**Build 2018**

**Scott Guthrie**

**May 7, 2018**

**ANNOUNCER:** Please welcome Scott Guthrie.

**SCOTT GUTHRIE:** Good morning, everyone, and welcome to Build.

Satya just talked about the intelligent edge and the intelligent cloud, and some of the incredible opportunities we now have as developers, and the amazing impact we can have on people's lives. I'm now going to go deeper into the platform and tools we're delivering with Microsoft Azure, and how you can take advantage of them to build truly amazing solutions.

One of the defining aspects of cloud computing is the ability to innovate and release new technology faster and at greater scale than ever before. This new set of technology, things like IoT, machine learning, microservices, AI, serverless computing and more, is all happening right now thanks in large part due to cloud computing. And this is an incredibly exciting time to be a developer, and the opportunities to really explore new approaches and technologies have never been greater.

But I also recognize all this stuff is cool, but it can also be more than a little overwhelming. I hear that concern a lot with a lot of my conversations with developers around the world. The expectation to know all of these new technologies, to be up to date with them all the time can sometimes leave you feeling like you're falling behind. And the expectations that your companies have on you to quickly deliver breakthrough experiences with all this new technology has never been higher. In a lot of cases now, the technology that we're talking about today is really being bet on to deliver radically new digital experiences that completely transform your organization's business models. And you know you have to be careful about security while also simultaneously trying to be an expert on all this new tech and deliver those breakthrough solutions and, needless to say, this isn't all easy.

And it's really this understanding that shapes how we try to build and deliver Azure. Every day my team comes to work to make Azure a powerful enterprise-grade cloud service. And the more important work that we do is, in terms of building Azure, is really trying to focus it to guide it around your success. Having lots of great technology and lots of features is necessary but not sufficient, it's really about how successful you can be using this technology in the cloud that ultimately matters.

And as a little bit on that, we focus Azure innovation really on your needs, on making cutting edge technology approachable for all developers and doing the heavy lifting to ensure that Azure uniquely meets enterprise scenarios. This means having an end-to-end experience across our cloud services, our management tools and development tools that provides an incredibly productive cloud experience, one that's hybrid and enables you to build solutions that run consistently within both our public cloud data centers as well as your own. It means having a cloud that enables you to use AI and data to infuse intelligence into all your solutions. And it means having a cloud that you can trust. Trust is a core value of Azure and we lead the industry in our work around security, compliance, privacy and responsibility.

And the focus we're now delivering on innovation, trust and results is leading to tremendous adoption of Azure right now. And you heard Satya talk about whole bunches of great customers using it today. Startups, governments, and over 90 percent of the Fortune 500 companies in the world are now running their businesses on Azure. These are just a handful of some of the logos of customers running on Azure today.

Let's watch a video of a few of them talking about how they're using Azure to drive their success.

(Video segment, applause.)

So Azure is a broad platform. You can use Azure for infrastructure and just take advantage of it for things like compute, storage and virtual machines; or you can also take advantage of over a hundred highly engineered services to build your apps even faster. And this morning I'm going to focus on five areas of innovation that we think are important for every developer as we move into this intelligent cloud and intelligent edge world.

Let's start by talking about the work we're doing around DevTools and DevOps. Now our mission in Visual Studio is to provide best-in-class tools for every developer. And we now have tools for developers who want a lightweight code optimizer as well as for those looking for a full development IDE. And with Visual Studio Team Services, we provide a suite of developer SaaS services that deliver a complete DevOps experience. And when combined with Microsoft Azure, these tools deliver a fantastic development stack with unparalleled developer productivity for every developer.

One of the great new Visual Studio improvements we're making available today is a new capability we call Live Share. We think this is really a game-changer in terms of enabling real-time collaborative development across teams. Live Share enables you to not only work collaboratively on code with your co-workers, but even to share debugging of an app across team members. It works across both Visual Studio and Visual Studio Code, and even allows developers to do Live Shares across Windows, Mac and Linux dev machines simultaneously across any language and code base.

And what I want to do now is invite Amanda Silver, who runs our Visual Studio IDE team on stage together with Jonathan to show off a demo of Visual Studio Live Share in action. Here's Amanda and Jonathan.

**AMANDA SILVER:** All right. Thanks, Scott.

Let's start coding. Now today developer collaboration usually doesn't start until after somebody makes the commit, and that's often really too late for teams that are motivated by time to market.

It looks like Jonathan needs some help. I guess now is as good a time as any to show you guys how collaborative development works on our team.

**JONATHAN CARTER:** So I'm supposed to deploy the Guest Book app for the Live Share booth in like the next hour, but the photos aren't showing up anymore, and I'm kind of starting to freak out. Now, to get Amanda's help, I could start a screen-sharing session. But the constant back and forth over control of the keyboard gets really tedious, and frankly I'm a little bit tired of people saying that my editor theme looks like the aftermath of Cinco de Mayo. So let's use something new instead. Let's use Visual Studio Live Share.

Now I'm on my Mac. I'm using Visual Studio Code, and I've already installed the Live Share extension. So what I'm going to do is go down to the status bar and click this Share button. What that's going to generate for me is a link that I can give to Amanda to get her help in my context right here.

So if we switch back over to teams, you can see from our thread that Amanda is already pretty used to me bugging her for help, kind of like right now. So let's paste this link in and then see what it looks like from her machine to join my session.

**AMANDA SILVER:** All right. So all I have to do is go ahead and click on this link, and it will automatically launch the preferred developer environment that I use, which is Visual Studio. Now notice, I'm on Windows, he's on Mac. I'm on Visual Studio, he's in VS Code. He likes techno-party theme colors, and I like a little bit more classic themes.

Now, as we join in Visual Studio, you will see it's populated by the context that's actually on Jonathan's machine. I have none of the dependencies that he has to run this source code. I didn't clone the repo. I don't have Mode installed. I basically have nothing installed that he has a dependency on, and yet I get all of the context and all of the language services here in Visual Studio.

Now, as soon as it connects, you can see that I'm automatically brought to the place where his cursor is currently at, and I can see his cursor move around. I can also see him select things, so that we can get a shared context really, really quickly, and start working together. Even further, I can independently investigate what's going on in the folder, so that I can look at other code to get my bearings a little bit.

All right, Jonathan, now that I'm in, what's going on?

**JONATHAN CARTER:** So the Guest Book isn't showing photos anymore. And I kind of think it's related to this code right here, but I'm not exactly sure. Any ideas what's going wrong?

**AMANDA SILVER:** Well, I have no idea looking at this, but I really can't think straight with red squiggles so the first thing I have to take care of is this linking error.

