

05182025 Pre-Build Keynote Kevin Scott

Pre-Build 2025 Keynote & Fireside Chat
Kevin Scott, Frank Shaw, Jay Parikh
Sunday, May 18, 2025

(Music.)

FRANK SHAW: Hey, everybody. Hey, we're back!

(Applause.)

For those of you who don't know me, I'm Frank Shaw. I lead Communications here at Microsoft.

And let me just start by saying welcome and thanks to everybody who is here in the room, able to join us in person. It's always great to be able to host people here at our Redmond campus. This is part of our brand-new facility, and of course, the Pacific Northwest was beautiful today. Thank you for bringing good weather from wherever you came. And it's great to see a room full of media, creators, analysts, founders and developers all in one place.

I'd also like to welcome everybody who's joining us virtually from across a variety of time zones. It's probably the middle of the night for many of you, but I'm confident tonight's speakers will keep you wide awake as we go along.

Now, before we get into tonight's event, I want to just touch on a little bit about what you can expect out of this year's Build, which kicks off tomorrow morning. And it's always exciting to see thousands of developers from around the world get together to share ideas, try new technology and dream about what they're going to build next.

I'm not a developer, but one of the most beautiful things that's been happening over the last set of years is that there's been so many advancements in the way people build apps that even people who don't have coding experience can start thinking about what they might build.

I'm a huge Bob Dylan fan. I have always wanted an app, a Bob Dylan random quote generator, and nobody has ever built it for me. And so, with a little help from my friend, GitHub Copilot, I built that app. And I can click on it and I can get a random Bob Dylan quote. It's a very small target audience for this one. And so, I don't know what the TAM is going to be, but I'm very happy with this.

And then just this week, I had some fun with our Copilot image generator, thinking about some of the things that you can do. Here I am, at an event with some of our international creators, sitting there talking and I thought, gosh, I want a new profile. And I really think I should have been in the Minecraft movie. And so, if I could make myself a Minecraft character, I should do that.

And so, you can see, I'm using Microsoft 365 Copilot. I said, turn this image to Minecraft style at the Microsoft Build Conference, and it applied the Build style sheet. It automatically knew what to do. And then here I am, see, same purple shirt that I was wearing, a little bit of a block head and I'm ready for my residuals when the next Minecraft movie comes out.

Then once you start going with something like this, you want to keep going. And so, I thought it would be fun to do it with some other people. I saw Todd Bishop here earlier.

Todd? Okay, there you are. Here's the prompt. Now, I'm just using Microsoft Copilot for this, and there's Todd with the GeekWire logo. Todd, you can recreate this. I expect to see it on your profile pictures pretty soon.

Tom Warren, I know he's always hard to pull away from Xbox. We thought we'd bring him to Build and still give him playing Xbox, so a little bit of a blocky head there, but we've got a very young Tom, by the way, looking good in Minecraft.

And then Tiffany Jansen (ph), I know, was unable to be here in person, but is watching live, and I thought we could put her in the event as a Minecraft character as well, right here in Seattle. I thought about doing this for everybody attending but ran out of time. We'll stop there and we'll get back to Build and what you can expect from Build.

And just as a reminder, the news embargo is Monday, May 19th, at 9 o'clock. That's tomorrow morning at 9 o'clock, Pacific Daylight Time. This is the same time the keynote starts. Satya will walk out on stage for the keynote, and he'll talk about all the ways that Microsoft is building an open agentic web, how we're making this possible across the Microsoft stack, from tools and apps to the platform and infrastructure. And you should expect to see Kevin Scott join the keynote to share some interesting content as well. Maybe you'll even hear some of that today.

And then on Tuesday, we'll have a technical keynote where our speakers will bring together some of the product news to life through a bunch of demos and code. Beyond the keynote, there'll be a ton of breakout sessions, live demos, interactive labs and networking opportunities. We really wanted to make this a code-first event, which is what we heard from our developer community. They want to get in there and get their hands on code as quickly as possible. And as a platform company, our goal for Build is to help developers turn their ideas into reality.

