How Intercom 2X'd engineering velocity with Claude Code | Brian Scanlan

78 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=BRDKFT0-DUU
https://www.youtube.com/watch?v=BRDKft0-dUU

Suddenly, you started realizing that you've to think bigger about things, or that your imagination is now the barrier, not the tool. How is this not happening in your organization? Like literally the physical limits of my ability to type code are unlocked by AI. >> Today, we're seeing twice the number of throughputs as we did compared to 9 months ago on our engineering team. Now it's like, why can't it be 10x? This is a little bit more of what my instinct tells me is possible, which is if you go all in, if you prepare your team, if you prepare your code base, I think your overall product quality is going to go up, I think your overall developer experience is going up. There's just so many good things that come out of using these tools and using them correctly.

Backlog zero is a realistic thing for teams to be able to go after. All the things that you wished you would ever wanted to do, it's now just achievable. >> I often advise a lot of CTOs and VPs of engineering when figuring out how to get their engineering team AI-pilled, say everything you hate about the code base, go spend a month fixing and see how fast we can speedrun that. That's going to feel really good. >> I've been having the most amount of fun in my career over the last 3 months.

>> [music] >> Welcome back to How I AI. I'm Claire Vo, product leader and AI obsessive here on a mission to help you build better with these new tools. Today, I am showing how Intercom [music] 2x the number of PRs that their R&D department is shipping in just a few months. Brian Scanlan is a senior principal engineer at Intercom and he's going to show us truly all of their secrets to getting a large product and engineering organization cooking on cloud code. Let's get to it. This episode is brought to you by Siligo.

[music] Every company today wants AI to improve how work gets done. The fastest way is building it directly into everyday business processes, automating employee onboarding, keeping customer data accurate, managing orders and inventory, or resolving finance and operations issues. When AI lives inside [music] the flow of work, it can update records, trigger approvals, route work, and kick off the next step across systems. >> [music] >> That's how teams operationalize AI and deliver measurable results. Celigo makes this possible. And now, with Celigo Aura, it's never been easier. Celigo Aura gives you access [music] to the entire platform through natural language, connecting your systems and turning intent into action. All of it under your control.

>> [music] >> Companies like Databricks, PayPal, and Olipop rely on Celigo to run critical business operations at scale. [music] Ready to operationalize AI? Visit celigo.com/howiai. That's c e l i g o.com/howiai. Brian, welcome to How I AI. Why I am so thrilled that you agreed to join the podcast is I think Intercom has done it, which is you all have met the moment in sort of two ways.

One, clearly met the moment from a product perspective, were one of the first companies that had Sorry, I don't want to say legacy business, but had a going concern business that saw AI coming and really transformed how your product worked for customers, and I'm a happy Finn customer. They did not tell me to say that. And then second, what we're going to talk about is the team met the moment in terms of really understanding AI was going to change how, in particular, product engineering and design orgs and engineering organizations were going to work.

And you just went full speed at changing how the team works. What what drove sort of the urgency around meeting the moment? How did that come to be? Was it a single person? Was it everybody? What was your experience? I think it's In some ways, it's been the easiest place to be driving our street adoption of AI in engineering and product. Um, because we've focused the company so much on our focus on product on adopting AI and being AI first in how we think about the product, future customer support, and all that. And we also have very clear expectations.

Like we, you know, we we've seen what's possible in the product space. And it's just very clear and obvious to us as like connoisseurs of AI, it's like this is clearly going to be huge in engineering and product and building. Um, and honestly, there's been a lot of impatience for like why why isn't this happening today? You know, we go back a few years and cursors picking up a bit of business and the models are getting better. Um, but it still wasn't transformative. It still wasn't like the whole business was changed and we're seeing vast amounts of extra productivity. We knew there was potential, but it still felt like we needed to have some sort of breakthrough moment or or something needed to to big had to happen for us to get to the kind of huge velocity winds that I think now we're starting to achieve. That said, we still want more, you know, we're we're proud of where we we're at, um, but we're we have we're not content with uh with what we've achieved so far. I feel like every 3 months I have a a breakthrough moment.

And in fact, I feel like Opus 46 I I don't know, something just like really inflected in what was possible when that particular model came out. Now, I think the GPT 5 4 uh models are also exceptional. And so it's something about that one moment with models that really inflected my own personal use of AI in engineering. Did you all see the same sort of inflection around that model point? Totally. I think you can go back to it's like November, December last year. Uh, and suddenly you started realizing that you're you have to think bigger about things or or that your imagination is now the barrier, not the tool. You're spending less time massaging the tool to get it to the right place.

Um and it's less about auto complete and more about just literally giving us your ideas and seeing what happens. Uh I think the Christmas break happened as well. I remember we we pretty much decided before Christmas like, "Hey, we're going to go all in on Claude Code." cuz up up to that point there was a bit of cursor here and there and augments in different tools. Uh and the Christmas break really helps. Like we uh but I just saw everybody go wild on Twitter X, you know, that uh people were uh talking about how this was like they were getting so much done and they were building all these things. You just come back to work after a Christmas break going like, "Okay, everything's changed." Like we we knew that there was something here and that we were starting to see the signs of it, but now the whole world is convinced. Or at least all of the influencers on

Twitter are like, "What?"

>> [laughter] >> That That would be me. Yeah, I'm I'm actually kind of convinced that companies should increase their PTO and parental leave policies because everybody I know right now in tech that is {quote} taking time off goes on their vacation and pops open Claude Code and comes back like 10 times more skilled than they were before before their time off. And so, if anybody wants a a little minor hack to AI literacy in your org, give people time off to hack and they will come back with more information than you expected.

Okay, I think we're going to skip to the punchline which I love which is we're going to see how AI has actually changed how you all ship at Intercom. So, can you just show us a little bit of how this has changed inside of the org and I think you all are measuring a lot of this. Yeah, so I think we've been diligent as you know, product owners inside of Intercom uh but we've been trying to uh get feedback from people and see how they're using the uh the tools and uh really like just doing everything uh that we would normally do with a regular product. Um and so, uh we've spent a lot of time hooking up Claude Code with telemetry both into things like Honeycomb um and the data also going into, uh, snowflake where we have our data warehouse. Um, we also store session data in S3. Um, and we mine this stuff for for useful, uh, insights. And, um, one one but one of the main things that we used to drive adoption, uh, of the tool was, uh, our CTO, Dara, setting a goal of us two-xing, like doubling the throughput of R&D. Uh, and we used pull requests as a crude, uh, simple measure. But, you know, there's uh, and you know, you can argue back and forth about what's a good measure, what's a bad measure, and whether measuring anything is appropriate or whatever. But, I think it's reasonable to just have the expectation that if you can get a lot more done and it's so fast and fun, then why aren't why isn't everyone just shipping more stuff? Uh, and so it's a basic measure that like the tools are being adopted, um, and that they're being used well. And, you know, of course, we don't tolerate lowering quality and we're high-trust environments, so we don't expect people not to game these stats or whatever.

But, our metrics and what I'm showing on the screen here is, you know, it's a classic number goes up, uh, kind of thing that where we have we started tracking this back, uh, like how many, uh, PRs and and what percentage of them were, uh, generated by either, uh, Claude or Cursor or whatever. And, um, yeah, since our major investment in Claude Code the platform and going all in on this and really pushing out like enablement and, uh, giving people, uh, freedom to explore and start to build skills and everything. Um, but also pushing them on on on we expect kind of throughput, uh, increase. We've seen a big, big increase in in the throughput, uh, of pull requests through our system. And, you know, like last year, like our CI system completely broke. It melted. It, you know, but not only did it get like 10 times as as expensive and, you know, we did the work, we fixed the bottlenecks, we improved the performance of our CI system. That stopped being the bottleneck. Um, now, uh, code review is our bottleneck, but um, but like we're still but today we are seeing uh, twice the number of throughput as we did compared to 9 months ago on our engineering team. Um, and like we're very proud of that and you know, now it's like why can't it be 10x?

