Context Ops will make or break your Claude Code setups

21 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=_7ALHJPJZUU
https://www.youtube.com/watch?v=_7AlhJpJZuU

SUMMARY

The discussion centers on the challenges of managing context operations (context ops) in agentic development, particularly when using cloud code and version control systems like Git. The speaker shares personal experiences transitioning to a more complex setup involving multiple machines and repositories, highlighting the overhead and visibility issues that arise from this complexity.

- *Context ops may evolve into a full-time role, similar to DevOps, due to the complexity of managing information generated by agentic development tools.*
- *Transitioning from a single Git repository to multiple machines introduces significant overhead in managing diffs and conflicts.*
- *The speaker's tool, Taste Matter, aims to provide visibility into session data and the evolution of the codebase through a context graph.*
- *A key problem encountered was a misconfiguration in repository names that led to confusion and lost changes, emphasizing the importance of clear instructions and understanding assumptions in agentic systems.*
- *The exponential complexity of adding users to a context operating system can lead to significant management challenges, especially with local drift in information.*
- *Recommendations include creating specialized skills for managing Git cycles and ensuring agents understand their assumptions when debugging.*
- *The speaker plans to separate unrelated functionalities into different repositories to reduce complexity and improve management.*
- *The overall trend suggests that as more agents and information are processed, the challenges of context ops will continue to grow, paralleling issues faced in DevOps.*

Okay, so today I want to talk about the issues that you run into that I think broadly could be described as context ops. the overhead around using agents for agent development for your go to market, for your generalized knowledge work where you're using some sort of we'll call it markdown files. you're using code obviously using the specific language so your Rust or Python, whatever, and then version controlling that software.

The this, you could say this is just an extension of DevOps. I think that's definitely an argument to be made. In fact I've actually seen this framed as, or one of my clients framed it as. DevOps is a full time job. it's seeming to be that context ops is going to become a full time job is the discussion we had. But by and large the overhead of using agentic development tools like your cloud code, you want to say open code, code, whatever, you're basically just spitting off all this information all the time and you're trying to keep track of it more often than

not or at least you should be keeping track of it using git and the complexity and overhead and pain points that creates the thing I just found.

So the main topic of this video is the issue I ran into once I moved from solely one git repo, only development, only doing development on one machine to setting up a virtual private server for myself so I could, so I could use cloud code on the go basically. and then having to manage the diffs and conflicts and all the overhead of just two computers or two machines.

and then I guess the second part of this is the antidotes I've heard from my clients around them set up context operating systems. So context OS is basically a context graph. mine are the ones out with clients. They're all focused on go to market. So we take all your transcripts, your CRM data, internal positioning documents, whatever and use that to generate a markdown based knowledge graph of your icp, your product positioning, your product marketing, whatever those end deliverables end up being.

Oftentimes it's, I mean it's entirely customizable. It effectively allows you to ship and learn significantly faster and go to market. but it's using markdown and cloud code. So the overlap between that agentic development and the issues you're finding with context operating systems are very similar. Okay, so what was the core problem that I experienced? it starts with the context you need is that I'm working on a command line tool for cloud code that is basically meant to give it visibility, into its own session data, into how the system has changed and evolved over time.

So sounds like memory. It's not memory. I think the core trade off with these agenda development tools is you get a ton of leverage. So you can Write your specifications 20 times faster or your product marketing 20 times faster, whatever, the code faster too. anyways, I think by definition you're getting the leverage. The trade off is visibility because you're not reading that white paper or that code yourself by hand more often than not. Anyways, the tool I built, Taste Matter, I just cut a new version last night.

So right now, or actually it's live now so if you could check out the new version. But anyways, the indexes all of your session data in your file access and builds into essentially like a, an index, index, index that allows cloud code to kind of see like hey, how has the system evolved and changed over time or how has the code base evolved and changed over time? That's the whole, that's the main value prop. I use it to restore context to me with the context all the time.

It's super valuable especially when I don't explicitly save my point in time. A context package. I could say hey what have I done on tastematter for example the tool itself and it could say you're working on you're messing around with a TARI app. And then you lost it lost focus on that and then you worked towards you know, iterating on these command line for example it gives cloud code the dictionary to do that or quite literally like you know like a book has you know the outline in front.

That's kind of what it does. Anyways, I cut a new version to release a feature called I'm calling graph execution. And this is more for people who have these context operating systems where you have a markdown based

knowledge graph with front matter at the start of all these files. So maybe I can show you one. This example. This is one of the context packages I created when I was working on, Tastematter. Excuse me. So this is like storing state of like, hey, what do we work on?

