

05202025 Microsoft Build Keynote Jay Parikh

Microsoft Build 2025

Jay Parikh

**Jessica Deen, Amanda Foster, Elijah Straight, Patrick LeBlanc, Mehrnoosh Sameki,
Seth Juarez, Jatinder Mann**

Tuesday, May 20, 2025

Voice Over: Please welcome executive vice president Core AI, Jay Parikh.

JAY PARIKH: Good morning, everyone. How is day one? That's awesome. All right. Well welcome to day two. My name is Jay and I support the new Core AI team here at Microsoft. This is my very first Build, so it's great to be here with you all.

Now, I'm excited because I've spent a lot of time in my career building infra, tools, platform for developers. Soon after I started Microsoft last year, Satya and I were chatting and we were discussing what it would take to reimagine the end-to-end developer experience for this new era of AI applications.

Now with this in mind, we formed the Core AI team to focus on empowering every developer to shape the future with AI. Now this is a big, ambitious vision and we have a lot of work to do. But there are three principles that guide us.

First is that we need to force ourselves to use AI driven tools and platforms instead of just AI assisted experiences. And we need to transform the entire development experience, not just a part of it. And finally, we need to continue to embrace openness and choice, given how fast the landscape of AI is changing around us.

Based on these principles, today we're going to show you a complete set of AI powered tools for building on top of a full stack agent factory with security and trust baked right in, extending from the cloud to the edge.

Now, we're focused on this because we want these tools and this factory to bring back joy in building and operating your new age of AI applications. We're going to make it easier for you to build. We're going to make it easier for you to test, for you to deploy and for you to monitor your applications. We want to help you modernize your applications, and we also want to help burn down tech debt.

Now, why is this important to us? Because as builders, our most precious resource is time. And we want to give you time back. So, as I tell our team, show more, talk less.

Now, as an engineer, I want to get in and start to understand how to use these tools. So, I'm going to jump right in here to Copilot. And first thing I'm going to do is pop into immersive experience.

Now, as a new member of the team, well, let's start off with just something basic that we all know how to do. So, we want to see how Copilot is going to help me ramp up in this new application. And I'm a new member of the team, and I'm trying to learn context about what this Build Events demo is that we're going to be using throughout the course of the day.

Now, this is what Copilot returned. It's pretty basic. I think we can do a little bit better. So I'm going to ask Copilot to help me out here now. Copilot is going to take this. I asked it to open up a PR so that maybe when a new engineer joins the team, the Copilot can actually help create a new Readme file with instructions that actually help me understand the code, understand the application, but also provide setup instructions.

Copilot got to work here, and I'm going to click on this to show you what it started to work on here. So here's the new request that it's working on. You can see that Copilot started work on behalf of me. Now it's going to take some time to go pull this together.

So, this was helpful. And it's a fast way for me to ramp up as a new person in the team. But let's dive into our tools a little bit more so that we can show you how to take these tools so that we can develop faster. And for that, I want to welcome my colleague, Jessica.

JESSICA DEEN: Hi, everyone. Yeah. Ready to do some stuff, yep. Good morning, everyone. My name is Jessica, and I'm a developer, which mostly means I spend a lot of time asking myself who broke this? And then realizing it was me. OK, I'm in VS Code, and with the GitHub extension, I've got all my issues and PRs right here in the editor, which is great until someone decides that I'm the expert and creates the kind of issue that looks simple until it most definitely is not.

So, let's do what every good developer does with a long backlog. You're absolutely right. Immediately look for something we can pawn off. All right. First up, we have input validation. What do you think, Jay?

JAY PARIKH: No.

JESSICA DEEN: Absolutely not. Yeah. I'm going to have to escalate this to, not me. Copilot, you're up. I believe in you. You love boring things. All right, next, type hints?

JAY PARIKH: Also no.

JESSICA DEEN: Yeah. Nothing says I trust no one like explaining every argument and return value in painful detail. Hard pass. Again, I'm going to let Copilot take this one. It lives for this kind of thing.

All right, next. To-do comments. Seriously? I feel personally attacked. Who opened this? Oh. Told you I break things. It was me. Yep. Copilot, my trusty sidekick, go fulfill past me's empty promises. So, yeah, instead of spending the rest of this keynote writing boilerplate validation and chasing down old to-dos, I can let GitHub Copilot coding agent handle the stuff that drains my will to code and actually focus on real product work like this one.

Updating the mic icon with a new design from Figma. I'll just click this arrow here that'll self-assign me to the issue, and it'll allow me to start work on a local branch. Let's also go ahead and bring up the simple browser here so that we can see what we're working with.

Great. I can see the current mic icon now. I'll just hop into the mic icon file and we'll start doing an onClick event. Perfect. Thanks, GhostText. Now notice that when I do, Copilot actually sees this, and it starts to suggest updates to other parts of the code proactively.

All I have to do is hit tab and then tab again and that's it. This is called Next Edit Suggestions, and it is Copilot taking completions to the next level. Now, Jay, I could go back over to the issue. I could grab the design from Figma. I could start refactoring and writing the route logic.

But honestly, you didn't come here to watch me code. They didn't come here to watch me code. We've got a packed keynote full of awesome things. So, let's actually scale me up and let's bring an agent in right here in VS Code. Simply pop over into Copilot and all I have to do is tell Copilot to begin to work on this issue.

Just like that, Copilot will actually take care of the rest. It's going to go out and fetch the issue details directly from GitHub. It'll even spot the Figma link and it'll grab the designs automatically. And check this out, Jay, look at this.

Under references, it's even sure to follow our organization and our code base's best practices. It picked the Copilot instructions file up automatically. Now you're probably thinking at this point, wow, Jessica, this is spectacular. And you are correct.

But you also might be wondering, how is this all working? I'm so glad you asked. This is all powered by Model Context Protocol or MCP, and to keep it simple, MCP enhances our chat experience by giving Copilot more context and access to more tools. You can see those tools here. We can do a wide variety of things with GitHub, and we can even grab the assets directly from Figma. OK, now I can see the agent mode is done. Let's take a look at it.

JAY PARIKH: Wow, that looks spectacular.

JESSICA DEEN: This looks pretty great. It's a pixel perfect match to Figma and it even has the raised effect. Now all I have to do is hit keep, and then if I go over to my source control, I want to show you something else that I really love.

