

05192025 Microsoft Build Keynote Satya Nadella

**Microsoft Build 2025**  
**Satya Nadella, Kevin Scott**  
**Monday, May 19, 2025**

**SATYA NADELLA:** Good morning. Good morning and welcome to Build. It's always fun to be back at Build, especially in times like this. We're just about getting into these middle innings of another platform shift. And these middle innings are where all things happen, all things scale. In fact, it reminds me of '91 and Win32 or the web stack in '96, 2008 and cloud and mobile.

And here we are in 2025, building out this open agentic web at scale. And we're going from these few apps with vertically integrated stacks to more of a platform that enables this open, scalable, agentic web. More importantly, it's all about expanding that opportunity for developers across every layer of the stack so that you all can build the apps, the agents that can empower every person and every organization on the planet. That's what we will unpack at this conference. But let us start where it all starts: the tools we use to build.

Software engineering has always been about having the right tools to bring your ideas to life, continually perfect and craft and tame complexity. We're continually evolving these tools. We are seeing incredible momentum, adoption and diffusion. In fact, Visual Studio family now has over 50 million users. GitHub has 150 million users. GitHub Copilot, in fact, has been used by more than 15 million developers.

And we're just getting started. Now we have a bunch of new updates we are rolling out at Build, starting with Visual Studio. It is the most powerful IDE for .NET and C++. And we are making it even better, .NET 10 support, a live preview at design time, improvements to JIT tooling, a new debugger for cross-platform apps and much, much more.

And we are moving, by the way, to a monthly cadence for stable releases as well. And when it comes to VS code, just a couple of weeks ago, we shipped our 100th release in the open. It included improved multi-window support and made it easier to view staged directly from within the editor, and GitHub continues to be the home for developers. GitHub Enterprise has tremendous momentum in the enterprise, and we are doubling down for developers building any applications. Trust, security, compliance, auditability, data residency are even more critical today.

Now, starting with tools you use to the infrastructure you deploy on to reach the users and the markets you want. Talking about trust, open source is at the core of GitHub, and we are taking this next big step. As GitHub Copilot has evolved inside VS Code, AI has become so central to how we code. And that's why we're open-sourcing Copilot in VS Code.

We're really excited about this. This is a big deal. Starting today, we will integrate these AI powered capabilities directly into the core of VS Code, bringing them into the same open-source repo that powers the world's most loved dev tool. And of course, we will continue to build out GitHub Copilot too.

In fact, over the past few years, we've gone from code completions to chat to multi-file edits and now agents. And this same pattern is emerging more broadly across the agentic web. You can ask questions, and AI assistants give us answers. You can assign tasks to agents and have them execute them, or work side by side with AI to complete jobs and projects. And you can mix and match all of these form factors, right? That's kind of what we care about as developers. It's not about any one of them.

In fact, we're building app modernization right into agent mode, right? So, Copilot now is capable of upgrading frameworks like a Java 8 to 20, Java 21 or .NET 6 to .NET 9 and migrate any on-premise app to the cloud. It creates a plan for your code and dependency suggests the fixes along the way; learns from changes you make and makes the entire process seamless.

And the next thing we're introducing is an autonomous agent for site reliability engineering, or SRE. I mean, think about one of the biggest pain points for any of us, right? Getting woken up in the middle of the night to deal with a life side issue. Take a PagerDuty memory leak issue. The SRE agent starts automatically triaging, route causing, mitigating the issue, and then it logs the incident management report as a GitHub issue with all the repair items. And from there you can even assign the repair items to GitHub Copilot. We're not stopping there. This is the next big step forward, which is a full coding agent built right into GitHub, taking Copilot from being a pair programmer to a peer programmer.

You can assign issues to Copilot, bug fixes, new features, code maintenance and it will complete these tasks autonomously. Today I'm super excited that it's now available to all of you. Gone are the days when I could just simply report bugs. At this point, I'm assigned bugs to fix. That's kind of called empowerment. Here I am with all the bugs that I have or issues that I have to deal with in GitHub issues.

The first one is adding a filter for user group size community page. Let's go take a look at this issue. It's nice. They say like I've got to go put some new filter up here. It also shows me where. I guess it needs to do a range small, medium, large by size or some kind of a percentile. It's like some kind of a group by case when type of thing.

Let's do the thing that is easiest for me. In fact, it even has some caching stuff, which I have no idea what that is, but I guess there's a staging cache and then a read is OK, fine, let's do the thing that I can do, which is assign it to my new buddy Copilot. So I'm going to assign it and there you go. Let's go and see. Let me scroll down. It's picked it up. It sees me, it creates a PR, and you see that nice emoji? It sort of knows that I'm here and it's sort of going on to work. And we'll come back and check it out later.

It's just so fun, to be able to go take care of issues like that, like email triage assigned to Copilot. What it's doing is setting up a branch. It starts GitHub actions in the sense it just generates the compute for you, or it creates a virtual machine using GitHub actions. It commits a draft PR to session logs. And in fact, you can go back to the session logs and continue to see all the draft PRs as it's working. Our coding agent respects all the security measures while delivering a great developer experience. That's obviously super important.

The agent works in its own branch. It only uses MCP servers configured by a developer. We can get other agents to do code reviews and keep people in the loop before they run any CI, CD or merge. And we're also making all these intrinsics available to partners to ensure that there is an open and secure ecosystem of agents, whether it's SRE, SWE, code review and many, many more agents that all of you will build.

And of course, we also want to ensure that individual developers as well as IT have all the controls here. And talking about ecosystem of agents, we're very excited about OpenAI's Codex agent that was just launched on Friday. And I'm thrilled to have Sam Altman join us virtually. Welcome, Sam to Build.

**SAM ALTMAN:** Thank you. Thank you for having me.

**SATYA NADELLA:** One of the things I know you've thought a lot about, all these various form factors that developers use for software engineering. Of course you did the CLI. And now, yes, last week you did the coding agent. You want to talk a little bit, Sam, sort of the vision you have for how software engineering evolves and actually how developers will use all these various form factors together.

**SAM ALTMAN:** Yeah. So, Satya, you and I have been talking about this for a long time. In fact, the very first version of Codex, I think it was all the way back to 2021, one of the very first things we did together in GitHub. And we've been talking about how someday we'd get to like a real agentic coding experience. And it's kind of wild to me that it's finally here.

I think this is one of the biggest changes to programming that I've ever seen, but this idea that you now have a real virtual teammate that you can assign work to, that you can say, hey, go off and do some of the stuff you were just doing and increasingly more advanced things, you know, at some point, say, like, I got a big idea, go off and work for a couple of days and do it. And then you can issue many requests in parallel that you can be fixing bugs, implement new features, answering questions about the code.

This is like true software engineering task delegation. And I think it'll only get better from here, but this is just a tremendously exciting moment. It integrates very deeply with GitHub. You can give it access to a repo in an environment, and you can get some pretty amazing stuff done.

**SATYA NADELLA:** Yeah, I know it's really exciting to see that. And it's also sort of great to have developers stay in the flow, work with essentially peer programmers, agents as well as other people that we collaborate with and just have the developer process itself in the life cycle get faster. Obviously, you also are working on a lot of models and are very sort of fantastic. In fact, we've had a chance to ship a lot of the models you guys have built. Just tell us a little bit about what's sort of coming as far as the model roadmap itself.

**SAM ALTMAN:** The models are already very smart. They will continue to get smarter too. But I think one of the most exciting things is the models will get simpler to use. You won't have so many to pick from, it'll just sort of automatically do the right thing. They'll get much more

reliable. You'll be able to trust them for much more. They'll be a lot more features like multimodality and great tool use and integration. It'll be closer to the 'it just works.' I can talk to it. I can do a complicated coding agentic thing. I can rely on it. And I think people are going to be surprised at how fast we're going to make progress in those directions now.

**SATYA NADELLA:** Yeah, I know we're very excited about your model roadmap. And obviously, when you look at ChatGPT, it's the most at scale stateful agentic app today that you guys built. And of course, Codex is another sort of agent app that you build.

This conference is all about unpacking so that every developer can, in some sense, build these new agentic apps that use the model, do their own model scaffolding, go on to even do multi-agent orchestration and so on. Any advice you have as people build out these high scale production, stateful agentic apps? Sam, based on obviously what you guys have been doing and leading.

**SAM ALTMAN:** I think one of the hardest, most difficult things to manage about this is just the rate of change. If you think about what was possible two years ago or one year ago or now and what will be possible another year or two years planning for this incredible increase in model power and how people are going to build products and software and companies in the near future and really leaning into the new tools and the new workflows that are possible. Again, we haven't seen many technological shifts like this in history, but every time one has come, like leaning in early and hard has been very rewarded.

