

Microsoft Frontier Tour – London, UK

Keynote

Satya Nadella – CEO, Microsoft

Wednesday, October 21, 2026

(Applause.)

SATYA NADELLA: Good morning!

(Applause.)

Good morning. It's fantastic to be back in London and in the UK, and especially at a time like this, where there's so much change in the technological frontier. The idea was to spend some time talking about what this means for the broader ecosystem. How do we, as individuals and organizations, move that frontier where we are able to see the progress locally for us ourselves, professionally, and in the firms we work in?

When I think about what keeps us grounded every day at Microsoft is our mission, mission to empower every person and every organization on the planet to achieve more, and especially talking about this frontier ecosystem that really manifests in, I'll say, three pretty distinct pillars that we have to go tackle collectively.

The first is we have to recapture that sense of empowerment individually, because this is very powerful technology. In fact, it can help take out some of the mechanized parts of the jobs that we do today, but one of the challenges is how do you design these AI systems, these agentic systems, as they become more and more capable, such that you are feeling that sense of empowerment? You still have agency. You have control. It's your ambition that's driving these systems, as opposed to the other way around.

And it's a pretty complex design challenge. How do you put the human in the loop? How do you put the human on the loop? How do you really ensure, ultimately, that sense of empowerment is not lost?

The second challenge is what I describe as the reverse information paradox that comes with the network effects of intelligence. This is the first time you have a digital system, which is a learning system. You deploy it. You use it. You even continuously start leaking value to the learning system.

And so, the goal has to be that as organizations, you are able to capture that knowledge and keep it locally for yourself. In other words, you have to own the knowledge that gets created, both with the human capital that you have as well as the token capital that you generate. And that's the second real challenge.

And the third is a challenge of trust, because if you look at these systems, one of the fascinating things is the scaling laws continue to work, and the AI models come out with more and more capability that is emergent, except we do have a challenge that alignment seems elusive. The mechanistic understanding of these models seems elusive. And there are people working hard on the science there. And I'm positive that we'll make progress, but as we speak here today, scaling laws are producing even more powerful models that are less and less trustworthy.

That introduces this other challenge, where I describe it as you need to treat, in fact, these models as new forms of insider risk, and really create a zero trust architecture around it, in fact, avoid nested black boxes so that you can have real trust in the systems that you're deploying, because ultimately, we, individually, have to be accountable for the outcomes of these systems.

These are the three foundational things that we are trying to tackle as a platform provider at Microsoft, and that's what we are building out all of these pieces. And I wanted to unpack each one.

The first is Copilot. We're designing Copilot with that goal of empowering people. The application itself, or the system itself, has gotten a lot more sophisticated than even the last time I was here. We now have multiple form factors that all are coming together. You have Chat. You have Cowork. You have Code. You have Autopilot. These are all the various ergonomics of how you want to get task done.

And then you have all the models available to you. It's not about any one model. You have all the form factors and all the models that then go to work on the task space, which is any task you need to get done, it's just about being able to do the match between the ergonomics of the form factor, the models that are available, and the task that needs to be done. And that's the goal of building out Copilot.

And of course, you want to be able to use the cloud, but all tokens coming out from the local PC, even. The capability now is such that you are going to have the ability to run fairly good models locally. And so, the ability to be able to blend both the cloud and the edge is something that we really want to build into the system.

And so, we conceptualize this as essentially a new type of an operating system for work. And so, that's what Copilot is all about.

To just give you a flavor for how we're building it, what you can do with it today, I wanted to invite up on stage, Efe, one of my colleagues. Efe, take it over.

(Applause.)

EFE ABUGO: Hey, everyone. My name is Efe Abugo, and I am a Senior Product Marketing Manager for Microsoft Copilot. And today, I'm going to show you all how we can use Copilot to pull together a product launch.

Now, in today's example, our leadership team has a critical decision that they need to make by the end of the week. For the launch of their new product, Alervio, should they go with their first initial five markets? Should they expand regionally, or should they go globally? And to tackle that today, we are going to use the new Microsoft Copilot app, my starting point for the day.

Here in the app, we are able to get answers to quick questions that we have in Chat, or we're able to delegate advanced tasks in Cowork. And if I come up here, if I want to build an interactive app or dashboard, I can use Code to do that. And in Autopilot, I can set this up to run tasks in the background and complete work on my behalf.

