CodeToDeploy · Follow publication

Member-only story

Claude-code vs Codex-CLI

7 min read · Oct 14, 2025

Kapil Viren Ahuja Following

The Showdown…

There is this trend on YT. A new model is released, and a barrage of videos conveys the standard message: this latest model outperforms everything else. Hardly any are posted with depth, and almost all are intended to gain viral views. While these are good as a knowledge source, I need to use these tools on a day-to-day basis, and hence, I need to check what works best.

I came up with five parameters to keep it simple.

1. Efficiency: How many follow-up prompts does the tool need
2. Token consumption: The lower the better
3. Dev Experience: Work within the tooling close to the developers
4. Accuracy: The output should give me the most accurate results
5. Speed: How fast was the model able to give me the output

. . .

Test 1: Working with GH Issues

Comparison Scenario

~~~
I need only open issues in the following format. Also fetch any epic without ch

  ```
  {Issue Type} #{number}
  ├── {Child Type} #{number}: {title} ({state}) (@{assignee if open})
  │   └── {Grandchild Type} #{number}: {title} (CLOSED ✅ or OPEN @{assignee})
  │
  └── {Child Type} #{number}: {title} ({state}) (@{assignee if open})
      └── {Grandchild Type} #{number}: {title} (CLOSED ✅ or OPEN @{assignee})
  ```

And as the command was running, I added another prompt.

Show me epics in the ascending order of the issue number
~~~

Claude-code

1. Demonstrated exceptional efficiency across multiple dimensions. It executed commands **4x faster.**

2. The tool created a custom script for repeated operations rather than making redundant API calls, resulting in **significantly lower token consumption**

3. From a user experience perspective, it required **minimal interaction**, adopting a more efficient approach that didn't burden users with unnecessary clarifications.

Codex-cli

1. It took a more straightforward but resource-intensive approach, as it repeatedly sent individual API calls to GitHub CLI instead of optimizing the process.

2. This approach also resulted in **substantially higher token usage** The tool handled all formatting operations directly within its invocation rather than creating reusable scripts.

Winner: Claude-code

Claude-code wins the comparison with **5 points vs 0 points**, excelling in all the metrics.

. . .

Test 2: Reviewing Pull Requests

Comparison Scenario

My team raised a PR for me to review. It was a straightforward one.

Claude-code

1. Gave me a very verbose output, but it covered all the dimensions of the code. It also gave me its interpretation of whether the PR should be approved or not.
2. The CLI command for starting the PR was simple enough. I gave it a PR number and it did the job.
3. It took a while, not too much though. I wish it were faster.

Codex-cli

1. The token usage was low, but the output was shallow.
2. The command in the CLI was also not good.
3. Codex discusses its GitHub integration, which automates review processes. (Claude has something similar to), But here I want to stay close to the developer's toolsets

Winner: Claude-code

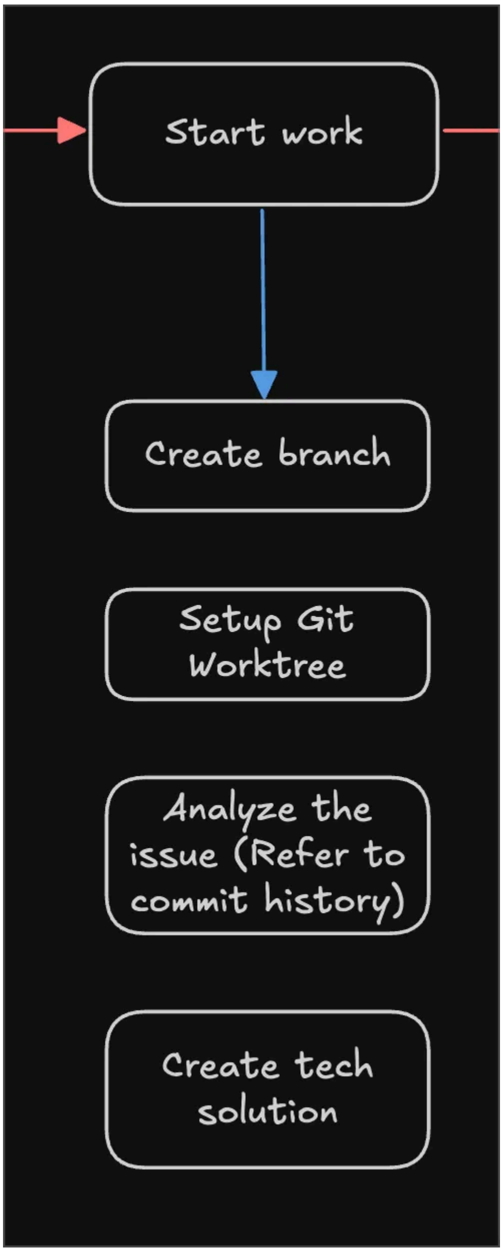
Claude-code wins the comparison with 4 **points vs 1 point.** Codex wins on the token usage.

