There's a next level feature inside of clawed code that once you truly see, it's impossible to unseen.

Once you see this feature, you'll understand that AI coding is not enough.

This feature lets you build in ways engineers using cursor, windsurf, and other easy mode vibe coding tools simply cannot.

Now don't worry if you're using one of these tools, you can still tap in to this capability.

Every single thing we do now is about scaling our impact by scaling our compute usage.

If you understand this one idea, you are setting yourself up to win in the generative AIH.

I've been shipping software for over a decade and every couple of years there's a tool with a feature that transforms what you can do.

This is one of them.

In this video, I'm going to share this feature with you.

We'll then breakdown the key difference between AI coding and a gentic coding so you can truly understand the impact that you can have with a gentic coding.

Then we'll look at 3 concrete ways you can take advantage of this capability for your engineering work.

Let's talk about programmable agentic coding for next generation engineering.

What's better than an agentic coding tool?

An agentic coding tool that you can embed inside your tools, work and projects.

Cloud Code is programmable.

What does that mean?

Here's exactly what that looks like in a single line of code.

Claude Dash P whatever your prompt is, and then you specify your allowed tools.

Why is this so important?

If you're a principled AG cutting member, you know that you can do this with Ader.

You've seen this already.

Here's the exact same version inside of Ader.

We use ader dash dash message.

We pass in the prompt, and then we specify the exact file we want changed.

Big whoop, who cares?

Not so fast.

The difference between these two examples is monumental.

So what's the difference?

To understand why this is so different, we must understand the difference between AI coding and a gentic coding.

These are two different animals.

What's the difference?

With AI coding, you're effectively using a single tool call, where the tool is a function where you pass in the big three context, model and prompt, and the side effect is your code being written by your AI coding tool.

We're all very familiar with this idea.

We're all very familiar with this concept.

Agentic Coding takes that a step further.

Agentic Coding has, of course, that AI coding tool, but you also have access to any tool you can think of.

But not all tools are created equally.

There are some that are exponentially more important.

We can see this by looking at the default tools inside of clawed code.

Here's a list, the current list of every single essential baked in tool that ships with clawed code.

You can see we have 11 tools versus traditional AI coding tool just has the one with tool 8 and 9, edit and write.

We already cover everything an AI coding tool can do with clawed code.

Once we cover edit and write, you can look at agentic coding as a superset of AI coding, a much larger superset.

All right, so with the edit and the write tool, we can write code with AI.

Now that's just the beginning.

With glob, grab, LS and read, our agentic coding tool can now act as we would inside of the code base, looking for important information to help us write the change.

Now, as we know, there are costs associated with using these tools.

Cloud 3.7 Sonnet is priced at $3 per million input and $15 per million output tokens.

All the searching you do stacks up.

This is the biggest problem with Claude code right now.

This is the price we pay for state-of-the-art agentic coding.

And then we have the game breaking bash tool.

This is where agentic coding really takes off.

With just the bash tool, your AI coding tool can now act like you would inside the terminal.

It can call any bash command that you could.

This tool and frankly many of these tools scaled directly with the intelligence of the model.

With just these tools, we have a brand new tool to wield and get tons of engineering work done.

But the cracked Claude code team at Anthropic has gone even further.

They added Batch, which lets you run tools in parallel to save time and task.

Now this is where we get into incredible agentic behavior, where the agent itself can launch a sub agent to perform a slew of tasks.

This is an extremely powerful capability in the state of AI coding.

We talk about this a little bit as a key process property of agents.

Powerful agents of the future will be self organizing and self replicating.

This tool paints a picture of what that will look like.

So this is just the baked end feature set from Clog Code.

As you know, Clog Code lets you tap in to MCP servers.

This means that you can not only use existing tools, but you can build out any tool set you can imagine.

MCP servers allow you to create tools for everything and this This is why effective tool calling with a smart model is so critically important.

It lets you scale your compute and step outside the simple bounds of AI coding.