Copilot can actually even finish my commit messages for me too. It knows exactly what changed and it knows my commit style. And quite honestly, it saves me from writing fixed stuff or did the thing for the third time today.

Now, we could go ahead and publish this and actually submit a PR, but we already know how that part works. So, I want to show you something else that's really cool. The issue that you assigned earlier, I know you have a lot of things to cover in the keynote. I'm going to do you a solid and I want to review this for you.

While we were off working in VS Code, Copilot actually already handled the Readme, and we can take a look and see if it's done. And I believe it is. There we go. We see the files changed, and I'm just going to jump over and look at the rich diff, and look at this. It added quite a bit more things. It's pretty. This is awesome.

I don't think it's perfect. Now we've talked a lot about accessibility and having some visualization and some other things to enhance it. I think it'd be even more powerful if we added an architecture diagram to this.

JAY PARIKH: Good idea.

JESSICA DEEN: So, I'm going to jump over to the conversation. And honestly, this is why we do code reviews. I'll just jump down here, down at the bottom. And I'll go ahead and tell Copilot that I want to add an architecture diagram using Mermaid.js. Now check this out. When I hit comment and I scroll up, we'll see Copilot should pick this up pretty quickly and it'll immediately get to work on the requested changes.

There we go. We could jump into the session, but we don't have time to do that. Let me show you what a completed one looks like. Let's take a look here. Look at that Mermaid diagram. That's pretty awesome.

JAY PARIKH: That's awesome.

JESSICA DEEN: Pretty great. Now, this is one of my favorite parts because in just a few issues, Copilot helped me move from issue to implementation quickly. And honestly, it handled the stuff that as developers, usually slows us down. Personally, as a developer, it amplifies me, and as a result, I get to focus on higher impact work and still move fast, confidently.

JAY PARIKH: How do I use that in my app?

JESSICA DEEN: I am so glad you asked. This is all powered by something called Models. And the best part is, is that I can use these same Models in my own apps. In fact, that's what makes this so powerful is it's not just GitHub. In fact, everyone in this room can build their own agentic applications too, using GitHub models and Azure AI Foundry.

We can even pop over and take a look at the different models we have in the playground. And when we click use this model, we can also take a look at the different languages. And when you scroll down, we'll even give you some code to get started. And I believe we actually have an issue open for that, that Amanda and Elijah and our team are going to dive more into, GitHub models and Azure AI Foundry in just a few moments.

Now, just to recap here, in just about five minutes we've tackled five issues. How are we doing so far?

JAY PARIKH: We're doing really well now. I was looking at last night, though, we had an incident.

JESSICA DEEN: We did.

JAY PARIKH: And I was wondering if we should take a deeper look at that, because we may have an issue with this old version of Java in the back end.

JESSICA DEEN: I think you're right. And honestly, one of the hardest parts about being a developer is things like tech debt like that and figuring out how we can address upgrading things. I believe our back end is using Java 8, right? And we've talked a lot about how we can upgrade that. Why don't I take a look at the incident, and do you want to take a look at the Java 8?

JAY PARIKH: Yeah. Let me get started on that.

JESSICA DEEN: Perfect. All right. Let me go ahead and switch over here. Now, let's talk about that issue that Jay mentioned. I can see right here that we do have an incident that happened last night. It looks like there was an unhandled exception. And what's cool is I can also see an incident timeline in summary.

Now, as I'm scrolling further, I can see that there's a root cause analysis. I can see that there's some attachments. And when I open that issue when I click the deep link to go over into Azure, I can see the Azure Monitor alert that was detected. I can see it was a Sb2 alert that was fired at 1:57 a.m.

Now, this is important, and this is really cool because the SRE agent from Azure actually did all the work for me. We as developers know that one of the worst things is being paged in the middle of the night and being on call. Mainly because at 1:57 in the morning, I don't know about you, but my brain is not booted up. It is sleeping. It is in a hibernation state.

Luckily the SRE agent was able to address that and handle that super quickly. I can see that it immediately acknowledged the alert at 1:57 and it began the investigation. I can see the activity logs for it. It's starting to form a hypothesis. I can see the log analytics queries, and scrolling down even further, I can see that it finished the investigation. It was looking at the total requests, memory utilization, CPU. And here's where it noticed the spike in those 500 errors.

Now, what's also cool is it figured out the deployment and the correlation. So, it was able to detect that these 500 errors were related to a faulty deployment. And what's even better is it actually handled the rollback for me, and it continued monitoring it after the fact.

Now, this is really important because if I look at the timestamp here, we're now at 2:02. So, in seven minutes, it was five minutes actually, I believe, it was able to pick that up, roll it back, handle the monitoring, and I didn't have to get out of bed.

What's even more powerful is because it commented in the issue, it only took me about 30s to get caught up on, so this is great. It saved me a lot of time. I didn't have to wake up and I could

go back up to the top of this issue, assign it to GitHub Copilot and let GitHub Copilot maybe address the code change that broke it.

However, Jay, I think we know what the issue is. We really can't put off that Java migration any longer. And as painful as it's going to be, I think we have to tackle it. What are your thoughts?

JAY PARIKH: I think we can tackle it. Actually, Jessica, it doesn't look that bad because I think we can call Copilot to help us. So, check this out.

I started this project to see what we need to do to upgrade this. And it runs this report for us. So, we get a just a good landscape as to what we got to do here. Then I can pop over to the migrate tab here. And it says here, we've got a couple of things to do here, but we've got to upgrade our version of RabbitMQ here to Azure Service Bus.

I'm just going to click the migrate button here, and GitHub Copilot picks this up. It starts working on this. And we'll see as it's analyzing the project here you can see that it's looking at different files here. It's coming up with a plan for what it needs to do to go change here.

Now it's going to start to go look at a bunch of files here. And you can see, hey, look, there's a couple of Java files here. We got to go do this. So, I'm going to hit continue here. And it's going to get to work here. Cranking through these files and figuring out what we need to change here, and you can see here it's going to jam through this. And look at that.

Here's the set of changes that Copilot puts out there for us to upgrade here to Azure Service Bus and off of RabbitMQ. And it looks like this is done like pretty fast.

JESSICA DEEN: That's amazing. I mean this just saved us so much time. And here we were thinking we kept putting that issue off on the backlog, thinking it was just going to take way too long. And I mean, I think that's basically now, what, seven issues in just about seven minutes. It's a pretty good success rate.