Now as you guys can see, as I move around in here, I get the same experience that you would expect from Visual Studio. You can see I get that toolkit. I can go ahead and get completion lists and select. And as I move my cursor off you can see that background complication kicks in and I have the confidence that that fix worked. But that still is not enough for me to know what's going on from a debut perspective. So I can go ahead and inspect other things here, look at a peak definition of this value and yet this method is definitely I think what's going on. So I'm just going to go promote that to the background.

**JONATHAN CARTER:** Now, as you can see, at this point Amanda and I are actually in entirely separate styles, which is cool and not something you could achieve with screen sharing. But if at any point I want to refocus with what she's doing I can choose to follow her cursor. What's great about this is it allows us to work independently or together, depending on what the situation demands.

**AMANDA SILVER:** All right, Jonathan, I think I know what's going on. But to be really confident I'd like to set a breakpoint on this right in the bud.

**JONATHAN CARTER:** All right. And as soon as Amanda sets that breakpoint it shows up on my end, as well. What's really important about that is that Live Share is not just an editing platform. It is real-time collaborative development, which we all know in development includes debugging. So because we're in the context of a Live Share session, if I go and press this Play button it's going to launch that app, share with Amanda, and hit the breakpoint that she just set.

**AMANDA SILVER:** All right. It's pretty cool. Now, as soon as Visual Studio detects that he is at a breakpoint I get moved into a debug session, as well. And you can see that I get all of the capabilities I would expect in a Visual Studio debug session. I can look at the call stack. I can look at locals and expand to understand what this is. I even had a shared debug cursor. I have access to the same execution point that he has. So I can go ahead and even step over and then get hover tips to understand what the values are.

And I can see that, Jonathan, I think I've figured out your problem. The signature is 44 and it really should be an object. So you're calling unshift on an array, but it really should be shift. So I'm just going to save that, and I think that should work.

**JONATHAN CARTER:** All right. Now, I'd really like for Amanda to be able to verify the fix that she just made before I let her off the hook. But the Guest Book app is running local on my machine and I'm not ready to deploy it yet and I certainly don't want to expose it over the internet. But because Live Share provides us with a secure connection between my and Amanda's machine I can actually share the web app that is running on local host with her so that she can see it as if it was running on her machine.

**AMANDA SILVER:** All right, so all I have to do is go ahead to our share session up here and look at shared servers. And you can see what I get access to is local host. I can go ahead and launch that in the browser and I can see the app straight away, even though it's local host hosted on his machine. It looks like we fixed the issue.

**JONATHAN CARTER:** All right. I'm feeling a lot better. Thank you. Thank you. So I'm kind of out of hot water at this point. But, let's make sure that we didn't regress anything else and run our test. Now, the Guest Book app, like many modern applications today, actually runs its tests from the command line. But Amanda is not standing right next to me to run that or view the results with me. But, once again, because Live Share provides that secure connection between the two of us, I can also share a Live Terminal instance with her that she will see on her end and can then run our tests and let us both see the results in real time.

**AMANDA SILVER:** So as soon as he starts that terminal it's shared. And this isn't just a terminal that's specific to this debug session. It's actually a shared terminal access to his machine. So I could use this to inspect developer configuration, for example, and see what version of Node is running. But in this case, I'm just going to use this to call our test. And we both see the results at the same time. It looks like it's good.

**JONATHAN CARTER:** So we fixed the app. I think the test results speak for themselves. Live Share is awesome. Thank you so much, Amanda.

**AMANDA SILVER:** Awesome, we're ready to go live.

**JONATHAN CARTER:** Yeah.

**AMANDA SILVER:** Great, thanks.

**JONATHAN CARTER:** Thank you.

**SCOTT GUTHRIE:** And let me tell you the share, that Visual Studio Live Share, is now available for you to start using today. You can start downloading it. Please wait until the end of the keynote. But you're welcome to start using it. And I'm also really happy to announce that we're making it free for all Visual Studio and Visual Studio Code developers on every platform.

(Applause.)

So one of the things you've probably noticed over the last few years is a change in how Microsoft approaches open source. We not only now support open source projects in a deep way, but we also have open sourced much of our own product development. Now, VS Code is just one example of the type of open source projects we help lead now. It's developed in the public on GitHub and it's had over 17,000 developer contributions to it to make it even better.

And VS Code is just one example of how we're really trying to change our development approach to be much more open and inclusive at Microsoft, things like .NET Core, NuGet, Power Shell, and our Azure SDK are other examples where we're taking Microsoft projects and are now developing them in the open and hosting them on GitHub.

And what I'd like to do is invite Jason Warner, the Senior Vice President of Technology at GitHub, on stage to talk a little about GitHub and some of the changes they're seeing as part of this engagement, as well as to talk about some exciting work that we're launching together today. So please join me in welcoming Jason.

(Applause.)

Welcome, Jason. Thanks for being here.

Can you talk a little bit about GitHub and the mission you guys have for developers?

**JASON WARNER:** Sure, thanks, Scott.

Yes, GitHub's core mission is to make developers’ lives better. We started this by revolutionizing code collaboration. Today the majority of the world's code, both public and private, sits on GitHub, with 27 million developers collaborating across 80 million repositories. We view ourselves as stewards of developers and we aim to reduce the friction in the entire software development lifecycle to get developers back to doing what they care about most, which is writing code.

**SCOTT GUTHRIE:** That's awesome. We obviously share your vision in terms of both how developers and specifically open source can really fuel innovation. And that's really, as I talked about, one of the core reasons why it's become such a critical part of our engineering culture over the last few years.

**JASON WARNER:** I think it's amazing to see what Microsoft has done in the past few years. The industry has shifted, and they realize the power of open source. And, in fact, I don't think it's too bold to say that open source now powers modern software development. And Microsoft might be the best example of a corporation embracing open source. We know from statistics that we have in GitHub that Microsoft is the single largest corporate contributor to open source on GitHub, and there by extension, in the history of open source. In fact, Microsoft has the largest open source community in the entire world with Visual Studio Code.

**SCOTT GUTHRIE:** It's pretty awesome.

(Applause.)

I know in addition to contributing to projects and hosting on GitHub, one of the other things that our two teams have been working on a lot has actually been on several other projects in terms of technology integration that we've done. I was wondering if you could share a little bit about those and from your perspective how it's helping developers.

**JASON WARNER:** Absolutely. We've been able to accomplish a lot over the past few years. Last year at Microsoft Connect we announced our combined work on GVSS, which allowed Git to scale, so the world's largest repository is across Windows and Linux. And with our shared developer focus our areas of interest seem to overlap more and more with each passing year. And I'm excited to see us working this way out in the open.

