساعت کاهش داده است و به ما اجازه می‌دهد مشکلات را در محیط تولید شناسایی کنیم و اصلاحات را در همان جلسه مستقر کنیم... وقتی با شرکت‌هایی رقابت می‌کنید که ۱۰ برابر شما نیروی انسانی دارند، چنین اهرم عملیاتی‌ای همه چیز را تغییر می‌دهد.»



اجزای روند ارزیابی عامل

نکته: وقتی تیم‌ها برای نخستین‌بار شروع به ساخت عامل‌ها می‌کنند، می‌توانند با ترکیبی از تست دستی، استفاده از محصول خودشان و شهود، به شکل شگفت‌آوری پیشرفت کنند. نقطه شکست اغلب زمانی فرامی‌رسد که کاربران می‌بینند عامل پس از اعمال تغییرات، عملکرد بدتری پیدا کرده است و تیم بدون هیچ راهی برای راستی‌آزمایی، جز حدس‌زدن و آزمون‌وخطا، عملاً در تاریکی حرکت می‌کند. در چنین شرایطی، تیم‌ها نمی‌توانند پسرفت‌های واقعی را از خطاهای رایج تشخیص دهند، تغییرات را پیش از عرضه به طور خودکار در برابر صدها سناریو تست کنند یا میزان بهبود را اندازه‌گیری کنند. برای اطلاعات بیشتر، مطلب «Demystifying Evals for AI Agents» را مطالعه کنید.

آخرین نکته‌ای که هم‌بنیان‌گذار Cainex مطرح می‌کند این است که ساخت چنین فرایندی به کار و تلاش قابل‌توجهی نیاز دارد. او می‌گوید: «این فرایند از همان ابتدا این‌قدر تمیز و منظم نبود. نسخه اول ما بیش‌برازش (Overfit) داشت. سیستم با واردکردن جزئیات همان مورد خاص، آن را «اصلاح» می‌کرد و ما به‌جای اینکه سیستم را هوشمندتر کنیم، دائماً در حال انباشتن وصله‌ها بودیم. رویکردمان را تغییر دادیم تا سیستم را وادار کنیم بر اصول کلی تمرکز کند و تعداد جزئیاتی که اساساً می‌توانند وارد یک تغییر شوند را محدود کردیم.»

نکته: عامل‌های هوش مصنوعی، قطعی (Deterministic) نیستند؛ اما بسیاری از فعالیت‌های به‌شدت قانون‌گذاری‌شده نیاز دارند که فرایندهایشان هر بار دقیقاً به یک شیوه انجام شوند. Claude Code قابلیت‌هایی دارد که می‌توانند به ترکیب هوش پیشرفته و فرایندهای قطعی کمک کنند.

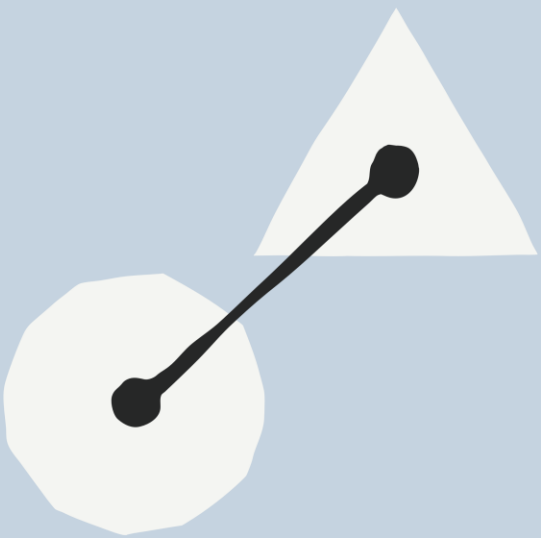
قلاب‌ها یا به‌اصطلاح فنی Hookها، فرمان‌هایی هستند که کاربر تعریف می‌کند و در نقاط مشخصی از چرخه عملکرد Claude Code اجرا می‌شوند و می‌توانند به‌عنوان دروازه‌های ورودی سخت‌گیرانه عمل کنند. این فرمان‌ها صرف‌نظر از اینکه مدل چه تصمیمی گرفته باشد، همیشه و هر بار اجرا می‌شوند. برای مثال، می‌توان از آن‌ها برای جلوگیری از نوشتن کدی که در Lint شکست می‌خورد، الزام به موفقیت تست پیش از اجرا یا حذف اسرار و اطلاعات محرمانه پیش از خروج هر چیزی از سندباکس استفاده کرد.

