MY WEIRD PROMPTS

Podcast Transcript

EPISODE #154

From Apps to Agents: Building Your Digital Workforce

Published January 04, 2026 • Runtime: 23:49

https://myweirdprompts.com/episode/ai-agentic-workflows-evolution/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn dive deep into the rapidly evolving world of agentic AI as of early 2026. They break down the crucial differences between reactive custom GPTs and autonomous multi-agent workflows, exploring how tools like Claude Code and N8N are reshaping productivity. From the architectural debate between serverless hosting and local "agent boxes" to the essential strategies for preventing token-burning infinite loops, this episode provides a practical roadmap for anyone looking to build a secure, scalable, and cost-effective digital workforce.

DANIEL'S PROMPT

Daniel

Agentic AI is a very relevant topic, but it seems like people are grouping many different things under the 'agent' umbrella. Where does the distinction lie between a custom GPT and a more complex agentic workflow that orchestrates sub-agents? Furthermore, where should these persistent agents be deployed, and what mechanisms exist for cost control when you can't always predict when or how often an agent will run?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. We are at episode two hundred sixty-one, and I have to say, the start of twenty twenty-six has been pretty wild so far. I am Corn, and sitting across from me as always is my brother, the man who probably has more browser tabs open than most small businesses.

Herman

Herman Poppleberry here, and you are not wrong, Corn. My computer is currently screaming for mercy, but the research is just too good to close. We have a fantastic prompt today from our housemate Daniel. He has been deep in the trenches of building things lately, and he is hitting on something that I think a lot of people are scratching their heads about right now.

Corn

Yeah, Daniel was actually showing me some of his setups in the kitchen this morning while the coffee was brewing. He has been working with agentic AI workflows, using things like N-eight-N and Claude Code, and he is feeling that classic tension. On one hand, you have these simple custom GPTs that everyone uses, and on the other, you have these massive, multi-agent orchestrations that feel like they are building a digital workforce.

Herman

It is a huge distinction, and honestly, the industry has done a pretty poor job of defining what an agent actually is. People just throw the word agent at anything that has a system prompt these days. Daniel is asking where that line is drawn, where these things should actually live, and how we keep them from accidentally spending our entire life savings on API tokens while we are sleeping.

Corn

That runaway cost fear is real. I remember back in episode two hundred thirty when we talked about the agentic dilemma and the idea of a kill switch. But today, I want to get more practical. Let us start with that first part of the prompt. What is the actual difference between a custom GPT and a bona fide agentic workflow? Because to a casual user, they might look the same, right? You type a prompt, and stuff happens.

Herman

Exactly, and that is where the confusion starts. A custom GPT, at its core, is basically just a wrapper. You have a large language model, you give it a personality or a set of instructions in the system prompt, and maybe you give it access to a few tools like a web search or an image generator. But it is fundamentally reactive. It waits for you to say something, it processes that one turn, and it gives you a response. It is a very linear, one-to-one interaction.

Corn

So it is like a very smart clerk. You go to the window, you ask for a document, the clerk goes and gets it, and then they sit back down and wait for the next person.

Herman

Precisely. Now, what Daniel is building with things like N-eight-N, that is where we move into agentic workflows. The key difference here is autonomy and the loop. An agentic workflow is not just waiting for a prompt; it is often triggered by an event. Daniel mentioned his news summary agent that runs once a day. That is a proactive system. It wakes up on its own, it looks at the news, it decides what is important based on its internal logic, and it executes a series of steps.

Corn

But wait, is a scheduled script an agent? If I just write a Python script that scrapes a website every morning, is that an agent? I feel like we are missing the intelligence component in that definition.

Herman

That is a great point. The intelligence is the brain. In a traditional automation, like a standard Zapier flow from five years ago, it is all if-this-then-that. It is rigid. If the website format changes by one pixel, the script breaks. An agentic workflow uses an LLM as the reasoning engine at each step. So, instead of saying "scrape this specific HTML tag," you are telling the agent, "find the top three stories about renewable energy today." The agent then has to reason through the search results, evaluate the content, and decide what makes the cut.