**SCOTT GUTHRIE:** I know our teams have been working hard on something new that we're launching today, and the GitHub is doing it even more broadly across your ecosystem. Can you talk a little bit more about what you guys are rolling out and how we're integrated?

**JASON WARNER:** Absolutely. Today we're announcing the public beta of the GitHub Checks API, which will allow our partner integrators to do very deep annotated rich data sets across all manners of continuous integration, including linting and code analysis. We're excited to see what you all are going to do with that today.

**SCOTT GUTHRIE:** Awesome. Thanks so much.

**JASON WARNER:** Thanks, everybody.

(Applause.)

**SCOTT GUTHRIE:** As Jason just mentioned, one of the things that we're excited to announce today is some of the deeper integration that we're doing between GitHub and our overall set of Azure and Visual Studio DevOps capabilities. And the first one we're releasing today is integrating the mobile DevOps tool support that we provide with Visual Studio App Center, directly within the GitHub experience.

Now, VS App Center is something we launched late last fall and it enables mobile developers to build, test and distribute mobile apps to a wide variety of different devices, including both iOS and Android, to monitor the performance of those apps, as well as to collect analytics and crash dumps from them so that we can quickly -- you can quickly improve your app with each release in a nice iterative way.

And we've made all this easy by integrating all of the different DevOps stages into one fluid workflow. And what's great with our GitHub partnership is we now have taken that and directly integrated it, as well, within GitHub as part of that experience. So what I'd like to do is invite Samina, who is the PM Team for VS App Center on stage to actually show off VS App Center and the great integration we now have with GitHub. Here's Samina.

**SAMINA PAST:** Thank you, Scott. I am so excited to show you today how with App Center and GitHub we take developer productivity to a whole new level. From pushing code to the repository to shipping the apps to the app store in under three minutes, ready? Here in GitHub is the repository for the Smart Hotel demo app. This is an iOS app written in Swift, but the same steps work for apps written in Xamarin, React Native or Android.

The dev team has been working on a new feature and they opened a pull request. And here in the pull request GitHub automatically detects that no continuous integration had been set up and take me to the GitHub marketplace for available GitHub apps. I am looking for a mobile CI solution and App Center is right here.

To start building my repository in App Center I need to install the GitHub app. App Center has a really great feature, so I'll go ahead and install it. I'm now completing my free purchase and finally I am giving App Center access to my repository.

Now I am in App Center. And when I select this, my code repository everything gets automatically detected. No need to enter anything manually. So I can simply continue. App center does the magic and it does all the build settings. So now we are ready to kick off our build. Behind the scenes Azure provisions a virtual machine with the latest tools, pulls the repository, starts a build, and when the build is completed securely deletes the VM. With Azure you can get from your repository to setting up continuous integration in under three minutes.

And back to the demo app, the build is already running. Now, this build will take one or two minutes to complete. But once the build is completed App Center will report the state back to GitHub. And here in the pull request I can see the build is successful.

Now, before making the pull request and distributing the app to my users I want to run a suite of automated UI tests and to make sure that no regressions were introduced. App Center offers thousands of devices with different operating systems and languages, all hosted in Azure. And here we can see how the test cases look for the iOS versions on a real iPhone. But remember, App Center also supports apps written in Xamarin, React Native and Java. So I run the test for the Android version of the Smart Hotel app on over 100 Android devices. And all you see here are real physical devices hosted in the Azure Cloud.

Now, if we look closer some of those are failing. It looks like on this device the app is not transitioning to the home screen. So later the team can troubleshoot the issue, look at the test logs, and fix the issues. We are now seeing how the app is continuously building on every change. UI tests are validating the user flows. So now we are ready to publish the app to the app store, to our end users.

In the past we might have done it manually, over and over again. But now we can use the same continuously delivery process and do it automatically with App Center and GitHub. So I will continue the master branch, but for every successful build it will automatically distribute it to the app store. All I have to do is enable distribution to production right here.

And with this every new pull request made to the master branch will trigger a new build in App Center. UI Tests will be running. And once everything looks green, the new release will be uploaded to the app store. Continuous delivery for your mobile app has never been easier. Together app center and GitHub make app developers more productive. Get started today. We know you are going to love it.

Thank you.

(Applause.)

**SCOTT GUTHRIE:** Samina's App Center is just one of the DevOps tooling components that we provide as part of the Visual Studio family and Azure set of DevOps tooling. With our Visual Studio Team Services family of tools we're making it easier to adopt a DevOps-based model and set your team up for success regardless of the application type that you're working on.

VSTS is fully integrated into Azure and includes everything that you need and works with every language and runtime environment that you already use. What I'd like to do now is invite Donovan Brown, who leads our DevOps Advocate team, on stage to show us how easy it is to get started using DevOps in Azure, using Visual Studio Team Services.

Here's Donovan.

**DONOVAN BROWN:** Thanks, Scott.

(Applause.)

Good morning, everyone.

(Audience response.)

Let's try that again. Good morning, everyone.

(Audience response.)

Perfect. You know what time it is. It's time to rub a little DevOps on it and make it better. When we combine the power of Visual Studio Team Services in Azure magic happens. We get to witness this magic every day as Visual Studio Team Services deploys itself into Azure. This gives our teams a unique perspective on how to apply DevOps best practices when deploying into the cloud. Let me show you how they do it.

This is an actual dashboard of one of the teams that bring you VSTS every single day. It's customized to show how many days are up in the script, and what work is currently assigned to them. We can also view this work on a Kanban board. This is a real time status report. To update your status you simply drag and drop. But this happens to be where our team actually works. So I probably shouldn't be moving any of these tiles around.

Our team uses Git. And when you use Git sometimes the branches get out of control. You don't really remember you created some of those branches, but not when you use our Kanban board. Here you're able to create a branch right from the board and this branch is now associated to this work. Every single pull request, every single commit, every single build, and every single release is automatically associated to this work.

So now you know exactly why you created the item and you get end-to-end traceability for free. Once we get a pull request made it's time for us to go ahead and collaborate over it. In a pull request, this is a way for you renew your code and have conversations with your teams to make sure that the code is quality. If you see that piece of code, that piece of code that gives you that warm and fuzzy feeling, you can actually share it with your teams right here in your pull request.

A pull request can also run policies and one of the policies that we run is a build. Our builds run a staggering 77,000-unit tests against every pull request and it does it 600 times a day to ensure that we only ship the highest quality code to our customers. When our pull request is done the code is now ready to be deployed into production.

Here we practice safe deployment. Safety deployment is where we deploy to one production environment, monitor the code before we deploy it to the next production environment. This first environment, ring zero, is where the VSTS team actually works. We dogfood everything that we give to our customers, because if it's not good enough for us, it's definitely not good enough for you. We monitor our telemetry and application insights. We run queries against our bugs. And if we find any we stop our release.

