Claude Code Skills + Computer Use Agents Will Make New Millionaires in 2026, Heres How

20 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=AQKV-GT9FRY
https://www.youtube.com/watch?v=aqKv-gt9FRY

SUMMARY

The video discusses the introduction of a new system that allows coding agents like Cloud Code and Gemini 3 to access their own browser tools, enabling them to automate tasks in ways that were previously impossible. The presenter, Kev, explains the framework he created called the II framework, which facilitates the development of autonomous agents capable of running workflows, improving themselves, and operating continuously without the need for coding knowledge. He demonstrates how to build an Instagram sales agent that automates outbound messaging to warm leads, showcasing the potential for significant business automation and profitability.

- *Coding agents now have access to browser tools, allowing for unprecedented automation capabilities.*
- *The II framework enables agents to run workflows autonomously and improve over time.*
- *The Instagram sales agent example automates direct messaging to users who have engaged with posts, enhancing lead generation.*
- *The system uses a headless browser (Playwright) for reliable task execution without visual errors.*
- *Agents can store results in a database for tracking and debugging purposes.*
- *The framework allows for easy adaptation to various applications (e.g., Gmail, Shopify) beyond Instagram.*
- *Users can follow along with a beginner-friendly tutorial to set up their own agents without prior coding experience.*
- *The entire codebase for the Instagram agent is available in the No Code Academy for users to customize and implement.*

What's going on guys? So recently I gave my coding agents like Cloud Code and Gemini 3 access to their own computer browser tools and now they can automate anything. This new system of giving coding agents access to their own browser tools and automated workflows are creating brand new automations that were literally never possible before. They also run 24/7 improve themselves on the fly. And the best part is I found a solution to do this without costing any money. And no other AI expert on the internet has shown you what I'm about to show you today. In this video, I'm going to explain how the Aentic browser tools actually works inside of the framework that I created called the II framework and why this is going to make a bunch of people really, really rich by applying these new type of agents to businesses.

After that, we're going to go ahead and we're going to build our first agentic AI workflow together that accesses its own browser and runs commands on its own and improves itself on its own. You guys do not need to know how to code to actually follow along in this tutorial. I designed this specific video in the most beginner-friendly way because after being in AI for the past 3 years, I'm telling you with absolute confidence that this is going to run the entire economy and is the future of AI automation. And by the end of this video, you'll set up your

Agentic computer use agents alongside me. My name is Kev. I'm the founder of Creator OS, which is an AI operating system that runs all of these Agentic workflows that are tailored designed for specific business use cases. And this is just going to take things to a whole another level. And I think the best way to explain to you guys just how powerful the future of automation is going to become with these browser tools, I'm just going to show you one of the agents that I'm working on to completely automate Instagram outbound reach. This is what I'm calling my Instagram sales agent. And the way that this agent works is it's going to use an entire database worth of warm leads. Now, these warm leads are people that have engaged with our posts in the past, aka from liking, from sharing, or from commenting. And that means that these people that my agent is now sending these DMs to are actually very warm leads that could potentially be interested in what I have to offer. And platforms like ManyHat or Instagram's API literally does not offer a feature like this. They offer like a comment to DM feature, for example. But this takes things to a whole new level because we are scraping the people that have engaged with our posts. We are then getting their profile information and enriching that profile information.

We're then feeding it to this agentic agent that has access to its own browser and then it's autonomously going to run this DM campaign on our behalf multiple times a day. Now, this is also a sped up version just to show you guys how this initial system works. But the agent also uses the browser as if a normal person would, adding certain blocks and certain breaks in between their outbound campaign so that we don't get flagged on Instagram and so that we can scale this to a whole bunch of clients. And the way that the agent then shows us all of its work is by having access to our database. Once it finishes up that workflow that you just saw, it just adds its results inside of this Instagram DM agent card so that we can track its performance without us actually having to watch the agent go and do that task.