So, what I love about this chart just for a moment is I had spent the last two decades of my career in product and engineering last decade of my career as a CPTO. And it's so funny, I want to go back to a couple things you said, which is one, you have to treat your org like a product. And I always thought that my job was not just the

product strategy and the capital P product that we were delivering to customers, it was to design our organization to I would say like output innovation on demand, which is that was the job. And less romantically wrote them less romantically put, my job is to invest R&D for positive enterprise value. That was like fundamentally my job as a CPTO.

And so what I love about this is it's merge PRs per R&D head. I'm presuming that includes Does that include product managers and non-engineering R&D or is that purely software engineers? Yeah, this is all of R&D and it's definitely the case that our designers and product managers and uh, TPMs, like every role in Intercom is really actively using cloud code on certain on shipping code and all that. Um, and also we've been hiring. Like this number has not been static. So, the number of PRs uh, the raw number is is dramatically higher than just 2x what it was a good way to go. So, this is everything from your newest hire to uh, your pro uh, product manager who's like adding uh, some copy or shipping like small changes whatever. Um, you know, that's all uh, based in this number. The other thing I want to call out for folks is every board meeting I have been in for the last 3 years have said how are we getting Well, actually, every board meeting I've ever been, period, has been how can we get more velocity out of R&D? Certainly in the last 3 years, it's been how is AI inflecting our velocity? And it's so funny. I talk to so many people that are like, "It doesn't really inflect velocity. We're not actually becoming that more efficient." And I'm like, "Is that true?" Because I look at a chart like this and I say, "This is a little bit more of what my instinct tells me is possible." Which is if you go all in, if you prepare your team, if you prepare your code base, if you have, as you said, I think a high-trust culture, people are going to look at this and say, "Oh, they're shipping these smaller PRs." Or like engineers are gaming the system. I just I have not worked at a place that has such kind of like bad culture that that would actually come as an outcome of setting some sort of ambitious fun target like this.

And so I I take this as at face value and I think, "How is this not happening in your organization?" In your Like literally the physical limits of my ability to type code are unlocked by by AI. You should get some inflection there. And so, you know, for VPs of engineering, CTO, even people that are on these R&D teams, look at this and think, "You know, this is possible." And it may be a crude measurement, but it's, I think, an appropriate one as a leading indicator of of what's happening in your org around AI.

Yeah, and we support this with not just telling people to move faster. Like that's that You know, we're we're really looking from first principles of how to how to do the work. Like we believe that like all technical work will become agent-first. And I'd like to set like a deadline for that so that, you know, at the end of the month, we're just going to go all in. And it's never going to be the first thing that happens, say, in response to an alarm or in a planning meeting that it there isn't like an agent in there kind of doing the the basic work. And I I think that's a realistic expectation, but it it involves not just We're not just moving faster for the sake of it. It's We're seeing that we're moving faster by looking at the fundamentals of where we're spending our time and reimagining how that work could be done in an agentic world. And honestly, if like the if the agents didn't get better, if the models didn't get better, the harnesses didn't get better, uh we've got the building blocks just today to be able to just continue going, moving around, looking at how we do our technical work today.

Uh on a By technical work, I mean everything in the delivery of product and and move it to entirely be agent

first and allow us to move up to a higher level, to be able to like work on higher level concerns or just getting more stuff built, more stuff out there, or higher quality. That's all within every org's grasp today. But you have to be very open to change. And I guess what's been fortunate is the end of the year over the last while is that we have been extremely open for change both in the product side of things and adapting the the company to how uh I think companies need to work now with AI and we're starting to see results.

Yeah, the other reflection I have upon looking at this chart is we're recording this in kind of the spring of 2026 and Anthropic just said that they crossed 30 billion in in revenue. I think up from 19 a couple months ago and I I I suspect their revenue chart looks a little bit like your merge PRs per R&D chart. So, how are you all thinking about the trade-off on cost here, right? Like we're all consuming Claude tokens. Yes, you know, efficiency or output is going up, throughput's going up, but is cost scaling proportionately? Are you all worried about Is that the problem right now? Are you even worried about it? How do you think about that?

Yeah, we're definitely worried. And that's the bill is looks exactly like this. >> [laughter] >> Um and um you know, I've I spent a lot of my career worrying about AWS costs and uh about our margins and stuff. And then suddenly you've got these costs showing up uh on uh and they're disproportionate to the growth that we've seen anywhere before. It It's like hiring whole new offices of people. Um and but at the moment our attitude has been look, everyone just turn on Opus for everything. We'll be done with uh context window, you know, we uh we just use the API plan so it's all just on-demand and we we think that there's enough alpha or benefit in at this point going as fast as possible and caring about the bill later because of the later benefits we'll get. And maybe that's the position where Intercom is. I don't think it's realistic or feasible for absolutely every single business to do it. And honestly, I do kind of respect when you have to actually think about your token use and how that can kind of force you to be more considerate or it sometimes even gets you better results. You know, you don't need Opus for everything. Like uh there's there's faster models out there. Um and so we're we've we're just kind of avoiding that optimization phase until the point of where we we until, you know, we've gotten serious benefits from investments in this platform. And so I think this investment and I think it's it we are treating it like an investment at this point um is worthwhile uh but you know, if this keeps going at this rate, yeah, we should all work for Anthropic, you know.

Um I think the way they're hiring we're all going to end up working for Anthropic. So Okay, and then one other thing because I think, you know, folks are going to look at this certainly engineers and they're going okay, like you're shipping more PRs, but it's all slop. It's all garbage. I you know, I know you all are measuring quality on the outside of this on the other side of shipping all this stuff. So how have you seen this inflect your measurements around quality or customer value or what you're trying to achieve at the end not just lines of code?

Yeah, I have a stand-alone graph that I can share which is kind of interesting. Um and so we've started to look at the uh the time it takes from the first line of code written in a feature to the time it gets posted on our news channel, like our updates. Um, and that's uh that that has decreased consistently over the last few months. Now, we're not optimizing for this, uh but we're interested in it. And the other thing is like the sheer volume of things we have shipped also appears to have kind of just rapidly increased in the last few months as well. And that's a

that should be a bit of a trailing metric. So, we believe that these numbers, these like increase in volume, is being born out in real features, real products that our customers are using. And even we've been running some experiments like how far can one person get on their own building um something that's plausibly a whole entire product area and feature to be able to sell. Uh so, this is something we're taking seriously um and is uh we also care a lot about quality. We've been working with a research group in Stanford. We've been giving them our data and uh you know, mostly just looking for any kind of insights to make sure we're not blind. You know, I join absolutely every single incident. I'm a ambulance chaser and I make like uh and I'm not seeing any increase in kind of regular kind of incidents or outages or customer facing problems. Um we've had a few kind of weird problems, but not related to production. Um and uh but also uh there's the interesting thing from the Stanford data when we checked back in on this last week was that their measures of code quality reckons that the code quality was improving. Um and you know, the models are improving, the agents are improving. We're adding more and more guidance and skills and all these kind of things, which I think do craft uh or do force people down a road which should result in higher uh quality output. But um it's great to see when when when tools kind of can independently pull that out. Uh now, devil's in the details, you've got to go into the weeds, you've got to actually really have a strong sense for what quality means in your own environment, but you know, we're not seeing some of the things that people are worried about out there. But that's it, we've got a mature environment. We're 15-year-old SaaS company, we've been doing this for years.