JAY PARIKH: Yeah. We should keep it going.

JESSICA DEEN: We should. We were actually talking about another issue over lunch last week. I have an idea for it.

JAY PARIKH: Do I get to work on that?

JESSICA DEEN: Yeah. Why don't I work on this?

JAY PARIKH: OK. Perfect. So just to recap here, what you saw from our agent was to help you migrate an old version of Java to a new version of Java.

Now, it goes through this phase of planning out what the migration needs to look like. Then it executes it, and then it'll summarize this. Now, this doesn't just work for Java. It also works for

.NET right here in Visual Studio. So, you can see here the agent is planning out what this upgrade needs to look like.

It then starts to execute the upgrade. Now if it gets stuck, it can prompt the user, me, for hints and I can take control back. It'll learn from my prompts so that it doesn't make the same errors going forward.

Finally, once Copilot is actually done, it can produce a detailed summary of what the changes are so that I can review them. And when I'm ready, I can commit them, test them and push them to production. Now we're taking the same framework of plan, execute and summarize to help also modernize mainframe code.

There is billions of lines of COBOL running significant parts of major companies around the world, and it's hard to know what all of this code is doing today. Now, not only will the agent help us document what all this code is doing, it will also help write a set of unit tests so that when you do the upgrade, you will be able to test the application and making sure it's working correctly.

Now, we're still working on this, and this should be available later this year.

Now, we have a lot of customers who are taking and getting benefits from the migration agent here, not only for .NET, but also for Java. They're saving time and they're able to move faster. Ford was able to also refactor their RabbitMQ code base and upgrade to Azure Service Bus. They did this three times faster with GitHub Copilot. So, Jessica, how's it going over there?

JESSICA DEEN: It's going great. You want to see something really cool?

JAY PARIKH: Yeah. Let's check it out.

JESSICA DEEN: All right. So last week we were talking about adding a virtual guest book called Memories with an agentic Spotify integration using Azure AI Foundry. All right. I thought this was going to be super hard. It turns out with agent mode, everything is possible.

Look at this. I just told Copilot that I want to get to work on issue 44 for the memory service. Began to have a normal conversation. It went out and got the issue details from GitHub. It started creating all of these components and it even told me exactly what it was doing.

Scrolling down even further, I told it I wanted to set up Azure Cosmos DB, some Azure Foundry integration, and then it continued with setting all this up. Do you want to see the finished product?

JAY PARIKH: Yeah, let's check it out.

JESSICA DEEN: Do you guys want to see the finished product? Absolutely. Let's do this. All right. Here it is. It's called The Memories. And check this out even more. Seth, who you guys met, is so on top of it, he was so excited to use it.

He's already commented, this event is awesome, and here's what's really cool because we know we're at Build. We also have a Windows sound. And guess what? I believe it really works. We can test Copilot code, but do you want to do the honors?

JAY PARIKH: No. Why don't you?

JESSICA DEEN: OK, I'll go ahead and do the honors. Everyone ready? Pretty awesome. Now, what do you think that is?

JAY PARIKH: Excellent, Jessica. So, thank you for showing all this off.

JESSICA DEEN: You're very welcome. One more thought I did have is what do you think about dropping the "The".

JAY PARIKH: You'd probably just call it.

JESSICA DEEN: I think it's cleaner. All right.

JAY PARIKH: All right. Thank you very much, Jessica. Great job. OK. So, that's just a glimpse of what we're doing to reimagine this end-to-end development process. We're focused on every phase of this plan, create, and operate cycle.

Not only are you going to have a set of AI-driven tools to help you move faster in your own work, but you will have a team of these always on agents that you're going to be able to orchestrate. Now, best of all, we want to do this and optimize for openness and choice.

You all saw the announcement yesterday where we're open sourcing Copilot extension in VS Code. Now, I'm super excited about this because I spent a lot of time in my career helping to drive different open-source efforts, both software and hardware and know what a great community can do to accelerate adoption of new technology and to drive innovation forward.

Yesterday, we also saw, and we announced that we are working with OpenAI to integrate the Codex agent directly into the GitHub platform. Today, I'm actually excited to announce that we are working with Anthropic to integrate Claude Code directly into the GitHub platform as well. Think of GitHub as the organizing layer for your teams of always-on agents, all of your SUI agents, all of your SRE agents, and any other agents that you might be using, it will be the platform where you orchestrate intelligence across the entire software lifecycle.

Now, it's the perfect place to do this because GitHub has valuable context and it has the right security measures. Your agents can operate effectively, efficiently and securely. Openness and choice, that's what all developers want, and that's what we're going to deliver. Now, to wrap up this section, I want to highlight a lot of customers here that are getting benefits from using GitHub Copilot.

Now, I especially like what the team at Ramp is doing. Ramp is using Copilot to integrate in their development lifecycle, and they are saving roughly 30,000 hours per month of manual work so that they can ship faster and have fewer blockers as they continue to innovate and build their product. Now, I also want to have you hear from our partner at Cathay Pacific.

(Video segment.)

JAY PARIKH: It's great to see how they're saving so much time and bringing joy back to their development teams. OK, so let's shift from AI-powered tools to the platform that powers your AI applications and agents. Building an AI-driven product today requires an entirely new set of infrastructure systems, services and tools. That's where Foundry comes in. Foundry is the AI app and agent factory.

Now, a bit of a side note here. Earlier this year I was sharing our vision for Agent Factory with Bill, and he lit up with excitement. He shared with me that when he and Paul started Microsoft 50 years ago, the vision was to create a software factory. It was a company built around smart programmers, capable of developing a broad range of software, from tools to applications to operating systems. This was unlike many other companies at the time that were focused on single-purpose applications. We're taking the same approach as we build out Foundry. We're building a full-stack platform for all of your AI applications and agents. It's pretty awesome to see the parallel here 50 years later.

Now, we announced Foundry last fall at Ignite, and we've been working really, really hard to ship features quickly. Now, as Satya mentioned yesterday, more than 70,000 organizations today use Foundry. We shipped so many things, it's impossible to cover them. I really hope you take some time to go play with all of the new features and try them out and see what you can build.

Now, we're using Foundry ourselves internally as well, and you're going to hear from Charles in a little bit about M365 and Copilot Studio as well, and both of those actually leverage Foundry. I'm not going to have time to cover everything, but I want to jump into three main areas of Foundry here. Today we're going to cover models and agents, and there's a lot actually to go explore in knowledge or data as well.