**SATYA NADELLA:** That's absolutely well said. Because at some level, one of the things we want to really unpack at this conference is what's that app server that allows you to take the latest new sample that comes and keeps moving at that pace, because I think that's the challenge we have as developers building these applications. But it's fantastic. Again, go ahead.

**SAM ALTMAN:** Yeah, I was just going to say it was amazing to watch over the last few months as we were working on codecs internally. There are always a few people that are the early adopters and how quickly the people who were just using codecs all day change their workflow, and just the incredible amount they were able to do relative to someone else was, was quite interesting.

**SATYA NADELLA:** No. It's fantastic. Thank you so much, Sam. Thanks for the partnership. See you again at Build again. All right.

**SAM ALTMAN:** Thank you for having me.

**SATYA NADELLA:** It's an incredible time to be a developer. And all these tools are getting richer. And more importantly, it's not about any one tool, any one agent or any one form factor. It's about the coming together of all of it for us as individuals and as teams of software engineers to be able to express our craft.

Now, let's go up the stack. Let's talk about the platform opportunity of Microsoft 365. I'm very excited about the latest update to Microsoft 365 Copilot, which is now generally available. It's really exciting to see this. This is the biggest update.

In fact, if you think about like, I don't think since Teams launched, we've had an update of this level and it really brings together chat, search, notebooks, create and agents all into this one scaffolding that's intuitive. I always say this is the UI for AI. And chat, for example, is grounded both on web data as well as your work data. And that's the game changer, especially with pages.

Search works across all of your applications, right? Whether it's Confluence or Google Drive or Jira or ServiceNow, not just M365 data. With notebooks, I can now create these heterogeneous collections of data, right? In fact, I can have chats and pages and any documents, emails, all in that collection. And then, in fact, I can get all these audio reviews or podcasts out of it. I can use create to turn a PowerPoint into a new explainer video or generate an image. And when it comes to agents, we have a couple of special agents like Researcher.

It's been perhaps the biggest game changer for me because it's synthesizing across both the web and enterprise sources. Right. Applying a deep chain of thought reasoning to any topic or any project. Analyst goes from raw data across multiple source files. I can just upload a bunch of Excel files. It will get the insights, it'll do forecasts, it will do all the visualizations. And these agents are all at the end about putting expertise at your fingertips.

A long time ago I talked about information at your fingertips. I feel we are at that age where we are now going to put expertise at your fingertips. And Teams takes all that and makes it multiplayer, right? And all of the agents you build can now show up in Teams and in Copilot, and you can ask questions, assign action items, or kick off a workflow by just mentioning an agent in a chat or meeting.

And with the Teams AI library, building multiplayer agents is easier than ever. It now supports MCP, and with just one line of code, you can even have it enabled A to A, and you can add things like episodic or semantic memory by using Azure Search and New Retrieval System, which I'll talk about later. And as a developer you can now publish. And this is the biggest thing right now.

You can build an agent, you can publish your agent to the agent store and have them discovered and distributed across both Copilot and Teams, providing you with access to the hundreds of millions of users and unlocking that opportunity. Over the past year, we've seen so many partners across every industry build agents that connect to Copilot and Teams.

And let me just share a few examples. With Workday's agent, you can ask Copilot what needs your attention. It gives you a summary of all the workday tasks, so you stay on top of the learnings, the approvals, the workflows. ServiceNow agent, you can ask it in real time about an incident, all the resolution metrics, and then you can use it to instantly create a PowerPoint presentation with all the results contained. With LSEG agent, financial professionals can discover, analyze, share any financial data that the exchange puts out right in Excel and PowerPoint. And with Copilot Studio, you can start to build your own agents.

We've shipped a lot, a ton of new features into Copilot Studio. Most recently, we put in a full core agent right into Copilot Studio MCP agent flows, and this is very key right now. You can mix and match both LLMs and deterministic workflows. Today we are making it easier to build even more complex multi-agent workflows in Copilot Studio using orchestration.

Take something like onboarding a new hire. It's actually a pretty complex process, actually. It involves agents from facilities, finance, legal, and each with their own expertise and workflows. And so, you can bring them all together and everything moves faster and the experience is better for everyone. That multi-agent orchestration, right. In Copilot Studio all up you have built 1 million plus agents that connect with Copilot and Teams in the past year. And we're not slowing down.

Today we're introducing a new class of enterprise grade agents you can build using fine-tuned models on your company's data, workflows and style. We call it Copilot Tuning. This is a big deal. I mean, at some level this is about really not just using Copilot, but it's about tuning Copilot for every customer, every enterprise, every firm. Copilot can now learn your company's unique tone and language, and soon it'll even go further to understanding all of the company specific expertise and knowledge. All you need to do is seed the training environment with a small set of references and kick off a training run.

The customized model inherits the permissions of all source control, and once integrated into the agent, it can deploy to authorized users. You can sort of go to groups that you have set up and distribute them across Copilot. For example, if you're a legal firm, it'll reason through past arguments and relevant literature and deliver answers or generate docs that are very specific to your firm. Or if you're a consulting company that works across, let's say, multiple vertical industries, you can now start to tune these models for each vertical industry to reflect sort of the specific know how you know about the workflows in that industry.

It's all about taking that expertise that you have as a firm and further amplifying it so that everyone in the company gets it, and also your products and services reflect it. And to sort of give you a feel for all of what I've talked about, let me invite up on stage Miti, to show you all of Copilot or ecosystem. Miti, over to you.

**MITI JOSHI:** We as developers can easily scale productivity solutions using the new Microsoft 365 Copilot app, Copilot Studio and Copilot tuning. Let's start with the new M365 Copilot app, where I'll show you how Copilot can reason over my GitHub backlog. This app is my 5 in 1 hub for work. It's got chat, search, agents, notebooks, and here I can call on agents like the Researcher agent. Researchers can reason over all my work data. And with the connector, it can even tap into GitHub.

So, let's ask for help analyzing performance issues and prioritizing my next steps. It goes ahead and it asks me a few clarifying questions, just like if I delegated this to someone on my team. I'll give it a bit more direction and off it goes. Because it is built on the OpenAI o3 reasoning model, Researcher shows its chain of thought as it pulls from both the web and my work data. I see here that it's giving me everything that I need, including some citations to help me get started.

Now, let me show you how I built this connector in Visual Studio. It takes just a few steps. I'll go ahead and add a name and a place to save it. And in seconds, the M365 Agents toolkit gave me fully scaffolded code that I can modify by creating my connection using the Microsoft Graph APIs.

And then I can index any type of data. So here I've defined a specific schema for my scenario. Next up, I'm using the GitHub API to fetch my issues. And finally, I'll ingest every issue into Microsoft Graph, so Copilot can now reason over this data. So, let's see it all in action. I see that all of my issues are being indexed, and I'm getting a full log of what's happening. As the log loads, I'll scroll up and confirm that my GitHub connection was made and that it has all the backlog items that I care about.

So, let's pivot to showing how the Copilot Studio can help us build agents with little code required. So here I've created an RFP response agent, which you can see right here in Teams. I see that it generated a proposal, and it even posted using its own Entra ID. So here you can see that we have a specific proposal that has the content, language and format that you'd expect, for example, from an experienced employee.

So, let's go ahead and see how I built it. And for that I'll jump over to Copilot Studio. Here I've described what I want the agent to do, and I've given it some instructions. The GPT-4 response model is selected by default, but I can choose other models via AI foundry.

Now I'll scroll below on the page and select which knowledge to ground this agent on, ensuring it's pulling from the right sources within my organization. We'll keep scrolling on to the tools section, and I've added this trigger that prompts my agent to begin working whenever a new RFP arrives in my inbox. Now let's link this agent to the dynamics MCP server to give it access to SKU and pricing data. And this will make for a more detailed proposal.

With just a few clicks it's now connected. This server will keep the agent up to date automatically. It can also use third party servers like DocuSign or custom MCP servers like this SAP one we created to access customer account data. The agent will need to be able to check compliance and contract terms. Now, instead of building this out from scratch here, I can just use the new multi-agent orchestration. As Satya said, this means that agents can now collaborate with each other to take on more complex work.

Let's go ahead and connect this agent to one that specializes in compliance checks. With orchestration, the RFB agent can connect to the compliance review agent to ensure there are no red flags. If it passes, it will now return to this agent to continue the process. And with that, our RFP response agent is up and running.

Now let's move to Copilot Tuning, which Satya just announced. This is a low-code way to do what would otherwise take a whole team of data scientists weeks to do fine tuning a new model specifically designed for contract writing. And here's what it looks like in action. I'll open the contract builder agent and ask it to draft a contract using a couple of example documents. It's going to reference these documents in the custom model I created to assemble the contract.