You know, AI and speeding up your velocity will will magnify all of your strengths and weaknesses. And thankfully, I think we've got a lot of strengths on the software delivery side of things that we've been able to take advantage of. One thing that I want to kind of call out here, which is you said that you've seen your code quality increase, which again, intuitively, I've always believed to be the ultimate end game of this. And every engineer, not every, many engineers that I've talked to just don't believe it to be true. But when you have the capacity to take on tech debt, when you have the capacity to take on the dragons in in your code base, you actually can do those things, whether it's developer experience, security and compliance, just general maintainability of your code base, flaky test, improving your CICD, all those things become very tractable, not just technically, not just can an engineer execute on it, but actually the business, and I feel like people don't appreciate this.

The business, capital T, capital B, only has so much capacity for internal projects. Meaning, we can only allocate so much of R&D towards improving code quality. Just just how we live. We don't generate ARR on code quality, unfortunately. But when the the um when the cost of doing that compresses, then you're able to say, yes, as a business, we should invest there. One, because we can, and two, because it'll unlock velocity on the outside for our agents and for our product managers and for engineers. And so I think this is actually a really important moment for folks to invest in code quality and I often advise a lot of CTOs and VPs of engineering when figuring out how to get their engineering team AI piled, say everything you hate about the code base, go spend a month fixing and see how fast we can speed run that. That's going to feel really good. Okay, we've chit-chatted, we've shown graphs.

The point of how AI is to actually ship some code. So, let's switch over to that. We can probably come back to all these topics. I think they're so interesting, but you're going to show us how you all again in your mature code base, your mature organization, are actually getting things live and some stuff you've done in the repo to make

that possible. Yeah, sure. So, I'm going to do a pretty trivial change in our majestic Ruby on Rails monolith.

So, this is >> it. >> millions of lines of code, all the tests. Uh yeah, it's it's the code base is older than Intercom. It it was uh created before Intercom was incorporated. Um I'm you know, it's got it's got its problems, but we love it and we we tend to it. Um and so, uh I'm just going to do a relatively simple change of adding a uh a lobster emoji Rails redirect to chat.prd.ai. So, I also I try and give hints to Claude when I'm actually demoing something.

Um I don't know if it actually helps, but it makes me feel better. Uh just trying to add a bit of urgency here, you know? I think that's everybody's prompting strategy, which is I don't know if it helps, but it makes me feel better. >> [laughter] >> Totally. Um so, that's a nice way to uh interact with the agents, you know? Um and so, what we're seeing here is I mean, it's already kind of figured out uh where to put a redirect. It's got the nice lobster emoji. Um and it's asking me if I want to open a PR.

So, obviously I do and uh I think it's actually got the URL wrong. I It's I thought it was intercom.com which which will have the URL, but we can tell Cloud Code later on about that. So, what we're seeing here is first of all an important point. So, I'm just going to scroll back up. One of the things we noticed early on when we started getting Cloud Code to write all of our code, and you know, we're well above 90% now, is that it would create pull request descriptions that were kind of terrible.

They it would describe the code, and that's the least interesting part of a pull request. You actually as a human or even as a an agent reviewing code, you want to know the intent behind the pull request. You want to know the interesting bits, what's kind of related to this, and you know, LLMs are very good at just regurgitating or rewriting code into English. That's fine, but it's not what we need. And so, one of the things that we noticed as well when people are using Cloud Code, we created an LLM judge to evaluate cuz we had suspicions that the quality of the pull request descriptions was going downhill. So, we created an LLM judge to evaluate what does a good pull request We decided what a good pull request description should look like, and then got an LLM judge to go through all like months and months of data. And yeah, the trend was awful. The trend was going in one direction.

And this is bad. And you know, look, humans aren't perfect at creating pull pull request descriptions. Sometimes they're just blank and whatever. But I think with our use of tools like Cloud Code and setting up these kind of platforms around us, you really have to pushing forward like higher standards. You want as close to perfection as possible, and this was clearly something that we're just not going to tolerate a lowering of standards in our in our environment. So, we created a skill called create PR. And what it does is it uses whatever context it can from the session to describe the pull request.

So, it's not quite rocket science, but often the session knows exactly why it's doing the thing. And so, uh then we had to kind of force it in. You know, we we started we told people like, "Oh, just use the create PR skill." And then people would want to use it. You don't really actually want to be have people remembering things. So, we added it as a hook. So, if Claude's decides to, uh you know, use the GitHub CLI to open a a pull request, uh we just block it.

We say, "Yeah, tough. You need to use the create PR skill. And also, you're probably going to have to uh like figure out the difference text description. Uh and then I might interview you if it is done off context there. Hopefully, uh there's enough context in this. But the point being that, you know, this is a platform. We want great outcomes, uh and we measure the inputs and outputs. And after we we put this in place, the LLM judged reckon we're doing a great job now. And so, we're we're at higher quality pull request um descriptions now.

Hey, this is not the most important thing in the world. Like, this is not going to get Intercom to 2x or or to 10x revenue or anything like that. But it's the all of the composite little jobs that like are when you assemble means you have an extremely competent engineer who works appropriately in our environments. And that's where we're putting our investments for each little skill and hook to do these things. And they look almost inconsequential, but you know, they result in better outcomes. And so, we look through here, it's uh it's creating a PR. I'm going to have to check on what it's going. This probably will be automatically approved as well, which is pretty cool. And you we might even see some pull request feedback as well in action. Um ah still building. We'll come back to it in a couple of minutes.

One thing I want to call out for folks as as you were describing sort of why you put in this skill to improve the PR. And for those who don't know, um a skill is basically just like a set of instructions and sometimes scripts that a LLM or a agent harness can invoke at a certain step in your flow. One of the things that I was thinking as you were describing why you put this skill together and got really opinionated about PR descriptions is in engineering we have been able to architect really opinionated CICD pipelines. So, how written code goes from being written to deployed in production and we have I mean you saw it in GitHub, we have all these checks and lints and pre-deploy, you know, pre-flight things and preview branches, all these things once the code is written. What I think is really interesting about skills is you can bring some of that determinism to as you write the code how you want that process to go. We used to not be able to do it because it used to flow through the hearts and minds and hands of humans which are much harder to put in these structured guardrails and we would do this by writing wikis or having, you know, SOPs where it said, "Can you please follow step A B C D E?" And now you can just make it really easy to enforce those standards across a team which I don't think is micromanaging.

It's actually just making everybody's golden path much smoother to production. And so I think there's this just very interesting parallel to how we've approached CICD to how we approach things more upstream even from the product management perspective. Totally. We're on this movement towards a software factory. And what factories are great at is, you know, like an IKEA factory or something. It's all the same furniture, all the different bits and you know how to assemble it and look, it's not your artisan stuff. It's not or it's not cutting edge or whatever, but it's very predictable and, you know, has a certain quality and meets certain standards when it comes out the other side of the factory. And so, well, pull request descriptions again, they're not they're not make or break for the factory or or the pull request or whatever. It's one of those qualities of just good quality work that's reliable, predictable and then when it all comes together, you've got your IKEA factory. Well, and people don't want to feel certainly engineers don't want to feel like they're part of a slot factory, right? And so these things that you can add into the flow that actually up level and meet the standards of the engineering team really help your human engineers on the team feel like they're working at a place that values quality. And so I appreciate that you've put the that effort into um into these behind-the-scenes hooks and skills because I'm sure it reinforces to a culture that's being asked to move very fast, to ship how you know, ship things differently than

they have before that you still do care about their experience reading pulled pulled you know, pull request descriptions. Um they're they're um you meet their bar for quality and I just think it makes everybody happier. Yeah, well, it's great when the robots just produce the work that you'd expect of your best engineers, you know.