And in this way, it's very scalable for our AI agencies. All right. So now I'm going to run you guys through a presentation that's going to break down exactly how this works, the tech stack that is needed to set this up. And I just want to also let you guys know that the Agentic operating system codebase that has that Instagram DM agent is actually available in my no code academy on school. So if you would like, head in there, grab the entire codebase and in that way you're going to actually have that Instagram example set up as well.

So today's overview is going to follow this six-step road map. We're going to break down the tech stack that actually creates the entire autonomous browsing agent workflow. I'm then going to explain how we're using something called a headless browser and how headless browser agents compared to like cloud codes computer use that we're seeing or any of these other mainstream solutions are actually worse than the headless browser solution I'm going to show you guys today. I'm going to run you guys through the II framework and how this actually makes our headless browser computer use agents actually, you know, do all of this work in a consistent way that also improves itself. So, I'm going to do my best to explain how this framework actually runs. And then we're going to go into a specific workflow and I'm going to show you guys how that specific workflow actually runs, the behind the scenes of how the agent thinks about the necessary steps to complete the actions. And then we're going to go through exactly what we're going to build out together, which was that Instagram sales agent that you saw in the beginning of this video. After that, you and I are actually going to build out this first agentic computer browser workflow. So, let's get into the rest of the breakdown. Like and subscribe and let's uh let's do it, baby. So, the text stack that I recommend using here, guys, is cursor as your IDE environment, Claude Code, and the anthropic API that we're going to use for our workflow. And then we're going to be using something

called the Playright MCP server. Now, Playright is a Microsoft application that lets you run any type of browsing task on the internet. Now, the cool thing about Playright is that it's completely free.

It's open- sourced and it works in a very unique way compared to the traditional way that everyone else is showing you guys today. And then when you have this entire setup actually inside of your cursor environment, you can log into any provider, any software application that you want to automate. So you want to log into like Gmail and then your Playright agents will have access to your Gmail login or Instagram or Facebook or literally any type of software that you want to automate. Now I also added this elephant because this is our database, our Postgress database.

So when our agents are actually conducting automations, we want them to store its results in a database because then we can show on the front end what's actually going on and in this way we have a very easy debug process because the output that we get is the only thing that we need to actually improve the workflow manually. However, the agent also autonomously will improve itself based on the goals that you provide. So now I want to explain why Claude Code's computer use tool is actually not what we're using and why we're using Playright plus anthropics API to have very consistent outcomes compared to the AI vision agents. So the first reason why that this workflow is going to change the game is because the playwright agent doesn't actually view a screen and it actually views the screen as code and its ability to understand code versus understanding pictures on a screen is much better. The reliability of this is 99.9%.

And it does this by understanding the different CSS selectors inside of a page and selecting exactly what we're instructing it to select. The other benefit of this workflow is that when your Playright agent runs into an error, it has a command to take a screenshot of the browser and it'll take that screenshot and feed it back to our enthropic API. And the enthropic agent will then read the browser. Let's say it was trying to click on a button and it can't find the button. So then it takes a screenshot, sends it back to Enthropic. Enthropic then reads the screenshot and instructs the playright agent on how to actually select the correct element on the screen. Once that solution is actually solved, then the workflow using the II framework saves that run. And so in a simplified way, this is how the II framework for autonomous computer browser agents actually works. So let's say that we were doing a email scrape to a instantly outbound automation. Inside of the information MD folder is where the cloud code skills will lie and in this skill we will be instructing it of exactly how we want this workflow to operate. So the so the agent would write something like you know this is a automated workflow to go and scrape leads. This is how you do it. And what this information MD folder also does is it's keeping track of all of the previous runs. So let's say I was running this agent a 100 times. After those 100 runs, the agent itself will be able to self-improve the information MD folder so that it'll run into errors left less often. The agent will also be given the implementation Python script.