That's a lot to look forward to. It's all happening starting tomorrow morning, but for now, let's get to this evening's program. And in just a minute, I'll welcome Kevin Scott, Microsoft CTO, who will provide us with some grounding on the state of AI in the agentic web. And then after that, Jay Parikh and I will have a brief fireside chat where we'll learn a little bit more about the work he and his CoreAI team are doing. And we'll close with an Off The Record reception for those who are in person. And I'll touch a little bit more on that at the end.

Last note, you are free to take photos, post interesting content from anything you see tonight, from the stage, from Kevin, anything I've said before if it was interesting, the conversation with Jay, etc. Please remember that any of the content shared on the embargoed news site, or the

interviews you have done with even some of the people that might be on stage need to adhere to the embargo.

And with that, I'll welcome Kevin on stage.

(Applause.)

KEVIN SCOTT: All right. Thank you very much, Frank. Thank you all for being here today. It's great to have this opportunity to chat with you before the Build conference gets fully underway tomorrow.

The thing that I wanted to chat about is thematically, this idea of the agentic web, which I'm guessing a bunch of you have been thinking about. I certainly spend most of my time in my role as CTO thinking about how all of the pieces have to come together in a very complicated AI software ecosystem to allow people to build things that fully utilize the capability and promise of some of these advanced AI systems we've been building over the past handful of years.

But I thought before we get into a bunch of jabber about architecture, that we could start a little bit with a review of the past 12 months. And one of the things that I like to do at Build every year, and I'll do a little bit more of this tomorrow, is to try to make some predictions about the 12 months ahead. And I think the thing that we can all see is making those predictions even 12 months in advance – which isn't a huge time horizon by normal standards – is incredibly difficult to do.

One of the big things that happened over the past 12 years that I think caught a whole bunch of people by surprise is the advent of these reasoning models, and all of the things that reasoning inside of these advanced AI models unlock. We have things like Copilot Researcher, things like Deep Research in ChatGPT, that can do just incredible, extraordinary things.

And I'll talk a little bit about one of my core beliefs around AI right now, which is that we have this thing that I've been calling the capability overhang where I think for a whole bunch of use cases, we're no longer necessarily reasoning constrained in the AI applications that we built. And in fact, there are just a ton of opportunities to have a higher ambition for the reasoning component of the agentic software that we're building.

One of the things that's probably obvious to all of you who are developers here in the room, roughly aligned with the development of these reasoning models, is that coding model improvements and software engineering agents and copilots have gotten extraordinarily good over the past year. If you look at some of these metrics like SUI (venture?) that are measuring the percent of tasks that the model of agent is able to solve for developers, it's just been extraordinary progress over the past year.

And the things that Frank talked about a minute ago, him building his Bob Dylan random quote generator, or just the constant stream of anecdotes that I hear from everyone—from developers who are doing cutting edge things on big, complicated, distributed systems -- to people inside of Microsoft Research trying to push the frontier of software engineering and algorithm

development, all the way to what I see kids doing in high school who have no prior programming experience at all.

All of that interesting activity is happening, just because we've had a real incredible set of breakthroughs over the past 12 months in these coding models. And I think the breakthroughs will continue over the coming year.

I thought it might be interesting, though, rather than me talking just about the progress, from my point of view, to go out and have a conversation with a bunch of people who are working on the frontier of AI. I spent a few minutes chatting with folks who have spent a lot of time over the years thinking about AI, who are playing leading roles in pushing this frontier forward for all of us. Let's take a minute and watch this video.

(Video segment.)

KEVIN SCOTT: Yeah. this is super good stuff from a bunch of our friends and collaborators and colleagues.

The thing that I would encourage you all to do is to think about the past 12 months and what has been surprising or eye opening to you.

One of the things that I will share is I've got two daughters, an 8th Grader and a 10th Grader. And my 10th Grader has an internship at the Cardiothoracic Research Lab at Stanford, where she does bad things to rats in service of science. And my daughter is working on this project that her PI assigned her, which is to figure out how to build a more accurate blood oxygen saturation meter.

And so, normal blood pulse oximeters take infrared readings from your finger. They're super inaccurate. The highly accurate pulse oximeters require arterial blood draws and are super slow. And if you're a heart surgeon trying to make moment-to-moment decisions when you've got a patient's chest open you really need accurate data, and you need it very, very quickly.