. . .

Test 3: Fixing Simple Bugs

Comparison Scenario

Specification generation for Bug fixing followed by Coding. I wanted to fix (not so complicated) bug in my backend Java code around Login, where the account would not lock out. Alternatively, it could be a new feature, depending on your perspective. I have a detailed workflow, but I will focus on just one part here. My goal was to generate the spec for my project, which is unique as I am writing MD files, not code.



BugFix (Start Work) tasks

. . .

Claude-code

Used Open 4.1, in Plan mode with Thinking On

1. It took longer than expected, but I expected that, as I had Opus and thinking mode both active.
2. The first time, it swayed from my standards and tried to generate the spec as it seemed fit. I had to course correct it by giving it two more prompts and providing a reference to my documentation.
3. The outcome was excellent and very close to the structure that I had created. I asked it to make a temporary set of files that I could review, and it was very close to the outcome.
4. Didn't use many tokens. Since the release of Claude-Code 2.0.0, I have observed an improvement in token usage, which may be a contributing factor.

Codex-cli

Used Codex version 0.44. Model: GPT-5 Codex with high reasoning.

1. They were eager to provide me with the details. It finished so quickly that I couldn't believe it.
2. The output was very crisp. It gave me five bullet points. And even when I referenced it in my existing commands, it wasn't generating details. Felt like Codex was in a hurry to provide me with the output.

Winner: Claude-code

Claude-code wins the comparison with 4 **points vs 1 point.** Codex wins on speed, but given how poor the output was, I was inclined not to provide it with that one point, either.

. . .

Test 2: Writing an end-to-end Login implementation

Comparison Scenario

I defined a feature for login that required a complete end-to-end implementation, encompassing frontend, backend, security, session management, and everything. The specification was a single liner sitting in GH, and I wanted to see how much the underlying model and harness would guide me through the implementation.

Claude-code

1. The sub-agent setup worked very well. I was able to delegate each task to a very specific sub-agent, and each sub-agent knew precisely what it had to do.
2. The context management was also excellent. I didn't run out of context a lot. I pretty much had to reset, I think, just a couple of times throughout the entire implementation.
3. Until I started writing code, the model worked quickly. Generating code, especially writing the tests, was much slower than I had experienced in the past.

Codex-cli

1. Codex continues to be relatively aggressive and quick. It's like it's in a hurry to get me through the answer.
2. Speed was nice, even when I was in high thinking mode.
3. The quality of the specification and plan on Codex was higher and better than what Sonnet gave me. Opus, I would probably go hand-in-hand, but Opus then really dunked my usage. So I would test more with Codex, but that seems relatively better.

Winner: Claude-code

Claude-code wins the comparison with 3 **points vs 2 points.** Codex wins on the speed and quality of output for specs and efficiency.

. . .

As I continue to use both, I would appreciate it if Codex provided me with slash commands. Sub-0 agents will be friendly too. However, I have found a workaround where I can force ptompts to recognize the agents and make them work, but having them run consistently all the time would be very nice.

