I need you to hear me out (it’s REALLY good)

30 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=NOO0NWD0GHU
https://www.youtube.com/watch?v=Noo0NWD0gHU

SUMMARY

The video discusses the creator's unexpected praise for Claude Code over Codex, highlighting the advantages of Claude Code's user interface, workflow management, and system prompts. The creator expresses frustration with Codex's limitations and inefficiencies, particularly regarding its handling of sub-agents and overall design capabilities.

- *Claude Code offers a superior user interface and better workflow management compared to Codex.*
- *The creator finds Claude Code's approach to sub-agents more efficient, allowing for clearer, structured workflows.*
- *Codex's system prompt is criticized for being overly prescriptive and poorly designed, leading to inefficient outputs.*
- *The creator notes that Claude Code's model can write its own code for workflows, enhancing flexibility and efficiency.*
- *Issues with Codex include a lack of updates, confusing sub-agent functionalities, and a tendency to produce poor design outputs.*
- *The creator plans to rewrite Codex's system prompt to improve its performance and usability.*
- *Claude Code's integration capabilities allow for the use of models from other platforms, enhancing its utility.*
- *The video concludes with an invitation for viewers to experiment with setting up Claude Code for their own projects.*

If you had told me a few months ago I would have to make this video, I would have looked at you very, very strange because I never thought I would be praising Claude code as much as I'm about to, especially in the context I'm about to praise it in. You see, I've been using 5-6 a lot, but I haven't been using it in Codex. I'm using it in Claude code. I'm sure you have a lot of questions as you probably should because this is kind of dumb. It is so dumb that I tried it out during my early access window and was impressed. And now that I'm getting more and more frustrated with Codex as a harness, a CLI and more, I decided to go explore using 5-6 soul in Claude code a bit more. And the results have absolutely broken me. It is not just cuz like Claude code is a better UI, it kind of is better than the Codex CLI, but the real benefits are the things Claude code does differently from how it thinks about sub-agents to its system prompt to its integrations and more. I already did a video where I praised the good parts of Claude code. I did not think I would be revisiting this topic so soon, but yeah, I have some issues with Codex. This video is going to be a bit chaotic. I'm obviously going to show you how to set up 5-6 soul in Claude code, but I also I'm going to talk a lot about the why, both why I'm so frustrated with Codex and also what Claude code is doing better to make me want to reach for it instead. And also a hint of why I'm not reaching for things like pie as cool as they are. No shots fired at pie at all, but I really need to focus in on Claude code and Codex because there is a new generation of capability with these models and it is very, very important that we understand what these harnesses are doing right and wrong if we want to get the most out of

them. But first, I want to make sure you get the most out of today's sponsor. AI has gotten really good at writing code and it's even gotten decent at reviewing code. So why is it that code review has never been worse? I'm so tired of just going through these PRs. Like look at this.

Why is it an alphabetical list of files in my code base? This PR has a thousand lines changed. Are you seriously going to tell me I have to sit here and go through every single one of them? Wouldn't it be nicer if this was broken up based on what's actually happening and not the order of the code in the code base? It'd be really cool if one of our code review agents could do this breaking up for us and give us a way better interface to review our code in, wouldn't it? Oh, it's open, isn't it?

Surprise, Code Rabbit has a code review platform now, not just a GitHub plugin, but an actual full website that is way easier to read changes from. They break up the PR into different layers, which describe specific changes that the PR makes, making it way easier to review. Each of these sections is a summary and the details that are important called out, and the pieces inside of the layer are prioritized based on what you should be caring about. They even have fancy little guides on the side here that show us labeled sections that might be worth checking out. This is particularly useful on gigantic PRs because they're basically impossible to use in the GitHub interface. Since this one breaks things up, it's way easier to find the parts that actually matter and give useful feedback to your team. Merge bigger PRs faster with more confidence at soidev.link/coderabbit.