>> Yeah. And I you know, maybe as you get this live, I also think there are just still such more interesting problems to solve in software engineering and we can talk a little bit later in the episode about some of the interesting problems that you are all solving on the product side, on the technical side. I think there is no lack of hard, intellectually stimulating, creative problems to solve for customers and coding redirects is just 100% not one of them. Um So did we get do we get my redirect live or are we close? It's still there. I'm waiting for an automatic review to kick in, but uh we can come back to this. So one of the things I would like to show next might be some of the telemetry that we have in place. So we saw that uh you know, there was different skills getting invoked and um and we don't like flying blind.

Uh to run a system like this, you need to know how well people are using us. Uh are people using these skills at all? Uh you know, the kind of basic information that you'd expect sort of like when you ship a product to your customers like, you know, where can I see the usage? How can I fight for the usage? What's what's going wrong or what's not going wrong? And so, we collect a bunch of telemetry using different mechanisms and have different homes for us.

The most open one that we have is we collect basic usage information for skills and and the like and we send it to Honeycomb. So, we just have a shared key that's deployed on our laptops and anyone can go in and kind of look through this data. So, if you're developing a skill internally in palm and like hundreds of people do this, it's very easy for you to go in to discover like, hey, how where is our who's actually using this? When are they using us and you kind of use this as a kickoff to like follow up on just like basic discovery of usage of your skills and all. And like unsurprisingly, the kind of main skills that we have are things like creating PORs, admin tools is our admin like internal tooling APIs or where we have an MTP in front of it.

Build cards are CI system. Snowflake logs is where we put Snowflake. So, you can see from this like a lot of work a lot of the skills are being invoked. They're all around the building and then seeing where my stuff is and maybe some troubleshooting type information as well. And so, this is the first kind of step. It's like you don't have this, it's hard to have a large system like all these hundreds of skills and hundreds of creators working in this area without having decent telemetry.

The The next thing we do as well is we also collect all of the session data and put it into S3. And so, we anonymize this. We do a few things to make sure we're not doing anything too private. You know, people put all sorts of stuff in their sessions. They yell at their sessions. Yeah, and yeah, people have personal relationships at times with with Cloud. And like we don't really want to know about that, but we do want to be able to dive deeper into uh things are going. You know, I think understanding like how what the drop out rate of sessions like dude, how quickly people got to something useful like whether it was a PR or something like that.

This kind of information is pretty interesting and so we're harvesting a lot of session data and we're doing

different things. This is what I'm showing here on the on the screen is like a very simple tool that we put together which just gives you some personalized insights and you know, you can do this inside Cloud these days as well. There's and there's plenty of skills out there on GitHub where you can do session analysis but I think we we just built a little tool on top of our session collection system to give people feedback and it's feedback that we're interested in giving feedback about how their sessions are going and and how they're kind of fitting in, how you should think about your own I guess use of Cloud Code compared to everybody else in the org and you know, I'm not doing too bad here. It's like 79% you know, someone has to be down the bottom of every percentile and um there's and there's some interesting feedback here like it's tell it's it's kind of getting annoyed at me. I want to I was getting annoyed at Cloud a few weeks ago because I'd set up GOG to interact with all of our Google stuff internally and but it kept on trying to do the wrong thing and I was kind of giving it twists and ended up adding stuff to Cloud and MD and stuff and it's it's kind of giving it to me here or it's reminding me that this wasn't a very effective way to interact with Cloud Code. So you know, it's a good prompt for me to actually go and fix up my memory or whatever.

And like we all like people are at different levels even at Intercom. People are different levels of adoption. People are joining Intercom. They may not have like seen a system like this before and they want to know how things are going and get feedback and so this is one example of how we're just trying to pull together this information to give useful actionable insights to people so that they can they feel supported and that we're not just throwing them an API key and saying best of luck. It's like, "No, we've got We understand what growth looks like and the progression that people go through um when they're using these tooling um and getting better and kind of self-improving." We want to support all that. So, this is one of the things that we're doing with the session data. There's loads of other things that's work in progress um like being able to like we want to get insights to which skills are are are the highest quality, which gets you which which skills get you to your results as quickly as possible, and then which ones need work. You know, which ones uh aren't working out so well and might need a bit of attention to improve.

This episode is brought to you by Cursor. If you all have been watching How I AI, you already know this. Cursor is my favorite way to code with AI. Whether I'm using plan mode to build out an ambitious feature, reviewing AI-generated diffs right in my editor, or kicking [music] off cloud agents to multi-thread our roadmap, I reach for Cursor as my favorite multi-model coding platform. [music] Even better than building myself in Cursor, I love collaborating with Bug Bot to fix PRs for code security and quality, [music] and have begun relying on Cursor's automated agents to keep our codebase clean. It's not just me. The most ambitious teams love Cursor, too, including engineers at [music] Stripe, OpenAI, and Figma. Ready to build more?

We're giving $50 in Cursor credit to How I AI listeners. >> [music] >> Claim your credits at chatprd.ai/howiai. That's $50 in Cursor credits by going to chatprd.ai/howiai. I have to pause before we look at your list of skills because I'm so excited about that part, but if folks aren't watching, it they may have missed how amazing what you just showed is. So, I'm going to reiterate it, which is one, you've instrumented all your internal skills with telemetry so that in in you're using honeycomb Um love the honeycomb team you're using honeycomb to see how often those skills are invoked over time. So this is just a tip for anybody building out a skills repository internally or even somebody who is maybe trying to get some visibility into their impact across the org. Let's say you build a skill and you want to go to your boss and be like boss my skill is being used by literally everybody

every day. Um find a way to put event level to telemetry invoked in the skill a little dashboard and you can track those over time. Again treating your org like a product treating your repo like a product treating your AI setup as a team like a product and all products all good products have tracking plans and so figuring out how you put that telemetry in I think is really smart and then the second thing for for those that missed it or how to do it is you're taking all the raw session presuming JSON files. So for folks that don't know Claude code stores all your chats with Claude code in on your computer in JSON and you can go look at those or query those at any time. It sounds like you all are uploading those files to S3 and then layering on top of it some anonymization some user level views and then you're essentially building sort of what I would call like an internal eval of how people are using Claude code and what problems they are having over time. So that individuals one can triage their own implementation as you said oh it looks like I need to do this or that or improve my agents MD.

But then if you're seeing consistent themes over the organization on it's never invoking this MCP when we need it to invoke this MCP or people are yelling no every time the create PR skill gets queued up you can fix that at a systems level but you can't do that if you don't have the visibility. So, again, my VPs of engineering, my CTOs, my friends out there, put some telemetry in your skills, and then do some meta-analysis on your Cloud Code sessions across the org, and you'll be able to identify places where some probably some high-leverage fixes are going to get your team unblocked over time. I do hope and expect that this stuff will get easier over time, you know. Um Uh I'm I'm happy to kind of invest the work uh so that we can move fast and kind of be on the bleeding edge, but there's something to be said also for being for having like last mover advantage, and just getting all this stuff for free whenever I'm talking or whoever Um I mean, maybe this is a product just that people should buy uh or build.

Um but for us right now, we've no choice. We just got to build it. We're We We're We're like We're fascinated with the insights that are locked away in these sessions, uh and so we just got to build uh stuff so that we can see what's going on. I I love it. Okay, can we see some of these skills? Yes. Uh so, it's a very exciting GitHub repo. >> [laughter] >> Uh Our lives are all GitHub repos and markdown files. Totally. Um and we have we have a lot of activity at the moment.