Now, we're going to actually start here in Cowork. And with Cowork I have model choice, and that means that I can pick from the latest OpenAI and Anthropic models to complete the work I need to get done. But for this situation today, we're going to select Auto, which lets Microsoft pick the right model for me.

Now, I'm going to ask Cowork to use the leadership briefing doc skill that I've created to put together a briefing for our leadership team, and once that's ready, to draft an email that I can send to my colleague, Jason, our manufacturing lead. We can see here that Cowork has gotten to work. It's using Work IQ to pull in that relevant context, and it's using that skill to build the Word document in our predefined format.

Here we have the email to Jason drafted, and all I need to do is hit Send, and we're good to go.

Now, because Office is built directly into Cowork, I can open the Word document right here. And something I want to call your attention to is the document automatically inherited the highly confidential sensitivity label based upon the highly confidential meeting that we pulled the data from.

And additionally, I can see my teammate, Jason, is already in the document and he's ready to collaborate. That's a good thing because there's actually one bit of information that Cowork does not know, and that is what capacity can we bring online in Q3.

I'm actually going to tag Jason in the document here, and I'm going to ask him to fill in that information on what the plant can bring in in Q3. And I can see that Jason has gone to work, and he's right now put in 25,000 units as an early estimate.

All right, that is great. The Word document is looking good.

Now, in my experience, leadership is not going to make a decision based upon just the document. They're going to want the opportunity to be able to interact with the data, to explore the data, and really dig in. And that is where Code comes in.

Now, I'm not a developer myself, but I can use plain language to tell Code the application I want to build, and Code's going to get me there. I'm going to ask Code here to build an interactive launch planner for our product. And I want it to pull in our market forecast context from that

planning meeting that we had, as well as other business context. And we're going to go ahead and send the prompt.

Now, this does take a little bit of time, so I'm going to jump to the same exact prompt that I ran earlier and walk you all through a few things.

First, when we're building with Code, it's an interactive iterative process. As I scroll down here in the chat, we can see I built the first iteration. And I looked at it and I said, hey, let's actually add in a slider that our leadership can interact with to see how much Q3 capacity that we can bring online.

And as we come down further, there are actually some risks that were surfaced in an email from one of my colleagues. And I want to pull those directly into the application as well.

Let's open it up. I'm going to full screen it here, and here we have the app that Code built. And right here, I can see that first option for Wave 1 going in those initial five markets, or what happens if we expand regionally? And I can see the numbers changing within this app. And we also have that view if we want to go global with this product launch.

And just as I requested in the chat, we have this interactive slider where I can pressure test different what-if scenarios, based upon how much capacity we're actually able to bring online. And if we scroll down here, we can see that those top three risks were brought in from that email.

Right from here, I can publish the app, I can share it, and I'm going to share it with my colleague, Jason, making this app immediately available for the team to collaborate in.

Now, you watched me build a briefing document in Cowork. You watched me just build an app in Code, but there's one more critical dependency that we need to get across the finish line. And that is I need Jason to bring in the updated capacity numbers.

We can see here in this Teams channel, I asked him to send them by 5p.m. and he promised me that he would. Now, typically, I would need to keep on coming back into Teams. I would need to check and see if he's actually sent that content through, but I'm going to delegate that follow through to Puck, my Autopilot.

Now, Puck has its own identity, its own memory, and its own workspace. And as I'm about to show you, it even has its own place within our org chart. As you can see right here, Puck, my Autopilot, reports to me.

Now, before we move forward, I want to call out something interesting. I'm not just going to use my Autopilot to check in and remind Jason. I'm actually giving it responsibility to drive a business-critical artifact through to completion.

I'm going to come over and I'm going to select my chat with Puck. And I'm going to tell it, hey, I want you to set up a routine where you're monitoring that leadership channel, and if Jason

doesn't get those slides to me by 5p.m., I want you to check in, and once you do get them, to draft an email to my colleague, Wanda.

Puck found the conversation. It proposes a routine, and that looks good to me. I'm going to go ahead and hit Confirm. And then what Puck is going to do here is it's going to confirm that that routine has actually been set.

Let me go back to that leadership review channel. Five o'clock has come and it's gone, and Jason has not sent the slides over. But we see here, Puck has followed up on my behalf, asking Jason to send those slides over.

And here we have it. Jason just sent them. We're good to go.

I'm going to close out here with Puck, and it looks like it's drafted that email to Wanda just as I requested. And it's included a link to the deck as well as a summary of the content in the deck, in case Wanda just wants to look at that before drilling all the way in.