And so, she's working on this medical device. And at some point, when you're building a medical device, you've got to write code. My daughter could not have less interest in coding, despite my efforts to get her enthused about it. But she's got to the point where she has to do something. She doesn't have money to go hire her own developer.

And so, I start getting these questions. She texts me one night and has, "Dad, what's a CNN?" And I'm like, "Why do you want to know, child?" And she explains the problem that she's trying to solve. She needs to segment out these retinal photographs and to get just the blood vessel pixels out of these images so that she can perform some other calculation.

And I'm thinking to myself, I'm in the car coming back from dinner, I don't have time to do this research myself. My intuition is that it's slightly the wrong technique to solve the problem. She doesn't have enough data to train a CNN. There's probably not one that's been a convolutional

neural network, by the way. There probably isn't one that's pre-trained for this problem. She probably should be using classical image processing techniques.

Rather than going off and doing a ton of research myself, like I would have done when I was a grad student, I just asked Deep Research how to solve the problem. And I asked it very specifically how it would solve the problem, and I asked it to write the code for it. And it asked one round of clarifying questions and then sent a complete solution back with 27 citations, including a bunch of medical literature where this problem had already been solved and a working bunch of pipeline code. And the Share button in the app let me send it to my daughter.

When I was in my 20s in grad school, having studied computer science for a whole bunch of years, including taking an image processing course at some point, that would have been a couple of days worth of solid work for me. And this AI system had solved it in four minutes.

And the point of that story is that there are these extraordinary things happening all the time, because the capabilities of these systems are increasing. And an even more extraordinary thing that's happening is you have people like my 16-year-old, who only know a world where you can ask an AI how to help you solve these complicated problems that all of us probably have some preconceived notion are too difficult for us to go tackle, and then she gets the problem sorted out. And the unlock that will come from that over the next handful of years, I really do think is going to be just absolutely extraordinary.

The thing that I wanted to chat with you all about now is this rate of agentic software adoption. If you look at what's relatively visible about agentic software adoption, we've increased the number of daily active users across the entire ecosystem of agents in Microsoft and out by about 215% over the past year. We have this really tremendous growth rate, not just to the capabilities of systems, but how people are using them in their daily life to do real work.

But still, we are not really leveraging the full potential of these AI systems. And it's why I wanted to chat with you all today, and why one of the big themes of the Build conference is going to be around this idea of the agentic web.

We likely aren't nearly as reasoning constrained as we have been in the past, but there's a bunch of stuff in this ecosystem that has to get built out so that agents can more fully deliver what it is we hope that they can deliver. You want agents to be able to delegate work to, and you want the work that you're delegating to these agents to be increasingly complicated over time, like the help me sort out an image processing problem for my medical device, or I don't know whether that Bob Dylan app is all that hard, Frank. Sorry. We want to solve everyone's problems no matter where they're at, because the thing you want is human imagination to be as unconstrained as humanly possible.

In order for this vision of these really powerful agents coming into existence, they have to do some things that they can't do right now. And those things largely are not about reasoning. The reasoning will continue to improve. We just are going to continue to see great progress there. The things that I'm seeing right now from our partners and the things that we're building here at Microsoft are just stunning. You all will not be disappointed by the rate of progress on the

reasoning frontier. But there are a handful of new things that have to start happening pretty quickly in order for agents to be the recipients of more complicated work.

We think about this runtime layer that sits below agents and there's some really important pieces in this runtime that need real work. And we'll talk about some of this at Build, and we've got some announcements to make. And some of these problems are going to get solved out in the open source community, because the important thing about them isn't that Microsoft's opinion gets expressed, or some big tech company's opinion gets expressed, or we've won some technical argument. Some of these things only matter when they're ubiquitous.

And so, it kind of doesn't matter which approach you choose, just as long as you get to the thing that's really ubiquitous as quickly as humanly possible.

One of the things that is quite conspicuously missing right now in agents is memory. And because of this lack of memory, most of what we're building feels very transactional. You interact with an agent, maybe it recalls past interactions, probably it doesn't. It almost certainly doesn't remember its scratch work over time, like the way that we would. We solve a problem once and we write it down somewhere on a piece of paper, or store it on disk, or we remember it.