When the document is complete, I'll receive an email with the draft. Let's take a look at it. I see that this contract uses our company's language, terms and conditions, structure and format. And here's how I built it. I'll start by creating a new model. I'll give it some basic information, a name, a description, and I'll select the task type, which in this case is document generation. Now I'll add the knowledge source. Here we'll use our contract database from SharePoint and I'll specify who should have access, so it's the contracts team and the procurement team. And then we'll ensure the Copilot ensures that all of this access to knowledge sources in the tuned model is aligned.

With that, a contract builder is now ready for data labeling. Now, once the subject matter experts have completed the data labeling process, we can complete the training for this fine-tuned model. We're done. Let's go ahead and publish it. From the M365 Copilot app, the team can select Create Agent, select a task specific agent, and select the contract builder model I just fine tuned. With that the agent is ready. I've now scaled the work that I can deliver as a developer, empowering those closest to the business to reimagine their workflows with AI. Back to you, Satya.

**SATYA NADELLA:** Thank you so much, Miti. I'm so excited about Copilot Studio, and with access to the reasoning models, the ability to do things like Copilot tuning these deterministic workflows. At some level, you can now think about these agents and multi-agent frameworks orchestrating the workflows in an agentic way for every role, every business process, especially once you start having even every business application just show up as an MCP server. You can imagine as a developer how you can start thinking about the next level of automation. It's just a complete game changer in terms of how we think about workflow and business process automation. Those SaaS applications and productivity software all coming together by role and process. It's just an exciting day.

To us, what we're doing is we're taking everything underneath Copilot, Copilot Studio, and making it available as a first-class platform for all of you as developers to build your own applications, your own application extensions. As models evolve faster and become more capable with new samples being dropped every couple of months, the apps will have to evolve to become these full stateful applications that are multi-model and multi-agent. That's the big departure now.

It's not about one model with just a request response API call, we're building real stateful multi-model applications, and they have to be production ready. That's what is the motivation for building a first-class app server. Think of Foundry like a production line for intelligence. It takes more than a great model to build these agents and applications. The system around the model, whether they are evals, this orchestration layer, or RAG, all really, really matter. Foundry is that complete app platform for the AI age.

Over 70,000 organizations are already using it across industries: BMW, Carvana, Coca-Cola, Nasdaq along with many ISVs like Gainsight and others. They're all choosing Foundry and enterprises are moving from just doing POCs to these enterprise-wide deployments to truly unlock the ROI of AI. In fact, over the past three months, we've processed over 100 trillion



tokens, which is 5x year over year. In fact, when I was in Japan just recently, I saw this app built with Foundry that's helping people with auditory processing disorder make sense of what they hear. It blew me away. Let's take a look at the video.

(Video segment.) (In Japanese.)

**SATYA NADELLA:** Really, a big thank you to all the developers like Ishin and Kato-San for sharing that story. It's really inspiring. Today we're going further. It starts for Foundry with model choice. We already support 1,900 models, whether they're response models, reasoning models, task specific, multi-modal, you name it, they're all there in Foundry. Of course, that includes the latest from our partners at OpenAI. Just this year we have sim-shipped 15 models from OpenAI on Azure, giving same-day access when a new model drops and Sora is coming next week.

As devs, we care about multiple dimensions: Cost, reliability, latency, as well as quality. Azure OpenAI is best in class. We offer enterprise guarantees like high reliability and great cost controls, things like batch or spillover, and of course, leading security and compliance and safety. But still, picking a model can be a bit of a chore.

You need to be able to route your queries to the right one fast, so we are making that easier too. Our new model router will automatically choose the best OpenAI model for the job, no more of those manual model selections. An approach today, though, goes from having apps that you built or agents you build only bind to one model to truly becoming multi-model. That's why today we are thrilled to announce that Grok from xAI is coming to Azure.

(Applause.)

Grok 3 offers reasoning, deep search, response models, all in one single model, and it is awesome to have a chance to chat with Elon over the weekend about it.

Let's roll the video.

(Begin video segment.)

**SATYA NADELLA:** Thank you so much, Elon, for being here at Build. I know you started off as an intern at Microsoft. You were a Windows developer, and of course, you're a big PC gamer still. Do you want to just talk about you and your early days with Windows and the kinds of things you built?

**ELON MUSK:** Well, actually, it started before Windows with DOS. I had one of the early IBM PCs with MS-DOS and I think I had like 128K in the beginning, and then it doubled to 256K, which felt like a lot. I programmed video games in DOS and then later in Windows. Remember Windows 3.1?

**SATYA NADELLA:** Yeah. It's wonderful. The last time I chatted with you you were talking all about everything, the intricacies of Active Directory, so it's fantastic to have you at our

developer conference. Obviously, the exciting thing for us is to be able to launch Grok on Azure. I know you have a deep vision for what AI needs to be, and that's what got you to get this built. It's a family of models that are both response and reasoning models, and you have a very exciting roadmap. Do you want to just tell us a little bit about your vision, the capability? You're pushing on both capability and efficiency, so maybe you can just talk a little bit about that.

**ELON MUSK:** Sure. With Grok, especially with Grok 3.5 that is about to be released, it's trying to reason from first principles to apply kind of the tools of physics to thinking. If you're trying to get to fundamental truths, you try to boil things down to the axiomatic elements that are most likely to be correct, and then you reason up from there. Then you can test your conclusions against those axiomatic elements.

In physics, if you violate conservation of energy or momentum, then you're either going to get a Nobel Prize or you're wrong, and you're all certainly wrong basically. That's really the focus of Grok 3.5 is fundamentals of physics and applying physics tools across all lines of reasoning and to aspire to truth with minimal error. There's always going to be some mistakes that are made, but we aim to get to truth with acknowledged error but minimize that error over time. I think that's actually extremely important for AI safety. I've thought a lot for a long time about AI safety, and my conclusion is the old maxim that honesty is the best policy.

**SATYA NADELLA:** Yeah.

**ELON MUSK:** It really is for safety. But I do want to emphasize we have and will make mistakes, but we aspire to correct them very quickly. We are very much looking forward to feedback from the developer community to say what do you need. Where are we wrong? How can we make it better? To have Grok be something that the developer community is very excited to use and where they can feel that their feedback is being heard and Grok is improving and serving their needs.

**SATYA NADELLA:** I'm really thrilled to get this journey started, getting that developer feedback, and then looking forward to even how they deploy it. Thank you so much, Elon, for briefly joining us today. We're really excited about working with you and getting this into the developer's hands.

**ELON MUSK:** Thank you very much. I can't emphasize enough that we're looking for feedback from you, the developer audience. Tell us what you want, and we'll make it happen. Thank you.

(End video segment.)

**SATYA NADELLA:** I had a chance to talk to him again, and then he was talking about all the things that they're doing with 3.5. We want to get the journey started, get the feedback on the API front, and help them with their roadmap. We're excited about all these new announcements. One of the things though, when you have multiple models, what you need is a new capability in how you use these models and how you can provision throughput once on Foundry and you can

use that provisioned throughput across multiple models including Grok. That's just a game changer in terms of how you think about models and model provisioning.

Now once you do that, you now have Minstral, which you can even provision with all the sovereign deployment in EU region, which increasingly becomes a massive consideration for people building applications around the world, because I think increasingly there will be models people will prefer in different parts. We're excited about Mistral LLaVa. I was at LlamaCon recently with Mark, and as he likes to say, he's bringing the full Llama herd to Azure. We're excited about all the advances that they're making in open source. Black Forest Labs and many more.

All of them all behind this one provision throughput. It's exciting for us as developers to be able to mix and match and use them all. Many of them have the same API signature, so even the ability to be able to make these model switches becomes even easier. We also have expanded our partnership with Hugging Face, so you'll have access to over 11,000 frontier and open-source models in Foundry.

But models are just part of the equation. For anyone building an agent, you need to be able to have truly great access to real-time web as well as the entire enterprise knowledge graph. We've learned that building RAG apps that you need a more sophisticated retrieval system. It's not good enough to just have a vector search and some embeddings, you really need a database or a knowledge engine or real query engine that's custom built for agents so that you can break down any complex query, run them in parallel, return synthesized results. Think of it essentially like a modern knowledge retrieval stack for agents that understand what users want and what your data means.

Of course, once you have that part of the system, the next layer is the orchestration layer. You want to be able to have all these agents or multiple agents orchestrated. The Foundry agent service lets you build declarative agents, in fact, just with few lines of code just in the portal. For complex workflows, it supports multi-agent orchestration. It integrates seamlessly with frameworks like Semantic Kernel and AutoGen, and you can use the agent service essentially like a managed service for the execution environment. More than 10,000 organizations are already using it, and I'm excited to share that now the agent service is generally available.

(Applause.).