As you just saw, you can go from an idea to a plan or a brief with Cowork. You can turn that brief or that plan into an app with Code. And you can set up your Autopilot to run in the background and complete work on your behalf, enabling you and your organization to achieve more.

Back to you, Satya.

(Applause.)

SATYA NADELLA: Thank you, Efe. Thank you very much. And what you saw there is Copilot with all its form factors and all the models it has, really helping you with your task.

And one of the things that Efe mentioned was how the Auto mode in Copilot is really bringing about that matching of the models to the task, to the form factor. And speaking of that Auto mode, one of the other places where we are really making great progress is in coding. In fact, with GitHub Copilot, we now have HydraFusion. Think of it as a learned model where you are now able to take advantage of even these powerful local models.

On an RTX Spark, for example, you now have very capable coding models, especially something like Code in Copilot or even GitHub Copilot. You can start using with HydraFusion whenever the task or one of the subagents of your task can be done and satisfied by just a local model. You can just have those tokens from essentially what's an infinite token factory locally. It's fantastic to see these learned models become the new Auto model.

Now, we have great usage of these systems across the UK. We have AstraZeneca in pharma. We have Barclays and Lloyd's in financial services, Marks & Spencer and Sainsbury in retail.

This morning, though, I had a chance to meet with folks at NHS, who have deployed Copilot to over 500,000 people. And of course, this is about really, if you look at healthcare, one of the

foundational challenges of healthcare is all around workflow, around care management. How do you connect the dots in a way that people can come together to improve patient outcomes and reduce waste and cost? And it's great to see that.

In fact, one of the applications, because it's not just the use of Copilot, it's using Copilot Studio to build agents that interact with Copilot and really create that workflow automation and connecting of the dots. And one of the places we've applied this is in ambulance services, again, really helping to be able to get the right services to the patient.

Let's roll the video to give you a flavor for it.

(NHS video segment.)

SATYA NADELLA: It's great to see that sense of empowerment that people at NHS have, and really, meaning in their work and the patient care that they're able to provide because of these tools.

The next part that I want to talk about is we have taken the entirety of the agentic system that was used to build Copilot, and we are now making that available for every organization, every developer to be able to use to build their own sophisticated agentic systems, agentic systems, let's say, in supply chain or marketing or finance across industry, business processes.

That's what we describe as Microsoft IQ. It has a couple of different parts to it.

The first is you really want to have the action space all safely registered with the agentic system. When we talk about the registry, that means all of the connectors, skills, plugins that you can use in the agentic system are all essentially vetted and available in a safe registry.

Then the next piece is the rich context and memory layer. This is where you are able to structure everything from what's underneath, for example, Microsoft 365. After all, underneath Microsoft 365 is very rich data. It's people, their relationships to other people, their documents, their files, their projects, their communications data in email and in Teams. All of that can come together now as Work IQ.

You also have the Web IQ, so this is everything the world knows. You also have, for example, all the dashboards you may have built in Power BI. Underneath all of those dashboards are very rich semantic models. You also have document knowledge bases. That's in Foundry IQ. You bring that entirety of context as well as the memory that is getting built continuously in the agentic system.

And then the last part is you want to have that multi-model system. Just like in Copilot, we were able to use auto routing with multiple models, you want to be able to do that in any application or agentic system you build. And the idea there is to have all the models available, and the key is to separate out the harness from the models, so that you can keep the action space as well as the context and memory outside of a model.

In fact, for me, the simple acid test is can you remove any model and yet still still climb on any eval of yours for your agentic system? If not, that means you now have a lock-in of a given model, and that could be problematic. What if the model went away? What if your memories got embedded in the weight of a model that you don't control? All of those can happen.

And so, that's why having an architectural approach where the separations are enforced, I think, is going to be one of the most critical considerations to deploying sophisticated agentic systems.

Inside of Foundry, we have lots of models, 11,000+ models. In fact, one of the exciting new models is this Decision-1 model, and the reason I'm excited about it is it makes everything much more token efficient, the idea that you can now have a classifier on whether it's tool calling, or for example, if you take that router, which model to call for what task is a decision. And so, you can now have a fast, quick, low latency model that is allowing you to pick the right model and make your routing that much more effective and efficient.

And the show you all of this in action and help build a business process agentic system using Microsoft IQ, I want to invite up on stage Marlene and to show you the demo. Marlene, take it away.