Corn

Okay, so the distinction is the degree of reasoning and the ability to handle ambiguity. A custom GPT follows a recipe. An agentic workflow is like a chef who knows the goal is a three-course meal and can adjust if they find out the store is out of onions.

Herman

I love that analogy. And then we go one step further into what Daniel called complex agentic workflows that orchestrate sub-agents. This is the manager-worker pattern. Imagine one agent whose only job is to be the architect. It receives a complex goal, like "build me a marketing campaign for a new coffee brand." The architect agent does not do the writing. Instead, it breaks that goal into sub-tasks and spins up or calls upon specialized sub-agents. One for copy, one for image generation, one for market research.

Corn

This is where it gets really interesting to me, but also where I start to see the potential for chaos. If you have agents talking to agents, you are basically creating a small, digital company. But in twenty twenty-six, how reliable are these sub-agent handoffs? I mean, we have all had that experience where you play a game of telephone and the message gets garbled by the third person.

Herman

It is the biggest bottleneck right now. The handoff. In twenty twenty-four and twenty twenty-five, we saw a lot of these systems fail because the manager agent would give vague instructions to the sub-agent, and the sub-agent would hallucinate a solution. But now, with models like Claude three point five and the newer iterations we are seeing this year, the instruction-following is so much sharper. We are using things like structured output, where the manager agent has to provide the sub-agent with a very specific JSON schema. It is less like a conversation and more like an API call wrapped in natural language reasoning.

Corn

So it is about adding guardrails to the autonomy. Daniel mentioned Claude Code specifically. That is a great example of a tool that feels very agentic because it is actually looking at your file system, running tests, seeing what failed, and then iterating on its own code. It is a closed loop. It is not just suggesting a snippet; it is trying to solve the problem until the tests pass.

Herman

And that brings us to the second part of Daniel's question, which is where these things should live. This is a huge architectural debate right now. If you are running a simple custom GPT, it lives on the provider's servers, like OpenAI or Anthropic. You do not worry about the infrastructure. But once you start building these persistent, long-running agentic workflows, you have to decide on a home for them.

Corn

Right, because if an agent is supposed to be monitoring something twenty-four-seven, or if it takes ten minutes to reason through a complex task, you can't just run that in a browser tab. You need a server. But do you want a persistent server that you pay for every month, or something serverless?

Herman

Let us break that down. For something like Daniel's news agent, which runs once a day, serverless is the obvious winner. You use something like AWS Lambda or a Vercel function. The code wakes up, runs for two minutes, the agent does its thing, and then it vanishes. You only pay for those two minutes. It is incredibly cost-effective.

Corn

But what about the more complex ones? The ones that need to maintain state? Like an agent that is managing a long-term project and needs to remember what happened three days ago without re-reading five thousand tokens of history every time it wakes up.

Herman

That is the challenge. For persistent agents, we are seeing a shift toward what people are calling agentic runtimes. These are specialized environments, often running in Docker containers, that stay "warm." They keep the agent's memory and the current state of the workspace active. If you are doing heavy coding work with something like Claude Code, you want it to have immediate access to your local environment. But if you are deploying this for a company, you might use a platform that spins up a dedicated virtual machine for that agent session.

Corn

It feels like we are moving back toward the idea of a personal server, just a very specialized one. I remember in episode two hundred fifty-eight, when we talked about the gigabit bottleneck and home networking. It makes me wonder if more people are going to start running "agent boxes" in their homes. Just a little high-powered NUC or a Mac Mini that sits in the closet and hosts all their personal agents so they don't have to pay cloud hosting fees.

Herman

I think that is a very real trend for twenty twenty-six. Especially with local models getting so good. You can run a very capable Llama three or Mistral model locally now. If you have an agent that is handling sensitive data, like your personal emails or finances, you probably don't want that running on a random server in Virginia. You want it in your house, on your hardware.