We We ran an AI day last week, kind of getting more people uh contributing to us. And so, well, what So, what this is is it's a plugin uh repo, and uh we have a series of plugins, and they've been They're growing daily at the moment. Um kind of every team will have their own kind of specific plugins. Actually, in general, though, we're very liberal. We want stuff to end up in here, even if it's not great. And but we do sweat the details on the core plugins, things that we think are fundamentals, foundational ones that go out to everybody. And so, where we start off with we have like these base plugin, which gets installed.

Oh yeah, so we distribute this not via the cloud cloud code plug-in mechanism. Uh we found it was just a bit flaky. It was, you know, sometimes it would update, sometimes it wouldn't. And it would end up kind of like trying to manage a Python install on on hundreds of different laptops. You know, it's you just don't want to do it. And so we ended up using our internal IT systems to synchronize all of the plug-ins to the disks of everyone's laptops. Uh so this is a great cheat code and yeah, strongly recommend getting very close with your IT team to be able to deliver things like this reliably and not have to rely entirely on the cloud cloud code plug-ins

mechanism. Just our experience is a bit flaky and it just gives us a lot of reassurance. We don't have to do certain types of debugging once it's all on disk. So So this is we know this stuff works anywhere uh because we've got our IT team pushing it out to disk.

Uh and so we've got some safety hooks. We have some uh some of the bait foundational old things like, yeah, merging PRs. We don't want our agents going off into AWS. And then just different settings and the telemetry things as well. So these are uh the core things that absolutely everybody gets and um will be, you know, these are minimalist. We uh we don't want anything that could be inappropriate in, say, a non-technical person's laptop or whatever. So uh that's this is like the the basic uh building block. The The next main bit for us is like what we call developer tools.

Again, like this would be things that we vend to all of engineering and beyond at this point. Uh and these would be generally skills that would be appropriate to be used by any engineer in the course of their work day today. And we would have a high quality bar again for all of these. These would all require evals. These would all require to pass different kind of tests or analysis that we do on the quality of skills. Uh and so we we try and maintain these and make sure that they're well updated and well used and we pay a lot of attention to to I can maybe go through one of these skills in a bit of detail. This one's near and dear to my heart. It's flaky specs. And I think the interesting part here is not the skill itself. The skill does reliably fix flaky specs. And so, I can pull up in the meantime like here is a list of flaky specs that we have at the moment. I'm going to open up the skill and just start to run it on this issue.

And so, while this is running, just walk through what's in the flaky spec skill. And so, there's a checklist here. And the fun part about how I built this was not that I believe was a world-class expert at fixing flaky specs. I roughly know the problem and you know, have fixed a few of them in my time. But there's different class of you know, in a large test that test environments like ours, we have hundreds of thousands of tests. And if you're not super careful about like data poisoning or race conditions and all these kind of things and kind of kick in when you're running millions and millions of tests a day, you know, you end up with these tests that end up slowing down your ability to deliver code to production fast and reliably and not confuse developers by things randomly breaking. And there's kind of known patterns and known known ways you would go about this. And I knew my goal which was to have a skill fixing all of these flaky specs.

And it was something that agents are pretty good at where you give them a kind of testable goal. You know, it's wasn't quite open-ended. And I also had this huge backlog or yeah, there was a backlog of probably a few hundred. But then also all of this historical flaky spec information. And so, you can just harvest all of this data in your environment to go, "Hey Claude, I'm going to build a skill. First of all, go and research every single flaky spec we've ever had. And then we're going to build a checklist.

We're going to build a mechanism. And then we're just going to crunch through them over and over and over. And mean, to this like 1x kind of, you know, it's doing a good job, probably as good as the job as I would do. But then as you keep building up all of these like little teeny steps, which are the kind of things that, you know, our best rails coders kind of do. They've got all this stuff in their heads and all the different classifications of flaky

specs and, you know, verifying against real data um and uh and then but the the really fun part is then you get So, you get something that's starting to be like 10x. It's fixing flaky specs that I'm not even sure if I could do. Um it might take me a day or something. Um and I probably wouldn't do it. But then you start to add in stuff into the the skill along the lines of like, okay, when you fix something and it's novel, you need to update yourself as well. So, in that session, it's updating the skill. So, the skill itself is kind of learning as it goes along. And we also fan out. So, it's like, okay, I'm very happy that you fixed that flaky spec. Now, find every flaky spec that got impacted by that nature of it. And so, I went from zero to like 100x in terms of this skill now is like, you know, senior distinguished engineer.

>> [laughter] >> Uh they're all are being able to fix uh these specs. But it was more like the process that got there. Um and sort of like working with a feedback loop, working with like a very clear goal, and then giving it the freedom to do it, you know, giving it access to the systems where it needs to pull in metadata, being able to run builds itself, um and having that feedback loop where it's learning and And then, you know, designing the skill as well so that it's You don't have to edit this every so often. Ends up taking up too much information, then might confuse things.

But then you break things out into uh like reference guides. So, you're doing this like progressive discovery thing. And And I've even accidentally pointed this skill as like a Python code base. And Claude has just gone, uh like it's just Python. I'll be able to go. Uh and it kind of uses the knowledge that's applicable to us. And so, that again, this skill is not going to make Intercom's revenue go 100x. But, it's now this like perfectly reliable thing that we really no longer have to think about. So, now we can expand out into many, many different areas. And we're we just have to maintain this. And the maintenance work for a skill like this just isn't much.

And we have evals and stuff so that when we're upgrading models or maybe even moving to cheaper models or whatever, that we can make sure, yeah, this thing isn't regressing. It's still working as well as we think it is. And we've got confidence and certainty that this is still a reliable building block. And again, the constituent part put when put together, you've got like a very senior engineer who's able to get any work done in your environment. And so, yeah, we can take a look at what it's doing.

Oh, it's asking me for permissions. Should have checked. >> to date make no mistakes dangerously skip permissions. That's the rule on our AI. One thing while it's running, I wanted to say is, you know, this skill is a perfect example of what I call the like and then AI workflow, which is I tell everybody like pull your skills and pull your workflows through a bunch of and thens. So, I want to fix flaky flaky tests. So, I go to GitHub, I find a flaky test, I run through the skill.

Let's say you fix it. And then what would you do? Well, I would document how I fixed it. And then what would you do? Well, I would go find all the other ones that are just like this and fix them. And then what would you do? I would go from you know, our Rails code base or a Python code base and apply the same you can just do that over and over and because the cost of running these is so low, you can actually pull the thread of a bunch of stuff any reasonable human would have quit at step one because you're not limited again by head count or coordination cost, you're limited by the technical capacity to solve the problem, which I think is is a really

interesting way to think about how you get from like the, you know, engineering intern that whose job is to go through and take a first, you know, gentle pass at all these flaky tests through to the distinguished engineer who has just speed run through 300 of them.

And has thought of a completely different way to architect your your testing overall in your repo. So, I think that's a really great model for for things. And then the other thing is like, again, engineers, go speed run your tech debt, fix your flaky tech. Like these are all things that as somebody who's run engineering organizations, I have heard over and over, we can't because our code base, blah blah blah blah blah. Like, can we pretty please allocate this amount of time to just fixing this really annoying front-end flaky tests?

Like, you don't have to ask permission for that stuff anymore because there's just a new way to solve it. And I think, again, just going back to some of the stuff we were talking about earlier, I think your overall product quality is going to go up. I think your overall developer experience is going up. There's just so many good things that come out of using these tools and using them correctly. Yeah, I think backlog zero is a realistic thing for teams to be able to go after. You know, all the things that you wish you ever wanted to do, you know, it's it's now just achievable.