(Applause.)

MARLENE MHANGAMI: Thanks, Satya. And hi, everyone.

Let's say I lead a small engineering team at a company called Tailspin Toys. My team works asynchronously. And so, sometimes this means work gets missed. A blocker sits for a day, or a new hire is too shy to ask twice. I'd like for Copilot to help me build an always-on agent to help my team with their business purposes. And to do that, I'm going to be using the GitHub Copilot app along with Microsoft Foundry.

You can see on the screen I already have the GitHub Copilot app open. And to the right here, I have a Microsoft Foundry Canvas open. A canvas is an interface I can use to collaborate with GitHub Copilots to build agents. And in this specific case, it's for building hosted agents in Foundry.

You can see in my prompt here, I'm asking Copilot to help me build a Foundry Autopilot called Mona. Mona will be an always-on business purchases assistant that lives in Teams, in Outlook and remembers everything my team is working on, so hopefully nothing gets missed. Let's go ahead and click on Create with Copilot to get us started with the building.

You can see that my prompt is going to be sent to the chat, along with extra information from Foundry for Copilot to integrate into my agent. We can also see that it evoked the Microsoft Foundry skill, and this will give Copilot the context that it needs to be able to use best practices when building out a Foundry agent.

I mentioned earlier that my agent, Mona, will be a Foundry Autopilot. And this is slightly different from what Efe showed, because we have more control over the code, so we can

customize our agent a little bit more, and we can integrate it into any of our business processes as well. And in this case, an agent, an Autopilot agent, is going to live in the cloud. It's going to be always on and run each of its sessions in their own VM.

While Copilot keeps building, I'll go ahead and click on this section here and open up models. And this is a list of all of the models that I currently have deployed to my Foundry project. I'm particularly excited about Microsoft Decision-1, which was just released a few days ago, like Satya mentioned.

Decision models are a new class of models that are built for frequent general-purpose classification. They're super-fast and super cheap, so I don't have to overthink where I'm using them in my project. And in this case today, our agent is going to use it for judgment call.

I can also go ahead and click on this toolboxes section, and this will show me a list of tools that I've already deployed to Foundry. These tools are going to allow my agent to take action autonomously. And so, that means when I deploy my agent into Foundry, it's going to be able to use these tools to keep working even when I close my laptop, which is fantastic.

Okay, the process of building an agent is going to take some time, so I'm going to go ahead and I'll switch to a different session where we already have a version of our agent built. I am going to click on this Inspect Locally button, and it's going to open up an environment for me to be able to test out my agent on my local machine.

I try to check in with my team probably every two weeks or so in terms of how we are making progress towards our OKR. I do this in a very particular way, and I'd like for my agent to be able to run the same check in the same way that I do it. And so, I'm going to send this prompt through.

And the first thing that you'll see is that our agent is going to call a tool associated with Foundry Memories, which is a memory store that helps to isolate my personal memories from those of my team and the wider org as well.

We can also see the Fetch tool is being called, and this is part of Work IQ. It's going to pull in a document from SharePoint, which has my OKRs in it, and it's also going to pull in updates from my teammates that live in Teams and Outlook, so that I can compare them against that document.

And here, we've also seen that the decision model has run a check to decide whether or not, based on all of the information it has, whether or not our OKRs are on track to be completed. And we can see that actually, Gwen's gift finder is running a little bit behind, so it's good to probably check on that later.

Now that we can see that our agent is working functionally as expected, the next thing I want to do is make sure the code is following best practices. And to do that, I can open up this validation skill from Microsoft as well. And this skill, once it runs, it's going to scan my code and generate a report. And this report will have lots of information in terms of where my agent code is looking good, is passing some checks, and things that I can do to improve my agent's code as well.

Once I'm happy that I've solved all of the issues in my validation report, I can go ahead and go back to the canvas and click on this Deploy to Foundry button. This is going to make sure that my agent is immediately available for use to be published in Teams, in Outlook, or in any other application that you can integrate its endpoint into.

For this example this morning, we are going to take a look at what my agent currently looks like in Teams. Here, I'm in my Teams general channel, and you can see my coworker, Dan, is asking Mona for help with an expense policy question. Julia is asking for help with a security review.

And I'm going to go ahead and I'm going to post in the channel. And I'm going to say to the team that I've noticed that everyone's working on a new security issue, and I'm going to plan some time to review this next week.