Corn

Speaking of paying for things, let us take a quick break for our sponsors. Larry: Are you tired of your dreams being just... dreams? Do you wake up feeling like your subconscious is underperforming? Introducing the Oneironautics Helmet by Dream-Stream. This revolutionary head-mounted device uses low-frequency vibrations and a proprietary blend of synthetic lavender and ozone to "guide" your dreams toward maximum productivity. Why sleep when you could be conceptualizing? Users report a forty percent increase in "imaginary output" and only a slight, temporary loss of the ability to distinguish the color blue. The Oneironautics Helmet. Sleep smarter, not deeper. BUY NOW!

Herman

...I really worry about Larry sometimes. Synthetic lavender and ozone? That sounds like a recipe for a very strange headache.

Corn

And the color blue? That is a high price to pay for productivity. Anyway, back to the world of agents. We were talking about deployment, but the real elephant in the room that Daniel mentioned is cost control. This is the thing that keeps people from hitting the "deploy" button on a multi-agent system.

Herman

It is the "infinite loop" problem. In traditional coding, an infinite loop might crash your program or max out your CPU. In agentic AI, an infinite loop can cost you five hundred dollars in twenty minutes. If two agents get into an argument or keep asking each other for clarification without a "stop" condition, they will just keep burning tokens.

Corn

I have seen that happen! It is like two polite people at a door saying "no, after you," but every time they say it, it costs ten cents. So, what are the mechanisms for control? How do we build "fiscal guardrails" into these systems?

Herman

There are a few layers to this. The first and most basic is the token cap per run. Every time you trigger an agent, you should have a hard limit on how many tokens it is allowed to consume for that specific task. If it hits the limit, it has to stop and ask for permission to continue. It is like giving your teenager a debit card with a twenty-dollar limit instead of your primary credit card.

Corn

That makes sense for a single run. But what about Daniel's concern about unpredictability? You don't always know how often an agent will need to run if it is responding to external triggers, like customer emails or market changes.

Herman

That is where budget-based rate limiting comes in. You need a middleman. Instead of your agent talking directly to the OpenAI or Anthropic API, it talks to a proxy. That proxy tracks your spending in real-time across all your agents. You set a daily or monthly budget, say, fifty dollars. Once that fifty dollars is hit, the proxy simply shuts down the API keys. Everything stops. It is better to have a broken agent than a three-thousand-dollar bill.

Corn

I also think there is a design element here. We need to move away from "open-ended" agents and toward "task-specific" agents. If an agent has a very narrow scope, it is much less likely to wander off into a token-burning forest.

Herman

Absolutely. This is the concept of "constrained agency." You don't give the agent the goal of "manage my social media." You give it the goal of "summarize this one blog post into three tweets." By narrowing the objective, you narrow the search space and the potential for runaway reasoning. Also, we are seeing more use of "human-in-the-loop" checkpoints. For any task that is estimated to cost more than a certain amount, the agent has to send a notification to your phone or Slack. "Hey, I can do this, but it is going to take about five thousand tokens. Do you want me to proceed?"

Corn

That feels like the most practical solution for most people. It is the "are you sure?" button for the AI age. I also want to touch on the "orchestration" part of Daniel's prompt. When you have sub-agents, do they all need to be the "smartest" and most expensive models?

Herman

Definitely not, and that is a huge cost-saving strategy. You use the "big brain" models, like Claude three point five Sonnet or GPT-four-o, for the manager agent—the one doing the high-level reasoning and planning. But for the sub-tasks, like formatting text, extracting dates, or simple data entry, you use much smaller, cheaper models. You might use a specialized Llama-three-eight-B model or a "flash" model from Google. Those are literally a hundred times cheaper.

Corn

So, you are building a team where you have one highly paid consultant overseeing a group of very efficient, specialized workers. That is a much more sustainable model than just throwing the most expensive model at every single comma and period.

Herman

Exactly. And in twenty twenty-six, the "routing" technology has gotten really good. There are frameworks now that automatically look at a task and decide which model is the most cost-effective one to handle it. It is like an automated triage system for AI.

Corn

Let us talk about the "persistent" part of the agents. Daniel mentioned persistent agents being deployed. What does "persistence" actually mean in this context? Is it just memory, or is it something more?