Here's a couple of things that are really, really exciting to help you understand and how to use these models in very sophisticated ways. First of all, we have this new leaderboard which will help you pick models. Second, we've actually done an integration with Hugging Face so that you get access to 11,000 more models in addition to the 1,900 models that are in the catalog today.

Then we also commit the same-time access, so as OpenAI releases the newest, latest, greatest models, you will have them accessible in Foundry as well. We've added a model router which will help you select the best model for your prompt in real time. Then there's an integration using MCP into Azure API Management Service, so whatever APIs you've built in your enterprise, you can access them via an agent or an AI application via MCP.

As you saw earlier, the GitHub VS Code integration is there, and it's used by 100+ million developers. We also have a GitHub models tab to test out your prompts right there. Then also

very rich set of functionalities to help you customize your models and bring your own data to do whatever you need to do for your specific use case. Finally, we have a distillation capability that will allow you to take a big model, break it down to a smaller model so that you can optimize for cost and for performance.

Now, a wide variety of organizations are already using and taking advantage of these capabilities. One example here is the Indiana Pacers. They've built an in-arena, real-time captioning system. Using Foundry, they were able to get the error rate down to just about 1%. Let's hear from our next customer here Manus on what they're doing to build with Foundry.

(Video segment.)

JAY PARIKH: I love what the team at Manus is building. It's really interesting stuff; you should check it out. Now onto the other core area of Foundry, agents. We're seeing a lot of customers build agents of all types to power their business and move forward in this new AI era. Foundry gives you a complete stack of agent capabilities. Let me highlight a few key services that we've been talking about this week that can extend Foundry so that you can provide and build a richer set of agents.

First up, Agent Service. Agent Service allows you to create, deploy and monitor agents at enterprise scale. Second is agent catalog. It provides a starter set of templates, so if you want to get started in building an agent, you can jump right into the catalog, use a template, use a recipe and start to build an agent and iterate from there. Agent knowledge allows you to bring data from Fabric, or from Databricks or from SharePoint, and put that together with your agent so your agent can produce or perform more sophisticated tasks.

Next up, we have agent tools. We support frameworks such as MCP and A2A, and you have access to 14,000+ services using those frameworks. Now we have also frameworks that will allow you to build multi-agent workflows, so you can build very, very complex workflows, automate decision making in a cohesive and intelligent workforce. Let's take a look at how easy it is to build an example of a multi-agent system with Foundry. Please welcome Amanda and Elijah to walk us through this.

AMANDA FOSTER: What better way to showcase our Foundry agent factory than by demonstrating it live, then taking you behind the scenes with us to show you how we built the agents and ensure they're safe and secure. Before we jump into our multi-agent application, let's first build a single agent together using Azure AI Foundry Portal. Meet our RCP agent. Here's how we'll build it.

First, we're going to give it a model. Here we're using GPT-4o. Next, we're going to give it a set of instructions. Finally, we're going to configure some tools. Here we're going to use the file search tool to manage my invite list as well as the Create Calendar Event Open API tool, which will call an Azure Logic app to create a calendar event on my behalf. But now let's see this agent in action. Right now, I'm going to ask it to create a calendar event for today, actually called Post-Build After Party, and I'm inviting Elijah.

ELIJAH STRAIGHT: Oh sweet. I'm excited for this after party. What is it doing right now?

AMANDA FOSTER: As you can see, in a moment it will start using its configuration, which we just did together to make the correct tool calls and then eventually create the calendar event on my behalf.

ELIJAH STRAIGHT: Awesome. Now, moment of truth, let's pull up my calendar and you guys can see I got a really busy day here today. There it is. It showed up in my calendar. Thanks so much, Amanda, for the sweet invite. —(Applause.)— That was great, but now let's jump into the code. I'm jumping into VS Code, and you can see here that I have the Azure AI Foundry extension installed. This extension is great because we can see the models, agents and threads that are associated with our project.

I want to take a moment to zoom in on threads, because transparency is a key part of agentic workflows, and through threads we can see each step the agent is taking along the way. Here we can actually see what Amanda just said, as well as the agent's response, as well as more in-depth details like the number of tokens that were used, as well as the tools that were invoked. Pretty cool, right, Amanda?

AMANDA FOSTER: Yes, Elijah. But where is this thread data stored?

ELIJAH STRAIGHT: That's a great question. Developers actually have the ability to use manage resources, but I actually today have them in my own Cosmos DB. What's great also is that we could use the data in Microsoft Fabric and Azure Databricks to power our agents, and we can see it here. Also, we're very excited that Cosmos DB is now available in Microsoft Fabric, so that's great.

Now, let's jump back into the code to see how we can actually build our Foundry agent. I'm going to go here into this file, and you can see we're using the Foundry Agent Service SDK to be able to initiate our agent client. Then from there we're going to attach the file search tool. What's important to note here though is that even though I'm using the file search tool today, we could use a variety of tools including external APIs, MCP or Model Context protocol servers, or even other agents using the Foundry Connected Agent tool. Pretty cool stuff. You can see here I'm giving it a model as well as a name and the system prompt. That's great.

Now let's see how we execute our agent. I'm going to go in here. You can see I'm executing it using the Foundry Agent Service SDK. But what's also important to know is that we're using Foundry agents today, but we could use any kind of agent including CrewAI, LangChain, or even multiple agents using agent-to-agent or 2A2 protocol. Pretty cool stuff. This is awesome but now let's see our agents in action. Amanda, do you want to show them what we built?

AMANDA FOSTER: Yes. Now let's dive into the application. Build events is a multi-agent event planner application powered by a voice-controlled agentic orchestrator, which will delegate tasks or tool calls, as we call them, to our subagents we built using the Foundry Agent Service. Together we just built our RSVP agent but now let me show you the power of multiple agents working together in one experience.

DEMO: Hello! How can I assist you with your event planning needs today?

AMANDA FOSTER: Hi, I'd like help planning a hackathon in San Francisco on May 22nd, 2025. My budget is \$15,000 and I'm expecting around 200 attendees. The event will be from 5:00 to 10:00 p.m. Pacific Time. Could you please help me find an event venue?