Memory is one of these problems that is really going to be important for us to solve. And it needs to be a form of agentic memory that probably more mirrors what happens with biological memory.

Biological memory is a funky thing, because it actually isn't like a database at all. It's very, very imprecise, but we have a set of cognitive tools that we use to have large recall, so that when we're trying to have a recollection about something, we can refine the recollection across a huge amount of information. And we can refine the precision of the recall using a whole bunch of tools, so that when we have slightly misremembered something, we can get to a more precise recollection.

And ultimately, we do need our agents to have precise and broad recollection of past interactions with us and with the information that they process over time, the solutions that they have previously made to problems. And there are a bunch of super cool things that we are doing right now to address some of these challenges.

Part of the challenge that we have right now is it's difficult to give an agent full context, given a prompt. The prompt sizes or the context windows are limited now. The fundamental computer science about expanding context length is kind of funky. There are a whole bunch of things in there that are quadratic. You can have longer context, but it gets really expensive the bigger the context windows are.

And we've got biological clues that tell us that that's probably not the way that a real reasoning memory works. We have fragmentation. Information that we have available to agents is distributed across a whole bunch of different stores of knowledge and memories. And we have these precision and recall problems where sometimes agents hallucinate. Sometimes, because of this context or fragmentation problem, you don't even have the information available to the

agent so that it can do recall. And it is iterative in the refinement of these memories and can't get to what's missing.

And so, some of the fun things that we are doing, that we'll announce and we'll have a GitHub repo for, are things like structured RAG, these memory exercises they would give us in school when we were a kid. And this is a core part of how you train a biological brain.

You don't brute force everything in your head every time you need to solve a particular problem. You don't go from first principles and just grind it out every time. You have a whole bunch of cognitive things that you learn, that we figured out over time that help us be more precise and recall from memory.

And so, structured RAG is one of those biologically inspired techniques where we are imposing structure on top of context to help have more precise recollections for models, so that memory can get better, and so that you can reason over larger volumes of information.

I've already said all of this good stuff. And this is the GitHub repo. This is live right now, so you all can go check this out and start playing around with it.

One of the other problems or set of problems that we'll have to solve with memory is it opens new problems when you have agents that have perfect recall and have perfect recall over every interaction that they've ever made. Context really is going to matter. You will have these things where you likely are going to have a personal agent, and you're going to have a work agent. And the work agent is going to have a whole bunch of your employer's information in it that is yours and the employers.

And so, we're going to have to really think hard about these enterprise scenarios to make agents really useful, and how you're going to deal with memory entitlements. How do you make sure that the agent has access to everything it needs to have access to, and IT doesn't lock the whole thing down and make the agent have some weird amnesia, because they're paranoid about where information is going?

And the systems here also have to be developers first. We can't have these super complicated systems. You all should go check out structured RAG. Tell us whether you think the code is good or not. It's one of the reasons why we're trying to do a bunch of this stuff out in the open, is I think simple, it's going to win here. And we've thought really hard about how to make the basic principles here as simple as humanly possible, so that people can jump in and help us, or help the entire world refine where we're going.

The other thing that's really interesting and conspicuously missing in the agentic web is, we've talked and you heard from the people in the video mention the fact that the reasoning model helps you do planning. You have this reasoning capability, which means that you can chain together a bunch of complicated actions, which is awesome. And I think those capabilities will get better and better over the coming year.

But one of the things is really hampering the ability of agents to fully leverage that planning capability is we don't have enough of the universe wired up in a way where agents can access those things. And this is where these protocols that are emerging are really super interesting.

You all have probably heard about MCP and A2A. MCP in particular, we're very, very excited about, and it's filling such a unbelievably big need in the ecosystem that the adoption of it – even though what Anthropic put out early is a really, really super simple implementation – it's really breathtaking.

And so, the way that I've been thinking about the agentic web and these protocols is it's a little bit like the unrolling or the emergence of the web back in the '90s. Some of these things have very similar equivalences to the web.

MCP is like HTTP. It is a protocol where you can connect to a service, or someone who's got a piece of content they want to make agent accessible. The protocol itself doesn't have a whole bunch of opinions about what ought to be in the payload. It's super simple. It's easy to implement. The structure of it is going to make it easy to amend over time, to layer things on top of it so that you can solve other problems, where you don't have to throw the whole stack away every time you need to make a minor tweak.