I generally reset the context after a significant chunk of code. CodeXS has a much bigger context window, but I'm in a habit of resetting it quite often, so I'm really unable to test if that big context window would help or not. I think I'll have to do that test completely separately, remembering not to hit the reset button. That's becoming a memory muscle for me now.

Conclusion

I will continue to use Claude-code with Sonnet and Opus in different combinations for now. The Codex model seems really good, but for my personal use, the Codex CLI struggles a bit. I don't mind switching over to FactoryAI and plugging Codex into it, which is where the Codex model really shines.

However, if I need to stick to a single ecosystem, then the Claude-code ecosystem works well, as FactoryAI does not currently have sub-agents. This makes my way of writing prompts and building my entire subsystem more effective, as it tends to fail elsewhere. I might need to find a way to contextualize everything to make it work across the board, which is possible—I absolutely can do it.

However, I feel I currently have an excellent system that delivers high-quality code, and for my workflow—where I can fire off two or walk away, and then ask it when to approve or to continue—it works really well with the Claude Code setup. So, that remains my preference for now.

. . .

Thank you for being a part of the community

Before you go:

👉 Be sure to **clap** and **follow** the writer 👏

👉 Follow us: **X** | **Medium**

👉 CodeToDeploy Tech Community is live on Discord — **Join now!**

👉 **Follow our publication, CodeToDeploy**

Note: This Post may contain affiliate links.

Generative Ai Tools · Thought Leadership · Code · Software Architecture · Technology

Follow

Published in CodeToDeploy

1.2K followers · Last published 7 hours ago

The First Publication That Blends Tech Insights + Real Job Opportunities

Following

Written by Kapil Viren Ahuja

95 followers · 38 following

Creating Architectural POVs | Nurturing architects of the future.

No responses yet

Bgerby

What are your thoughts?

More from Kapil Viren Ahuja and CodeToDeploy

In Generative AI by Kapil Viren Ahuja

Top-Down AI Adoption Beats Bottom-Up Every Time (Except When It Doesn't)

I have been watching leaders struggle with AI adoption for years now, and I need to tell you something that might sound contradictory...

Sep 15 57

In CodeToDeploy by TechToFit - Master Your Life with Tech

I Took Google's 9-Hour AI Course. Here's What You Need.

The four frameworks that unlock expert-level AI conversations and save you hours of guesswork.

Oct 26 685 11

In CodeToDeploy by Neha Singh

Best Free Alternatives to Paid AI Tools 🚀

Amazing free AI tools that are just as powerful as their paid counterparts...

Oct 12 524 10

In Towards AI by Kapil Viren Ahuja

CTO’s ADAPT System: My Five Bets for the Agentic Engineering Era

A Principal’s Bets for Thriving in the Agentic Engineering Era

✦ Oct 5 68

See all from Kapil Viren Ahuja

See all from CodeToDeploy

Recommended from Medium

Ashish Sharda

I Tested GLM-4.6 for 2 Weeks and Went Back to Claude. Here’s Why.

The $200/month AI model beat the $30/month alternative. Sometimes expensive is worth it.

✦ 2d ago

In JavaScript in Plain English by Marshfire

MCP in the Age of LLMs — How VS Code, AI, and Dev Tools Use Context

If Part 1 was the "why," this one's the "how."

Oct 24 22

Larry W Jordan Jr.

Building at the Speed of Thought: A Claude Code Plugin That Actually Gets How You Work

Why I Created a Full-Stack Claude Code Plugin with 81 Commands and 12 Specialized AI Agents

Oct 12 22 2

In Vibe Coding by Alex Dunlop

I Stopped Typing and Started Speaking My Way to 4x Productivity (99% of Developers Need This)

The Voice Revolution

✦ 4d ago 53 2

In Better Dev—NextJs/React by Melvin Prince

Next.js vs SvelteKit in 2025: Ecosystem Power vs Pure Performance

How React’s heavyweight compares to the compiler-first challenger.

✦ Sep 29

In Level Up Coding by Swathi Ganesh

13 Rare Python Packages That Make AI Feel Like Magic

Rare yet the best python packages to discover

✦ 3d ago 👏 43

See more recommendations