Because our agent is an Autopilot, it doesn't have to be explicitly tagged to help me with my work. The agent is going to use the decision model to be able to make a choice to figure out whether it could be helpful in a specific case.

Here, the agent has decided that it should remind me. It's been able to judge my intent and turn that into an action. It's going to set up a reminder for next week Tuesday, so I can keep my commitment of reviewing my team's work.

Okay, so we've seen that our agent is going to be having multiple conversations in Teams or wherever the surface is that you're using. We want to make sure that it's performing well in production, and we need to evaluate the agent's work. And to do that, the best way to do that is through traces, but traces are often dense and difficult to parse and understand.

And so, now, in Microsoft Foundry, we have a new feature called Insights. This feature will be able to find all of the issues that consistently come up in your traces and give you a way to optimize your agent so that your agent improves over time.

Okay, in the past few minutes, we've seen a lot. We started with an agent that we built with Copilot in the GitHub Copilot app and with Microsoft Foundry Canvas. We then tested our agent locally and were able to validate that the code follows best practices. And then we saw that our agent works in Teams, and we're able to see how we can improve it over time with Insights. Microsoft Foundry lets you go from prototype to production, and we're excited to see what you build.

Back to you, Satya.

(Applause.)

SATYA NADELLA: Thank you so much, Marlene.

That gives you a flavor for how you can use the IQ platform and Foundry to build out your agentic systems.

In fact, I've had a chance to look at and see these agentic systems getting built by many of our partners and customers here, and one of the ones I saw this morning was from EasyJet. They built out a system for their contact center. It's a multi-agent system that's really fully deployed now for their contact center operations.

Also, I had a chance to meet with the folks at 1st Central, where they're building out what is an end-to-end auto insurance, which is all being driven by an agent system that they built. They have a goal of targeting something like 400,000 insurance policies next year. It's just great to see the entire system was conceived and built as an agentic system to start with.

Rolls Royce, each one of the engines has over 30,000 parts. There are eight types of engines in service at or any time. Just planning for scheduled maintenance is a pretty complex optimization problem, given the supply chain of those parts and the placement of those parts in each of the facilities in which the maintenance happens. They have now built an agentic system to automate all of that.

I had a chance to meet with folks at Unilever, where they have built an agentic system to take both orders that are coming in, order to cash and automation of that, but then also tied it to trade claims, which is a very manual process today, and for them to be able to now have the agentic system that just increases the throughput and then the cash flow that then they can put to work and then reduce, in fact, all the burden on the folks who are doing that work today as well, which is so manual. It's great to see again, I'd say, the ambition level and the real-world impact that these agentic systems are having in the economy as they're getting broadly deployed.

Now, the next stanza is all around, if this was about building agentic systems, the next is about ensuring there's trust around these agentic systems, and that's where Agent 365 and Perception are critical. It starts with having a set of principles that are driving the architecture of your trust system, and I describe them as principles of observability and governance.

But you have to be able to ID every one of these agents. You need to be able to log every one of the activities of the agent. That means its tool calls, its CoT. When it uses a model and it has Chain of Thought, you want to be able to log that Chain of Thought not in neuralese but in English, and it has to be tamper proof, the logs themselves.

You need to be able to set policy around it such that you can contain the actions. You need to be able to behaviorally monitor those actions, so if it has abnormal behavior, you need to be able to block it and contain it. These are the principles of building out the trust system.

Agent 365 essentially brings together the control plane for your ITOps. This is where you can have every agent ID'd, managed, governed, set policy. You also have SecOps. That's where you can do containment. Then it also has FinOps built in because one of the other top considerations for any of us deploying agents today is to ensure that our token budgets are being used where we

want them used. The idea that you really now have one control plane for IT, security, as well as financial controls, is great to see.

Then you also have Foundry. In fact, Marlene showed that optimization of the traces of an agentic system, and the idea is that when you collect all this observability data, that means you should go back to optimizing and continuously learning and improving your agentic system.

That's why the AI engineers are able to use Foundry with the same data of observability. It's not just observability for IT or security or financial operations; it's also to continuously improve the system.

And lastly, Perception, between defender and Perception, now that you have these sophisticated systems, you need to be able to manage these systems with essentially an agentic system of its own, which is driving security.

The best way I think of it is you now have red agents that are continuously simulating the attacks, you have blue agents that are continuously defending, and then you have green agents that are continuously remediating any vulnerabilities they find. You have essentially an agent swarm that forms Perception.