So, with that, let's get started with what's wrong with Codex? There are layers to this one. I don't want to crash out again about the rebranding of Codex as the Chachi BT desktop app or about the show of the CLI not getting updated as much as the desktop app. Generally speaking, I still think for most people, the best way to start using and taking advantage of more agentic engineering is the Codex desktop app on one of the Codex subscriptions. This hurts because obviously, I want to be pushing people towards T3 Code, but the Codex app is a really good starting point. So much so that it inspired me to build my own open-source alternative, but I have some issues right now. The biggest issue by far is how Codex thinks about sub-agents. I just filmed a video about Ultra mode that's already out. I would recommend checking that if you want a deeper dive on this. The simplest TLDR I can give is that there's two versions of sub-agents in Codex. The V1 is stable and very simple. It lets the top-level agent spawn a bunch of sub-agents one level deeper with specific work and tasks to do and then gets a response when they're done. V2, which is the new unfinished version that you have to manually enable, has the ability to copy the whole context window over by default, which is annoying, and also let the sub-agents spin up their own named sub-agents beneath them with layering and message passing between them. It takes a very simple hierarchy and makes it a bit absurdly complex and adds a ton of tools that are necessary for all of these things to communicate, and the result's a bit chaotic and clearly still being tuned. That's why it's off by default. Go check out my video I just did about Ultra if you want a deeper dive on how sub-agents are currently working in Codex. The thing I want you to know right now is that I don't like it and I find that the sub-agent workflows, specifically the workflow feature in Claude code, is much better. Because instead of having sub-agents spawn randomly when the model decides to, the model will upfront define a programmatic workflow with different stages, different prompts, different sub-agents at each stage that can spawn work for other stages, all in a single JavaScript file that is executed top to bottom because code is a great way to define something dynamic.

Since workflows are code, they actually end, which I have found to be a huge win for token efficiency when using the 5-6 models. They tend to just kind of go forever with Ultra, and I have seen way, way less utilization on my limits when I'm doing this through workflows instead, even for the same task. Ultra can just go forever, workflows won't. The output quality's the same if not better from workflows, and the token usage is like a fourth as much. So, much more efficient way of doing things.

OpenAI is shipping the code in Codex itself, and the model has to call the code correctly. Claude Code lets the model itself write the code for its workflow and for its sub-agents, which I have found to be much better overall. I found it to be so much better that I wanted to try it myself, and that's what inspired me initially to start using 5-6 inside of Claude Code. Not Fable in Claude Code calling 5-6 over the Codex CLI, but with 5-6 is the actual model doing the work in Claude Code itself.

Since then, I've been using it a ton. And the more I use it, the more impressed I get. I have a question for you guys. Which model do you think designed this page? When I look at this, it screams Claude to me. This obviously looks like something that Claude made, right? Like Fable or Opus. Clearly. As someone who's looked at a lot of pages that were generated with Codex, it could be something GLM or otherwise distilled off of Claude, but this screams Claude to me. For reference, here's a similar page I did with Codex in 5-6 Soul. I don't know about you guys, but the difference in quality here is pretty aggressive to me.

I think both are slop, to be clear. This one hurts me a lot less than the one that Codex did. It shouldn't be too surprising to you, based on the topic of this video, that this page was made with Soul inside of Claude Code. So, why the hell is 5-6 Soul making designs this much better in Claude Code? You probably are going to have the same assumption I had, which is that they have some things in the system prompt for Claude Code that make it more likely to do good designs.

So, I investigated. I read the full Claude Code system prompt, trying to find this front-end guidance so that I could bring it over to Codex and maybe have Codex design better. So, I hunted, and the word front-end appears three times. First, it appears with the instructions that for UI and front-end changes, start the dev server and use the features in a browser before reporting the task as complete. This is telling the model to confirm its work.

Cool, makes sense. Not really front-end guidance. Don't worry though, the word front-end appears two more times on the page. Oh, they're both here as examples of memory. Huh. Strange. Maybe UI appears in here? Oh, yeah, it does twice in the thing we were just app for UI or front-end changes. Huh. Does that mean the system prompt has nothing about design? Why is it designing things so much better then? Something has to be up. I also remember that OpenAI has been bragging about 5.6 being way better at design even though I never saw that. But when I was testing 5.6, almost all of the work I did in it was done in Codex.

Turns out I was checking the wrong place. If you were waiting for the crash out, it begins now. This is the official Codex system prompt. You are Codex, an agent based on GPT-5. You and the user share one workspace and your job is to collaborate with them until their goal is genuinely handled. You can tell from the first sentence what you're in for here, can't you? Oh boy. I don't want to spend this whole video reading this to you

guys and explaining how horrible it is. Okay, that's a lie.

I do kind of want to do that. It is very tempting because I did read this whole thing. And however bad you think it is, it is comically worse. I think they might have listened to me and removed the part I want to about the most, so I'm going to go find a cached version so I can do that there. Thank you to Felipe for uploading this for me. It is very helpful to have. Here is the official front-end guidance that was in Codex up until recently, like I think at least.