گردش‌کارهای پویا نیز زیرعامل‌ها را با استفاده از توالی قطعی، پنجره‌های زمینه مجزا و اهداف متمرکز هماهنگ می‌کنند. دستور goal/ برای وظایف پیچیده و طولانی مفید است؛ وظایفی که در آن‌ها Claude ممکن است زودتر از موعد اعلام کند کار تمام شده است یا هنگام بررسی کار، یافته‌های خودش را بر شواهد دیگر ترجیح دهد یا از اهداف اولیه خود منحرف شود.

۴

برای بازسازی بسازید

قابلیت‌های مدل‌ها دائماً در حال تغییر است؛ بنابراین بسیاری از چیزها موقت تلقی می‌شوند و تقریباً هیچ‌چیز دائمی فرض نمی‌شود.



برای بازسازی بسازید

بسیاری از این استارتاپ‌های AI-Native در وضعیت بازآفرینی و بازطراحی دائمی قرار دارند.

هوش مصنوعی اغلب هم در قلب چیزی قرار دارد که این شرکت‌ها می‌سازند و هم در قلب شیوه‌ای که آن را می‌سازند. ازآنجایی‌که قابلیت‌های مدل‌ها به طور مستمر تکامل پیدا می‌کند، قابلیت‌های تحول‌آفرین و زیرساخت‌های حیاتی به‌محض اینکه به هزینه‌های غرق‌شده (Sunk Costs) تبدیل می‌شدند، کنار گذاشته می‌شدند. بسیاری از این سازمان‌ها این بازسازی مستمر را بخشی از مزیت رقابتی خود می‌دانستند.

«کریم امین» دراین‌رابطه می‌گوید: «کاری که ما در Clay انجام می‌دهیم این است که چیزی را می‌سازیم، بعد دوباره آن را می‌سازیم و باز هم آن را می‌سازیم؛ سپس بار چهارم که آن را می‌سازیم، همه چیزهایی را که لازم است می‌شناسیم و این بار آن را درست انجام می‌دهیم. بنابراین ما لزوماً چیزها را دور نمی‌اندازیم؛ فقط دوباره آن‌ها را می‌سازیم، این بار با شفافیت بیشتر.»

هم‌بنیان‌گذار Commure نیز می‌گوید: «یک بازسازی زمانی تمام نمی‌شود که یک مسیر جدید ارائه شود؛ زمانی تمام می‌شود که مسیر قدیمی دیگر وجود نداشته باشد. حذف و جمع‌کردن مسیر قبلی همیشه در رقابت برای اولویت‌بندی شکست می‌خورد؛ چون کار خسته‌کننده‌ای است و هیچ قابلیت جدیدی عرضه نمی‌کند.»

«اکنون یکی از مهندسان Commure صرفاً یک Skill از Claude را با این مضمون فراخوانی می‌کند که: «برای هر Feature Flag که قبلاً برای همه منتشر شده است، یک PR برای حذف آن و کد مرتبط با آن باز کن»؛ سپس خروجی را بررسی می‌کند. جابه‌جایی‌هایی که قبلاً بخش زیادی از چرخه‌های توسعه را به خود اختصاص می‌دادند، اکنون به یک برنامه و یک اجرای موازی تبدیل شده‌اند و در عرض چند ساعت انجام می‌شوند.»

نکته: از Git Worktree برای اجرای بازسازی در یک کپی ایزوله از مخزن استفاده کنید، درحالی‌که نسخه فعلی دست‌نخورده باقی می‌ماند. Claude Code می‌تواند Worktree ایجاد کند؛ در نتیجه نسخه ۲ در کنار نسخه ۱ اجرا می‌شود، ارزیابی‌ها روی هر دو اجرا می‌شود و تنها زمانی ادغام می‌شوند که نسخه جدید عملکرد بهتری داشته باشد. همین ویژگی است که «چهار بار ساختن» را کم‌هزینه می‌کند.