This used to be a manual process, but now I can use Release Gate. Release Gate allows you to have release management automatically run queries against my bug, automatically watch my telemetry and application insights, and if it finds any issues Release Management will automatically stop our release and protect our customers. We have automated safe deployment.

Now, I know what you're dying to ask me right now, Donovan, how do we get started? Well, I'm glad that you asked, because you get started in the Azure portal. We have a new feature called DevOps Project that allows us to bring the power of Visual Studio Team Services right inside of the Azure portal.

You get started like you would any other resource. You go to creative and there is DevOps Project. Gone are the days of you having to manually deploy code or FTP. You don't even have to right-click inside of your IDE. And as Damian Brady likes to say, friends don't let friends right-click publish. (Laughter.) And now they never have to. And if you look at this page here, you instantly see that this is not your daddy's Microsoft. This is a Microsoft that understands Go and Ruby, PHP and Python.

Let's have some fun and let's do a Java application. I get to choose my favorite framework, which happens to be Spring. Now this is where the fun begins. I want to reassure you, if you're not ready for containers, don't let anyone pressure you into them. We can run your app the way that it is today inside of App Service. But if you are ready for containers, so are we. We're able to set up an entire Kubernetes cluster for you and run your app inside of there. Let's have some fun with Kubernetes.

At this point you have to wire up your VSTS account. I see some of you getting anxious thinking but, Donovan, we don't have a VSTS account. Shhh! (Laughter.) We can create one for you right here on the portal. But if you happen to have your own, let's go ahead and choose one that I already have for myself. So we'll choose this one. I need to give the product a name, and now I click on done.

Now I just sit back and relax while Visual Studio and Team Services work together to build us entire CICD pipeline for our Java application. When this is done, I'm going to have a portal just like this one. This is a blade that shows me everything from Visual Studio Team Services in Azure in one place. I can see my commits to my repository, the builds and the releases. I can see my cluster, and even infrastructure information coming from Application Insights down here at the bottom. If I click on this link, I get to see the sample application running in my cluster right now.

Chances are you have your own code. So why don't we go ahead and show you how you can actually use your own code as well. Luckily, I can use the deep links here on our build pipeline, jump right inside of Visual Studio Team Services where I have complete control over this pipeline. I could simply come in here and say, why don't we edit our build definition. Let's get your code from where it exists today, maybe your code lives inside of GitHub. So I come over here, and then I authorize my account, and then once I do I have access to all my GitHub repos right here inside of Visual Studio Team Services.

I'm going to choose my Smart Hotel 360, click on select, and now I'm going to save and queue this build. I am now pulling code upon every commit from GitHub, building this Java application and packaging it as an image. I'm then going to take that image and deploy it all the way inside of Kubernetes.

I hope you realize that this is not just a new era for Microsoft, this is a new era for all of us to really combine the power of Visual Studio Team Services in Azure. Here at Microsoft we can implement DevOps for any language targeting any platform.

Thank you so much everyone.

(Applause.)

**SCOTT GUTHRIE:** Donovan just showed us in that demo, it's now easier than ever to set up a DevOps model using Azure. Using the new DevOps Project Support, you can set up CICD with a full application setup for .NET, for Java, for Node, for Python and PHP, with literally just a few clicks, or you can drive it an automated way from the command line. And with today's update, you can now easily do that both for web-based applications as well as container-based Kubernetes clusters using our new Azure Kubernetes Service, or AKS, as we call it for short. It supports Git-based source control repositories in both Visual Studio Team Services as well as GitHub, and you can get started with it now for free.

Now that we've covered some of the developer tooling improvements that we're shipping this week, let's now jump into discussing some of the Azure platform improvements that we're delivering as well.

One of the things that makes Azure really powerful is the coherent set of highly engineered services that enable you build, deploy and scale your applications even faster. For example, our Azure Web Apps service provides an incredibly optimized way to run web applications as well as expose web APIs. Our Azure Serverless Functions Service makes it incredibly easy to host serverless apps and code that can scale from just handling one request per second up to literally millions of requests per second simultaneously. And with both our Azure Kubernetes Service and Service Fabric offerings, we enable you to build modern container-based microservice applications.

Let's talk about the new AKS service I mentioned. So our new AKS or Azure Kubernetes Service, provides a fully managed Kubernetes-based orchestration service. It provides built-in auto-patching, auto-scaling and update support, which enables you to take the full-breadth of the Kubernetes ecosystem when you're doing your development.

And we're releasing a bunch of great updates with this this week at Build. One of the capabilities we're really excited to share is a new feature we call Dev Spaces. Dev Spaces enables a fast, inner loop development experience when building Kubernetes-based microservice applications. It enables you to edit and deploy code instantaneously as well as debug and set breakpoints across multipole containers running in a microservice-based environment. And it's designed to work both great for individual developers, but where it really shines is when you have a team of developers working together and you want to share a common team-based development environment and be able to test and iterate independently as appropriate.

What I would like to do is invite Scott Hanselman on stage to show off some of the new improvements we're shipping this week with Azure Kubernetes Service, as well as demonstrate how you'll be able to take advantage of Dev Spaces to get a super-smooth inner loop development experience with Kubernetes on Azure.

Here's Scott.

(Applause.)

**SCOTT HANDELMAN:** Hey, friends. So we saw Donovan take a Java app through Azure DevOps Projects, and he deployed that to AKS. Fast-forward 30 days later, and the project is a lot bigger, the company is a lot bigger, and let's say that I'm a new developer. So I need to get up to speed on how to work on this as quickly as possible. This is a site that I'm working on, and I'm going to come down here because I was told that here was a big when people go and search for hotels. I can type in New York, and I find my New York, but when they type in Seattle it doesn't work. So something is wrong. However, this application has a lot going on.

I'm going to switch over to Visual Studio, that's where I live as a Visual Studio developer. And this is where I feel most comfortable, and I see that there's eight different projects here. There's eight different projects here, and there's maybe dozens and dozens elsewhere. And I'm told by Brady, who set up the AKS environment, that each one of these projects within Visual Studio maps directly to a microservice that's running in a container inside of AKS.

So if I switch over into Azure, I can see all the different containers, not just the ones that are in Visual Studio, but the ones that other teams are working on that might be running in different languages, in different environments. I can see there's my hotel service. This is all running as part of Azure Kubernetes Service, which is pretty amazing, because it is a fully managed service. I don't have any VMs to worry about. I didn't create a bunch of VMs and install Kubernetes on them. I just have Kubernetes as a service. I can go and upgrade at any time. I can hit scale, and I can go and say I want eight CPUs, or I want hundreds of CPUs, totally up to me.