Herman

It is both memory and state. A persistent agent has a "soul" that lives on between sessions. In twenty twenty-five, we saw the rise of "vector database memory," where an agent could look up things it did months ago. But in twenty twenty-six, we are moving toward "graph-based memory." Instead of just finding similar text, the agent understands the relationships. It knows that "Project X" is related to "Client Y" and that "Client Y" has a preference for concise emails.

Corn

That makes the agent so much more useful over time. It is not just starting from scratch every morning. It is building a knowledge base of how you work. But that also adds to the cost, right? Because every time the agent wakes up, it has to "load" that context.

Herman

It does, which is why "context caching" has been such a game-changer this past year. The API providers finally realized that if we are sending the same three thousand words of "background info" every time, they should just cache it on their end and charge us a fraction of the price to reference it. It makes those "persistent" agents actually affordable.

Corn

That is a huge technical detail that I think a lot of people overlook. If you are building an agentic workflow today, and you are not using context caching, you are basically throwing money away.

Herman

Precisely. Now, looking at Daniel's specific setup with N-eight-N and Claude Code. N-eight-N is fantastic because it gives you that visual map of where the data is going. You can see the nodes, you can see the logic, and you can inject "manual approval" steps very easily. It is the perfect bridge between "pure code" and "pure autonomy."

Corn

I think that is the sweet spot for most developers right now. You don't want a "black box" where you just hope the agent does the right thing. You want a "glass box" where you can see the reasoning, intervene if necessary, and have clear visibility into the costs.

Herman

And Claude Code represents the other side of that—the "embedded" agent. It is right there in your terminal. It has the context of your entire codebase. The synergy between those two—a high-level orchestrator like N-eight-N and a deep-dive worker like Claude Code—is really the blueprint for a modern AI-augmented workflow.

Corn

So, if we were to give Daniel some concrete takeaways for his project, what would they be? I mean, he is already doing the work, but how does he level it up for twenty twenty-six?

Herman

First, I would say: define your "Agentic Boundary." Not everything needs to be an agent. If a simple script can do it, use a script. Save the "reasoning" for the parts that actually need it. Second, implement a "Proxy Layer" for cost control immediately. Don't wait for a surprise bill to realize you need a budget cap.

Corn

And third, I would add: diversify your models. Don't use a "god model" for everything. Look at your sub-tasks and see which ones can be handled by smaller, faster, cheaper models. It is better for your wallet and often faster for the workflow.

Herman

Also, don't sleep on "Local Hosting" for the persistent parts. If you have an agent that needs to be "on" all the time, consider a dedicated home server or a low-cost VPS rather than a high-end serverless function that might get expensive with high invocation counts.

Corn

This has been a really deep dive, and I think it is such a perfect snapshot of where we are right now. We have moved past the "can it do it?" phase and into the "how do we make it sustainable, reliable, and affordable?" phase.

Herman

It is the "industrialization" of AI. We are building the factories and the supply chains now, not just the prototypes. It is a very exciting time to be a builder, even if it is a bit confusing with all the jargon.

Corn

Well, I think we have cleared up at least a little bit of that jargon today. Daniel, thanks for the prompt—keep us posted on how that news agent evolves. If it starts picking up stories about Larry's helmet, we might need to have a serious talk about its filtering logic.

Herman

Or we might need to buy some helmets, Corn. Think of the productivity!

Corn

I like the color blue too much, Herman. I'm not risking it. Anyway, if you have been enjoying the show, we would really appreciate it if you could leave us a review on Spotify or whatever podcast app you are using. It genuinely helps the show reach more people who are interested in these kinds of deep dives.

Herman

It really does. And remember, you can find all our past episodes and a way to get in touch with us at our website, myweirdprompts.com. We love hearing from you, whether you are a long-time listener or just joining us for episode two hundred sixty-one.

Corn

Thanks for listening to My Weird Prompts. We will be back next week with another exploration into the strange and wonderful world of technology and beyond.

Herman

This has been Herman Poppleberry and Corn. Until next time, stay curious!

Corn

See ya!