We just announced it this week, and it's great to see now both Agent 365 and Perception. For example, one of the biggest things we do in Perception is really do this scoring of risk of any given agent. It's a dynamic risk score using all of the trillions of signals we see that we ascribe to any agent, and then based on that score, you dynamically set the ACLs. For example, the agent ACLs are not inheriting the ACLs. For example, when you take autopilot that I may have, the autopilot should not inherit all my permissions. In fact, it should be based on what its activity is, what its trust score is, and therefore get more dynamic ACLs. That means least privileged access to things that I have access to. This type of system is what will allow us to contain these sophisticated agents, of course the attacks coming from the outside, but also this insider risk issue is being tackled.

That's the systems that we are using to build in Copilot, Microsoft IQ, Agent 365, as well as Perception to build out, deploy, diffuse these agentic systems. Of course, you need world-class infrastructure underneath all of these agents, and that's where Azure comes in. Ultimately, it's about that performance per tokens per pound per watt, and that goal remains for us top of mind as we build out Azure across as the world's computer,

We have 500-plus data centers around the world, 80-plus Azure regions around the world. Right here in the UK, we have two live, we have a third one coming live soon, and we are committed to what we announced last year, which is the \$30 billion of incremental investment in bringing up all of this data center capacity in the UK and making great progress.

It's not just about bringing compute, but it's also ensuring that we are readying the ecosystem by upskilling and skilling around AI. In fact, we had a commitment to train a million residents of the UK around AI, and now we have already crossed that, and we're at two and a half million. It's

great to sieve and the appetite to upskill and have the ambition to really make professional progress using AI in the economy broadly spread, and it's great to see.

The other element of infrastructure is the fleet and the systems and the silicon. and we have an approach that we have taken. Again, it's an ecosystem approach. We have the best systems coming from NVIDIA and AMD. We're also building our own. It's not just the chip, but it's a full rack and it's a full supercomputer being powered by Maia, and it's producing close to 40%-plus additional tokens for us for the models that we are serving compared to the leading systems that we have in our fleet. This, again, ultimately will allow us to do that performance optimization, which is can we get more tokens per watt per pound? That's ultimately the goal of having a heterogeneous fleet.

We want to be open to innovation. I see even in the UK, there are lots of new hardware companies and systems companies that are being born, and we look forward to working with the broad ecosystem to bring all of that innovation to bear in order to drive that performance frontier.

Now, the other element that people talk a lot about when it comes to infrastructure is sovereignty. Fundamentally, it's about making sure that you have the control.

When you talk about sovereignty, I think the best way to conceptualize it is that you are trying to manage risk. One of the best ways to manage risk is to have a portfolio approach. So you have a public cloud, a public cloud with sovereign control, you have a private cloud, and then you also have, in some cases, partner ecosystem clouds that are nationally built. This combination is what will give you the choice and the controls and the resilience.

In fact, in the public cloud itself, with sovereign controls, we just announced many, many features this week, and one of them is the ability for you to have, for example, clean rooms for you to be able to have things that allow you to have the ability to detect any tampering of logs during administrative process. These are all capabilities that we are building into our sovereign cloud, sovereign controls in the public cloud, as well as giving you full disconnected operations with your private cloud.

To make all the complexity of all this come together, what we have done this week is to announce the coming of a new control plane to manage all of these various clouds. Because how can you take any one workload that you may use in the public cloud with the sovereign controls, and you may decide that I want to move it to a local cloud, and you should be able to do that using this control plane. That's what we have enabled, and I wanted to have Stuart come up on stage and show you a demo of this new control plane for managing all of the sovereignty controls that are available to you.

Stuart, take it away.

STUART RUSSELL: Hi, everyone. I'm Stuart Russell, Vice President of International Sovereign Clouds at Microsoft, and I'm here to talk to you in a little bit more detail about sovereignty.

Sovereignty means different things depending on your organization, your workloads, and the risks and opportunities you are trying to balance. What I want to demonstrate is how we provide the sovereignty controls, so you can manage your resources and data as you see fit. And it all comes back to something fundamental, oversight. There's oversight of what we do, but what about your oversight of what you do, the oversight you need to understand your sovereignty posture?

Now, usually, that information is in services, in policies, and in people's heads. Today, the Sovereign Control Panel brings it together into a unified view, so that you have the visibility and insight to act.