I mean, of course you got to balance it with, you know, all of the extra stuff that you can just deliver at the same time, but it's so sweet to be able to think that, hey, we actually have a path to getting rid of our all of our backlogs and all of the kind of architecture changes or whatever, you know, we we can Recently, I was taking a Go microservice and re-implementing it in Ruby. That was a single cloud code session. Before November, this was something that I would have had to advocate for on a road map and like, you know, plant some seeds in different engineers' heads and kind of build people towards this and kind of blame a lot of problems on the existence of this micro-friend.

Um but now we're Wait, trigger warning first before you talk about that process. >> [laughter] >> Uh I'm sorry. I'm giving you the secret sauce here of how to influence an org. But also, yeah, and but now it's like, well, I don't even have to think about this now. It's a single session. On the fact I can I can get Claude to implement it five times and compare the styles or compare the you know, get us review them and figure out what the best way of of implementing the thing is. And this is just like this level of kind of creativity and freedom that where like your imagination is the blocker, not not the the time it takes to actually knock out one of these things, which was months in the past, you know?

I I completely agree and I I feel this at Char Period where people are like, what are your I mean, I'm a product tool for product people. They're always asking what my road map is. And I was like, I literally don't have a road map. We burn down the road map every week and then we figure out what we're going to ship next. And of course, we have thematic ideas we want to pursue and things that are larger. And one of the things that I do to keep myself from over shipping absent product market fit is literally constrain the ideas to what I can do in my brain, which is there's like a natural throttle on not getting slop out because it's not engineering throttling me. It's actually just good commercializable ideas. And I think that's where we're going to see some of the limits start to come into play. I again referring to Anthropic.

And another big news piece came out is that they're hiring a bunch of PMs because they have so much engineering capacity, they're actually limited at the PM capacity. And so it'll be interesting to see where the bottlenecks in your business, you know, end up and which bottlenecks are appropriate. It's probably good to have a product bottleneck a little bit because then you're not shipping anything, um which customers can't absorb. And so I just I I think it's going to and it's going to evolve over time and then, you know, product is going to have a whole set of skills, and then I don't know what we're going to do with our time, hang out on the beach, but I think it's it's a pretty interesting time to to run orgs.

Yeah, you know, I think engineers, designers, product managers, maybe it's just all going to be one blob of builders or something like that. That's the beauty of it. Everyone Everyone just does things Everyone just does things. Yeah. And uh you know, that it's great. It's uh it's it's lowering the barriers to like just getting a lot of stuff done. And it's like so much fun when you can when you don't have to ask somebody or get something on the backlog or whatever, you can just get it done yourself uh or even just get it done very fast in a small group. Doesn't matter what your discipline is. It's just like a great leveler at the moment.

So yeah, so we're live. I think our lobster is live, and it should be on app.intercom lobster emoji. Look at that. It's amazing. I need to get you all an affiliate code, you know. >> [laughter] >> Uh yeah, I mean, lobster emojis, they're they're the new thing. They're the new um growth hack. They are the new growth hack. Okay, so we have seen your PR per R&D employee go up. We've seen how you can get from kind of cloud code to production very, very fast with a bunch of guardrails. We've seen your list of it looks like hundreds of skills, but at least dozens of skills that you're invoking via hooks. You're using that to not only ship customer-facing product, but you're also using that just to make developer experience better, burn down tech debt, all those things we want to see. You all are You're measuring it both from a telemetry perspective um both like quantitative and qualitatively, you're measuring your cloud code sessions. And you know, 2x isn't enough. You're going to get to to 10x. So you all are on the edge, at least for for folks that I talk to, and I'm sure you're like me where you're like sure, you think we're on the edge, but then I see people and they're really on the edge. So, we always have ambitions to move forward, but my question now to you is, how has this impacted how you think about your customers, product? You know, I'm an Intercom customer, I'm a Finn customer. I interact with Intercom code and Intercom UI literally every day. My Open Claw has an Intercom API key. How do you How do you think about, you know, now that you have this experience with Claude code internally, how do you think about what that customer experience is going to look like?

Yeah, there's a few things going on. One is as people are outsourcing a lot of decisions to their agents. And I know this is a good thing in many cases, but you know, there there was good research done recently about what does Claude code pick? And certainly I've had the experience in the distant past where I'd ask an agent to add something, except do it behind a feature flag, and then it would start to go and implement its own feature flag system. This was >> No, no, no.

in our code base, which has a pretty sophisticated old school uh home-rolled feature flag system. So, you know, nowadays, mostly we'll stick to whatever's in the code base, and that's fine. Um but, you know, SaaS products, they're really good at their jobs. They're actually worth paying money for. And uh getting back to the feature

flag uh situation, you know, uh if you're building a new business, you're you're um relying on your agents to make decisions, often an agent will when prompted, say, "Hey, how should I solve the feature flag problem? I want to make sure I'm doing all these safe deploys." And that uh the agent will just go, "Yeah, I'll do it myself." And the kind of build over buy decision, uh and you can see why the agents do this way, because they can achieve this. They can get it done.

They don't have to rely on the human. Okay, like Open Claw changes things here a little bit, and maybe computer used does as well, but still we're not really we haven't really adopted um SaaS businesses to be agent-friendly. And that means well, all sorts of things around uh how do we position our websites and content and how do you get updated in their in their knowledge and how do they really discover it? And but also can they actually just get it done?

Like can you ask an agent, "Hey, could you just sign me up to Intercom and get me uh straight and working on my website?" And so uh like this goes alongside was having to make more more APIs for things. I think I think uh I'm I'm kind of like omnichannel as such. I think like there's a future for CLIs and MC fans like REST APIs. I think I I'd like us to be get more comfortable around things like ephemeral APIs or multi-step APIs. I think CLIs are good at wrapping these kind of things. And but the whole point of all this where I'm getting at is like, you know, you want to be able to just help agents out at the at the time uh when they're interacting, they're in discovery mode, and you want to give them clues, you want to give them hints, you want to give them help to be able to do things like sign up for something fully without having to go back to the user and say, "Yeah, sorry, I can't help you there. You got to go away and like figure out how to sign up for something." Um so uh I've been working on something in the last few weeks which hopefully should solve a problem and I can I can paste in a uh a prompt and then see how far it gets. I also, [snorts] just while we're running this, I have to go back to your feature flag example because it you know where I used to work, it broke my heart that build it yourself was at the top of the feature flagging list. But I do think I have I have a a paranoia moment about this, which is model providers and harness providers are highly incentivized to build it yourself consumes lots of tokens versus buy it maybe it consumes less. So, I'm I'm just really interesting to see how this all shakes out. You know, people people are very anti-SaaS is dead.

And I'm a little bit more like, yeah, but like the current form factor of SaaS really is has something coming for it in a particular dev tools because these models are so good at writing code. I think you're in a real pickle to try to figure out how to find the right value wedge at the right moment, how you can allow agents to not just sign up and set up things, but purchase it, you know, like what is your trial experience look like if your first user is an agent. I think all of that is super important. And then, you know, to your point earlier where you said, you know, are we APIs, ephemeral APIs, CLIs, MCP? I think the answer is yes right now, which you cannot predict the the medium by which a user is going to come to your site.

They could come through a search and hit your website and download things and look through your docs. They could come through Claude code. They could come through an open Claude. You just really don't know. And so you sort of have to meet your customers and your non-human customers where they're at. And I think it's a really smart for teams that have any part of their product that needs to be implemented via code to be thinking

about this problem yesterday because you will be left behind, I think, if your agent experience isn't there.