What's next? Why? And you can see here it's like connected to a bunch of other, nodes in this gigantic context graph. that's what a context operating system is. Obviously mine is like gigantic at this point in here in a while. And actually there's some client graphs in here. but that gets to my point of how do you manage all this information in, version control and like, what's the right repository pattern to use?

Do you version control everything? Do you ignore some stuff? How do you nest repositories? Do you use a monorepo? These are all big meta questions, that tie into the main problem I was having today. So getting to the actual problem I had, I cut a new version that added this new functionality where basically allows cloud code to compute the graph itself. It's super cool, right? You can add. It'll actually go in and compute, run right.

Code and run it in a sandbox to compute this stuff. Right? So the front matter, it makes it. So instead of having to run like 10 grep queries to find a specific file or a pattern you're looking for against the front matter, it'll do it in one or one or two. I got it down to. It was a 91% reduction is what I got in terms of how many tool calls steps it took. Anyways, I wrote a post about it.

Sandbots code execution goes brrr, right? I thought it was cool. People were like hell yeah, great. I'm like great, awesome, cool. And then I go and check my. It's this one. Where is it? Yeah, I go and check the public repository because this is where the skill is. So I have the skill and the command line. Two different repositories allows me to kind of control and let people download the skill to use the command line most effectively.

and I go well why is this here? There shouldn't be a release on the public repository. That doesn't make any sense. And then I went to the other repository. This is a whatever other repository. And it's here now because I fixed it. But there was. The version wasn't there. And I'm like well what's happening? what had happened was a context problem that me using cloud code and my cloud code like checkout process didn't check or didn't catch automatically.

So the, the reason this happened is not some like crazy convoluted, engineering problem or things like that, but something simple. And I want to highlight this because more often than not, the, the root cause of any seemingly super complex problem is like once one thing misconfigured somewhere upstream. This happens more often than you think in like organizations where you think they should be like super squared away, perfect enterprise software, whatever.

Right? It just, it happens. and what my version of this was as a solo solo engineer founder was I changed the repository names because I wanted this one to be the public one and I wanted this one to be my private one. Previously this was just called Taste Matter. So what had happened was I didn't update the skill for cloud code. I have a skill called tastematter Release Ops that tells cloud code, hey, here's the repository she pushed to.

Here's the CI cd. It's going to run through. It's going to test it on Windows Linux and Mac OS, virtual machines in GitHub Actions is. It's going to install it, make sure it works and make sure that a cloud code version in there can call it. Okay, cool, great. I spent 40 minutes more or less. I had other things running at the same time.

the reason I realized was I was working on, my. I was working on my virtual machine. So this is my virtual private server. And the agent in there was basically like, hey, I can't pull it down, buddy. Like, where is it? Where's the code, big guy? and I like copy and pasted the, thing in here. It's like, where is it? Because I was trying to look at something else related to it. While it's messing around and stashing changes in, trying to fumble its way through fetching and rebasing and pushing and all this other stuff like that.

I have a internal dashboard I'm building myself basically. And literally it, it, it just lost the code. It didn't actually lose the code, but because it switched branches locally to. And it figured out how to do that, it just wiped all of the changes. So thank God the code, was local here or it was in the session data. Excuse me, but it wiped the changes for a completely different feature.

And this is my mistake. I should have realized that saying, yeah, go for it, do whatever you need to do to fix it, was going to change it up, was going to impact the rest of the code locally because it's a repository. but you can just see here I'm like just losing my patience. And this is my fault by the way. This is not. This is on me because you see effectively what I was, I was giving bad instructions.

I always had it being like, yeah, go, go make the changes, push them up. Whatever, codes here, local on this machine, we can make sure it's in the repository committed to the master and then we can release the, release the right version. this is why I harp on making it enumerate assumptions and prove it understands better.

Because up until like right here, it didn't really intuit things. But if I wanted to anthropomorphize it, it didn't intuit that, we just changed the names of this repository and that's like the whole, the whole source of this problem. Meanwhile, it's like make doing everything in God's green earth beyond just like the simplest possible solution.

right here, this is the thing I was like, that's actually not at all what you need to do. So this is the new graphic deck, is the new feature. And I'm like, you did not push it up. It's not done. so then it's where it starts trying to do this get jiu jitsu. And it's just like completely. This is where I was wasted like 40 minutes just waiting for this thing to like try and figure it out.

Before I was like, this is actually not working. Because what it's trying to do in here, in this section is it's basically trying to make like little line by line changes. I don't know why, it thought to do that because when I go down to here, I stopped it. I was like, prove to me. Yeah, it's also lighting tokens on fire. Anyways, I was, had it ground itself.