Let me double-check. What was an easy way to do this? Yes, they changed it. They listened to me. I crashed out at my friend who works on front-end stuff at Codex to get this fixed and they did. But can we take a moment to appreciate that it takes a YouTuber to get slop removed from Codex? Why is this my job? Let me show you what I made them remove. This is the previously present official front end guidance. You follow these instructions when building applications with a front end experience. First off, build with empathy. If working with an existing design or given a design framework and context, you pay careful attention to existing conventions and ensure that what you build is consistent with the frameworks used in design of the existing application. It gets so much worse from here. I see people in chat already freaking the out. You have no idea how bad this gets. This probably the worst counter psychosis I've ever had was reading this. This feels like it was written for GPT-41.

You think deeply about the audience of what you are building and use that to decide what features to build and when designing layouts, components, visual styles, on-screen text, and interaction patterns. Using your application should feel rich and sophisticated. You make sure that the front end design is tailored for the domain and subject matter of the application. For example, SaaS, CRM, and other operational tools should feel quiet, utilitarian, and work focused rather than illustrative or editorial. Remember that thing that opening eye models do where when you tell them to do a thing, they do the thing? This just told the model if you're doing anything that looks like a SaaS, that it needs to look utilitarian, quiet, and work focused.

Always. And it does, always. But this isn't even the bad section of the front end guidance. You make sure to use icons and buttons for tools, swatches for color, segmented controls for models, toggle/checkboxes for binary settings, sliders/steppers/inputs for numeric values, menus for option sets, tabs for views, and text or icon plus text buttons only for clear commands (parentheses) unless otherwise specified. Cards are kept at 8-pixel border radius or less unless the existing design system requires otherwise. You do not use rounded rectangular UI elements with text inside if you could use a familiar symbol or icon instead. You build tool tips with names and describe unfamiliar icons when the user hovers over it. Cool, fine.

Point three. You use Lucid icons inside buttons whenever one exists instead of manually drawn SVG icons. If there's a library, use that icon library instead. Whatever. The fact that it's prescribing specific libraries for icons and specific border radiuses for cards is absurd. But, we're just getting started here. You build feature-complete controls, states, and views that a target user would naturally expect for the application. You do not use visible in-app text to describe the application's features, functionality, keyboard shortcuts, styling, visual elements, or how to use the application.

You should not make a landing page unless absolutely required. When asked for a site, app, game, or tool, build the actual usable experience as the first screen, not marketing or explanatory content. When making a hero page, you use a relevant image, a generated bitmap image, or an immersive full-bleed interactive scene as the background with text over it that is not in a card. Notice how all of the pages that Codex designs are the same. It's cuz it tells it exactly how to make it look. Here, it tells it how to structure the home page to make sure that H1 tag is the brand, product, place, or person name, or a literal offer/category.

Chad is begging me to stop reading because this is hurting them so much. This is an absolute show. And sadly, Never Drax, this isn't a skill. This isn't a thing that you can add to Codex. This is the official Codex system prompt directly. Up until yesterday, every single time you sent a prompt to any OpenAI model in Codex, this all came along with it. You have burned hundreds of thousands, if not millions of tokens on this slop.

And it took me, probably the first person to ever actually read this, to get it removed. I'm not trying to talk myself up here. In fact, I'm trying to do the opposite. I don't get why it's my job as a YouTuber to be the one to get this fixed. And I'm at the point now where I am working on building a whole new Codex system prompt from scratch by hand, no AI-generated anything in it. Prompts should be by hand, especially global ones and skills and things like that. My system prompt will be written by hand by the end of this coming week.

And if it works out well, I will share it with all of you and harass OpenAI to copy my changes. But there is so much more in here. I don't want to just sit here and complain because I God, you do not put UI cards inside of other cards. Do not style up page sections as floating cards. Only use cards for individual repeated items, models, and genuinely framed tools. The word cards is on this page six times.

The word card is 12. I feel sick. I'm going to ask you guys a question to the power users of Codex. Have you ever noticed that when you set it off to do some work, like babysitting a PR or keeping track of sub-agents, or just doing something long and difficult, that it loves to set timers? Specifically, it really loves to set 30-second timers. Have you guys seen this weird fixation on 30-second time limits in Codex?