Once you have these agents running, the next thing you need is the compute environment. We are making everything from serverless Azure Functions to Azure container apps to the extreme control that you need with AKS and how you scale. We're providing full spectrum of compute so that you can hit that right price/performance for any of your agentic scenarios. We're making it straightforward, for example, for you to connect Foundry to your container app or functions and deploy any open-source model into AKS, whether it's in the cloud or in hybrid mode with Arc. Increasingly, you want to even have these models that get deployed at the edge, and Foundry will support that.

We are closing the gap and the loop between Foundry and Copilot Studio. This is pretty, pretty key for us as developers. You can now take a model, fine tune it or post train it in Foundry, and then drop it right into Copilot Studio so that you can now use that post-trained model to automate a workflow or build an agent. That's the type of stuff that you can do.

One fantastic example of this is what Stanford Medicine orchestrated using multiple agents to connect. From patient history to radiology to getting PubMed data, clinical trial data and many, many, many other scenarios. But one goal, which is how can they take something mission-critical like cancer care and a tumor board meeting and orchestrate it all with AI? It's pretty incredible. Let's take a look at the video.

(Begin video segment.)

**STANFORD MEDICINE:** Stanford Medicine is at the forefront of cancer research. In the context of treatment, tumor boards are a really important meeting of many different clinicians who convene because the patient presents in a way that they're not familiar with. You have to pull together information about medications, any procedures, radiology labs, a patient's history, as well as the medical literature. There's not just a lot of information that you need to bring together for the tumor board, it's that that information is fragmented in a bunch of different places.

Stanford runs about 4,000 tumor board meetings per year. Pulling through real world evidence, pulling literature, pulling clinical trials. Those are things we do manually, and we can't do them 4,000 times per year. The healthcare agent orchestrator is a way of bringing all this together at the beginning, so that we can help make patient decisions more efficiently, faster and perhaps more accurately.

This is an agentic AI solution deployable through Azure AI Foundry. We've been able to build, customize and deploy our own agents, with this first use case being a tumor board. There is an orchestration layer that allows all of the different agents to dispatch and to provide a comprehensive report to a clinician that brings together all of these disparate information sources. This really powerful thing that Microsoft has incorporated into its agents is grounding within specific clinical notes.

They're already using Word to summarize things. They often make PowerPoint slides. This enables us to put everything in an integrated setting into one summary. It took just a few lines of code to deploy these agents into Teams, so that we could start interacting with them directly. The more integration between all these systems, the less effort they're spending on those mechanical things, the more effort we can spend at actual decision-making that we're experts at.

It's being delivered as a platform on which we can build. We can package things to share with others so that community hospitals get access to the same kind of capability that an academic medical center does. We work faster and we come to better decisions with better information. That itself, that's a revolution. We want to develop tools that will not only help our physicians at Stanford but help physicians all over the world. I think it's going to be transformative.

(End video segment.)

**SATYA NADELLA:** I'm really thrilled to share that this healthcare agent orchestrator that Stanford used is now available to everyone in Foundry. It's pretty awesome. Quite frankly when I see that video is when I know things are becoming real. Think about it, you use multiple models in Foundry, you do this multi-agent orchestrator, you bring it together in Copilot and in Teams, and it's a real-world example of some workflow orchestration done in an authentic way. That's when you know you're going from one little app to something that can scale throughout the enterprise.

Speaking about scaling to the enterprise, the other important consideration for an app server is observability. That's why we now have new observability features coming to Foundry to help you monitor and manage AI in production. You can track the impact, quality, safety, as well as cost all in one place. It goes beyond that. In the future, we believe every organization is going to have people and agents working together. That means the systems that you are today using ubiquitously for things like identity, management of endpoints, security will need to extend to agents as well.

That's a big deal. You want the same rails that you use today at scale to work across people and agents. That's what we're doing. With Entra ID, agents now get their own identity permissions, policies, access controls. The agents you build in Foundry and Copilot Studio show up automatically in an agent directory in Entra. We're also partnering with ServiceNow and Workday to bring automated provisioning and management to their agents via Entra. When it comes to data governance, Purview now integrates with Foundry. When you write an agent, automatically because of Purview, you can ensure end-to-end data protection. Another massive safety consideration.

On the security side, Defender now integrates with Foundry. That means your agents are also protected, just like an endpoint would be, from threats like wallet abuse or credential theft in Defender. This is just a quick look at Foundry, your agent factory. Now, to make all this real, I would love to invite up on stage Kadesha, one of my colleagues, to show you how it all comes together. Kadesha, take it away.

**KEDESHA KERR:** Thanks, Satya. Today I'm going to show you how you can use Azure AI Foundry and GitHub Copilot to make your dev life a whole lot easier. This is Vibe Travel, a chat-based app that uses Azure AI Foundry to provide an AI travel agent to help people plan their trips. I'm planning a trip to New Zealand in about eight months, so I asked Vibe for places to ski. This mask isn't mapping. I'm pretty sure eight months from now is January 2026. Think about it, that's going to be the summer season in the southern hemisphere.

We definitely need to make our AI agents smarter and give more grounded responses. I can pop into Foundry and see all of the AI agents that we have for this and our other apps. Our Vibe travel agent uses GPT-4o and has a pretty great prompt, but we need to add some knowledge. We can do this by giving it files with reference data or connections to other services like Microsoft Fabric or TripAdvisor. But I think we can get a long way by just adding grounding

with Bing search, so let's choose that option. Really simple. In addition to giving it access to knowledge, we also give it access to our flight reservation API. Because it's an open API, it just works.

Now, let's go back to our app and try our query again. Being able to add data sources or search capabilities makes the agent smarter and far less likely to hallucinate. I could make it even smarter by adding information about hotel discounts or weather forecasts. The possibilities are endless. "In about eight months, it's January 2026, which is the summer season. No skiing possible." That is much better. You saw how easy it was for me to improve.

Now, let's have the agent actually book a flight on my behalf. Adding actions that the agent can perform upgrades the app from a simple question answering machine to an autonomous agent that can actually do things on your behalf. I could use an action to put that booking in my calendar, or request to book a hotel, or even to request leave from an internal system. Now, these flights look great. You just saw how Foundry made it easy to improve our agents, but there's still plenty of work to do in the actual code.

I'm going to use agents in GitHub Copilot to get this done. Let's open Copilot agent mode, which is new, by the way, and I can choose between the most popular and most powerful models from all the major LLM providers. Agent mode takes GitHub Copilot beyond just answering questions or suggesting code to actually helping me do the work. I can come in here and I can ask, "can you show me the issues that are assigned to me in this repository?"

Traditionally, I would have opened the browser to look at the issues on github.com, but with MCP or Model Context Protocol, this allows you to give tools like GitHub Copilot the context and capabilities it needs to help you. GitHub Copilot will notice that we're asking about GitHub issues and uses GitHub MCP server to get that information. It looks like I have a few issues assigned to me, so let's have a look at the first one.

Now, when I click this reset button in the app, it immediately clears our chat session without warning. That's not ideal. We should really warn people first. Let's go back to the app and I can ask GitHub Copilot to help me implement the details for issue No. 1 and also fetch the details for issue No. 1. I also have this beautiful design that my manager gave to me last minute that I need to add as context to the prompt as well. I hope you can see it. Let me pull up my screenshot for you so you can see that. It's pretty sweet, wouldn't you say?

(Applause.)

Let me add this to the context window and send the prompts. We'll see Copilot asking for a specific permission to use the GitHub MCP server to fetch the details of the issue. Let's hit continue to allow it to implement the changes in our code base. You can see the GitHub MCP server fetch the details and with agent mode's new vision capabilities, it means that Copilot can even understand the sketches of what I want. Then it starts working out the issues of what I want to change and figure out how to change it.

While it's doing that, let's take a look at the other issue. Actually, I cheated a little for this one. Before I came out, I assigned the issue to Copilot, just like you saw earlier with the Azure Dev Community website. Copilot has looked at this issue and opened up a pull request for us. Let's take a look at it. You can see it here. It has a description with implementation details and the code changes that it recommends. We can also see the full session log of how GitHub Copilot implemented this change by clicking this view session button and scrolling through the changes.

Let's take a look at the actual code, though. This looks good, but if I want to add some feedback, I can do that by popping in a comment right in the issue or in the pull request, and GitHub Copilot is going to take a look at it and get started on the work. You can see the little emoji right here. It actually feels really good working on two issues at the same time. Now, let's go back to our editors to see the progress on issue No. 1.

As you can see, Copilot Agent mode was able to examine the proposed changes, look for relevant files to change, make the changes as appropriate, and it even stuck to the styles and coding standards that I wanted it to. When I go back to the app, it was able to implement the UI changes that my boss wanted. Let me see if it's running. Isn't it cute? As you can see, folks, this is live. Don't have time to debug, but once I click this issue, I'm pretty sure it's implemented.

(Laughter.)