I was like, hey, this is the actual architecture works. And I finally got, I finally had the insights, finally used my

mind, my brain, to say, hey, I'm going to use a. I have a debugging complexity assessment skill that whenever cloud code is kind of over engineering or trying to debug something in a way that I think is just dumb, it's backwards. I say, hey, this skill literally just says just pause and think about the problem.

What is the smallest thing you could do to fix it? And then from here it makes a couple of changes that has to go up and done. This is the tag process. Boom, done. I know for a fact this is partly me being dumb, partly me learning, just how to do CI CD better, how to think about get better.

Right? There's all this stuff where it's this lot, a lot of this is on me. But this is also part of a larger trend that I'm seeing where agents spin off so much information all the time and we're trying to like manage it. when you use the agent to then manage the information that it is like spun off, it is just a recipe for disaster unless you do it in very specific, specialized ways.

So I guess to share, share my learnings. well let's take a step back for a second actually thinking about the git in context problem at large. I've heard about companies that are like have so much. Have so many agents running all the time in work trees and, and virtual containers and just everywhere, all the time. It is becoming like untenable even to use other agents to monitor and like improve changes, prove up PRs, whatever.

Right to the point where I think we are like we will eventually really hurt, not hurt, we will push git to its limits. and I think that's the broader trend here. But I guess in terms of the practical steps that you can take away from this, make sure that you're one I would say definitely create a skill to manage your git cycles. So I have two, I have the context GitOps, and I have the specifically one for cutting releases.

The other one I would say is. Make the agent prove it understands the assumptions it's making inherent to how it's treating a problem or like debugging a problem. I released a skill around this called the epistemic context audit basis. Has the agent enumerate, hey, this is what I think. this is, these are my assumptions about this problem. And then like raids how well can actually know that thing?

Epistemology is the study of what you can know. then so it's make the agent prove it have specific skills for managing git and like the like the hard parts like a release cycle basically. But then I would also say the strategy I'm going to go with now probably is I need to. So that thing I showed you where I was working on Dashboard, I'm thinking I might make that a separate repository or a sub module or something where it's just not tied to something that really has nothing to do with in regard to functionality.

They don't talk to one another. It's just they're part of the same like business case, business project, whatever you want to call it. So that's how I'm going to think about it and I guess getting more to the multi user coordination where I'm seeing my clients. The big takeaway is the more people you add onto your context operating system. It doesn't get.

It's not like it's not summative. That's the right word. Yeah, it's not summative. So one person, it's. Yeah, you

have one person on it, then you add another person. It's not twice as hard or three persons three times as hard. It is like exponentials. that is like the theoretical version of where if you have ever been pushing and pulling changes from the thing all the time, the only reason is not like a pure exponential all the way up is because most people on your system will not be pushing and pulling from it.

Generally you have power users that read and write from the thing all the time. And 80% people just take the work that they're doing, whether it's around new positioning documents or code or whatever it is and they're reading from it. so that makes it a bit more manageable. Now I like example of this. I know I have. Excuse me, I have a client called Pixie. Victor's their head of go to market. He's awesome. He's like their power user.

And like it's a word from open source like benevolent dictator of the repository, managing the thing. But I also know they got their 20 person team or whatever, the exact number is all up on cog code, all using the context os, on top of their main core product, which is their code base. Right. And the problem is twofold. So it's easy because not a lot of people are pushing and pulling. But then you have also no idea how much local drift is happening.

People are using it to do event planning and like whatever the deliverables they're responsible for is, then you get just a ton of drift. So In this by and large parallels most with this by and large parallels DevOps pretty well from my understanding having never been a DevOps engineer. But I think, I think this is different. If nothing else just the sheer amount of information being processed.

and what I would say is so number one it's the amount of people but most people can just read and write. That's the summer so far for the first two. and the third thing is it's this is a new problem. Everyone's figuring it out from the very top 0% when they're generating I'm sure a zillion times more code and context than I am for my systems all the way to people just being like what's clock code they're going to eventually run into?

Well I'd like to save my files but I want to change them. How does this work? Right, so that's my story time of the context ops problem I had for myself. mostly around git but I'm happy to dive more into specific context context graph oriented stuff if people would be interested in that. But yeah, I just wanted to share what the pain points are of modern agentic development at least my experience and how I'm going about fixing them.

Which by and large it is the same principles of Rubber Ducky. It work through, work through the problem step by step and slow down. so I will be releasing the are happy to release the debugging complexity skill if people are interested in that. I find it super valuable. It's. I use it for everything, not just this git issue. so if you're interested in that, leave a comment. I'll push it up to the, to one of the repositories I have up.

But thank you so much for your time. I hope you have a wonderful rest of your day. Bye.