Because I did, and I was suspicious of this. Would you like to know where that came from. You provide user updates frequently every 30 seconds. I cannot say enough bad things about this system prompt. It is enough of a reason to never trust Codex again. The fact that this has been shipping for as long as it has and is burning our tokens, wasting our time, and making our outputs worse is hilarious. And with this, I must issue an apology to my friends on Pi. I underestimated your game. To be fair, this is your fault, too, because you never told us how bad the Codex system prompt was. Because the problem with Codex isn't even the implementation. A lot of it is just this. It is so bad.

Don't worry, though. It mentions goblins twice. The Codex system prompt mentions goblins more than the Claude code one mentions front end and UI. And it mentions cards six times more than the Claude code prompt mentions front end at all. My chat is crashing out so hard over this right now. What in the This has been with it interacting with Claude 2. Please God, stop this. Yep, you've all noticed this and I'm the first person to figure out why somehow. I'm sure others have and they just don't have a platform enough that they can be loud about

it online.

But like what the I did mention before that you shouldn't be using agents to write your system prompts, your skills, and all of those things. Like just do it by hand. They will come out way better and you can adjust and learn as you go. That said, even most models should be able to see the problems with this system prompt. So, I asked 56 all in Claude code, as well as Fable 5 in Claude code, and of course 56 all in Codex just to double check and be sure to describe all of the things that are wrong with this system prompt and help me rate it. 56 in Claude code said the following.

Seven out of 10 is a prompt for one tightly controlled Codex runtime, four out of 10 is a general coding agent prompt, and three out of 10 is a portable prompt for modern models like 560. Obviously, it is coupled to Codex. It's built for that. That's fine. It calls out that modern models generally understand tool schemas well without the main behavioral prompt repeatedly describing implementation details. These runtime instructions should be generated by the harness not embedded into the reusable agent prompt. Yeah, sure, whatever. But it redescribes tool calls multiple times throughout. The front-end section is far too prescriptive. It's effectively a separate front-end design constitution. It consumes about a quarter of the prompt even when the task concerns a CLI database back-end or a library. Cards must have a 8-pixel radius or letter spacing must always be zero. All of these things that I went over before. A capable model will obey these, but this doesn't mean the results improve. It may ignore the existing design system user request or domain convention to satisfy arbitrary global constraints. And then there's a bunch of actively questionable UX guidance. Like it's weird call-outs for icon-only controls, especially for touch devices.

Cuz touch devices don't have tooltips, remember? The ban on visible instructional text is more concerning. Empty states, onboarding, validation, guidance, accessibility, hints, and contextual help are often essential product UI. Yep. Have you ever wondered why Codex never does good empty states? It's the system prompt. They also have a bunch of weird stuff in there like continue until solved that is probably part of why 56 doesn't stop when it should sometimes. Have you ever wondered why Codex loves to just start building when you're in the early stages of planning? It's right here, autonomy and persistence.

Unless the user explicitly asks for a plan, asks a question about the code, is brainstorming possible approaches, or otherwise makes it clear that they do not want code changes yet, you assume they want you to make the change or run the tools needed to solve the problem. In those cases, do not stop at a proposal, implement the fix. If you hit a blocker, try to work it through yourself before handing the problem back. I am very thankful opening I let you use your Codex on other places right now.

I love that I'm seeing in chat as I do this, people saying, "Oh my god, that's infuriating. I've been by that exact prompt." I forgot it includes callouts about limiting dominant purple and purple blue gradients, beige, cream, sand, and tan, dark blue and slate, and brown {slash} orange {slash} espresso palettes. Scan CSS colors before you you get the idea. They tried to make 5.5 better at design by putting a bunch of rules in, it made it worse. They tried to make 5.5 more persistent by putting a bunch of rules in, it made it worse or different. And all of these things were still applied on 5.6 where they should not have been. Apparently, the changes they made for the

front-end design stuff is only for 5.6, it still applied for 5.5, even better. We're unlikely to change the current system prompt for 5.5 because it was used during training, but I'll pass your feedback along to the team.

Even 5.6 Soul in Codex had a lot of negative things to say. Specifically called it four out of 10 prompt implementation and tore apart a bunch of things that are wrong with it. I think there's even more than that. The point I'm trying to make here is that the system prompt feels more thrown together than crafted. And although I've on Claude Code and the insane length of their system prompt and whatnot, it's a lot better here. I think they're actually writing a decent bit of it.