Yeah, agree entirely. And I think there's a whole craft in how to make a CLI like agent-friendly. I think like MCPs obviously get that right a lot a lot a lot of the time. But, you know, for example, one of the things that we do in in the help is like pretty just give a hint to the the agent. It's almost like prompt injection to a certain extent except it's not malicious. You're just trying to get us along to what it's trying to achieve. It's like, well, maybe you could check email. And if if an agent has access to your email >> That's what I was looking at. Yeah. So, it's it's just there going, "Oh, yeah, I can probably get this done."

Or like you can hint to them like I've kind of cheated with this. This is on my own personal website I've hosted in for sale. And it's is I've kind of pre-populated a few articles so they can upload and Finn has some content to answer questions with. But, you can also just you know, return in the the help going like, "Hey, you know, you should probably think about creating some articles if you want Finn to actually start answering questions." And that can be extracted from, you know, the code base or whatever.

Yeah, being like I've been asked to think like a lot of interfaces like CLI interfaces. Like I use Gog, you know, it's part of the open claw universe. And I think it's a lot better than the official Google AWS one. And but I think if you would start to use it, it's it's actually just more human. As in it's the interface just kind of makes more sense to a human. I think the Google one is like, I kind of get what they're getting at and it's kind of jason in there and stuff like that.

It's not that it but it feels more human-friendly. Or something that things that are effective for agents can often be things that are more human-friendly because they're discoverable these verbs and words and not just kind of inscrutable weird stuff going on in command line options. I think I've confused Claude here. I'm not sure what where this is >> That's That's okay. I'm going to I'm going to narrate it for folks what's happening here which is you basically said like install Intercom on this site.

There's an Intercom CLI that's like, "Cool, I can access the Intercom APIs and do a lot of this." My favorite part of it though is signing up, getting a verification email in your email address, invoking via like this hint basically of like, "If the user has email access set up in however you're accessing it, go check for this verification email because we have we have a code in there that we got to snag and because you're using Gog which is a command line tool to access Google Workspace.

I you you can go do that. Pull that code in and what I think it's so interesting about that particular flow is you know, I I think AI is creating sort of race conditions in shipping across the org which is like you can Yolo a CLI probably faster than whatever team that manages email authentication can change how email verification works. And so you're like, I'm not going to let that break my product. What I'm going to do is create a flow that I can I can use that sort of sticky part in the flow AI brains and and get through it. And so again, your product doesn't have to be perfect for an agent to traverse it. And this is one of the things I'm actually really excited about SAS. Is all those things that are just so complicated to do as a human. Multi-step forms and like nested fields on nested fields and finding, you know, categories and just those things that I would say UX designers and product

managers have written their most tedious PRDs on and done their most detailed specs on. Like you don't actually have to worry about making that quote unquote usable cuz you can just brute force intelligence against it and and solve the problem.

And so I think that's interesting because the core value proposition can get bigger without being constrained by the surface area of a website or a UI or any of those things. And so I think if you're not thinking about what does that CLI look like for you and what adjacent systems does your product butt up against? It may be email, it may may be it some other dependency and how an agent might traverse those systems, you're just going to get less and less adoption cuz this is going to be more how people install products.

Yeah, I'm if I don't poke holes and if I don't rate make a CLI that kind of bypasses some of the ways that product works, somebody else will. You know, they'll just put their own agents on it and they'll burn more tokens, they might get frustrated and you may as well short cuz them and give them an interface which just works. May not be the perfect interface, but that's the beauty of these things you can get updated over time, you can agents can just pull down the latest version and and yeah, like hopefully I have something to show here though. Well, the the other thing that I I want to call out while you're talking about that, which is as I'm watching this and it's taking some time to build your conversion rate drop off point is somebody pressing the escape button and just saying forget it. Like this is clearly not working. What if we built it ourselves? And so I think it's a really interesting moment for product managers who right now are not getting the visibility of the drop off, right? When you were going through a website, you could put telemetry in it.

You could say, okay, user is going to the sign up page drop off, email verification drop off, go into the docs drop off. You could build this nice little funnel that identifies where your users are having problems. You can put some telemetry in your CLI, but at the end of the day some of that drop off and the alternatives is very invisible to you here and the the switching cost quote quote unquote is like pressing escape and saying do it a different way.

And so again, how quickly you can speed run to a zero to one installation in an agent, I think is something that everybody should be running right now. Um and it doesn't just have to be a code product. Like I think more and more people are doing non-technical tasks and interacting with non-technical SaaS in Claude Code, in Claude Co-work. And so it you know, even if you're not dev tools, if you're not thinking about how can a user do this quickly in in in a third-party harness or or system or an agent can do this quickly, um you're really missing out on customer growth.

Totally. Okay. How are we doing? It's on its fourth attempt. That's fine. And you know what? Let's Let's press Let's press the escape. Because, you know what? Let me tell you how cheap that exercise was. It was like 5 minutes and some tokens. And you're going to spin up a fresh Claude code. You're I don't know if you put make no mistakes. That was probably what we missed. Make no mistakes. Um and it and it could have done it. And again, this is just learning. Like, why why aren't Why isn't every engineer, every PM doing this once a week or once a month just to figure out how it can work. Um and then it's great. So, Ryan, you've shown us everything. You've given us all all the secrets. Let's get out of the terminal and let's do some lightning round questions.

So, my first question for you is how does it feel? Because what what I observe from our conversation is it feels fun. Like, culture has in fact gotten better, not worse, because of this investment. And so, you know, as a company that has really put in the effort both on the on the customer side and internally, how do you think it shifted culture? Has it at all um what have you observed? Yeah, everything is just faster and more exciting. You know, I mentioned feedback loops a good few times and you know, you can just get stuff out there so fast now. And uh I've been having the most amount of fun in my career over the last 3 months or something like that. And like, it's it's fun in many ways. It's fun because I can do stuff that again, I would have had to convince other people to do or they were just things on my wish list and I could never get around to them, and I just kind of complain about them.

Um but now they're just realizable. But also the fun aspect of like making other people productive, like leveling people up, getting like removing work. I had like Intercom has pretty good culture around resisting like the kind of slow movements towards being a large company and all this process and stuff like that. So, we're kind of in denial that we're like a large company. And I think it's a healthy way to work in many ways. And uh but this has kind of got us back to our roots in in a lot but you know, you can make fast decisions and get them delivered and get that feedback super fast. Um and I've been able to like ship actual features, like not just the CLI, but I shipped shipped some webhook features and um it's been a long time since I've done that. I'm just I've been in the weeds and platform space for a long time. Um and but it wasn't even a big deal. It was like just a couple of hours just kind of get something done.

It was like something a customer asked for. So, my job has become more varied. Um I'm able to kind of see more and get more done and help other people get a lot more done. So, you get this kind of excitement and velocity increases. And you know, we have all those measurements, and that's all kind of good stuff, but just the excitement of waking up in the morning going like, "I'm going to get a lot done today."

Like that is a fun way to go about your day. I I completely agree, and I hear this over and over and over again. I certainly feel it myself, which is this is the It brings me back to why I learned learned to code. It's like that same moment of I didn't learn to code because I like to type code. I learned to code because of the magic of you running like Hello, world and it it shows up somewhere. And that feels so it's just a very creative experience, which leads us to my second question, which is I see all the time that one of the most impactful change agents inside an engineering organization can be a senior principal engineer saying, "Let's go ham on some AI code, and the single most blocking person in the organization can be a senior principal engineer going, "I don't believe it. Absolutely not. Not me. Not here. Not No way." And in fact, last week I heard a story of somebody who had their most senior staff engineer quit. Says, I and I quote, "I do not believe in AI.