This is the big difference between AI coding and agentic coding.

Agentic coding puts together 3 essential ideas.

A model capable of calling the right tools, arbitrary tool calling from baked in tools to tools you create via MCP servers, and then finally the right agent architecture that lets the agent be more autonomous and get work done on its own.

To be super, super clear, clog code is an AI agent.

When you put this all together and you're able to program this, you get a tool that is infinitely programmable.

Clog code is infinitely programmable.

What does that mean?

What does that allow us to do?

Here's a simple UV single file script with 10 lines of Python where we can see this in action.

We're not just writing code anymore.

AI coding is not enough.

Engineering is a lot more than just writing code.

We need to be able to call arbitrary tools in many, many, many different workflows to get real engineering work done.

Let's look at the simple to do example.

Here's a prompt get check out a new branch, create todo dot TS, then commit your changes.

We have at least three tool calls happening here that we're handing off to Claude code.

You can see the list of allowed tools we're passing in edit bash, create, and then we're kicking off this sub process.

We're going to run this in a moment.

I'm going to show you exactly what this looks like with the three essential examples that you can use to get ramped up with this incredible capability.

Claude code is infinitely programmable.

This means we can build reusable workflows.

Principal AI Coding members know exactly what ADW's are.

You're already aware of this capability inside of Principal AI Coding.

We use Ader to really showcase what this looks like.

Cloud Code takes us to the next level with arbitrary tool calling in natural language.

Speaking of, we can call any tool and many tools in natural language in any sequence that our work requires.

This is part of why Cloud Code is so incredible.

The Cloud Code team has built out the tool that you you need to get real engineering work done.

Huge shout out to Anthropic.

They are actually building with this stuff.

They know what's going on, unlike the insanely weirdly rushed responsive codex release from Open AI.

Check out the previous video to see my thoughts on that.

Moving on, we can embed this inside scripts.

OK so that means we can automate engineering work and DevOps work.

This is big.

Your infra work can now be automated in natural language with any tool you want.

Now we can push this even further.

OK, these these next two ideas are big, and we're going to spend a lot of time on the channel digging deeper and scaling up what you can do with compute.

Make sure you're subscribed so you don't miss these big ideas.

This is just the tip of the iceberg.

We can stack up multiple claw code instances for insane impact.

Here we're running a single prompt, but we can set up a reviewer claw code instance to review this code.

We can set up another one to do the next series of work.

We can set up all types of automation.

OK, this is just a simple example.

I want to get the gears turning in your mind.

I want your engineering brain to turn on and start understanding how you can scale your impact with cloud code as a programmable agentic coding tool.

The last big idea here is truly the future of engineering.

This is the next, next phase.

You can build cloud code into your agents.

OK, I'm going to take a little pause here just to let this idea air out and really take hold.

You can scale your compute with clawed code inside of your AI agents.

This is a next level idea we're going to be talking about on the channel in the future.

So again, make sure you're part of the journey you don't want to miss.

What's next?

We can take this and improve it right here.

This is a static script.

It's going to do one thing, it's going to do it well, but it isn't as useful.

OK, we can easily build up a workflow with this one idea.

OK, so we can take that exact same workflow and add just one dynamic variable.

We can parse the CLI args and place it as the second step.

In this 3 three-step workflow.

Git check out a new branch, do whatever the prompt is, then commit your changes.

All we're doing here is internalizing the capability of using clawed code in a gentic coding tool as a programmable tool.

You can build this into your scripts, into your workflows, into anywhere you can write code.

Then you can commit this file.

You can commit whatever workflow you build and iterate on it across every single one of your code bases.

Let's execute some real examples.

Everything we do here on the channel is hands on.

I always aim to hand you as much value as I can every single Monday.

Let's first look at the essential files of this code base.

If we run as a programmable, you can see we have you know 7 files here showcasing how ader is programmable and how clog code is programmable with increasingly more agentic examples to help you understand what you can really do with this technology.

Here's a simple example.