You're an interactive agent that helps users with software engineering tasks. Use the instructions below and the tools available to you to assist the user. They added some inserts here about what to not do with like hacking and whatnot. It starts with descriptions of what the user will see. It's All text you output outside of tool use is displayed to the user. Output text to communicate with the user. Cool. That is fine. Tools are executed in a user-specific permission mode. Yep, makes sense.

Doing tasks. The user will primarily request you to perform software engineering tasks. This may include solving bugs, adding new functionality, yada yada. You're highly capable and often allow users to complete ambitious tasks that would otherwise be too complex or take too long. You defer to user judgment about whether a task is too large to attempt. For exploratory questions, respond in two to three sentences with recommendations and the main trade-offs. Presented as something the user can redirect, not a decided plan. Don't influence until the user agrees. This is the opposite of the Codex one, and it's actually quite nice to have the model ask some questions.

Prefer editing existing files to creating new ones. That's nice. Just be careful to not introduce security problems. Probably doesn't need to be inserted anymore, but not the worst thing. Default to writing no comments. Only add one when the why is non-obvious. A hidden constraint, a subtle invariant, a workaround for a bug, behavior that would surprise a reader. If removing the comment wouldn't confuse a future reader, don't write it. These are reasonable guidelines, and all of these are simple and pretty general. Executing actions with care. Carefully consider the reversibility and blast radius of actions. Generally, you could freely take local reversible actions like editing files or running tests. But for actions that are hard to reverse, affect shared systems beyond your local environment, or could otherwise be risky or destructive, check with the user before proceeding. Just like the way this is written, the tone of it, the actual details, of course, as well. It's not that bad. It's actually decent.

It gives a list of examples that are things it should verify before doing, like destructive operations, hard to reverse operations, actions visible to others. Meanwhile, I just saw poor Matt Schumer get his entire system, like user folder, deleted by Soul running an ultra in Codex. So, maybe Codex needs these warnings now, too. There's a simple tone and style section. It says only use emojis if the user requests it. Your responses should be short and concise. When referencing specific functions or pieces, use the pattern file path colon line number.

Don't use a colon before tool calls. Your tool calls may not be shown directly in the output. As things to not save, memory is complex. I'm happy that they gave it enough instructions here to do it right. It gives info on the

environment it is in. It gives the tools. Has instructions on how to create a pull request, which are a little more specific than I like, and make it in formats I don't love. This is one of the few things I would actually change here.

Their edit tool, I still hate how Claude code does edits, but that's a rant for another time. Schedule wake up is cool. It doesn't specify to wait 30 seconds. It gives it info on how to pick a good amount of delay, which is useful. It then just gives a description of skills, tool search, and that's it. It's not that long. I read through the whole thing and it wasn't too bad. Meanwhile, the more I read the Codex system prompt, the more I questioned my whole life up until this point. Yeah.

That's enough of the Codex crash out for now. I think a cleaning of house is in order on that side though. This system prompt is so bad that it should just be deleted and rewritten from scratch, and they need to make some really, really intense benchmarks to measure success and failure, as well as auditing the histories to see how cringe its interactions are. This is bad beyond what is acceptable. So, we established what's wrong with Codex and a good bit of what's better in Claude code. The system prompt is better, workflows are awesome. It's UX in the terminal is better overall. The CLI for Codex is rougher than it should be right now. One last thing before we do the setup though, I want to talk about the gotchas that I've been experiencing using Codex models inside of Claude code. Admittedly, there have been fewer than I expected. First off, I've noticed it sometimes loses track of what it's doing a little bit more. This could be changes in how context is managed, but I honestly think this is just something small in the system prompt. Here is an example where I had it making changes alongside two other agents. All three were making HTML files for me to go review the feedback it was giving.

And two of them happened to come up with the same name for the HTML file. One was using Fable and it wrote fine and made the URL fine. This one was using 56 Soul and it tried to write the file. It already existed, so it said it would leave that file untouched and publish a new report with a distinct file name. So, it did its own different file instead. Of note though is that I told it to use my HTML plan skill. And my HTML plan skill explicitly tells the model to use the web host through the CLI I gave it to host this model. This skill is meant to give the model the ability to upload the HTML with a link that I can click and look at. It's one of my favorite things I've built, my silly little post plan site. And it even says here, I'll keep it standalone and responsive, then publish through post plan as requested.