And all of those are super, super important to getting to ubiquity, because that's the important thing here. In order for agents to be as useful as they could be, they need to be able to talk to everything in the world.

It's also important, by the way, I think ideologically as well, because if you imagine this counterfactual scenario of what would have happened with the World Wide Web if you picked a player in the ecosystem, say, the browser makers, and the browser makers had decided, we want to control everything in this ecosystem. And so, we have to have a bunch of vertically integrated protocols. And our imagination is going to be the imagination that decides everything that's going to be the web.

Obviously, 20-30 years of hindsight tell us that that probably would have been a very uninteresting or less interesting version of the web than the one that we have right now. And so, the important thing about the openness of these protocols and simplicity and trying to get to ubiquity as quickly as possible, is that it means that your imagination gets to drive what the agentic web becomes, not just a handful of companies that happen to see some of these problems first.

Yeah, I talked a little bit about the capability overhang a minute ago. I think it's kind of an obvious thing at this point. The thing that we have observed over and over again, and Bill hinted at this in his part of his video interview, is we already have extremely useful things that agents are capable of, like medical diagnosis, for instance, that are not available to literally billions of people in the world who do not have access at all to any quality of medical diagnosis.

And it is just fact right now that models are more capable than what they're being used for. And part of our job is developers, like you all in the room, and developer advocates and journalists

and people who are writing and thinking about all of this stuff, is to try to figure out for those really urgent problems that we collectively have, how can we help solve this capability overhang?

You've got a ton of space for innovation, and we're seeing it. Many of you here in the room today are entrepreneurs and building new companies. And I'm just seeing some extraordinary stuff happening at extraordinary speeds.

Anyway, it's going to be a really, really interesting time.

The thing that I will say for the developers in the room is there are a handful of antipatterns here, with the capability overhang. Dario mentioned one of these. It was his advice to folks. I hear these all the time, by the way, from Microsoft developers, when they will look at a system. They say, eh, this is right on the margin. The quality of what's coming out of the model isn't quite good enough to solve this problem.

GitHub Copilot, by the way, was one of those things. When we first showed this to people inside of Microsoft, they looked at the system, which at the time, was at about a 30% accept rate, I think, for developers accepting the code that came from a coding prompt that was given to GPT-3.0.

And some people saw that and they were like, oh, this is horrible, 30% is nowhere near good enough. And some people saw that and were like, holy crap, this worked at all? Oh my god, what's the trend line look on this? And that's how things like GitHub Copilot happen.

And so, I think you all have to really set your product ambition to 11. You've got to really be targeting things that actually seem impossible to you. And you really do have to ignore the marginal cost. This is one of the things that's been so true over the past six years. We're getting about 10 times cheaper price performance wise, on inference, every year. I wish I was 25 years old again, because all I had was Moore's Law, and it's a piece of crap compared to what's happening right now.

You really, really should never be telling yourself that this is too expensive to do, because stuff is getting cheaper faster.

I'm over my time. We already showed you type agent.

(Break for direction.)

KEVIN SCOTT: Let me just close by saying that the last time that I was as excited about being a software developer or a technologist as I am now, was in the '90s. And one of the reasons why is I had this kid in a candy store set of building blocks that even a dummy like me could get my head fully wrapped around. I could understand how each of these individual pieces worked and I understood how they composed. And I could just go play and I could play because all of that stuff was permissionless. There was no gatekeeper sitting there telling me, you cannot do this,

because you have to sign a BD deal with us, or you have to get reviewed in this particular way or whatnot.

And so, that's the moment you all are in right now. All of these building blocks are available to you. They're relatively simple. I'm watching people do just nutty stuff.

My last story I will tell before I stop ruining Frank's show features my 16-year-old, who, by the way, is not a programmer or a developer. I went to this review that they were doing of their end-of-year social entrepreneurship projects at our school. And I knew she was doing something to try to help kids who were on hormone blockers with calcium supplementation. How do you get a teenager to do anything that they don't want to do?

And I show up to this thing, and she demonstrates this app. And I'm like, "How the hell did you write an app?" It's like, "You don't know how to program, and moreover you never came to Dad and said, hey, I'd like some help with this app, not a word."