DEMO: I'm searching for a venue in San Francisco to host your – Sure, I can assist with that. Would you like help with creating invitations or setting up an RSVP system for the hackathon?

AMANDA FOSTER: Our first agent is now hard at work, which will help me find an event venue.

DEMO: Here are some venue options in San Francisco for your hackathon. One, the Fillmore. Capacity up to a thousand guests.

AMANDA FOSTER: Hey, this looks great. Could we go with the Fillmore? Could you now create the calendar event and send out my invites to my invite list?

DEMO: I'm in the process of creating the calendar event for the hackathon at the Fillmore and sending out the invites to your existing invite list. I'll update you once it's done. Is there anything you'd like me to arrange for the hackathon? The hackathon event?

AMANDA FOSTER: Yes. Now I'd like your help generating an image to promote my event on LinkedIn, as well as drafting some content. I'd like you to make the image convey excitement, but other than that, take full creative liberty on both tasks.

DEMO: I'm working on generating an exciting image and drafting content to promote your hackathon on LinkedIn. I'll have them ready shortly. Is there anything specific you'd like to include in the post?

AMANDA FOSTER: As you can see now, two Foundry agents have been kicked off at the same time working simultaneously to help me get my LinkedIn post ready.

DEMO: (Inaudible/crosstalk.)

AMANDA FOSTER: Here is the draft we have for the hackathon.

DEMO: I'll have everything ready for you to approve the draft now.

AMANDA FOSTER: Here is the image. Hi. This looks great. Could you now post both the image and the draft to LinkedIn?

DEMO: I'm posting the image and draft to LinkedIn now. I'll let you know once it's live.

AMANDA FOSTER: Now, our final agent has been kicked off and it will give the response back and tell us the status of our posts. Moment of truth.

ELIJAH STRAIGHT: Getting it there. This is a live demo.

AMANDA FOSTER: Yes, everything is happening live. That is why.

DEMO: The LinkedIn post has been successfully published.

(Applause.)

AMANDA FOSTER: OK, now let's go to LinkedIn. Again, this is live, so we're making sure everything went through. As you can see, the post is now live. Check out our Build Events by Contoso page after this if you want to see my practice. Now, Elijah's going to show us how we ensure our agents are always doing as we program them to do.

ELIJAH STRAIGHT: Amanda, that was great. You guys are all invited to our hackathon coming up. Now that we know our agents are up and running, it's also crucial to make sure that they're delivering high-quality results. In order to do that, we've integrated the Azure AI evaluation SDK directly into our CI/CD pipeline with GitHub Actions. What's actually even more exciting is using GitHub models, which Jessica presented earlier, we generated a summary of all the evals that we're running here. We can see that right here, which enhances our overall DevOps experience.

Then you can see here are our agents that we're evaluating as well as operational metrics doing pretty well, and then some AI quality metrics. It looks like, Amanda, we have got to do some work to improve the AI quality. Today, you've seen how to create, use and evaluate agents with the Azure AI Agent Foundry service. Amanda and I actually have a breakout session today at 4:30, where we'll be taking an even deeper dive on the app you saw here today. We're super excited to see what you build. With that, Jay, we'll kick it back to you. Thank you, guys.

JAY PARIKH: Thank you, Amanda. Thank you, Elijah. Love the capabilities you are building for developers. Let's fast forward just a couple of days and pretend that this hackathon happened and it went really well, but now I want to dig into the data about the event. I want to bring on Patrick to show us how.

PATRICK LEBLANC: What's up Jay.

JAY PARIKH: How are you?

PATRICK LEBLANC: How's it going? All right. The event team, they want to search. They want to do a better search on their app. They came to me, and they were like, "Patrick, we want to look for demo presenters that are good at live demos like us, that are good with GenAI and data." I'm like, "All right, I could do that. Let me see what you got." They showed me this app, and they said, "Oh, by the way, we're only using the vector search and a keyword search." I thought wow, like 1900s again. Let's switch this up.

JAY PARIKH: Maybe we can do better.

PATRICK LEBLANC: We can do better. There's a new hope. We can use semantic operators in PostgreSQL. Now, I'm going to amp this search up a little bit, and I click search. What you're going to see is that it narrows it down to some very specific speakers.

JAY PARIKH: Including myself, Jay.

PATRICK LEBLANC: Including myself. All right. But you may be thinking, hey, how? How did you do this? Enough of all this talking. Let's head over to VS Code. We're going to head over to VS Code, and I'm going to introduce you to these semantic operators. The first one I'm going to show you is the AI generate. Basically, I'm going to pass the events in, and it creates a description for all my events. I don't have to do anything. It just works for me.

It's easy, but let's take a look at a query that resembles or reflects what we did behind the scenes. The first part of our query, we did go back to the 1900s. We used the vector search and a keyword search to handle the GenAI and the data part, but then we use the semantic join down here in our view, to handle the live and the good part. Then you can see right here it returned the exact same speakers you saw on that page.

JAY PARIKH: That's excellent.

PATRICK LEBLANC: It's excellent, all right. You heard Amanda and Elijah say that "hey, we're storing all of our thread data in Cosmos DB running in Fabric," which means they keep all their data in a OneLake. The OneLake is amazing, so we decided that we're going to replicate it. We're going to mirror our Postgres SQL database over into Fabric, which puts our data in the OneLake, and now we can combine these two different sources into a single query to create a semantic model. Watch this.

Let me show you something else. We're going to create this really beautiful report right here. There we go. Create this really beautiful report that they can use so for example, they can look at their hackathon in San Francisco to see all the data, and they can gain some insight and how great that event was, but they can also take a look at this and possibly make this event even better.

JAY PARIKH: That is awesome. Thank you for sharing those insights.

PATRICK LEBLANC: Yeah. Do you think we can get should we get this in the app?

JAY PARIKH: Yeah. You want to, Copilot?

PATRICK LEBLANC: Hey, Copilot, do what you do. Let's create that easy, Jay. All right. Thank you everyone. Have a great Build!

(Applause.)

JAY PARIKH: Lots of customers are already firing up their agent factories, as you can see, Carvana actually used Foundry to build an agent to analyze all of their customer interactions. And from there, the insights that they gained, they were able to make a 40% reduction in inbound calls for sale.

Now, let's hear from Nasdaq and what they're building.

(Nasdaq video segment.)

(Applause.)