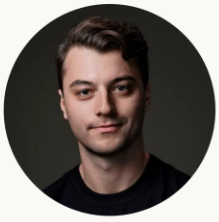
```
~/projects — zsh

$ tree -L 1 ~/projects
~/projects
├── acme-web ← main worktree [main]
├── acme-web-feature ← linked [feature/checkout-flow]
└── acme-web-hotfix ← linked [hotfix/payment-bug]

One repository, one object store — three checkouts you can work in simultaneously, each on its
own branch.
```

هر Worktree متصل‌شده، یک دایرکتوری معمولی با شاخه بررسی‌شده مخصوص به خود است؛
هر سه Worktree نیز مخزن git. را در acme-web به اشتراک می‌گذارند.

«کریم امین» همچنین بخشی از سازوکار دفاعی Clay را توانایی این شرکت در بازسازی، تکامل و ایجاد حلقه‌های خودبهبوددهنده مستمر توصیف کرد.
به گفته وی: «فکر می‌کنم سازوکار دفاعی هر شرکتی در حال حاضر این است که باید خودبهبوددهنده باشد. بنابراین Clay یک موتور درآمد خودآموز است. هرچه بیشتر از آن استفاده کنید، بیشتر می‌فهمیم بهترین مشتریان چه کسانی هستند، چه چیزی باید بگویید، چه چیزهایی جواب داده و چه چیزهایی جواب نداده‌اند و این موضوع در طول زمان تغییر می‌کند. رقابت واقعاً بر سر این است که چه کسی می‌تواند سریع‌تر به بازار گسترده برسد... تا بتوانید به هر مشتری کمک کنید و در نتیجه خودتان نیز بهبود پیدا کنید.»



Niko Grupen
Harvey

در رویداد Code with Claude در ماه مه ۲۰۲۶، **«نیکو گروپن»، مدیر Applied AI در Harvey** درباره این موضوع صحبت کرد که هر موج جدید از قابلیت‌های مدل‌ها از جمله استدلال نوظهور، خودکارسازی عامل‌محور، برنامه‌ریزی و ارکستراسیون به یک بازمعماری کامل پلتفرم نیاز داشته است.

گروپن گفت: «اگر شش ماه پیش از من می‌پرسیدید معماری ما چگونه است، پاسخی که می‌دادم اساساً با چیزی که امروز داریم متفاوت بود. اگر حاضر نبودیم بگوییم «باید این را کنار بگذاریم و به سمت معماری بومی عامل‌ها برویم»، امروز به‌سادگی نمی‌توانستیم این قابلیت‌ها را در پلتفرم خود داشته باشیم.»

Walden Yan
Cognition

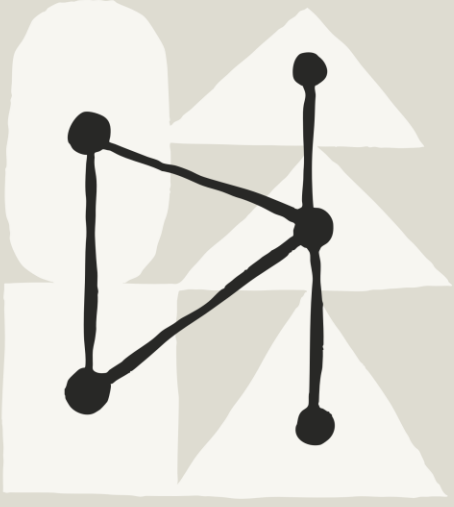
«والدن یان»، هم‌بنیان‌گذار Cognition نیز در همان رویداد گفت: «شیوه کار در ساخت سیستم‌های هوش مصنوعی کنونی این است که بپذیرید چیزی که امروز می‌سازید، به‌احتمال زیاد ظرف شش ماه تا یک سال کنار گذاشته خواهد شد... [Devin] با مجموعه مدل‌هایی که دو سال پیش داشتیم، اساساً امکان‌پذیر نبود؛ اما شرط‌بندی ما این بود که شاید امروز کار نکند، اما به‌زودی کار خواهد کرد.»