Make hello dot JS a script that prints out hello.

OK, super straightforward.

Nothing complicated going on here so we can go ahead and run this.

I'm going to use my file reference hotkey.

I highly recommend you set something like this up so you can quickly reference files for your AI tools.

I've mapped this to a command shift R and then I'll just kick this off like this.

So to be clear here, let me actually first do this.

If we look for hello, you can see there is no hello file.

So hello dot JS nothing there.

And now we can go ahead and kick this off.

So over on this file successfully created.

If we do LS grab hello, you can now see that file cat hello.

You can see exactly what we wanted there.

We can run whatever your favorite version of JS is to kick this off.

There's bun, node and of course, Dino.

Let's push this further.

Shell scripts are kind of nasty.

Not all of us like to use these.

Trust me, I'm one of those people.

I prefer using Python or JS for my scripting.

This is a single file Python script.

We can kick this off and get this file generated in a brand new branch.

Let's go ahead and kick this off same flow.

And this is going to be really cool.

And you know, while this is running, just to be super clear, you know, LS grep CC to do this does not exist.

You can see here our branch changed.

We're now in to do CLI app.

And so after this completes, we're going to have a stage commit and then we're going to switch back to the main branch.

Claw code is going to switch this back to main after it completes its work.

You know, you can see here at least 4 distinct tool calls with one or more of our allowed tools enabling this functionality, right?

OK, so there it is.

We're back on our main branch and claw code is printing its output.

Here we can look at the changes.

And you can see here we have that new branch, 0 dependencies add list update toggle.

Fantastic.

So we can check this out, right?

If we open up our directory and we get checkout this branch, you can see that CC_to do now exists.

We can open this up and we can see our file exactly as we asked, right?

Create this file, a zero library CLI to do app.

We can click into this and you can see we have a clean TypeScript application.

Looks like we can kick this off a node.

I'll use BUN since it usually works out-of-the-box.

No problem.

And there you can see we have our nice, simple To Do List application built out for us with our agentic coding tool.

We're not going to go through the code.

It's A to do app.

No one cares.

But what we can do is clear and type git log.

And you can see a clean commit here from our agentic coding tool.

Add CLI to do app.

This is that next step.

And then it's switched back to main.

OK, so GC main, now we're back in Maine.

Now that code is gone.

This work happened with a script with our agentic coding tool on the channel.

We're on a mission to build living software.

I hope you can see how this is directly in line with that goal.

We need these arbitrary tool calling capabilities inside of small to large workflows in order to build out full on agents, full on agentic software that operates on its own while we sleep.

If that journey interests you, you know, make sure to like this video, let the algorithm know you're interested, and let's move on to our final example.

So things just scale up from here, right?

What you can do is only limited by what you can design into your systems.

OK, so if you want to run the raw JS version, I have that here for you as well.

You can run this in JS with the spawn command.

And if you're interested, you can also check out the aider versions.

I just want to add this in here for you in the code base, just to make it clear how you can do this with Aider and how you have to do a lot more work with a non agentic coding tool.

For that exact same bulk of work we have to do something like this, right?

You can see we have several get commands.

Aider does have auto commit functionality built in.

It does have some nice web searching and you know, image pacing some niceties.

But it's not agentic, it's not an agent.

It's a single tool with built in functionality with raw code.

And by the way, I like to use Aider just as a comparison.

Aider is still the best way to teach principled AI coding.

This is why I use ADER inside of Principled AI coding.

I built this from scratch for you and all the ideas are still relevant.

I'm really proud of that.

I'm really proud to say that usually courses lose value over time.

The value of principal AI coding has been steady and frankly a little upward with everything that we continue to get right and, you know, predict essentially by making bets and sticking to them.

So that's there if you're interested.

For all my existing principled AI coding members, I've got some exciting news.

Stick around to the end of the video for that.

Let's move on to another large example to see how we can use cloud code as a programmable AI coding tool.

So this one is really, really cool.

This is going to blow your mind a little bit.