It failed to write. It noticed the file already existed. It made the new file. Oh, wait. I was wrong. I thought it didn't host it. This is a link. It's just not familiar with the formatting here. I'm used to the model spitting out an actual link. This is somehow a clickable link even though it doesn't look like that here at all. I was wrong. I thought it just didn't do that step. It did. I asked it and here it gave me the actual link in text that makes more sense.

I was wrong about that. Mostly it's good about these things. Like it does stay on task largely fine. It just doesn't know how to use the formatting great. And I have noticed this in other places, too. Like here when it did the number-based formatting, it screwed it up a little bit. Like here it's number one, it's heavily coupled to one specific runtime. Has a chunk and then one again, modern models generally understand yada yada. Then it has

two twice, then it has three twice, and then four twice. I think there's something weird about how markdown formatting works in Claude code, but 5-6 isn't doing it right, clearly.

Remember, the reason I wanted to do this initially was for ultra code and specifically for workflows, because I wanted to see how good 5-6 could do workflows. And it does them great. There was a problem, though. It is an append to the system prompt that says, "Hey, please please please use workflows for this task." This works great with Claude models. You can tell it to use them for your workflows and orchestration. For example, here I told it that I want to better understand this code base. It's the Codex code base, and I want it to analyze with a workflow. Use 5 6 Soul, 5 6 Terra, and Fable 5 all on high reasoning levels to analyze the code base. I want it to have the info when I ask it a vague thing like this so that it will orchestrate properly. So, I'll probably put it in my system prompt in some way or my global agents MD. There's a lot of places you can put it, but as long as you give Claude code the ability to know what you mean by Soul Terra and other things, or you just specify them correctly here, you're good to go.

There's a couple other small catches. Like for example, here the 5 6 models don't report how many tokens they're using until the work is done. Fable will give you live updates about how many tokens it's using. It's a small thing, just one of the few I noticed. Overall, I've actually been really impressed with this. And I already see the comments coming in, so I'm going to address them now. Why not open code or Pi or O my Pi or whatever harness is popular today?

The reason is because they don't solve the problem I care about. They're sure their system prompts are less bad than Codex is, but they don't solve the sub agent orchestration problem, which is the thing I am most excited about here. O my Pi, absolute slop. It froze my terminal for 2 minutes when I installed it because they thought it was a good idea to post their entire 150 page change log in formatted markdown in my terminal, destroying my entire scroll back, and just destroying it entirely.

It was real bad. Pi is great, but you're going to have to build all of these workflows and orchestration tools yourself. Open code has a bunch of hard-coded sub agents that aren't very good. The V2 of Open Code might be better. I have not explored it at all yet, but none of those have workflows. And to be real, and to be very, very clear cuz I started here, I did this for workflows. The workflows in Claude Code are the best implementation I have seen of orchestrating sub-agents on real day-to-day work for people who are employed that I've ever seen, by far. If you want to have an agent break up other agents with other models to do complex work, to take turns and have phases throughout, to kick off things that matter and ignore things that don't, and have an actual end, workflows are the best solution I have found for this by far. And nothing else comes close. To be very clear, I don't think workflows are the super complex feature that no one else can implement.

In fact, Codex told me to just add it to Codex myself when I asked it about that versus using it in Claude Code. You can absolutely do this. That all said, Claude Code has this figured out. They have a good system for it built in. They have good UI built for it, too, in the terminal, at least. The desktop app, less so. They have a system prompt that properly steers models to do it right, and they have a harness that is good enough. One last cool thing that I'm sure you probably could have guessed, CLI proxy API supports things other than Claude Code

and Codex and subs from those platforms. So, if you want to bring in subscriptions from other places, like a cough cough, Grok Build, because you have an existing Twitter Premium account, you can here.

It's actually a pretty good value. You get a slightly absurd amount of usage out of the $20 a month Twitter plan if you use it with Grok Build. And if you don't want to use Grok Build, and you want to use it in other things, you can now get Opus 4.5 in Claude Code by just adding it here. I had a lot of fun with this one. And again, to be clear, I don't think everyone should go exclusively use their Codex sub through Claude Code. I just thought it was a fun experiment with results that were much better than I expected. And now I'm just curious if you guys are interested enough to do the same yourself. Am I insane for going this deep, or is this actually kind of fun and interesting?

Are you going to set it up yourself, or maybe you're just going to point your agent at this video and ask it to do it for you? Let me know how all of that goes. I'm genuinely curious. I'll keep an eye on the comment section, but until next time, peace nerds.