And so, what had happened is they had spent about an hour and a half with one of the dads from their school who taught them how to use one of these AI app development systems. And these three girls, who are not programmers and who don't particularly like the idea of programming, built an app that any of us in this room who were active developers in 2008 -- when you could write your first mobile application -- would have been super proud of this thing, if we had built it ourselves back then.

And that's the moment we're in. They didn't have to ask anybody's permission for anything. There was nothing too complicated about the building blocks for them to understand. They just went and did something cool.

And that's our challenge to all of you at Build. We're going to give you a ton of tools to go work with and then we hope you go do something cool with it. Thank you very much.

Frank?

(Applause.)

FRANK SHAW: Okay. Jay, let's come on up. Are you going to tell us all about some crazy, cool things some random person built that's going to make me feel worse about my programming skills, because I've got serious work to do?

JAY PARIKH: We can do that later, Frank.

FRANK SHAW: Just first, welcome. This is your first Build.

JAY PARIKH: It is, yeah.

FRANK SHAW: For a lot of people in the room, this is the first time hearing from you since you joined Microsoft. What have you been up to here?

JAY PARIKH: Yeah, so I joined Microsoft at the end of October last year. It's been, what, seven months, almost?

FRANK SHAW: Yeah.

JAY PARIKH: And what have I learned? Well, I'd say it's been a just epic, fast pace, I think, in terms of trying to understand everything going on in this company, which I've probably not even scratched the surface, then also organizing the new team that I think we're going to talk about a little bit. And to keep up with the fast changing, crazy world that Kevin told us about, it's really easier to say what did I actually got done today, because there's so much changing around you all the time.

But I would say one of the things that is very, very striking to me as somebody who's never worked at Microsoft before -- but who knows a lot about Microsoft, grew up with Microsoft in all of our lives-- is the amount of innovation that you see from the outside versus what you feel and see going on inside the company is just profoundly different.

Getting inside the company, you just meet tons and tons of smart people doing all sorts of things, from big, distributed systems work to cutting edge security research and dealing with nation state attacks, to building all sorts of different types of products, serving all of these enterprises out there, the work that's going on Microsoft Research, the work that we're doing to serve developers, and the list just goes on and on and on.

It is awesome. It's energizing, one, to just be around so many smart, diverse people, but, two, just to see this pace that needs to pick up everywhere, but inside of Microsoft, to try to put these building blocks together and solve these problems that we've been talking about.

FRANK SHAW: Yeah, on Tuesday, you'll have a chance to show some of the stuff you've been working on, too. Before you came to Microsoft, you were the CEO of Lacework. And then before that, you were at Facebook, where you spent a decade scaling from something that was very small to something that was very, very large. And now you run what we call CoreAI here at Microsoft.

Tell us a little bit about what the group is and what the vision behind it is.

JAY PARIKH: Yeah, absolutely. I would say ~~that in order to maybe to~~ just tack on to what Kevin was saying here, is that this entire way of building these AI applications is just changing so fast. And the difference here is that maybe in the past, ~~past~~ couple of decades, we built software where all of us, we'd go out there, we'd talk to customers. We'd come up with some requirements. We'd come back. We'd design some database schema. We'd go write some code around it, maybe some other systems, and then we'd put some user experience around it.

And that's what we did for decades, and we built lots of great products, great companies, etc. But now, when you look at the paradigm shift here, and that drawing that Kevin put up there, now the models are at the center of this stack. And these models, they plan, they reason, they think.

They can call different tools. They can call each other and they're also advancing at such a rapid rate.

The goal for the CoreAI team really is to build the new stack, the stack that allows somebody to build these AI-driven applications, or these new agents in a way that really tried to take advantage of the capabilities that are in these models. You have lots of models, big ones, small ones, different special purpose ones, very, very general ones, and then you want to build these AI native applications. But the way we thought about building applications and any systems in the past without AI just profoundly have changed for us. And they're changing really rapidly, the ecosystem, the frameworks, etc.

CoreAI is really trying to reimagine the end-to-end developer experience using AI and then building on a platform that becomes your factory for these AI applications and agents. And then we're embedding and we're making sure that security and trust is baked in right from the start. We'll talk more about that in the next couple days, and then also give you an option around how you deploy this from the cloud all the way to local or to the edge.