We can use Cloud code with arbitrary MCP servers.

So inside of this code base, I have an MCP setup.

If you click this, you're going to see an MCP setup for Notion.

Notion is your second brain.

It's an application you can use to store to take notes and do a lot of knowledge work.

All the setup you need is in here, and the README will explain exactly how to get it set up.

You can see I have dot MCP dot Jason get ignored because my Notion API key is in that file.

But now that we have this, and now that we know that we can use Claude code as a programmable tool, we can do something incredibly.

If you want to understand an agent or an MCP server, first look at its tool set.

I like to call this the tool belt.

We can see we have the essential built in Cloud Code standard tools, and we have a huge list of Notion API tools.

If we collapse everything here, you can see all you need to see.

We have a nice prompt here that we're going to look at in a second, and then we have our programmable log code line.

We're going to use the output format flag to stream the output as this program runs.

So what is this going to do?

Cloud Code is agentic, so that means that we can use it with any tool we want.

Move this to half screen, open up the terminal, prep this to run, and this tool takes in a single parameter and it's going to be the name of a Notion page.

So I'm going to copy this Notion page and pass and this parameter here.

I built out a complete workflow here that reads a Notion page, analyzes the instructions, and executes on the task 1 by 1.

Let's kick this off and I'll explain exactly what's happening as we progress.

So let's full screen this, let's move this over a little bit more, and there we go.

So claw code has started to work for us.

So what is it doing here?

Like I mentioned, it's going to read this file and then it's going to start calling tools to actually write this code.

You can see this directory just got created Notion to do's, and now Claude Code is reading this file and it's going to execute every one of these tasks top to bottom.

And now we're tapping into another key principle of AI coding and of agenda coding.

Great planning is great prompting.

We have detailed everything we want out here in this Notion page.

And now we can use our Notion page as a document, as a spec, right as a plan to get work done.

So you can see here, Claude Code just created this add method.

We can click into this and you can see some of that code here update just got created there.

As Cloud Code moves through these features, it's going to make a tool called back into Notion.

Cloud Code had to use a Notion read tool to get all this information inside of its context window.

We have every Notion API tool that we need to interact with Notion.

So here we're using Notion as our, you know, planning document that we can use to get work done.

So whatever you can write in here, whatever you can communicate with a great prompt, which we'll look at in just a Cloud code can pull off, right?

We're quite literally reading from Notion.

Then Cloud Code is going to review these tasks and check them off one by one.

There we go.

So this is really cool, right?

Our top level to do is got checked off.

You can see that exists our add just got checked off and then update, delete and list are going to follow, right?

Because all this functionality exists.

Our agentic coding tool is running Notion tools right?

It has the Notion tool belt attached to it, so it can perform arbitrary work inside of our Notion page as well as in our code base.

This is the key difference between AI coding and agentic coding.

Agentic coding is not limited to AI coding.

It can call any tool.

It can act as if you or I were calling the tool.

We're clicking the buttons, we're hitting the API calls ourselves, right?

You can see a nice output there that cost us, you know, $0.50 to run.

Not too bad considering I literally just set this up, right?

I just wrote the plan.

Now this can be rerun over and over and over, and we can run anything.

We can put in a Notion.

Page anything we can communicate to our agentic coding tool OK, so I really hope you can see the value in this.

Let me just briefly show you the prompt that I used to write this.

You can see here I'm writing a clean markdown prompt.

You can also use an XML ish prompt if you like, but all we're doing here is clearly stating how to process this type of Notion page.

The key thing here is that we're defining how cloud code should behave here in a loop right?

And then we have a process flow.

All right, step one, find the Notion page, get the page content.

Remember, we've passed in the page that we wanted to run.

OK, So this can run on any page you can set up.

Then it processes each to do.

And then it wraps up with a summary.

We have some additional important notes.

And then we list a couple of key tools that are going to be helpful for our agent.

And that's it, right?

200 lines.

We have a reusable asset that we can use to deploy cloud code over and over and over to get real engineering work done.