I will not work at a place that does this." So, what is your appeal, sort of engineer to engineer, of of why to invest in this? Why Why you think it's the way that engineer organizations are moving, and how you kind of come to meet skeptics where they are, um and hopefully see things a little bit from more from where kind of Intercom is approaching them? I mentioned that Intercom kind of had it on easy mode, um we didn't have to convince leadership that there's something to this AI stuff.

Like we were pretty much had decided the direction of the company the week ends that ChatGPT came out. So, uh so we already had this expectation that this will be transformative across many parts of our work, including all of building products and engineering. We were just kind of mostly annoyed about how long it took. Um Um but I think uh for sure it does need strong advocates, and it needs to push uh boundaries. Like one of the biggest things that I've been able to do successfully was kind of push through the barrier of like, "Should we let an agent connect to Snowflake?" Like what Like And there's all these things can go wrong. Or should we let our agents run real production code in in our Rails console over API? And the easiest thing to answer there is like, "Well, you know, I'm not sure." Or like this is this is risky. Or we we should think about this. But we've been largely pushing through it. So, now like not recklessly. Like we've lots of good controls, and we're a mature business, and we have like I've been on our security team, but definitely not trying to do anything too wild. But there's still even then, I have apprehension. Just like, is this like I think I I think we should do this. But it seems weird or it seems hard.

But then I just have to give myself permission. And then I realize if I have to give myself permission, there's lots of people out there who just need need permission. And honestly, like one of the biggest things I do at Intercom is just telling people they can do things. Just just a pre pre post AI and or telling them like look, whatever you do, just blame me if it all goes wrong. And I guess maybe we can blame blame Claude's now, but but ultimately it's that like permission and just like there's a level of ambition which comes from as well as like if you if you're out there saying I'm not sure if AI is going to take or have a big role to play in all of our work. And if you keep on saying that, that kind of will permeate through the culture and people say that. But if you're very clear, you're saying you're saying that like look, all work is going to be agent first like at some stage in the near future. And so we're going to figure out the path there. And so we're going to break down every barrier as we come across them. And look, it's your job, it's my job and if anything goes wrong, blame me. Like that's largely been how I've been approaching it. But not just me, like this has been a very large collective effort. But giving that kind of permission thing, but also the kind of like freedom to like explore or push things or whatever, it's kind of necessary and look, it might be a less stressful way to go about it to like just take a nap for a few years and come back and then will all the problems have been solved and we've got these perfect agents running and working in our environments, then then that that would avoid some of this.

But like I think all places have to get through that kind of apprehension and initial kind of issues that some of these can some of the introduction of agents into environments can have. And I think our job as leaders, whether it's as an engineer or as a manager or whatever, just has to be on that like enablement hunt people space to to to go deep on the work, enjoy us, and like have that moment where things click and you start realizing like, "Oh my god, this is something that will transform how much I can get done." Say it again for the people in the back. I love I was like, "Oh my gosh, I love this so much."

And you know, I it is absolutely those two things, which is like give permission. You you can. Please just go Please, by all means, go ahead. Designer, hit me with the PR. No one's going to get mad at you. Like go ahead. And then the second thing of just accountability can roll to the top, and not in a scary way. Let's not do irresponsible things. But I you know, I've seen I've seen a couple incidents in the past months, some big ones. And what you see is CEOs or big leaders coming out and saying like, "The team's shipping, and we want to keep shipping, and we're going to be careful with our customer data, and we care for the customer experience, and

stuff happens, we've learned from it.

It's ultimately on me. I'm going to call the customers, and we're going to we're going to move on and deliver great innovation for you." And you know what I tell people to you know, to get them over that hump, which is like, "You really got to know what your existential problem is." And I love what you said is the second that ChatGPT came out, Intercom changed, because that is an existential problem. Who writes the code in your code base?

Agents or humans? Not an existential problem. Like, will you be fundamentally disrupted by a new technology? That is the real problem in your business. So, I always tell people like, let's differentiate the real problems in our business from problems that we can tolerate, and then go go go use the problems we can tolerate to move fast. Um and so, it sounds like you have a really good call I mean, I think at the end of the day, the results speak for themselves. And again, you all are not asking me to say this. Intercom has met the moment.

You went all in on AI-assisted, you know, customer support and experience. You're now building models. And so it's not just a one-and-done. Chat GPT's here, we need to change how our product works. Or AI assisted coding's here, so we need to change how our engineering team works. It's, you know, models are going to be how people differentiate, we need to go there. CLIs are going to be how people use products, we need to go there. And so I think this sort of like fearlessness and what I would suspect is like just a fun, nice, high-trust culture, good people. Uh, you actually see the business results on on the other side. So I'm going to hype you up. I see a lot of teams, I see a lot of leaders.

Um, and I think people can take a lot of inspiration from this. But let's uninspire them really quickly before I get you out of here. Which is my last question, which is when um, Finn takes 15 solid minutes on a live live podcast to do a very basic task that you know it can do. Or not Finn, when when Claude Code. Yep. What do you do? Do you yell? Are you a yeller? Um, What does your What does your meta-analysis on this internal dashboard say?

The human needs to improve on. I I I do lapse into giving Claude Code like just like smiley faces or unhappy faces or, you know, not over the top. I I certainly haven't cursed at us. Um, Very polite. >> kind of not my style. But I do like the odd kind of like attaboy kind of smiley face and I don't know if it knows like that I'm deeply thinking about this and like these little subtle kind of hints or whatever, but um, yeah, no, I think like professional with a few emojis is is my style with Claude and, you know, hopefully that'll come back to me someday with an emoji.

Same. I waste the tokens on telling it it did a good job. I somehow in my mind I'm like, that's going into into its own sense of it itself. And it's going to know what good looks like. So I I am there I am there with you. All right, Ryan, this has been one of my favorites. Y'all, if you have gotten to the end, there is so much alpha in this episode. I cannot believe it. This is a cheat code to winning friends and influencing SAS um through AI engineering. Brian, where can we find you and how can we be helpful?

Uh I can be found on the internet as a nice friendly URL, which is brian.scanlon.ie. Uh I've got a few links here to some of the talks and some of the writing and different bits and bobs. As you can tell, I'm not a designer. I asked Claude to design this as if I was a Unix systems administrator writing a little webpage and it kind of shows. Um I'm active on X Twitter um brian_scanlon. Um I'm on LinkedIn scanlonb or something like that. I think I'm the most famous Brian Scanlon on the internet. So, generally if you search Brian Scanlon, then that tends to work. Um and I tend to be active in like showing up to different conferences and uh just like getting good word out about what we do in Intercom. Mostly these days AI, but I've also given lots of talks about many other different topics.

And um yeah, I'm also a big believer in just saying yes to a lot of things. Um so, if you look me up, you've got a good idea, you want to get in touch, uh you want to run stuff past me or whatever, chances are I'll say yes and we can I'll just keep on doing this until things break and then I start saying no. So, but I'm still not there yet. So, bring them on. Great. So, search for Brian and ask him to do something for you. That's it. Well, thank you so I thank you truly for sharing all this information. People are going to get tons of value out of this.

It's going to be a hit uh for sure. And I just really appreciate you joining How AI. Of course. This is so much fun. Thanks so much for watching. If you enjoyed the show, please like and subscribe [music] here on YouTube or even better, leave us a comment with your thoughts. You can also find this podcast on [music] Apple Podcasts, Spotify, or your favorite podcast app. Please consider leaving us a rating and review, which will help others find the show. You can see all our episodes and learn more about the show at howiaipod.com.

See you next time.