FRANK SHAW: Yeah, one of the things that I hear you talk a lot about is the need for speed on some of this stuff. How do you ramp up the speed of development and test and experimentation? How do you motivate and think through change like that?

JAY PARIKH: Absolutely, inside Microsoft, this is all about, for us, around learning faster. Speed is all about learning faster. The world is changing so fast with all the technologies, with all the different applications, with the different companies out there. And so, it is really just about, how do you create that learning loop, or how do you have a learning loop that's just faster and faster and faster over time.

And so, part of that is cultural. I'd say part of this is maybe 25% to 33%, the cultural thing. And then I would say the rest is, how do you build the tools, the infrastructure that allow your product makers to go fast, to be able to iterate, be able to learn fast.

FRANK SHAW: We're here for Build. And so, Satya kicks off tomorrow, and then you're up there on the big stage on Day 2. I know you've been sweating over the demos and whatnot. How are you feeling about everything? And then what should people expect to see there on stage?

JAY PARIKH: Yeah, absolutely. I think what you're going to see in the next couple of days is just a lot of things that we want folks to try out and to build with. And that, I think, is the most exciting thing for me and for the team, is we've been working hard for many months here to really push to build stuff that's working, that's ready to try to compose, to put together. And then we want your feedback.

And it cuts across the entire portfolio, from things that help your employees with maybe M365 Copilot, or Copilot Studio, or other parts of that stack, to developer tools and GitHub, GitHub Copilot, different types of agents that we're going to talk about, that can really help you move through your work as a developer faster and faster and faster, but then also the underlying

platform that we call Foundry, that really is the place where you get to tap into these different models, these primitives that Kevin talked about.

And we work really closely with Kevin's team and other parts of the business here to really try to embed those primitives into Foundry so that our internal product teams can take advantage of those. But then our external customers can take advantage of those, so support for things like MCP, accessing your data, securing that, etc. those capabilities, we've actually got a lot to go try out.

FRANK SHAW: A lot of demos.

JAY PARIKH: Yeah, I've got a lot of demos to figure out.

FRANK SHAW: Yeah.

JAY PARIKH: It better work.

FRANK SHAW: Real demos, live on stage. What could go wrong?

JAY PARIKH: They are live.

FRANK SHAW: You're going to be great.

One of the concepts that I hear you talk a lot about is around turning the Azure AI Foundry into this agent factory. Steve talked about that. And when Bill and Paul started Microsoft, Bill had the idea of the software factory. You wrote about it on LinkedIn, which I thought was great.

Obviously, a lot has changed over the 50 years, but feels like the essence of that vision is still hanging out. Talk a little bit about why you've picked this up as a drive and a metaphor.

JAY PARIKH: Yeah, absolutely. I actually didn't know about the software factory tidbit from Bill back in 1975. When he and I were chatting, I think it was earlier this year, I was explaining to him this concept that we had for the team around building this set of tools, this platform to, one, internally help Microsoft become and have this agent factory, but then potentially having companies that use Microsoft and build on Microsoft to turn themselves into their own agent factories.

I was explaining this to Bill, and he stopped. And he got excited, and then he told me this story about how he had this vision when he started Microsoft, about it being the software factory, and the software factory wasn't just about building one application, that it was going to be a company where he and Paul and others were going to hire a bunch of really smart programmers.

And they were just going to build a lot of very useful software and do it all connected, packaged as one company. And they could learn and iterate and really solve these problems. Instead of having a company buy all these different point solutions, Microsoft could be that software factory for you.

In some ways, that vision just fell into this agent factory thing, and then it was an odd parallel from 50 years ago, but I think it just is more apropos for the AI age, where really, we went from building these simple applications around chatbots, and now we're seeing more and more organizations moving into these applications that are really driving business outcomes for them, where there's real ROI to them. And they're taking advantage of more and more agents.

Even just prepping some of the Build sessions today, talking to some of the customers, the startups that I work with, they're moving into these very sophisticated multiagent orchestration systems that are actually functioning as entire mini-departments, mini-functions, or running an entire application area.