Copilot was able to add the modal in our app, and that's two issues that we worked on in a handful of minutes. Thank you. GitHub Copilot and Azure AI Foundry.

(Cheers, applause.)

Actually, while I'm here, it's not often that I get the chance to check on Satya's work. Let's take a look at the issue that he assigned earlier — do you remember it? To add that group size filter to the site? Let's take a look at it. Oh, we're already here. OK. This is the issue as you see, and Copilot was able to open up a pull request to start implementing these changes. Let's take a look at the PR.

As you see here, Copilot was able to add the group size filter property. It was able to add the logic for Reda's caching. It also added some testing with instructions. That's pretty sweet. Now, it also deployed the app to our staging environment, so let's take a look at the deployment. Fingers crossed. Let's scroll all the way down and you can see the group size filter was added.

I do see a couple of things that I want to tweak, but I work on that backstage, and then push the production later, so you can take a look.

That's three issues in a handful of minutes. Back to you, Satya.

**SATYA NADELLA:** Thank you so much.

(Applause.)

Talk about working hard in parallel.

It's really exciting to see the progress that we're making with really building out. Each one of these platform shifts is needed an app server, and building out that fullness of the app server is when you know that the platform shift is now taking scale.

And so far, we have talked about everything we're doing in the cloud. Now, we want to bring the power of this app server and app building capability to the edge and clients as well, with Foundry Local, which we are announcing today. It includes a very – yeah.

(Applause.)

It includes a fast, high performance runtime, models, Agents as a Service, and a CLI for local app development. And yes, it's fully supported on Windows and the Mac.

(Applause.)

And so, now, let's just step back and talk about Windows for a moment.

If you're a developer, Windows is the most open platform with massive scale, with over a billion users and billion devices that you can reach. And we are constantly improving it, making it more secure, more reliable, better for your app, no matter whether you want – where you want to run, in fact, the Windows image, whether it's on the cloud or on the client machine.

In the past year, we have seen developers from Adobe to Zoom use these on-device AI capabilities on Windows to ship some amazing applications. If you go to the store and the AI hub, you can start seeing many of these things light up. And today, we are taking another step to make Windows the best platform for AI. We're excited to announce the Windows AI Foundry.

(Applause.)

Windows AI Foundry is what we used, in fact, ourselves internally, to build features on Copilot+ PCs for things like Recall or even Click-to-Do. All of these now are built using the same runtime and the SDK. And now we're extending this platform to support the full dev life cycle, not just on Copilot PCs, but across CPUs, GPUs, NPUs, all, and in the cloud. You can build your application and have them run across all of that silicon.

And Foundry Local is built into Windows AI Foundry, so you can tap into this rich catalogue of these preoptimized, open source models that you can run locally on your device. And with Windows AI Foundry, you can customize – this is very, very cool – our built-in Phi Silica SLM using LoRA adapters to basically meet any specific need of your application. Think about it. It comes with the image, comes with Phi Silica. You just do a LoRA adapter in your application, and it just works.

And if o1 and DeepSeek mark the beginning of basically, inference or test time compute in the cloud, I think Phi Silica is going to completely revolutionize what inference compute on the PC



is going to look like. And you all as developers are going to truly exploit that to build some amazing experiences.

And beyond the model itself, you now have all these rich semantic APIs to embed and index your local data into a vector store, with all the privacy protected. And then you can do these hybrid RAG applications where, when you do a RAG, you can actually get to the local data and have it relevant, contextually relevant. And if you're building your own models, Windows ML gives you a path to deploy high performance AI across a variety of silicon without the complexity of all these device specific tuning.

And today, we are taking the next step, modernizing Windows for the agentic web. We are announcing native support for MCP in Windows.

(Applause.)

Windows will now include several built in MCP servers, like file systems, settings, app actions as well as windowing. And we are adding native MCP registry that lets MCP-compatible clients discover the secure MCP servers that have been vetted by us for security performance, all while keeping you in control.

Let me hand it over to Divya to show you all of this and how it works and how it comes together, both as an app developer and also as a user of applications on Windows.

Divya, over to you.

(Applause.)

**DIVYA VENKATARAMU:** Thank you, Satya. Today, I'm going to show you how easy it is to set up a new Linux distro, build a complete website project and update with my Figma design, all with just three sentences in VS Code, using GitHub Copilot and MCP on Windows.

First, let's make sure we're set up correctly. For agents to connect to MCP servers, apps need explicit user permissions. In settings, you will see that I have already enabled WSL and Figma so their MCP servers can be easily accessible to agents. Other apps remain in their disabled default state, putting me as a user in full control.

Now that we're all set up, let's dive into VS Code and use my first two sentences.

I have the VS Code open with GitHub Copilot, and this is also in the agent mode, as you can see. I'm using my first two sentences, and I'm asking GitHub Copilot, can you search for the latest version of Fedora in WSL and install it for me? Once it's installed, can you also set up a simple website project in Fedora using Node.js?

I hit Enter, and Copilot here is trying to establish a secure connection with the WSL MCP server. And it gives me first of my several quick user consent prompts. I hit accept and allow it to proceed.

Behind the scenes, it's also trying to query the list of available online distros, finds that Fedora 42 is the latest version and proceeds to install. This installation, as you can imagine, can take you roughly about two to three minutes. So let's switch over to a device where this setup is complete.

On this device, I've got the environment set up. Let's scroll back up and see what steps it executed for me.

As you can see, it's the exact same prompt. GitHub Copilot and the agent mode was smart enough to parse this multi-step request pretty seamlessly. It also queried the WSL online distro, set up the Fedora environment for me. It also installed the Node.js NPM directly inside Fedora using the DNS Package Manager. The best part here is it also knew how to run specific commands directly interacting with WSL on my device. That's pretty neat.

Now, let's go take a look at the simple website project it created for me.

Well, this is a start. I'd like to take it one step farther and make it look like the design I've selected in my Figma. Here's my Figma desktop app, and here's the design I've selected, Penguin Pen Pals. Now, I'm ready to go back to VS Code and use my third sentence.

I'm going to ask GitHub Copilot here, can you make my website look like the design I have selected in Figma, and pull the design details and apply them to my project? But you can see the detail prompt in the chat. Now, I enter and Copilot, here again, is establishing the secure connection with the Figma MCP server and asked for my consent to proceed. And I obviously accept.

Before doing that, it queried the list of available servers in my MCP registry, found that Figma was the most relevant one to perform this task, and it proceeds to connect. Once connected, the agent pulls the design details directly from Figma, integrates them into my project and updates the site, layout, design, style and content accordingly.

This could take roughly about two to three minutes. Let's switch over to another device where this final step is complete.

On this device, the whole setup is complete. On the left side, you will see the code changes it applied by pulling the design details from my Figma desktop app. Now, it's the right time for us to go take a look at the website it created for me.

Boom, this is exactly the design I selected in my Figma desktop app.

(Applause.)

It did all of this in just three sentences in a single chat interface.

This is just one example of a developer centric scenario I showed you today. Imagine what other incredible opportunities this can unlock for you, enhancing your productivity, all while

remaining secure. This is the power of MCP on Windows, secure, agent-driven and developer first.

Back to you, Satya.

(Applause.)

**SATYA NADELLA:** Thank you so much, Divya. As you saw with VS Code, GitHub Copilot and WSL, we are really making Windows the best dev box for the agentic web.

Now, speaking of WSL, we first announced Bash on Ubuntu on Windows nearly 10 years ago. It subsequently became what we obviously call today, WSL. Looking back, in fact, the very first issue in the repo was a request to open source it. At that time, all of our logic for the project was quite inseparable from the rest of the Windows image. But since then, with a lot of the changes we made, WSL 2 distros are more separable and they're standalone.

Let's go back and reopen that very first issue and close it as fixed, because today, we are making WSL fully open source.

(Applause.)

So far, we've talked a lot about the agentic web and the ecosystem around it coming together, whether it's Microsoft 365 to Foundry, all the way to Windows. And I thought it would be great to have Kevin Scott, our CTO, to give you a broader perspective on this ecosystem and our collective opportunity to build a more agentic web.

Welcome, Kevin.

(Applause.)

**KEVIN SCOTT:** All right. Thank you, Satya. It's super good to see you all. I think this is my ninth Build as CTO of Microsoft, which seems like a really long time. And I get to come out here once a year to talk to you all, forget that I'm an introvert for about 15-20 minutes. And it's not like there's some magical thing that happens every May that makes me forget that I'm an introvert. It's just that there have been so many exciting things that have happened in the universe of technology over the past year, and there's so many things that are coming in the next year that I just want to share some of that enthusiasm with you all.

And the thing that is really exciting me right now is something that is emerging, not solely because Microsoft wants it to happen, but because the components and the protocols and services that we've all been collectively working on for the past few years are forming this thing that we've been calling the agentic web. And the thing that's really, really encouraging about it is that it's happening in an open way, which I think is the thing that we all should want a whole lot, given where we're headed.