Now within this cluster, I can also peer into the health of individual containers. So that's green, that's good. Okay is good. I'm going to open that up. Here's all these containers. And I can scope my name space. Name line is where I'm told that the bug is, and all the services can be picked from, I'm going to pick hotels because that's the one that I am currently working on, I click on hotels, and then open that up.

I see some interesting stuff. The CPU is about 4 or 5 percent. There's my hotels container that I want to debug, because I think that's where the problem is. And I'm going to hit View Logs. Now, when I click into View Logs, I didn't have to install anything. This all comes as part of this service. I've got this rich logging experience right off the bat. And Brady gave me some queries to try out, so I could learn this as a new developer, and I put a couple of those into the clipboard.

So here I can go and do a search and say run, and now I'm looking at aggregate CPU over all of the containers instantly. You see there's about 4 or 5 percent, so obviously my problem is not a CPU-related problem, but I'm here to fix a bug, and I want to see if I can use the analytics to do that because, again, there's dozens of containers.

So I've done a query here where I'm going to look for places that return zero results, because when I typed in Seattle I got back zero results, and I know that it's written in C#, because that's my language and that's the thing I'm working on, so I'm going to look for places here it contains the word "controller." So hopefully we'll see if we can narrow down where this bug might have happened.

And I can zoom in here and see the places where the bug has happened in the public web. And I can even change this and see how often that bug happens and then render it as a charge. Again, all of this built into the service. It lets me do some amazing stuff. So I can see that clearly Brady checked into production at 5:00 and then just went home. So that's when the bugs started. So he won't be working here much longer -- (laughter) -- which is good for me as a new developer, because I'm moving up in the company.

So I know that it's in the city's controller because I saw that in my logs. I'm going to switch over into Visual Studio. Now within Visual Studio I could go looking for it and hunting around in here, but I'm just going to hit Control-T, because this is my Cs, this is where I live, I know how to move around in the Visual Studio.

So I'm just going to hit Control-T and it goes to cities controller and see the microservice that pulls that data back. Here's my Git. Git defaults to these, I'm going to right click on it and say go to definition. These are the default cities that are hard coded. And then up here the actual microservice, Git, that goes and talks to the backend.

Now, we have 10 microservices here I can see in Visual Studio, but just so you can get a real sense there's dozens elsewhere and some of them are in different languages. But some of them are actual Azure services like Cosmos DB, or Postgres or SQL. There's a lot going on to make this happen. So I go back over to cities controller and I say, all right, we're looking for default cities or, okay, and there's the bug, they didn't call get, they called get default cities. So that's a problem. They're actually only looking at the hard-coded cities. I'm going to change city queries to say get, pass in name and go like this.

Now, I'm going to set a breakpoint and you'll notice that this is all just the stuff that I ordinarily do as a Visual Studio developer, right. Nothing interesting is going on here. However, if I were to hit F5 ordinarily I might need to start up all of this stuff. Think about the amount of work it takes when someone says, hey, welcome to the company. Here's a laptop, let's get this set up with a tiny production environment on your underpowered laptop and then next week we'll make you more productive.

That's a complicated experience. I don't want to have that experience. Instead I'm going to use Azure Dev Spaces and I'm going to run this stuff up on AKS. I have my own personal space up there. Now, I think it's my own personal space, because there's at least five Scotts that work at the company and a number of women with the last name Scott. So there could be seven potential Scotts that are going on here.

But I think this is my --

(Break for direction.)

**SCOTT GUTHRIE:** Let me add some value.

(Laughter, applause.)

**SCOTT HANSELMAN:** I'm Scott. I'm new here.

(Laughter.)

So I've got my own space here within Azure Dev Spaces and this is really significant. I want to point this out. Notice how hotel is bold-faced that's the microservice that I want to debug. That's not the website. It's a microservice as part of the website that then talks to other microservices, like the profile at the backend database.

When I hit F5 and go and debug this, it's going to do a lot for me. I want to point out that I don't have Kubernetes installed. I don't have any virtual machines here. This is not running here. It's running in Azure Dev Spaces. And this is my own private space. I'm just running that one hotel.

Now, we've seen demos before when you hit F5 and then a miracle happens. But I really want you to feel this in your chest. We are not hitting F5 on the website. I just want to debug that one microservice. I'm going to compile all of this into containers, put it into Azure Container Registry, drop it into Kubernetes, and then when we hit the URL we'll take a look at the URL and you notice that it says, Scott, that's my space, but I'm not affecting anyone else in my team, because I want to make sure that I get in an environment that feels like production, but isn't.

I type in Seattle and now we're going to hit a breakpoint. And I can hover over it. And I get the great experience in Visual Studio as if I were local, but I have my own private space with dozens of microservices all working together and I just pressed F5.

(Applause.)

So what did we see? We have a fully managed Kubernetes. We've got auto-patching, scaling. We've got upgrades. We've got an integrated container help. I searched terabytes of logs instantaneously. I've got the ability to rapidly debug and fix apps in a complicated environment like this with just Visual Studio and my Azure subscription. And thanks to AKS and Azure Dev Spaces, Azure has the best Kubernetes experience in the cloud.

Thank you.

(Applause.)

**SCOTT GUTHRIE:** Scott, I've been joking for years that we've got to do some kind of pair demo in a keynote and I guess we just finally did it. So that's awesome.

(Laughter.)

So, you know, as you look to modernize your applications and adopt a microservice based architecture, one of the things you ultimately want to take advantage of is serverless-based computing. A serverless-based approach allows you to execute code in an event-driven way, which can help guide you to build applications that can scale better, but serverless computing can also help you save money by enabling you to avoid having to pay for server resources that you might not be fully using.

And Azure enables you to really build a serverless based solution that can respond to really millions of events per second. Our Azure Function service enables you to execute on-demand serverless code written in a wide variety of different languages, including C#, JavaScript, Python and Java. And our Azure Logic Apps service enables you to execute on-demand serverless workflow.

These workflows can use more than 100 built-in data and app connectors that we provide out of the box and integrate your workflows with data and actions that are in Office 365, or Dynamics 365, Salesforce, SAP, Adobe, Twitter and more and secure processing around data across long-running processes. These workflows can also then invoke Azure Function now, giving you a really robust way that you can integrate both code as well as declared workflows together.

One of the new capabilities of Azure you'll hear us talk a lot more about this week at Build, is our Azure Event Grid service. Event Grid provides a routing service that makes it easy to route events from a wide variety of different event sources and connect them to event handlers to process them. Event Grid guarantees reliable event delivery. And it will automatically queue, for example, messages for later delivery if your event listener is not available. And it can scale to support millions of events per second.