JAY PARIKH: Now, these new capabilities are great, but adopting these in the enterprise is going to be limited if we can't trust it and we can't secure it. Foundry offers these capabilities around trust and security baked in from the start. It's a core principle for how we're building Foundry.

Now, we're also integrating Microsoft Security, products like Entra, Purview and Defender, right into Foundry. As you're building agent, you're going to be able to secure it.

To show us how all of this works, please welcome Mehrnoosh.

(Applause.)

MEHRNOOSH SAMEKI: Hello, everyone. I am tasked with ensuring that my AI agent is safe and secure. There's a lot that I need to do from identifying and measuring for the right risk, putting the right control in place, and perform ongoing monitoring for any new threats and issues. Luckily, Foundry enables me to do all of that.

Today, I'm just going to show you a small part of three steps, measure, protect and monitor. Now, I know that once my AI application goes out there, bad actors will try to misuse it. Here on my computer, I'm going to use the AI red teaming agent to adversarial-ly test my AI application against such attacks before I ship it.

As you can see, I have the right risk categories and attack strategies set up. And now my AI red teaming agent is attempting to attack my system. Let's take a look at some results from an earlier run.

Here, I can see that 42% of attacks have gone through. Double-clicking on that, I can see that my model is getting confused by some jailbreak attempts. I need to activate additional guardrails in order to protect against these.

Luckily, Foundry enables me with agent, data and model guardrails. I can create a content filter, attach my prompt shield and activate it to protect me against direct and indirect jailbreak attempts coming through grounding data, agents and user interface. I set my output filters here, attach it to the endpoint that I'm using in this agent. And just like that, I have my guardrail set up.

Now, it's time to monitor against new threats to my AI system.

I can see the new threats against my AI system in this view. As you can see, several jailbreak attempts have happened against my AI system, but the great news is that prompt shield has already blocked them for me.

I would like to double-click on that and understand the source of the issue. For that, I head to Defender.

Defender is a product for security operation to fully investigate security incidents. I can see the whole incident here, including where it is coming from. The attacker appears to be part of a known attack group. I think it's safe to remove this agent's access to resources, just in case it's compromised.

I go to Entra for that. Using Entra's Agent ID, I can track and manage all my agents.

Here is the compromised one. Let's remove its access from resources until I address the issue.

Now, my system is secure again.

As you can see, Foundry plus Microsoft Security enables you to identify and measure for new risks, put the right controls in place and perform ongoing monitoring post production. Thank you.

(Applause.)

JAY PARIKH: All right, great. Thank you, Mehrnoosh. Here are some customers that are seeing significant benefits of building with Foundry, because security and trust is baked in.

Using Foundry, Heineken has implemented regular evaluations and personality checks to safeguard its chatbot from violating its business value.

Let's hear from Accenture on how they're using Foundry.

(Accenture video segment.)

(Applause.)

JAY PARIKH: OK, so we've seen our tools. We've seen our agent factory, and it's also protected with industry leading trust and security. But we shouldn't be restricted to just deploying in the cloud. We don't want two different platforms, one for the cloud, and a different one from the edge. That would be a real pain.

We're building Foundry so that we can extend it from the cloud all the way to the edge. Now, this is really cool, so we should just jump right into the demo. Please welcome Seth.

(Applause.)

SETH JUAREZ: How's it going? I'm so excited to be here. Everyone should know that all those demos are live. We're not faking any of this stuff.

JAY PARIKH: They're all live.

SETH JUAREZ: Except for this. Just kidding. Let's go ahead and let me show you Foundry models, but locally.

Here's a number of models that are available to me directly inside. I'm just running the command because that's cool. But the thing is, there's a bunch of models that you can download and use locally. And that's the thing, because when you're moving from cloud to edge and you want to have some of this benefit, you can have it directly on your actual machine, which – can you not hear me?

Oh, now I can hear me. Thank you. I could have just spoken to this. Would that have been awkward?

JAY PARIKH: Maybe. (Laughter.)

SETH JUAREZ: OK, so let me go ahead. (Laughter.) Let me go ahead and just say, write me an app. I think you asked me to do that.

JAY PARIKH: I did. It was homework.

SETH JUAREZ: Yeah.

JAY PARIKH: We'll grade later.

SETH JUAREZ: I'm going to get it done as soon as I can, and I'll get it to you in an email.

JAY PARIKH: I can see you're working hard.

SETH JUAREZ: Well, yes, I'm typing it up right now. Notice that this is working locally, directly on my machine, and you can see that it's hitting my GPU.

JAY PARIKH: Excellent.

SETH JUAREZ: Yeah, which is really – wait, I did the wrong GPU. It's this one. Sorry, you're like, that's not the right one. (Laughter.) That's the one. But the thing about this is that this is going to tie into something that you heard about yesterday, the Windows MCP registry. And now imagine what your operating system might be able to do for you later on.

JAY PARIKH: That's an interesting idea.

SETH JUAREZ: That's my demo. Thank you so much, my friend.

JAY PARIKH: Thank you. Can you hang out for a second, just one more second?

SETH JUAREZ: OK.

JAY PARIKH: OK.

SETH JUAREZ: We'll do that.

JAY PARIKH: All right. Thank you, Seth. That's really cool stuff.

Now, we have seen the AI-powered tools that we have. We've also talked about our agent factory. We're securing this whole thing with the industry leading security and safety.

Now, we've also seen how we can take it from the edge to the cloud as well, but we want to keep moving here. And we want to keep exploring the frontier of what we can do with our tools and our platform.

Before we go, we want to share one more thing. We're teaming up with the MSR team to really explore what could be possible to build and to incorporate into our tools and our platform. Now, these are not new products. We don't have dates in terms of when we're shipping these things, so we have a lot to figure out still, but we want to give you a sense of what capabilities we're working on.

Seth, let's show them what we've got.

SETH JUAREZ: Let's do it, but we're going to make it awkward here. OK, this is Project Amelie, and for those that don't know – yeah, we'll hang onto that. (Laughter.)

For those that don't know, I used to be a data scientist. I'm not just this beautifully coiffed beard. I used to be a data scientist. And what happened is there was a lot of work that we needed to do as data scientists, when we did an AI thing, or machine learning. That's what we called it before it was cool.

Imagine you're asked to build a model, like predict accommodation rental prices in Seattle. What you would have to do is you would have to get some data. You'd have to analyze it a little bit. Again, I think here's some homework here that I was maybe supposed to do for you. We can upload the data set.