I'll explain a little bit of why that's important in a minute, but before we go into the architecture of this open agentic web, let's talk about what an agent is.

I know you all probably have a bunch of different definitions in your head. It's a little bit like a Rorschach test, but the thing that I always come back to and what we're pushing towards here at Microsoft, both in terms of the agents that we're building ourselves as well as the capability that we are trying to create to help ourselves and to help you all make your own agent, is that an agent is a thing that a human being is able to delegate tasks to, and over time, the tasks to which you are – the task that you're delegating to these agents are becoming increasingly complex. And that's the thing that we're aiming for over time.

The thing that we've seen over the past year is just an explosion of agents themselves. This is their diagram of what the agentic web looks like. And at the very top of the stack is a richer and richer ecosystem of agents that we're building, our partners are building, that you all are building. And we're just going to see more coming over the next year.

And the striking thing about what's happened with these agents over the past year is that they're being used more and more frequently than ever before. Across the agents that we have visibility on, the daily active users since I was here last time chatting with you all at Build, have more than doubled, so just, people using them at an incredibly increasing and accelerating rate.

But the bigger thing that I think has happened is that these agents, because of the new reasoning models that have launched over the past year have been able to really take on extremely complicated task. And you've seen a whole bunch of this stuff already from Satya and from the folks who've been on stage to provide demos, just the incredible things that some of these agents can do right now, particularly in software engineering. And you guys are going to see more and more of that over the course of Build.

But sitting on top of the agent layer of this agentic web stack is a thing called the runtime layer. And the runtime is, again, an emerging set of components that we are building. The thing that I just mentioned that's gotten really good over the past year is the reasoning layer, or the reasoning component inside of the runtime layer here. The reasoning capabilities of these models is going to continue to get better and better. If anything, right now, I think we have a little bit of a capability overhang with reasoning, so the models are more powerful than what we collectively are using them for at the moment.

And one of my big challenges to all of you this year at Build is to just think about how you can set your ambition level to 11 to try to target some things that you think are barely possible right now in terms of reasoning capability, because the models are going to get so much more powerful and so much cheaper over the course of the next 12 months that we don't want that capability overhang getting bigger than it already is.

But there are a whole bunch of things in addition to reasoning that need to be built that are just basic software engineering. A thing that's been conspicuously missing for a while with agents is really robust agentic memory. We've approximated it a bunch of ways with things like RAG and with long context Windows. But the thing that you really want your agents to be able to do is to

have a rich memory over which they can recall with real breadth and where you can have high precision in the recollection, so that the things that are being pulled back out of memory are accurate, and you can trust them and rely upon them.

And so, there's just super interesting stuff happening. We open sourced the system called type agent that you all can learn more about later here at Build that is one way that we're trying to tackle this agentic memory, which is just really important to make them less transactional. They can solve a problem once and recall the solution. They can remember their interactions with you, so that they can become personalized and understand your preferences, the same way that you would expect from someone that you were delegating a task to or that you were collaborating with.

This whole runtime layer, the way that you can think about how Microsoft is delivering these components and participating in the community is these things are going to be offered in Azure Foundry. Azure Foundry is going to get richer and richer and richer over the coming year, and you're going to hear my colleague, Jay Parikh, and a bunch of other folks talking about that boundary layer in this runtime in much greater depth.

But the thing that is super important, if you think about what an open agentic web could be, is you need agents to be able to take actions on your behalf. And one of the really important things about agents being able to take actions on your behalf is they have to be plumbed up to the greater world. You need protocols, things like MCP and A2A, and things that likely are going to be emerging over the coming year that will help connect in an open, reliable, interoperable way, the agents that you are writing and agents that are being used so actively now by hundreds of millions of people to be able to go access content, to access services, to take action on behalf of users in fulfilling the tasks that have been delegated to them.

The super exciting thing that we really want to talk about here at this Build is our real serious commitment to MCP. MCP has just taken off like crazy over the past handful of months, and there's a good reason for it. It's filling an incredibly important niche in this open web, open agentic web ecosystem. It's a very simple protocol, akin to HTTP in the internet, where it allows you to do really sophisticated things, just because the protocol itself doesn't have much in the way of opinion about the payload that it carries, which also means that it's a great foundation, which you can layer things on top of.

And if you think about how the internet itself emerged, that composability of components and layering of protocols was really important to get to the richness. It gives you good starting places, and it gives you the ability to solve problems as you discover them, as things are developing over time.

We have heard from Satya just now about the MCP registry in Windows. We're doing a ton of work to enable Microsoft's first-party platform components and first-party services to be MCP uplifted. We've chosen it as our standard protocol for agent to communication, between agents and from agents to services.

And then we're going to be doing a whole bunch of work over the next handful of months with our partners and collaborators at Anthropic to make sure that the really tough enterprise problems that need to be solved on top of a protocol like MCP get resolved and solved, like how an agent identifies itself, how you build a really rich entitlements infrastructure so we can permission agents to have access to the systems they need to have access to. And then we are going to be just really, really supporting the open community here.

Partially, that is because when you think about the utility of a protocol at this layer of the stack, the most important thing is ubiquity. We can get into all sorts of arguments, and I know you all are perfectly capable of this as am I, as engineers, where you've got really sharp opinions about pieces of technology, and this thing's a little bit better than that thing. But what's better than all of that is just getting something standard that we can all use and build on top of. And we hope that thing is MCP.

The thing that I want to talk about next is I mentioned layering. And so, if you think back to the web, we have HTTP, and then we had things that sit on top of HTTP, like HTML, mainly, that are opinionated about the payload.

And so, we're announcing today, and you all should go check out the code in the GitHub repo, NL web. And the idea behind NL web is it is a way for anyone who has a website or an API already to very easily make their website or their API an agentic application. It lets you implement and leverage the full power of large language models to enrich the services and products that you're already offering.

And because every NL web endpoint is by default an MCP server, it means that those things that people are offering up via NL web will be accessible to any agent that speaks MCP. You really can think about it a little bit like HTML for the agentic web.

We have done a bunch of work already with partners who have been really excited and been able to really, very quickly get to quick implementations and prototypes using NL web. We worked with Trip Advisor to O'Reilly Media, a ton of really great companies that offer important products and services on the internet, to add in a web functionality to their site so that they can have these agentic experiences directly on their sites.

And one of the things that I wanted to show you all to just ground you in the capabilities is how you all might be able to go out right now, although you shouldn't do it right now – wait to hear all of the amazing finish of Satya's keynote, please, but right after that, you can go grab code from this depot.

And what we're showing here is a way for you all to put an agentic NL web user interface on top of any GitHub repo that you have access to. What we're showing on the screen is you run this Python script. It talks to the GitHub API. It extracts a whole bunch of data from your GitHub repo. Here, you're seeing Jennifer Marsman, who's a member of my team in the office of the CTO, who's the best demo builder on the planet, who is getting all of the meta data and data out of her repos into a web repository.

The second thing that you do is you go run a little script that takes all of this data, computes a bunch of embeddings for it, in this case, and you have a ton of choice here about which model you're using, which embeddings you're using, which vector store that you're sticking the embeddings in, in a web server to have access to. And as soon as you compute the embeddings and import them, in this case, into a local vector store that's running on her laptop, then you're able to make some really interesting queries on top of it.

In this case, you can do more than what you would get just with a database query or some information retrieval thing. You're able to say, hey, I wrote this piece of code. It was in some PR I don't remember. I need to recall the name of this algorithm. And that's the thing that NL web can do, and things that we honestly aren't going to even be able to imagine, which is why we're making this open source.

And we're hoping to hear a ton of feedback from you all about how we can improve and make NL web better, because, again, we think that the sum total of your imagination operating on the agentic web is going to make the agentic web a fundamentally more interesting thing than we would have, if one entity was trying to do everything alone.

You can learn more about these things from these two talks in particular. There is a fantastic talk at 4:30 to 5:30 today on how you can turn your website into an AI app. And then the memory stuff that I was referencing, we've got a talk from 1:45 to 2 p.m. today that is going to be really awesome.

The last thing that I want to say before handing things back over to Satya is to just press on these two points about why open is so important here.

It is unbelievable what can happen in the world when simple components and simple protocols that are composable with one another are out there exposed to the full scrutiny and creativity of every developer in the world who wants to participate or who has an idea.

This thought game that I play with myself all the time is trying to imagine what the web would have looked like if one of the actors in the early development of the web, say the browser manufacturers, had decided that they wanted to vertically integrate and own the entire web. A hundred percent of the web would have been dictated by the limits of their imagination. And it's just obvious, with 30 years of history, that that wouldn't have been a very interesting web that we would have, because the web is interesting because millions, tens of millions, hundreds of millions of people are participating to make it into this rich, dynamic thing.