And it's easy to build really robust, event-driven architectures that enable you to eliminate polling and the associated costs of latency within your application. And it runs both in the cloud with Azure, but you can also, it's included as part of our Azure IoT Edge, as well as Azure Stack offerings.

Now, you can set up Event Grid routes programmatically using code. One of the cool things we're also doing starting this week is making it crazy easy to set up events using the Azure Portal, as well. For example, this is a screen shot of a storage account in the Azure Portal, so our standard Azure Blob Storage service.

And you'll notice now we expose an event link on Azure resources directly inside the portal. If you don't have any events registered when you click that link we actually provide sort of a nice, helpful UI that enables you to easily create and run serverless Functions or Logic Apps that will execute when a particular event occurs.

For example, here this sample tutorial is built in now. We make it super-easy to create and wire up a serverless Logic App workflow that can execute, for example, every time a video file is uploaded into my Azure storage account. Here's what the Logic App workflow looks like. You can see you kind of composed this workflow graphically, using our designer and notice how with the built-in workflow it's taking the uploaded file and in turn calling an Azure Cognitive Service that we call our Video Indexer service that will automatically transcribe the speech in the video into a text file into that storage account.

And the great thing is I didn't have to write any code. I didn't have to spin up any VMs or other services. Instead it all executed serverlessly using my declared Logic App workflow. This is just one of literally hundreds of new Event Grid and serverless scenarios that we're making super-easy to accomplish this week and all of our Azure services will expose events that can be managed using Event Grid with this type of serverless composition model.

Now, when this container and event-driven serverless technology becomes incredibly powerful is when handling new types of massive scale solutions, like IoT. Now, IoT is now experiencing just exponential market growth. Two years from now there will be more than 20 billion connected devices in the world. That's approximately three times the number of human beings on the planet. And the need for developers who can build IoT-based solutions is rapidly growing.

We already have thousands of Azure IoT-certified hardware devices in the market today, supported by hundreds of different partners. And what's great is that means as a developer you can easily build Azure IoT-based solutions and take this hardware, take these solutions and easily integrate them with our Azure IoT cloud services. And let's take a look at how we can do this.

So our Azure IoT Hub service enables you to securely connect and manage up to billions of IoT devices. It was the thing that was on the backend when Sam was showing the drone, for example, in Satya's keynote this morning. What's great about Azure IoT Hub is it can work with any IoT device out there and includes built-in security management capabilities.

So for example, you can tell it to patch the device automatically, as well as device screen support, so that you can actually query and update the configuration of any device, even when the device is offline. And when it comes back online you'll automatically update. And as a developer it provides a great, event-driven, serverless programming model built on Azure Functions, that enables you to both ingest telemetry from any device, as well as interact and program it from the cloud.

Since I bet many of you here have never built an IoT app before, you're probably wondering, okay, it sounds good. How easy is it to do? What I would like to do is invite Jeff Hollan on stage to show off literally from scratch how easy it is to build using Azure IoT Hub and our Azure Serverless Functions programming model.

So here's Jeff.

(Applause.)

**JEFF HOLLAN:** Thank you all very much and thank you, Scott.

All right. So I want to show you just how easy it is for every single developer to get started building applications across IoT and the cloud. And to do that Scott has given me a challenge to build an IoT application from scratch live in front of all of you now in less than five minutes. And the application that I want to build is going to center around this IoT button.

Now this is a very simple IoT device created by one of our partners. And how it works is that whenever somebody presses this button it automatically wakes up, connects to Wi-Fi and sends some data up into the cloud. Now the application that I have to build centers around Twitter, so I'm showing here a Twitter account, #AzureIoTDemo if you want to follow along. And what I need to build is an application so that if you're at the conference and you use this button or buttons just like it and you see something awesome that you can press this button and let the world know through a tweet that something awesome is happening and where they can go to check it out.

All right. So with that, let's start the clock and see what we can do. All right. The first part of this application is how these devices securely connect to the cloud. And for that I'm in my Azure portal and I can create a brand new IoT Hub. IoT Hub is an all-in-one service in Azure that allows me to connect, configure, manage and monitor all of my IoT devices at massive scale.

Now I already have an IoT Hub in my subscription, so let's open it up here and see what it's doing. You'll notice that there's actually a number of devices already connected to this hub, and it's even been sending some data throughout the day today. And I can come in here and drill into the devices that I've paired with this IoT Hub. And I want to point out right here at the top of this, this is my IoT button in my hand.

Now we need some code to run whenever these buttons are pressed. So to do that I'm going to come here into Visual Studio and create a brand-new Azure Functions Project. This will enable me to have serverless code run on demand to all of these IoT events, and serverless and IoT pair great together because as I continue to add more devices and data, my application automatically scales. So in this new project let's go ahead and add a new function. You'll notice there's actually a number of triggers that I can use to start this code, but I'm just going to use this IoT Hub trigger right here, and let's associate it with my IoT Hub and my subscription.

Now here's where I can write whatever code I want to run in response to events and to prove this out, let's just change this line of code and have it say, button press, and we'll click save here, and click run. Now, I'm going ahead now, and what's powerful with Azure Serverless is that I can actually run Azure Functions locally on my machine and debug it. In fact, I've gone ahead now and just pressed this button, and you'll notice that my function on my PC was able to securely connect to IoT Hub, see that trigger, and run my code. In fact, just to prove it again, I'll press this button again, and you'll notice here we got another log message. So I can very easily debug and make sure my code is working before I deploy it.

Now there's two pieces of information that I care about for my application to work, and the first is the location of the device. And that's because I want to include that location in the tweet of where something awesome is happening. And the second thing that I want to know is the type of message that's being sent. As you notice, my IoT Hub is actually filled with a bunch of devices sending lots of data. And before I go and send a tweet, I actually want to make sure that the type is, in fact, equal to a button press.

All right. We've written about six lines of code here. This is where I know I need to post a tweet. But I've only got about two minutes left, there is no way that I can look up the Twitter API and figure out how the crap to grab an OAuth 1 token that that amount of time. So instead what I'm going to do is leverage another piece of Azure Serverless which is Logic Apps.

So I'm back here in the Azure portal, and here I can create serverless workflows. Now I'm going to add an action here, and what's powerful is that not only can I add other Azure functions that I've written, but there's over 200 connectors here out of the box to enable me to integrate with services like Cosmos DB, Cognitive Services, GitHub, Google Docs, the Microsoft Graph, even on-premises SAP.