And what you would do as a data scientist is you would look at every row of these data sets. You would try to figure out if the column was right, sometimes someone –

JAY PARIKH: Maybe.

SETH JUAREZ: – it was probably one of you – would put a string dollar where there should have been a number. Do you know what I’m saying? Some of you would actually do this, very frustrating.

Notice here that it’s already done some analysis, and what I can do is I can start to ask it things as if it were a mini-data scientist in a box. And notice that it’s trying to look at this information, and it’s doing some of this stuff. It’s still accessing this. This doesn’t take – this is not a five-minute demo. It takes about an hour. But for a data scientist, that’s like a whole day and a half.

Let me show you something that’s already done. This thing is already done. You can see it goes through – oh!

JAY PARIKH: Oh, there’s the error.

SETH JUAREZ: You want to know what error it was?

JAY PARIKH: I do. Hmm.

SETH JUAREZ: Hmm. Oh, the dollar sign. Who did that? (Laughter.) OK, but notice that it was able to iterate and finally get to the actual model – it self-corrected itself – the actual code and symmetric, because it’s using two different models, one for reasoning and one for generating code, so that you can actually use this actual code to get started as a data scientist. How cool is that?

(Applause.)

JAY PARIKH: Pretty awesome.

SETH JUAREZ: Yeah, but there’s more. There’s an OpenAI benchmark that they put up. This RD agent together with MSR is actually at the top of the leaderboard.

JAY PARIKH: Top of the leaderboard.

SETH JUAREZ: That’s right. Thank you so much. This is awesome.

JAY PARIKH: That’s awesome. Hey, do you think we have time for one more thing?

SETH JUAREZ: Well, I’m not dressed like the fun director at a cruise for nothing.

JAY PARIKH: Right, right.

SETH JUAREZ: Let’s do one more thing.

JAY PARIKH: OK.

SETH JUAREZ: Let's do one more thing. It turns out that I went and read a paper this weekend on something called Graph RAG, and Microsoft Research has done some amazing things with this.

JAY PARIKH: It is.

SETH JUAREZ: Graph RAG is a situation where instead of just taking chunks of text and putting it into vector and searching, it actually takes the chunks of text and uses the LLMs to extract entities and relationships, and it makes a graph so that other LLMs can query it. And the SQL Server team actually did this with their code.

Let's take a look at this thing. They weren't going to let me look at their code here in person. (Laughter.)

Here is an error log of something that actually happened to one of their engineers. Now, C++ is super scary to me, even to this day because of pointers and such.

JAY PARIKH: Yeah, me, too.

SETH JUAREZ: Yeah, I know. Looking at this thing is stressful, but now imagine being able to talk to this new data structure, figure out where the errors actually live, understand the actual code and maybe even get to a fix.

JAY PARIKH: That's pretty amazing.

SETH JUAREZ: Yeah. And for example, whenever we get into a new organization like you are, you can literally talk to the knowledge of a senior engineer.

JAY PARIKH: And how big is this graph?

SETH JUAREZ: This graph has about 14 million nodes and 24 million edges. It really is –

JAY PARIKH: A big code base.

SETH JUAREZ: It is a big code base. And there you see, it's literally fixing the actual code of our actual production thing. That's why I brought the video. We're not going to give Seth access to code. That's it. Isn't that great?

JAY PARIKH: That is awesome.

SETH JUAREZ: Yeah.

JAY PARIKH: So should we do one more thing? What do you think, audience?

(Applause.)

JAY PARIKH: I mean –

SETH JUAREZ: I love being on stage.

JAY PARIKH: OK. (Laughter.)

SETH JUAREZ: Do you all remember that –

JAY PARIKH: Do you want to – we should do something more fun.

SETH JUAREZ: More fun?

JAY PARIKH: More fun, yeah.

SETH JUAREZ: OK. Do you remember this game? Let's play the video. Do you remember this game here? Let's play. Oh, get it up here. Anyone remember this game? (Laughter.)

Yeah, I used to play it all the time myself.

JAY PARIKH: Me, too.

SETH JUAREZ: One of the things, though, you couldn't do something.

JAY PARIKH: You couldn't do two things.

SETH JUAREZ: And in fact, if we go to my – if we go to the code, they actually released the code, and there's a love letter here to all of the programmers.

JAY PARIKH: Do you guys remember this?

SETH JUAREZ: He said, you could add some game features like weapons or jumping. Someone used the exact same in Graph RAG, indexed this code and then made the coding agent, they said, hey, can you make it jump?

And you can see here – (laughter). Someone said, people tell me that all the time, and they can't even make me do it, right?

JAY PARIKH: (Laughter.) Right.

SETH JUAREZ: You can see that it actually created this capability for real-zies. I don't know "for real-zies" on purpose on stage, but for real-zies. You can see it actually did it. And now if we can roll the second one, we're going to actually going to see the –

JAY PARIKH: Look at that!

SETH JUAREZ: Look at that! Come on, how cool is that?

(Applause.)

SETH JUAREZ: There you have it, my friend.

JAY PARIKH: All right. Thank you, Seth. That's pretty cool, Seth. Thank you for that.

Now, we want to wrap up here. And as builders, we know how important these things are, our tools and our platform. They matter for perfecting our craft. They matter for shaping the culture of our teams, and they also matter for shaping and realizing our ambitions.

When I think about the opportunity here, the thing that gets me most excited is the idea of breaking free from the constraints of the old ways of building and operating. To break out of this single paned, linear, deterministic view of what's possible and into a multiverse where you can build and operate in multiple dimensions simultaneously, and to do so in parallel with effectively, a limitless number of agents acting on your behalf.

There's a lot of work to do here, still, to realize this vision, and it's something that we'll need to do together and in the open to make AI superpowers ubiquitous. But if we can do it, we can create a world where we are no longer constrained by the number of developers you can hire and the number of hours in the day. It's a world where only constraints are imagination and our drive and maybe looking for a few GPUs, but that's still a world worth building. Let's go build it.

Thank you, and please welcome Jatinder.

(Applause.)

JATINDER MANN: Thank you, Jay. Windows AI Foundry is our next step forward, building on the foundation of Azure AI Foundry and Windows Copilot runtime to bring the full power of AI to the client. It gives you a consistent system for running, customizing and building AI models directly on Windows at scale. You get access to a growing catalogue of models, tools to customize them with your own data, APIs for natural language and vision, and the support you need to run custom models consistently, regardless of the underlying silicon.