That's what we think we need with the agentic web, and that's what we're hoping you all can get inspired to go work on a little bit and to riff on and to use the full extent of your imagination to help make this thing interesting.

Thank you all for being at Build, and I'll see you next year.

(Applause.)

**SATYA NADELLA:** Thanks so much, Kevin. The agentic web vision that Kevin just described, in some sense, gets us closer to even the original vision and the ethos of the web. Both the content as well as the intelligence now can be more distributed and discoverable across the web. NL web, if you look at it, it's a pretty fascinating thing. I'm really looking forward to seeing what all of you make of it.

But it democratizes both the creation of intelligence for any app, any website, but here is the interesting thing. It democratizes the aggregation of this intelligence for any developer. You can easily see it. Give me a reasoning model and NL web, and I can take an intent and start composing, using the reasoning model to go compose and synthesize across that distributed intelligence. It's just a completely – what is search? What is a feed? Any of these things gets completely changed in terms of how one goes about building it.

This is a platform that we want to create together. I think something big is going to shake out of this, because this is not just the repeat of the past. The last 10-15 years, I know, has always been about the aggregated power. I think something big is going to shift.

Now moving to the next layer of the stack, data. After all, for any AI app, the data tier is super important. We are building out the full data stack. There are so many great examples, whether it's Lumen using Fabric to save over 10,000 hours, because they can access data faster, or Sitecore delivering 10x speed with our data stack. But one of my favorite examples is NFL, which used our data stack to run the recent scouting combine. Let's take a look at the video.

(NFL video segment.)

(Applause.)

**SATYA NADELLA:** Yeah, sports and data, it's the most fun, just love that. And we're making a ton of data news at Build, starting, in fact, with SQL Server 2025, that's launching. And more importantly, we're bringing the data layer and the intelligence closer than ever before. For any real-world agent, you require the storage, and that's why we're integrating Cosmos DB directly into Foundry. That means any agent can store and retrieve things like conversational history, and soon they'll be able to also use Cosmos for all their RAG application needs.

And we're taking it further with Azure Databricks, connecting your data in genie spaces or in AI functions to Foundry. The other very cool capability is now inside of a Postgres SQL query, you can have LLM directly, LLM responses directly integrated. Basically, you can have a natural language and SQL mix together and have the orchestration happen and a query plan happen in a very different way.

And when it comes to Fabric, which we launched here at Build two years ago, it's at the heart of our data and analytics stack. And Fabric brings together all your data, all your workloads together into this one unified experience.

Last fall, we put SQL into Fabric. And today, we're taking the next big step. We're bringing Cosmos DB to Fabric, too, because AI apps need more than just structured data. They need semi-



structured data, whether it's text, images, audio, and with Cosmos and Fabric and your data instantly available alongside SQL, you can now unify your entire data estate and make it ready for AI.

And there's a lot more. In fact, we are even building our digital twin builder right into Fabric. Now, you can very easily take digital twins with no code, low code. As you can see here, you can map the data from your physical assets and systems super-fast.

We're also announcing shortcut transformations in OneLake. You can think of this as AI-driven ETL. You can apply all these pre-built, AI-powered transformations, audio to text or sentiment analysis when data is coming in, summarization, all powered by Foundry, straight into Fabric, and all with just a few clicks.

Lastly, I want to talk about one of my favorite features now, which is coming to Power BI, which is Copilot in Power BI, so that allows you to chat with all of your data. You can ask questions about data, visually explore, analyze it across multiple Power BI reports and semantic models. It'll also be available. This agent will be available in Microsoft 365 Copilot. The power of all the work you did, building out those semantic models, building out those dashboards in Power BI, and now being able to put reasoning models on top of it in a chat interface, I mean, think about what a game changer that will be. And so, we are very, very excited about that.

(Applause.)

Let's get to the bottom layer of the stack, which is infrastructure.

As a developer, you face that classic optimization problem between delivering the best AI experience in terms of performance and latency, and then, of course, you've got to deliver it with monotonically decreasing cost. And that's why we are taking a systems approach, working across the entire industry to optimize the full stack, whether it's the data center, silicon, the system software or the app server, bringing it all together as one system, and optimizing, using the power of software.

And our aim is to offer the lowest cost, higher scale infrastructure to build both cloud and next generation of AI workloads. And it all comes down to delivering the most tokens, per watt, per dollar, and that's the ultimate equation for us. And we're writing multiple S-curves across silicon. Think of it as classic Moore's Law, plus system software optimization and the model optimization. It's the compounding effects of those three S-curves.

And we're doing all of this with real speed. When anything gets ready, you want to bring it into the fleet. In fact, Azure is the first cloud to bring Nvidia GB200 online, and do so at scale. Azure now leads the performance from a single GB200 with NV link, 72 rack, generating an astonishing number of tokens, 865,000 tokens per second. It's the highest throughput for any cloud platform.

And of course, there's nobody better to talk about this than Jensen Huang. Let's roll the video.

(Begin video segment.)

**SATYA NADELLA:** Thank you so much, Jensen, for joining yet again our Build developer conference. At the end of the day, our joint goal is to be able to deliver more intelligence to the world. In some sense, you can even say it's tokens, per dollar, per watt. And so, I just wanted to be able to maybe get your perspective on all this, maybe starting, in fact, with the very start of what has been our industry, which is Moore's Law.

**JENSEN HUANG:** So excited to be here Satya. In fact, two years ago, we had just launched the largest AI supercomputer in the world together on Azure. This is the big change in how computing works now, with our new CUDA algorithms and new model technology in your new AI infrastructure on Azure, together, all of that together, 40x speed up over Hopper. That's just an insane speed up in just two years.

**SATYA NADELLA:** Do you want to talk about the compounding effects of software that then add to all of what we're doing together?

**JENSEN HUANG:** We still want architecture compatibility. We want the rich ecosystem to be stable, so that software developers' investments, and all of the AI developers' investments and all your infrastructure customers' investments get to be distributed and amortized across the entire fleet.

**SATYA NADELLA:** Do you want to speak a little bit to the what happens to utilization and the diversity of workloads that get all accelerated?

**JENSEN HUANG:** One of the benefits of CUDA is that the install base is so large. Another benefit of CUDA is that on the one hand, it's an accelerator architecture. On the other hand, it's fairly general purpose for some of the heaviest workloads.

Our two teams are working on accelerating data processing, accelerating data processing by 20x, 50x, transcoding video, image processing, models of all kinds. All of those different types of algorithms map wonderfully on top of CUDA. Our job is to make sure that that fleet and that workload, all of the workloads of your data center, are fully accelerated, the fleet is fully utilized for the entire life of the fleet.

**SATYA NADELLA:** It's not just tokens, but it's all workloads, per dollar, per watt. Can we really accelerate them all, as we continue to innovate across both the hardware and the software boundaries?

**JENSEN HUANG:** And Satya, thank you for your partnership and leadership, and the alignment between our two organizations to build the most advanced infrastructure in the world, the most advanced AI factories in the world. And the best of times are yet to come.

(End video segment.)

(Applause.)

**SATYA NADELLA:** In fact, the largest GB200-based supercomputer is going to be Azure. We're very, very excited about scaling this and making it available to all of you as developers. In fact, we are continuing to expand Azure. We now have 70+ data center regions, more than any other provider. Over the past three months alone, we have opened 10 data centers across countries and continents. Of course, we're building a complete AI system. That means it includes cooling to meet the demands of AI workloads.

With MAIA, we had introduced and engineered the sidekick liquid cooling unit that also supports now GB200 in this closed-loop fashion so that you can really consume zero water. On the network side, our newest data centers built for AI have more fiber optics than we added in all of Azure before last year. Think about that. The data center design is just mind blowing for me when I go back and see them. It's unlike anything that at least I grew up with.

We're connecting our DCs with this AI WAN, 400-terabyte backbone. It's about all the type of workloads, whether it's inference or training needs when you distribute them, the AI WAN connects this data center footprint, and of course, the AI applications themselves. It's not just about having an AI accelerator or a GPU, you need all of your compute, and every app needs storage and compute. That's also a place where we are driving a lot of the S-curve efficiency there as well. Last October we launched Cobalt, our ARM-based VMs. They're already powering a lot of our own workloads with massive scale. Teams runs on it; Defender runs on it. We have our partners Adobe, Databricks, Snowflake, all scaling on Cobalt.

In fact, talking about new workloads that are running on using our fungible fleet of AI accelerators, GPU, compute and storage, Met Office is an unbelievable example. They're using Azure to better predict the weather, ensuring timely and accurate forecasting for millions. The supercomputer they built on Azure is the most advanced supercomputer of any kind for weather and climate science. Let's take a look.