Now this is incredible for me as a developer because this is all function code that I don't have to write, so that if I want my application to do something like post a tweet to Twitter, I just select the Twitter connector, say I want to post a tweet, I've got some text here ready to go, let's paste this in. And we'll even pass in the location of that device.

All right. So that's all we need for the workflow. The last thing I need to do here is wire it up to my Azure Function. So let's grab this URL here and write one last line of code. And if you think it's hard to write code when someone is looking over your shoulder, imagine doing it in front of thousands of people. That's okay. Make sure to make this ratable. I'm going to add this. Quick save, run, time.

All right. We've got time to spare. Let's test this out now. I'm back here at Twitter. I'm going to go ahead now and press this button. The Azure Function is kicked off and triggered, it's sent the location data to that Twitter connector, and when I click refresh here, you will see here now is a tweet sent to the world letting them know that something awesome just happened here at Build. Thank you so much. (Applause.)

Now what's great here is this even includes a sample on GitHub, so that if you go check out this tweet, you can build the same solution in a few minutes later today. And I was able to build this entire application that could now scale to thousands of IoT applications in less than five minutes.

Thank you so much.

(Applause.)

**SCOTT GUTHRIE:** One of the things that Jeff mentioned, if you go to our booth, we're actually giving out thousands of these Internet connected buttons. And so it's super-easy to build this exact same app yourself, and you can become an IoT developer.

So, you know, Jeff showed you how easy it is to connect a simple Wi-Fi connected button to Azure and how easy it was to basically write code in the cloud that interacted with him. And this is great for many, many different scenarios where you really only need to run the logic in the cloud.

But as Satya talked about this morning, the trend that we're seeing is where we're also increasingly moving to one where you also want to be able to run logic on the edge as well and interact both with the device and the environment in even richer ways and be able to do it locally while still retaining cloud connection.

Azure IoT Edge runs on both Windows and Linux devices and provides a really easy way to build and deliver application functionality on the edge. It supports a container-based programming model that you can use to both develop and deploy apps on the edge and supports running serverless based Azure functions directly on the device. In fact, we've taken the Azure functions programming model and it's just part of Azure IoT Edge, as well as Event Grid.

And this enables you to use the same container and serverless skills and technologies that we just talked about running in the cloud to be able to run on the edge device as well.

And you can basically deploy and run both code, as well as your own AI models using our Azure Cognitive Services or by building and training your own models, which we'll talk about in a few minutes, on the edge as well and run it in the context of an Azure IoT Edge device. And this gives you both an incredibly powerful and a very elegant way to build IoT solutions.

What I'd like to do now is hand it back over to Jeff to show off how you can take this first IoT scenario that we just demoed and now extend it to build and deploy custom code and AI models using Azure IoT Edge.

**JEFF HOLAN:** Thanks, Scott.

All right, so let's extend that application we just started, and what if instead of requiring somebody to press a button when they see something awesome, what if the device itself actually knew that there were awesome things around it and automatically sent a tweet.

Now, to do that I want to talk about Azure IoT Edge. I'm back here in my Azure portal, and this is my IoT Hub. This is the same hub we were just using for my button. But I want to call out here that there's a section here for my IoT Edge devices.

Now, these are devices that actually run pieces of the Azure cloud directly on the device itself. And these aren't fancy devices that you have to special order, these are devices that you're likely already using today. In fact, two that I want to call out here is my Windows PC that I'm developing on right here and this Raspberry Pi that's next to me, which is actually running Linux.

Now, both of these are edge devices and both of them have cameras. So I want to leverage those cameras to be observing the environment and figuring out if there's awesome things around.

So let's check out the solution and what it looks like.

Now, this inside of Visual Studio code is my IoT Edge solution that I can completely build and manage here.

A few things I want to call out. This same solution and same code is running both on my Windows PC and on this Raspberry Pi running Linux, and it's made up entirely of containers.

Now, this is great because it means as I'm building these, I get the benefits that containers bring like portability and consistency, regardless of the form factor that it's running on.

Now, this solution is just three simple parts. The first one is this startup container. I'm just going to start it up on the device, grab the camera feed, and push that feed to a machine learning model.

Now, the second piece here, this is that machine learning model that needs to identify are there awesome things around or not.

But what's exciting here is just how easy it was for me to build this. In fact, to build this model I leveraged the Azure Custom Vision service. This is one of the Cognitive Services that enables developers to upload samples of images with tags, and after you've trained the model, you can actually upload new images that it has never seen before, and it will be able to predict for you what this image should be classified like.

So let me show you the project that I've built for this demo. You might be able to see here from the text here is my custom vision portal, and I want to build my own Scott or Not app, because chances are if Scott is somewhere at Build, there's something awesome happening.

So here's the sample images I've uploaded. It's a wall of awesomeness. Some of these images are tagged with Scott and some of them are tagged not. And I can go ahead and tag these and then click this train button right up here.

But I don't want to have to be sending the feed from all of my devices constantly up into the cloud to identify if it's Scott or not.

And what's great here is that after training the model, I can actually click this export button and export this model to run on any device, even locally on the IoT Edge.

So I've gone ahead now, I've trained this model. Let's choose that we do want to run this as a Docker container, say I want this to be Linux, click download, save, done. This is the exact container that I have now imported into this solution, running now on these devices, looking at that camera feed and identifying is this Scott or not.

Now, that's generating a lot of data, and in order to deal with that data, I have my third piece, which is an Azure function running on a container on the device.

Now, I actually want to show you Azure function code because it should look familiar. This is the code here and this is the exact same code that I wrote for my button a few minutes ago. The only difference is that this is running on the device locally and you'll notice instead of looking if it's a button press or not, I'm actually detecting if it's Scott or not.

Now, I've built this whole application, I've deployed it and works on my Windows PC. So now let's talk about this Raspberry Pi that I also want to run it on.

If you're not familiar, this is a Raspberry Pi right here. It's a computer the size of a credit card. It has an ARM processor, it only has a gig of memory. It can run Windows IoT core but for right now I'm actually running Linux on it.

Now, I want to pull in all of that power of the Azure Cloud and run it on this simple device right here. So let's test it out now. I want you to pay attention to these lights up here, because this is going to show us what the device is thinking.

So here's the camera feed. I'm going to go ahead here and give a nice smile for the camera. And you'll notice that right away, this device has rejected me with a red X, letting me know that I am not, in fact, awesome or Scott. Getting flashbacks of high school.

So we'll go ahead now, and Scott, if you will join us, let's see if we can get something awesome to happen, if you'll go ahead and give a nice smile to the camera. And you'll notice that right away this device has seen something awesome is happening. Thank you. (Applause.)

It's actually posted another tweet with Scott's picture. Hopefully, it was a flattering shot that it took of you. Which also includes a link to this code.