The growth in available models over the past year has been exceptional. Today, you can explore a wide range of local models, tune for Windows and source from the AI Toolkit extension, as you can see here, or pulling directly from leading repositories. These span CPU, GPU and, of course, NPU unlocking powerful SLMs across the incredibly broad Windows install base, but it's not just about the number of models, it's about the quality.

Just a year ago, small language models were good at summarization and basic chat. Today, those same models, running entirely on your device, can solve math problems, reason through multistep tasks and power intelligent agents with real contextual understanding. That capability used to require cloud scale compute. Now, it runs locally on a Windows PC.

Take our own Phi models as an example. We went from Phi-3 to Phi-4 in months, and just last week, we introduced the Phi-4 reasoning family. These models are compact enough for the edge, but smart enough to rival the kinds of experiences you'd expect from GPT-4.0.

But this advancement also creates more complexity for developers. You're now managing more models across more device types under tighter performance constraints. That's exactly why we built Foundry Local into Windows AI Foundry. It gives you a Model as a Service experience to access, run and integrate high quality models optimized for real Windows hardware.

Let me show you what this looks like.

First, let's see what models are available in the Windows AI Foundry catalogue. I have my command line open, and I'm going to type, Foundry model list, like Seth had shown before. And this gives me a list of models optimized for this Windows PC, across CPU, GPU and MPU, like Mistral, Phi and even DeepSeek.

I can also see which models are already downloaded, or I can run one by typing, Foundry model run, the name of the model, Phi-4 mini reasoning, and everything here is running locally on an Nvidia RTX GPU.

Let me start by interacting with a model. I'm going to ask it a riddle to test its reasoning abilities. And you can see this is running really fast.

Now that I'm happy with the Phi-4 model, I want to integrate it into my electron agent app. And all I have to do is change this REST endpoint from the Azure Cloud to a local host, and I have a fully local AI-powered chat experience. Now, I have a new agent app powered by Phi-4 running locally, and let's ask it the same question.

Now that you've seen how simple it is to access and run models with Foundry Local, let's talk about the next step, optimizing those models to suit your data, your domain and your users.

Just like with Azure AI Foundry Windows, AI Foundry gives you a flexible path to tailor model behavior using two proven approaches, parameter efficient finetuning with LoRA and retrieval augmented generation using semantic search and vector indexing. Let's start with LoRA.

You finetune a base model, like Phi Silica, by training a compact adapter layer on your private data. This finetuning runs in your Azure environment and outputs a lightweight adapter file that can be deployed with a base model on the client. At inference time, the adapter runs in parallel, improving output quality without modifying base weights or introducing latency overhead. And the whole process, from data prep to job submission, is integrated into the AI Toolkit for VS Code using familiar tools like Bicep and Azure ML.

Let's use the Phi Silica model with LoRA to build an education agent app using my own training data. I'm going to start here in the AI Toolkit for VS Code. And because I want to finetune the Phi Silica model with my own custom data, I'm going to start with the finetuning tool right here.

Let's start the project. Step one, let's give it a name. Let's call it Phi Silica LoRA. Then we have to select a model from the model catalogue. I'm going to configure the project. And next, let's select the training data, the test data, and the rest of the hyper parameters look good. I'm going to generate the project.

This is going to contain two key resources, my training data and a Bicep file used to deploy my training job in Azure. The training data consists of many samples of how I'd like the model to respond. It contains sample input prompts and output, and LoRA is then going to learn from this data. The Bicep file contains my Azure subscription information, so my data can run, my job can run privately with my data.

Let's start a new finetuning job. Let's call it job one. Let's select the Azure subscription. There we go, and finally, a resource group.

And now you can see that the job is provisioning. And if you look closely at the console, you can see that the job has already started to run in Azure. And because training takes time, I already trained LoRA for the purposes of this demo.

Let me open up the AI dev gallery. This is an interactive sample app to help developers integrate AI into their Windows apps. And I've already selected my downloaded LoRA adapter, and I've generated the response. Now, you can see the difference in response output, both with the LoRA adapter and without the LoRA adapter. And in order to integrate LoRA into your app, all you have to do is copy these few lines of code.

(Applause.)

In other cases, you don't need to finetune. You just need the model to reference the right knowledge at the right time. That's where retrieval augmented generation, or RAG, comes in. Windows AI Foundry includes a full semantic indexing and search pipeline similar to what you use in Azure. You can convert structured or unstructured content, like documentation, FAQs or support tickets into an embedding based vector store that the model can query during inference. This gives you up to date answers tailored to your domain without retraining.

Now, I'm going to use Filmora, a video editing app native to Windows that uses the Windows semantic search and knowledge retrieval APIs locally on a Copilot+ PC. Let me show you how this works.

Here, I have the Filmora app pre-loaded with a video clip that I want to edit that I'm playing right now. I'm going to go to the Filmora AI Mate, and I'm going to ask it to make this video look vintage from the 1950s.

And thanks to the knowledge retrieval API, Filmora AI Mate creates a knowledge base of all available assets, like effects, filter and music. The semantic search API helps AI Mate provide contextual suggestions, making it easy to achieve my desired outcome. Not only can I see the type of treatment I can apply to this video, I can also see the top assets that are most relevant to the vintage outcome I just described.

I really like these last two filters. I'm going to select the film damaged one, and let's see what this looks like.

I was able to create an amazing video without specialized knowledge, thanks to the Filmora app, their AI Mate agent, and all while running locally on a Copilot+ PC.

Everything we've shown so far assumes you're working with preoptimized models, but when you bring your own custom model, things get more complex. Windows ML changes that. It's a unified execution layer that simplifies custom model deployment across Windows. It handles hardware mapping, power aware execution and fall back, so your model just runs with no need to tailor it for individual devices.

This is only possible because of the breakthroughs from our silicon partners, AMD, Intel, NVIDIA and Qualcomm, who have each pushed AI performance on their platforms and are now coming together to make those capabilities accessible at scale through Windows ML.

Next, let's hear directly from them some of our developer partners who have a private preview and share what their experience has been with Windows ML. And after that, we will welcome Charles Lamanna to the stage.

Thank you.

(Applause.)

(Video segment.)

END