OK, so this is just one simple example and I hope you can see where this is all going.

There are only a few AI coding tools that have this agentic capability.

This is why I stress the importance of Aider and now clog code.

Okay, the fact that this is a programmable tool puts it in its own category.

So if you're an Aider user and you know, I completely understand using Ader.

It's a great tool.

You can script with Ader.

As mentioned, there is potential for Ader adding MCP support.

This could really open it up to become an agentic coding tool instead of just an AI coding tool.

I've been keeping a sharp eye on this PR, but I haven't seen Paul, the creator of Ader, jump on this.

So I'm a little pessimistic that this is going to get out and really transform Ader to become an agentic coding tool.

But I'm also personally on the fence.

I think Ader is a beautiful, fantastic tool because it is simply an AI coding tool.

It does one thing and it does it well.

For anyone that's interested, for anyone that's inclined, I have a bonus directory here with a couple additional assets in addition to, you know, a bunch of kind of clear, concise examples of how you can get set up very quickly and how you can understand how to use cloud code as a programmable agentic tool.

I have a couple of bonus files in here for you, and one of them I show exactly how you can use no inside of the open AISDK.

And then in this example, I'll show you how you can build a full on AI agent with Claude code inside of the Open AI Agent SDK.

Links going to be in the description for you.

We're going to be talking about agents a lot on the channel.

Stay tuned for that.

The Claude Code team is one of the few teams really shipping value into the agentic coding space.

If you ask me.

They are the leaders here and it's very, very clear.

I recommend you pay the price it costs to use this tool.

It's just too valuable to miss and it truly is paving the way for next generation agentic coding.

After you get a hang of this tool, you can start using it as a programmable agentic coding tool inside of your work and projects to get asymmetric results.

As mentioned, to scale your impact, you scale your compute.

This is how you win.

And Cloud Code is the agentic programmable tool you can use to scale your compute.

And when you scale your compute, you scale your impact.

This is how you win.

Hit subscribe so you can follow the channel and stay up to date on in depth guides like this.

The Cloud Code team briefly mentions this as a feature inside of Cloud Code.

They give it just, you know, 3 headers and just kind of move on from it.

The value you can create, the systems you can build with this one feature is massive.

So what's the big announcement?

I'm super excited to announce that the next course, the next phase that I'm going to be building on top of principal AI coding is officially in the works.

I'm starting to build up the essential ideas for what's going to pave the way for the next essential set of lessons that's going to help you accelerate what you can do with AI coding.

And now with a Gentic coding.

I'm massively prioritizing all existing principled AI coding members as I'm building out this next generation course.

So if you're not a principal AI coding member already, definitely jump in and start getting ahead.

Start setting up the foundations you need to survive with AI coding and what's coming next.

We detail this transition from AI coding to agentic coding in the state of AI coding.

I'll also link this in a description for you.

The big three ideas here is engineering with Exponential AI coding is transitory.

I hope after you see this video you can kind of understand why.

And then Part 3, agentic coding is the end game.

I have audio files at the top of each one of these posts so that you can really understand and digest this information as quickly as possible.

Things are moving fast.

You want to have every edge.

You can have packed a ton of value into this for current principal AI coding members.

I'll mention it on the channel as we move closer to the launch date, but look out for a late Q2, early Q3 launch of the next course where we accelerate what we can do with next generation agentic coding tools like Clog Code.

The final blocker for this course is that I'm waiting for Clog Code to get out of a research preview.

As soon as it's out of research preview, we can really dig in and you can be sure that we're going to take this tool and leverage it to the Max to ship not just code, but real engineering work.

Remember, we're moving away from AI coding into a gentic coding.

AI coding is not enough.

It's just the beginning.

It's the tip of the iceberg.

As you now know.

It's a single tool Where we're headed.

We're going to be calling 10's and hundreds of tools to ship real engineering value in record time.

You know where to find me every single Monday.

Stay focused and keep building.