Now the Python script is actually the real code that the agent needs to run to actually make this workflow operate. And in this case, it's actually sending it the code instructions that the Playright MCP has access to. And so to have your agent set up in this example, it'll have the code instructions from the Python script. It'll have database access via the database API and it'll have login credentials to any type of application that we need to give it access to. So now when this agent actually runs, it's either going to succeed or fail. If it succeeds, then that

means this entire workflow actually conducted and the output of the workflow will be stored in the database. And so an example of a success run for an email scrape to email outbound is to keep track of all the leads and which leads have been contacted so far, which leads opened up their email so far. But let's say the agent runs the automation and runs into an error. Well, in this case, the agent can just take a screenshot and send it back to the anthropic API, which will then rewrite the information MD folder once it figures out the solution. And in this way, the system just keeps going and keeps going. Now, let's break down this exact example, but in in depth. So, we're going to start off with the file.

Uh, so we're going to start off with the information MD file. Another way to think of this is this is the brain for our AI agents, and it's going to have the goal of the workflow, the context of the workflow, and learned constraints or memory. it'll have persistent memory based on the previous runs. And so in this example, let's say we were scraping dentists in Toronto. And the dentists that we're scraping, let's say after running it 10 times, has come up with a couple errors in some of those runs.

Well, the agent will actually send the information to the AI agent architect, which will then use the implementation Python script to run the code. The Python script explains how to use the playwright MCP tools which uses the live internet to conduct any task. If the process works, it goes back to the architect agent and this loop just keeps running on whatever schedule that we want. If it doesn't work, then the agent takes a screenshot and sends it back to the information MD folder for it to write the new constraint and try the entire workflow again. So now let's talk about the exact use case that we're going to be building out today, which is going to be the Instagram lead engagement system. So the lead generation database lives in the same database that this agent will be performing its tasks and exe and and storing its results in. And this database can be storing all of the people that have liked our Instagram posts, for example. So this system will store leads on the agents behalf and then we can enrich those leads by running it through like another appy type of scrape to then get their profile information, their location, uh any other type of details that could be associated in their bio. This allows us to then write customized outbound messages to these new leads. And so the way that this system works is by giving the architect agent login access to our Instagram account and then it'll use the cookies and save the cookies so that every time we want to run this Instagram DM system, it'll just pop it up and we don't have to log in or do any 2FA ever again. And once again over time this specific workflow will identify the certain constraints. So after running this for a little bit, I can tell you guys about the constraints. So each Instagram account can send up to 200 outbound messages per Instagram account.

Up to 10 DMs per hour so that you don't get flagged. Now let's talk about how this Agentic browser tool system is going to expand into any automated workflow that we can possibly think of. So, in the previous example, I showed you that we logged into Instagram and then we worked on that workflow until it was solidified, until it was working perfectly, like the example that we saw in the intro, and then we just let it run. Well, now imagine we do this for Gmail, we do this for Shopify, we do this for our investments, we do this for we do this for Only Fans creators. Like, the list goes on and on. And the exact same setup works for each one of these workflows where we pass the access credentials aka API keys or login information to the architect agent which will have the goals, the context and the constraints of said workflow. It'll then use the session token that it that it has access to to log in to this service provider and conduct any solution that we want. And this right here is what is turning what was previously impossible with software automation into possibilities for the first time ever.

Right? So these were all things that were not possible through a simple API key. We needed to have API keys plus user credentials and access to a computer browser. Now guys, we get to go ahead and build your first Aentic computer use agent together from scratch. Now, there's going to be two ways that you guys can follow along from this moment on. If you're part of the No Code Academy, you're going to have access to the Agentic OS codebase. So, you can go in and grab that. And if you don't want to join the academy, then in the description down below, I have the system setup prompt to set up your IDE in cursor, in replet, in VS Code, whatever you want to actually get your II framework ready to go so that you guys can follow along. This is the name of the repository you'll have access to called the Agentic OS Academy. And I'm just going to click create repository.