Here, I'm demonstrating the Sovereign Control Panel, which enables you to outline your use of our sovereignty controls, such as data residency, confidentiality, and business continuity. It provides that single view into your posture, allowing you to drill into details and take action where necessary. If your policy dictates that all resources must be in a specific region, we show your compliance with that policy, in this instance, 80%.

The breakdown of the underlying detail is simply a click away, where out of region resources can be investigated, policy compliance issues identified. In this instance, two resources need to be redeployed into a different geography, and through this, remediation activity undertaken; or if data confidentiality is your concern, encryption at rest and in transit can be evaluated in much the same way as residency, and the detail drilled into when necessary.

But for some organizations, the question is much, much broader than data. Can you keep operating if access to the public cloud is disrupted? Now, Azure already offers resiliency through things like availability zones, but we've heard you loud and clear that you need extra confidence from us that you can continue to operate in the most extreme of circumstances. That is why today, I'm really excited to demonstrate how the Sovereign Control Panel goes even further with sovereign business continuity.

Now, just like the other controls, the top-level view provides the way in. In the case of business continuity, it shows where your resources are built in such a way as to enable transition from Azure to Azure Local, your private cloud that you own and operate.

In this instance, 460 resources are ready to go, but 150 are still needing changes. Good statistics, but these did not ensure business continuity. Only you know which workloads are central to the smooth running of your business. And once again, by drilling into the detail, we can show the resources that are central to your business that still require changes.

And based on our sovereign business continuity architecture patterns, you can see that to move this specific resource from Azure to Azure Local, you need to adopt an alternative messaging service. Through this, you can drive concrete remediation processes to enable your business to operate locally when necessary.

Now, having manually transitioned these workloads into Azure Local, let's look into that sovereign private cloud environment that you own and operate. Here, you can see the resources

that we just moved are now running on premises, on Azure Local, and importantly, still connected to Azure. You can manage them alongside your wider public cloud estate.

But if disruption escalates further, then now you can go further, too, and disconnect from Azure altogether through a simple script. This script we're running moves critical workloads from Azure Local connected to operate completely disconnected with a local control plane, without interrupting what the workloads are actually doing.

Transition complete, your business stayed running throughout, and you are now completely under your control.

Now with that control, let's make the most of what you can do there. With Foundry Local and GitHub Enterprise Local, you can now bring models and developer tools right alongside your most sensitive data and your most critical workloads. You can make the most of these opportunities at your fingertips.

Because sovereignty isn't a one-size-fits-all, your requirements are unique, and Microsoft Sovereign Cloud, through the Sovereign Control Panel, gives you the choice and control to meet these requirements.

Back to you, Satya.

(Applause.)

SATYA NADELLA: Thank you so much, Stuart. I mean, that was a lot of, I would say, sophisticated distributed systems that really had to come together to make this possible. I mean, if you think about this, you are using Service Bus, and now you want to disconnect and deploy it locally. You can then switch to an open source package like Rabbit MQ.

But then the last part that Stuart talked about is pretty cool, which is now you have, let's say, one of those agentic systems you built in the cloud, and you want to fully run it disconnected. You now have Azure Local, you have Foundry Local, you have GitHub Enterprise, and you have the control plane that can also move completely disconnected in local. And then, of course, you can use an open weight model that can be running and powering your local systems. And so, that ability is fantastic to see.

And one consideration, though, is cybersecurity, because you do not want to sacrifice your ability to protect these sophisticated agentic systems when, for example, you moved it to some local data center in which Azure Local and Foundry Local are all running, because at the end of the day, you're only as good as your ability to take the global signal and then use that to protect anything locally. And that's, I think, the nuanced discussion that needs to be had.

And in fact, when you think about even the risk, you need to not only think about the risk when it comes to sovereignty and the options of resilience, but you also want to talk about and think about how do you ensure cybersecurity continues, so the way we want to make sure is the cyber signal itself is available even for things that are disconnected and local.

And so, that's really a little bit of a walk across the entirety of the stack and the platform that we are building, such that you all can take AI and its ever increasing capability, and fundamentally start building systems that are empowering people in your organizations, that are enhancing the intelligence and the learning inside of your organizations. And ultimately you're able to build the trust architecture around these sophisticated agentic systems.

That's really our mission, is to deliver the platform that allows you to get the benefits of this technology and create that local surplus.

Thank you all very, very much. Thank you.

(Applause.)

END