(Start video segment.)

**MET OFFICE DEMO:** Met Office has been in existence for over 170 years, delivering the most trusted weather and climate intelligence in our radically changing world. Every day we are taking on board more than 50 billion observations, 200 to 300TB of operational data today, so we need very big storage and very big compute to be able to do that numerical weather prediction.

The resolution of model is very important. They require a lot of supercomputing power. The more supercomputing power we have will help increase the accuracy of our forecasts. That will also then help people make better decisions. The weather and climate science is big science. The data that those models produce underpins all our products and services. Our ability to repeatedly and reliably deliver these massive supercomputing services is of paramount importance, and we can only really do that in something like Azure.

This program really puts a scientific supercomputer in the heart of the Azure wider infrastructure, and it's fully integrated, which means that as we go forward, we can start to develop new and currently impossible products and services. It's going to be the world's first

cloud-based, solely for weather forecasting supercomputer. It enables us to run more complicated models at higher resolution to provide more timely and accurate predictions. Our customers will notice improvements to our forecast accuracy.

**ANNOUNCER:** Good morning. Welcome to our UK Met Office weather update.

**MET OFFICE DEMO:** We do weather forecasts and climate predictions for the public, for the government and for businesses. It's about that honesty and about building that trust so that when the weather is severe, people do believe what you're going to say and most importantly, they act upon it. Looking further ahead, we've got really exciting plans where we're extending our forecast range, higher resolution modeling, all of which will benefit our weather forecast accuracy, timeliness, and usefulness. It's just going to enhance everything that we do here at the Met Office and improve everyone's lives.

(End video segment.)

**SATYA NADELLA:** It's of course more than just performance and efficiency. It's about digital resilience. We have the most comprehensive set of compliance capabilities in the cloud. We offer sovereign controls such as data residency, confidential computing for cloud infrastructure, including your AI accelerators and GPUs, as well as the past workloads. This ensures that you maintain that complete control and how your systems are governed, who has access to them.

Now, while our cloud offers this comprehensive coverage, there are still many critical use cases that need extremely low latency and explicit control where and how your apps and data are stored. That means the ability to run things disconnected. That's why we also offer Azure Local. Now, so far, we've talked about developer and knowledge worker workflows on our AI platform, but there's one more domain that I want to talk about to close out, which is science.

One of the most exciting things I think that will happen over the years to come is we'll have real breakthroughs in the scientific process itself, which will really accelerate our ability to create a new material, a new compound, a new molecule. That's why we are bringing together the entire stack I talked about today and saying, "Look, let's apply it to science and the scientific workflow, the scientific process." That's our ambition with Microsoft Discovery, which we are announcing today.

(Applause.)

If you think about GitHub for software developers, Microsoft 365, Copilot and Copilot Studio for knowledge work and business process, Microsoft Discovery is for science. It's built on a very powerful graph-based knowledge engine graph RAG. Instead of just merely retrieving facts, this is the key thing. It understands the nuanced knowledge in the scientific domain from public domain as well as your own data if you're a biopharma company.

Discovery is built on Foundry bringing advanced agents that are highly specialized in R&D, not just for reasoning, but for conducting research itself. Instead of really static pipelines, these agents cooperate in a continuous iterative cycle, generating candidates, simulating them and

learning from the results. Think about that Copilot agent or coding agent, this is the science agent. Let me hand it over to John to show you all this in action. John.

**JOHN LINK:** Thanks, Satya. I'm John, chemistry product lead for our science platform. Today I'll show you how I lead a team of agents to make a discovery for immersion coolant. These are an interesting line of research for cooling data centers. Unfortunately, most of them are based on PFAS, or forever chemicals, which are harmful for the environment. How could I discover a better coolant? I'll show you three steps today: Reasoning over knowledge, generating hypotheses and running experiments, all in an iterative loop. Let's dive in.

My first step is to conduct research. I want to be up to date with the latest knowledge on this topic, so I'll start by asking about coolants and their properties to help me identify starting candidates. The platform uses a network of agents that reason over scientific knowledge in both public sources and your trusted internal research. Behind the scenes, it uses a knowledge graph to provide deeper insights and more accurate answers than we get from LLMs on their own, which can struggle to connect the dots.

This process will take a few minutes, so let's skip ahead. What we see here is a summary on the left and a comprehensive report on the right, covering the state of the art in coolant research, with links throughout to citations for trusted research. Now, I can validate these findings and iterate if I choose to before moving on, but our goal today is not just to reason over knowledge, we want to make a real coolant discovery, so we need to move on to step two, which is generating hypotheses.

I am going to ask for a plan specific to my investigation, informed by this research. Knowing, for example, that I should target a specific boiling point and a dielectric constant that won't fry my electronics. We'll submit that. Now, my team of agents is working to build the right workflow. Now, you'll see I didn't specify what methods to use or write any code, Microsoft Discovery handles it all for me. The agents can use tools and models from Microsoft, they can integrate open-source or third-party solutions, or even ones from my own organization. I can always adjust this plan, either adding additional steps or changing the criteria so that I'm always in control.

If we look right here, we can see the plan that my agents returned. It starts with a generative chemistry step here, where it creates millions of novel candidates that are more likely to meet my criteria, and then it uses AI models to screen these down quickly. Finally, we use HPC simulations for validation of our findings. As I look at this, I like this plan. It looks like a good approach, so let's go ahead and run it.

We click proceed right there, and with that, we're in step three: experimentation. Microsoft Discovery executes this full plan, choosing and managing the best HPC resources in Azure. In the future, it will also integrate our advances in quantum computing. We can see here now the Discovery agents working together in real time, driving all of these intense computations.

With traditional methods, getting to a short list of candidates could take months or years of trial and error; Microsoft Discovery can compress the time to days or even hours. Let me show you the final results of the process. These are the set of candidates that Microsoft Discovery has

identified for PFAS-free immersion coolants. Now I can analyze these results to decide if I'm ready to head to the lab or if I need to run another iteration. If I just take a look at a couple of these, I can see that indeed, the boiling points are in line with my expectations, and the dielectric constant is great. This is very promising.

But I know what you're wondering. Did we actually make a discovery? Well, this is not just a demo; we really did this. We took one of the most promising candidates and synthesized it. They didn't let me bring a new material unknown to humans onto this stage, but I've got this video from the lab. We can see my coolant there, and we dropped a standard PC in it. It's running Forza Motorsport, and it is keeping the temperature stable with no fans. It's literally very cool.

(Applause.)

We found a promising candidate for an immersion coolant that does not rely on forever chemicals. Imagine using Microsoft Discovery across domains, designing new therapeutics, new semiconductors or new materials. It worked for us; the next great breakthrough is yours to discover. Thank you. Have a great Build. Thank you, Satya.

**SATYA NADELLA:** Thank you, John. It is indeed very, very cool. It's great to see how companies across every industry are already using our discovery to accelerate their R&D. I can't wait to see this in the hands of more R&D labs all over and what they can do. That was a quick, comprehensive, whatever you want to call it, walk through the full stack and how we're creating new opportunity for you across the agentic Web.

We're talking we're taking really a systems approach, a platform approach, which you can expect from Microsoft across every layer of the stack. Whether it's GitHub and GitHub Copilot, enabling an open ecosystem for the software development life cycle, Microsoft 365, Copilot, Teams and Copilot Studio enabling agents for every role and business process, and an agent factory in Foundry enabling you to build any AI app, any agent using any data, all running on world-class infrastructure. All of this on a robust set of rails for management, identity and security.

Ultimately, though, all of this is about creating opportunity to fuel your ambition. I think back to some of the developers I've had the chance to meet over the course of the last year, like a father in Spain using Foundry to help speed diagnosis of rare diseases like the one affecting his son. A startup in South America building an app to gamify wellness. An airline in Japan using Phi to help flight attendants complete reports even when they're 30,000 feet above. Examples like this is what I'm excited about and take much pride in.

After all, the big winners are going to be people like yourselves who are going to build these applications, not just people who create platforms like us. The winners need to be across every sector of the economy, in every part of the world. That's our goal. For us, it's never been about the technology, it's about what people can do with it. That's what is core to our mission. I want to leave you with this one really powerful example.

Earlier this year, World Bank conducted a study for students in Nigeria using Copilot. What they found was pretty remarkable. By their analysis, it was the best, most-effective educational intervention involving tech. This is something I've been working on throughout my life at Microsoft. To see finally, in a country like Nigeria, putting something in the hands of people and having it really register statistically as an intervention that had positive impact. The World Bank has expanded the program to Peru, where hundreds of public-school teachers are now using Copilot to enhance lesson plans, improve student outcomes and grow in their own careers. Let's roll the video. Thank you all very, very much. Enjoy the rest of Build.

(Video segment.)

END