نکته: برای بازنویسی‌های غیرساده و اساسی، Claude Code را در حالت برنامه‌ریزی (Plan Mode) اجرا کنید (plan-- یا میان‌بر Shift+Tab). Claude پیش از شروع به نوشتن هر قطعه کد، کدبیس را بررسی و رویکرد پیشنهادی برای بازسازی را ارائه می‌کند؛ سپس شما آن را تأیید یا مسیرش را اصلاح می‌کنید. این ارزان‌ترین نقطه برای شناسایی بازسازی‌ای است که در آستانه فاصله‌گرفتن از معماری شما قرار دارد.

۵

نمونه‌سازی، استفاده داخلی، تجاری‌سازی

ساخت به کمک هوش مصنوعی به این استارتاپ‌ها کمک می‌کند محصولات تحول‌آفرین مبتنی بر هوش مصنوعی ایجاد کنند؛ این همان چرخه تقویتی‌ای است که در قلب فرایند آن‌ها قرار دارد.



نمونه‌سازی، استفاده داخلی، تجاری‌سازی

بسیاری از این استارتاپ‌ها یک چرخه تقویتی کلیدی در قلب فرایند توسعه خود دارند. ساخت به کمک هوش مصنوعی به آن‌ها کمک می‌کند محصولات تحول‌آفرین مبتنی بر هوش مصنوعی ایجاد کنند.

وقتی توسعه‌دهندگان شیوه‌های کدنویسی عامل‌محور خود را ارتقا می‌دهند، درک عمیق‌تری از قابلیت‌های مدل به دست می‌آورند و بینش بیشتری درباره چگونگی تکامل سازوکارهای حفاظتی در خط مقدم فناوری پیدا می‌کنند. سپس می‌توانند با الهام‌گرفتن از آن، در عامل‌ها و محصولات خودشان نیز از آن استفاده کنند.

«کریس مریک» دراین‌خصوص می‌گوید: «ما از رویکرد آنتروپیک در زمینه فایل در برابر تعبیه یا همان Embedding الهام گرفتیم؛ این موضوع ما را تشویق کرد که در محصول خودمان نیز همه چیز را ساده نگه داریم. از بسیاری از پیچیدگی‌هایی که یک خط لوله فرایندی RAG ایجاد می‌کرد، اجتناب کردیم. همچنین دیدیم که سازوکارهای حفاظتی مربوط به Claude Code چگونه به کاربران امکان می‌داد کارها را به‌صورت موازی انجام دهند و برخی از این مفاهیم را در رابط کاربری خودمان به کار گرفتیم.»

این رویکرد همچنین به آن‌ها کمک می‌کند نسبت به عملکرد محصول خود حساس و همواره به‌روز بمانند.

«موکوند جا» نیز می‌گوید: «ازآنجایی‌که سازنده اپلیکیشن ما نیز در پشت‌صحنه از مدل‌های آنتروپیک استفاده می‌کند، اگر زمانی رفتاری را در محصول خود مشاهده کنیم... می‌توانیم به‌سرعت با استفاده از Claude Code در محیط لوکال باگ‌ها و اشکالات را شناسایی کنیم تا مشخص شود این رفتار ناشی از خود مدل است یا یک مشکل در سازوکاری حفاظتی ماست. این موضوع چرخه‌های تریاژ ما را به‌شدت بهبود داده است.»

الگویی که بارها شنیده شد این بود که: با Claude Code یک عامل داخلی بسازید (Prototype)، از آن در داخل سازمان استفاده کنید (Dogfood) و بسته به نتیجه، آن را به یک محصول مشتری‌محور ارتقا دهید (Productionize)؛ این کار اغلب با استفاده از Claude API، SDK یا Claude Managed Agents انجام می‌شود.