That's what the agent factory has to support for all of us builders out there. This is what it's for.

FRANK SHAW: One of the things about Microsoft is it's always been a platform company. And a lot of times, sometimes the world tends to look at things in the technology sector, in particular, through the lens of competition. And so, people say, well, wait a minute, how can you be supporting OpenAI and Mistral and 54 and Sonnet side by side on those.

Spend a little time talking about the value of that ecosystem, and why it's a feature, not a bug.

JAY PARIKH: Yeah, absolutely. I think I would say maybe three things here. One is that it is just super early right now, and things are changing super-fast. As Kevin talked about, also, when you think about the web, if we had picked some very specialized stack by a bunch of vendors or one vendor, I think the web would look very different than it does today. It is a world, one, that's moving fast.

Two is, I think there are a lot of use cases that we yet can't imagine. We talk about these, maybe healthcare benefits. We talk about these other agentic applications, or these ways to assist folks and different types of teams out there, but I think there's just so many use cases out there that we can't even see and understand today, in that capability overhang that Kevin mentioned as well.

We've got to stay open in being able to iterate and being able to swap in different types of frameworks, whatnot. And by definition, to me, a platform is one that does provide choice, that it is open for developers, for builders, that it also morphs and evolves over time. As somebody who spent a fair amount of my career in and helping drive open source efforts, both in software and in hardware, you really see what a great community can do to drive that innovation and drive that adoption. And that's what it's going to take to get to this ubiquitous state of accepting, adopting, using and, -benefitting from AI.

FRANK SHAW: I heard you say one time, every engineer is a performance engineer.

JAY PARIKH: I still want to know where you heard that from. (Laughter.)

FRANK SHAW: What does that mean when you think about tokens as the new currency?

JAY PARIKH: Yeah, absolutely. I think for me, performance is just something that we do to be great engineers. You care about this from a user experience perspective. You want your latency, you want your response time to be super-fast in whatever you do. People are impatient. People don't have time. People want to do more with per unit time. We, as builders, owe that responsibility to our users that everything has to be fast.

Now with performance and just doing hardcore performance engineering, we do want the models and these applications that we're building to be interactive, to be fast, to be efficient, because the more capability we can give per unit time and the cheaper it is, that is going to drive more and more adoption, more and more use cases.

This goes hand in hand. We want people to consume more AI. We need that to be cheaper. To do that with less dollars, we're going to need to give them more functionality at the same time. But this is also just part of what we're riding, is this 10 ~~times~~ reduction in cost per year, which is a crazy fast change.

FRANK SHAW: Kevin said that tomorrow, he's going to give us a look at the next 12 months, but he hasn't done it yet. ~~(Laughter.)~~ You have the opportunity here to just get ahead of him a little bit. ~~(Laughter.)~~

Looking past Build, what are you excited about for the next year? And what are developers ~~(crosstalk/inaudible)~~ looking forward to?

JAY PARIKH: ~~I would be shocked if Kevin and I are at all not going to — w~~We would say the same thing. I'll leave it up to Kevin to share his 12-month roadmap.

What am I excited about? I'm excited about a few things here – well, lots of things. One is I believe that we have a really, really great set of updates coming out over the next couple of days of news. And I think there are lots of different places as a community here that you get to go and be curious and try and experiment with.

And we're still learning a lot. These things are not done. They're not finished, they're not perfect. We want your feedback. We're picking up the pace in lots of places in the company, to just iterate, to learn, to ship a lot more frequently as well. That's going to be really fun to just see what the community does with everything that we're talking about in the next couple of days.

And then I also think that when we start to tease some of these probably further out there innovations that we've been working ~~out~~ on, we're going to set, I think, some pretty far, bigger ambitions in terms of what the technology can do. ~~And that should... w~~We may not know what to do with it right away, because it's going to sound very, maybe academic at first, but I'm excited to see what problems people try to solve with it.

FRANK SHAW: Okay, I think that's probably all the time that we've got. I think for that, it's super interesting. And thanks to Kevin for his presentation as well. And now is when we say goodbye to everybody who has joined us remotely.

Thank you again, and I hope you have a chance to see us tomorrow during the keynotes and again on Tuesday.

| ~~(Music.)~~

END