Once your email has access to this codebase, you're going to be able to use it in the exact same way that I'm showing you right here. And so once you gain access to this codebase, all you have to do is click use this template, create a new repository, and in that way, you're going to own the intellectual property to this codebase all by your own. So you actually I won't have access to what you build from here on out. and you'll have this entire codebase set up which has the Instagram agent. So once that happens, we want to head over to the code section. We want to click the HTTPS code snippet and we're going to copy that URL. Once we head into cursor, you guys want to click clone repo. You want to paste in that link that you got from GitHub. And the agent's just going to go ahead and build this out. Now the next thing we have to do is make our folder.

So I would call this your Aentic OS folder. select it as the repository destination and it's going to clone the entire repo. If we open up the implementation folder, we can see that we have the Instagram DM configurations and if we open up the instruction folder, we can see that we have the Instagram DM agent instruction guide. So now all that we're missing is some dependencies. Now what I also did guys is I added a set of Instagram leads that you guys are going to be able to use to get your agent actually set up and running. And once it's using these leads correctly, then you can set up an automation to feed it the leads that you are actually wanting to send DMs to. So I'd recommend using the ones that are in the codebase first because it just saves you time to getting your agent fully working. So the first prompt we're going to be sending in is set up the Instagram agentic DM workflow. We need to install the dependencies for Playright. And then I need to log in with the 2FA once you start up the browser for the agentic workflow. And we're doing this because then once the agent actually, you know, does this for us, it's going to save that JSON session into the code. And so in that way, this agent can open up Instagram repeatedly. So now we're just going to go ahead and install the dependencies. So installing Playright, Python, Chromium browser, and that's going to create the ENV folder or the secrets folder where our Instagram credentials will actually be. And then it's going to launch the entire browser for us. So we can pop out this channel here and that should show us exactly what's going on in the terminal with the coding agent and what it's actually doing. And so we can actually make that a little bit bigger for you guys as well. So we can see everything that's going on. The Chromium browser is now installed and it just created the ENV folder. And so in the ENV folder, it's asking for our Instagram username and our Instagram password. So I'm going to quickly add that in and then we'll move on to the next step. All right, guys. So now what's going to happen is the agent is literally just going to open up the Instagram page. And I had to stop the agent because I wanted to show you guys how this works. So now what we're going to do is I'm going to click go. And the agent should just open up this browser on our behalf. All right. So here we're seeing on the screen that the agent is signing in with one of my clients websites. So m77.shop is the Instagram handle. And we can see here that it's now signed in successfully. And so if we actually close down the screen half size and we look at what's going on inside of the

cursor now, we could see that the browser has now started and it's going to use the DMs that we've given it or the leads that we've given it to immediately go ahead and start sending DMs. So I'm not touching the screen anymore. It's just going ahead. It's picking out newest account. And we're going to see that this agent is now going to find an account and start sending DMs. Right? So, it found this account, Aush Tayagi or whoever this is, opened up their Instagram DM conversation, wrote our customized message, and our customized message currently says, you know, username, we built the instantly for Instagram. It automates DMs at scale. It personalized it. It sends personalized messages using AI and it books calls while you sleep.

Now you guys can go inside of the Python script in the II framework for this agent and you can change exactly what you want it to be saying. But now that you guys have that set up, this agent will then go to sleep for about 60 seconds. So it's now refreshing the page on the right hand side. It's sending itself back to the Instagram DM's uh homepage. It's now finding the next lead in the sequence, which is IG. Let's see who this person is. We got IG by Gazelle or Gazelle. It's going to now send the exact same DM to them. There we go. And it's just going to do this again and again until we run through either all of the leads or it reaches the uh constraint limit of Instagram, which is 200 messages a day or 10 messages an hour. And all of this information is being relayed in the logs on the lefth hand side. And so that is the agentic framework that I built for browser tools to actually be very powerful when we give them to coding agents. And now we can program any automation on the internet, which is what I'm going to be showing you guys a lot more of now that you know how it works. Hope you guys enjoyed. I'll see you next one.