«الکسی میلوویدف» هم دراین‌رابطه گفت: «ما عامل‌های هوش مصنوعی خودمان را در محصولمان ساخته‌ایم که تیم‌ها مستقیماً با آن‌ها تعامل دارند؛ از جمله یک عامل در کنسول SQL و یک AI SRE. ما از Claude Code برای ساخت و بهبود تکرارشونده خود این عامل‌ها استفاده می‌کنیم. ابزارهایی که تجربه‌های هوش مصنوعی مشتریان ما را شکل می‌دهند، خودشان تا حدی با هوش مصنوعی ساخته شده‌اند.»

چک‌لیست

ازآنجایی‌که این راهنما موضوعات زیادی را پوشش می‌دهد؛
در ادامه، نکات کلیدی در یک صفحه یکجا گرده‌آوری‌شده است.

چک‌لیست

بخش اول: همه دست‌اندرکار هستند

- ❑ Claude نمی‌تواند چیزی که نمی‌بیند را درک کند. آن را از طریق MCP یا CLI به منابع حقیقت و ابزارهایی که تیم شما به‌صورت روزانه استفاده می‌کند، متصل کنید.
- ❑ یک بازار داخلی افزونه‌های سازمانی (Company Plugin Marketplace) ایجاد کنید تا بهترین رویه‌های یک کارمند بتواند از طریق یک Skill، بلافاصله در اختیار سایر کارمندان نیز قرار بگیرد. در هر زیرشاخه از مخزن خود از فایل‌های CLAUDE.md استفاده کنید تا قراردادهای کدنویسی مخصوص همان زیرشاخه را تعریف کنید؛ قراردادهایی که هر بار اعمال می‌شوند. برای گردش‌کارهای رویه‌ای درخواستی از Skillها استفاده کنید.

بخش دوم: کارهای خسته‌کننده را خودکار کنید

- ❑ قابلیت Code Review را روی یک مخزن راه‌اندازی کنید تا یک مرحله بررسی خودکار روی درخواست‌های ادغام انجام شود.
- ❑ Claude Tag را به بخشی از فرایند پاسخ‌گویی در حالت آماده‌باش CI/CD و تریاژ باگ‌های خود تبدیل کنید.
- ❑ از گردش‌کارهای پویا می‌توان برای اجرای موازی چندین زیرعامل به‌منظور تحلیل حجم زیادی از داده یا انجام یک بازبینی خصمانه روی کار عامل دیگر استفاده کرد.

بخش سوم: اعتماد کنید، اما راستی‌آزمایی کنید

- ❑ آنچه نباید تغییر کند را در فایل CLAUDE.md در ریشه مخزن خود قرار دهید.
- ❑ برای کارهای مستقل‌تر یا طولانی‌مدت‌تر از Loopها، یعنی عامل‌هایی که چرخه‌های کاری را تا رسیدن به یک شرط توقف تکرار می‌کنند، استفاده کنید.
- ❑ فرایندی برای ایجاد و نگهداری ارزیابی‌های عامل‌ها ایجاد کنید.
- ❑ Hookها فرمان‌هایی هستند که کاربر تعریف می‌کند و در نقاط مشخصی از چرخه عمر Claude Code اجرا می‌شوند و می‌توانند به‌عنوان دروازه‌های ورود سخت‌گیرانه عمل کنند. هرجا بخش‌هایی از کار باید قطعی و قابل‌پیش‌بینی باشند، از Hook استفاده کنید.

بخش چهارم: اعتماد کنید، اما راستی‌آزمایی کنید

- ❑ از Git Worktree استفاده کنید تا بازسازی را در یک کپی ایزوله از مخزن اجرا کنید، درحالی‌که نسخه فعلی دست‌نخورده باقی می‌ماند. همین ویژگی است که «چهار بار ساختن» را کم‌هزینه می‌کند.
- ❑ برای بازنویسی‌های غیرساده، Claude Code را در حالت برنامه‌ریزی اجرا کنید. Claude پیش از نوشتن هر کد، کدبیس را بررسی و رویکرد پیشنهادی برای بازسازی را ارائه می‌کند؛ شما آن را تأیید یا مسیرش را اصلاح می‌کنید. این ارزان‌ترین نقطه برای شناسایی بازسازی‌ای است که در آستانه فاصله‌گرفتن از معماری شما قرار دارد.