Now, that's fantastic and just an example how as a developer I can use these powerful tools to build applications at massive scale on the Azure Cloud. But when I need to, I can actually pull pieces of that cloud down and run with that power on a device as simple as this Raspberry Pi.

Thank you so much. (Applause.)

**SCOTT GUTHRIE:** Thanks, Jeff.

So everything you've seen in all of these IoT demos is now available for you to use starting today, and you can build incredibly powerful IoT solutions using Azure and really take advantage of this intelligent cloud, intelligent edge world.

Now, one of the most important decisions you'll make as you start to build large scale cloud applications, whether it's for IoT or for any other scenario, is how to store and represent your data. So let's spend some time talking about how you can do that with Azure and some of the cool new things coming out this week.

Azure now provides an incredibly flexible choice of operational data services. Our Azure SQL database, Postgres, MySQL, Redis and Cosmos DB services are databases that we provide as a service, which means we provide them as a database service with an SLA, and it's a fully automated managed service. This means you don't have to manually configure your databases for high availability or have to manually patch or update them. These are capabilities built into the services that we provide for you.

We similarly provide built-in backup, as well as point in time restore capabilities, the ability to easily scale up and scale down capacity at will very quickly, and a whole bunch more capabilities that are just built-in features that you can leverage. This enables you to be much more productive and build applications faster, while still preserving the flexibility of your favorite database of choice.

At last year's Build Conference we released Azure Cosmos DB, which is the world's first globally distributed, multimodal, NoSQL database service that delivers horizonal scale-out with guaranteed single-digit millisecond latency.

We've seen incredible adoption of Cosmos DB over the last year, and it is today one of our fastest growing services across all of Azure.

And the reason for this incredible adoption is because it really enables you to build amazing cloud solutions.

With Cosmos DB you have a horizontally scalable database that puts data everywhere your users are.

Cosmos DB enables you to automatically replicate data across all Azure regions around the world, giving your users lightning fast performance regardless of wherever they're accessing your application.

Cosmos DB allows you to scale your storage and performance throughput across one or multiple Azure regions with zero application downtime.

You can start with just say gigabytes of data and then scale up to manage petabytes of it.

You can start by processing just say a hundred operations per second, but then scale to tens of millions of operations per second around the world.

And best of all, with Cosmos DB you pay only for the storage and performance throughput that you provision.

And Cosmos DB is unique in that it allows you to program against it using a variety of different APIs and data models. We support MongoDB, Gremlin Graph, Table, SQL APIs, Spark APIs, as well as Cassandra APIs. And this gives you maximum programming model flexibility and enables you to easily reuse code in libraries that you already have.

This week at Build, we're announcing a bunch of great new improvements to Cosmos DB. These include new pricing options that help developers achieve in some cases up to 10x cost saving over their solution. And it also includes an incredibly powerful new capability which is multi-master write support.

Cosmos DB now supports unlimited read and write scalability by virtue of a highly decentralized master-less replication protocol support. This guarantees single-digit millisecond reads and now writes response time at the 99th percentile anywhere in the world, which is something no other database in the world delivers today.

And with its unique adoptive entropy protocol, Cosmos DB significantly reduces write conflicts, which makes it much easier to build multi-master write scenarios in applications.

In the event of a write-write conflict, Cosmos DB supports both a well-defined, policy-based programming model that's built into the transaction system, as well as the ability for developers to add their own custom logic conflict. These are incredible innovations that really enable you to build truly planet-scale applications.

And what I'd like to do now is invite Rimma, who's one of the core architects of Cosmos DB, onstage to show off an example of how easy it is to use this multi-master write support to build amazing cloud solutions. Here's Rimma.

**RIMMA NEHME:** Thank you, Scott.

Good morning, everyone. (Applause.)

It's really, really exciting to be at Build here this year.

Now, imagine you want to build a planet-scale intelligent app serving millions of users all around the world. In this demo I'll show you how easy it is to take any app and scale it globally using multi-master capability in Azure Cosmos DB.

Now, to show you the benefits of multi-master for global-scale apps, we have built a multiuser interactive canvas with millions of pixels, inspired by the Ready Place experiment.

This app is running across all Azure regions with the data stored inside Cosmos DB and replicated across all Azure regions as well. The app needs to deliver instant writes and instant reads to millions of users worldwide.

In this demo I'll show you two canvases, one pointing to replicate in West U.S. and another one that's pointing to a replicate in Japan West. Users in those regions can draw and view each other's drawings in near real time all over the world.

Now let's see this in action.

So I will go to the canvas in West U.S. and pick a color. And while I'm drawing, Ben is going to be drawing in Japan West. And I can see Cosmos DB, and as you can see, we can see each other's drawings in near real time. I can even draw maybe a tic-tac-toe game and play it with Ben.

So every single update from every single user gets written to the nearest region and becomes available all around the world in near real time.

This canvas is stored inside Cosmos DB. So let's take a look at how this data gets stored inside Cosmos DB.

I can come to the portal, click on the data explorer, and view this data, all of the pixels, the locations of the pixels and the colors that get chosen by the users.

Now, we're very excited that right after this keynote, we're going to open up this application to all of the users worldwide so all of you could participate in this globally distributed drawing experiment, powered by the multi-master capabilities in Azure Cosmos DB.

Now, remember, this is not just a regular app, this is a globally distributed application. Building such apps is really, really challenging. Azure Cosmos DB has been natively built in the cloud service, was carefully engineered with multitenancy and global distribution from the ground up.

With its turnkey global distribution capability, I can come to the world map, I can pick the regions where I want my data to be, click the "save" button, and that's it. The data gets seamlessly replicated and becomes available for querying locally in those regions.

And as I'm adding and removing those regions, my application does not need to be paused or redeployed, it continues to be highly available thanks to the multi-homing capability that Cosmos DB provides.

Now, the key thing that I want you all to remember is that independent of the data scale, whether you're working with a petabyte, or hundreds of terabytes of data, independent of the data model or the data distribution, or potential failures that may occur when you're running on such a massive scale, Cosmos DB continues to provide you with a single-system image for all of these globally distributed resources. This is something that no other database in the world can offer you today.

Now, the multi-master. And not just any multi-master, but the multi-master at true global scale. With its natively built-in multi-master capabilities in Cosmos DB, applications developers get numerous benefits. No. 1 is you get unlimited and elastic writes scalability all around the world. You can perform trillions of writes and reads every single second on the single globally distributed Cosmos Database, whether it happens to be a graph or a table or a collection of documents.

With the natively built-in multi-master support, you get five-nines write availability, all around the world, in addition to the five-nines read availability backed up by the industry-leading SLAs.

With the natively built-in multi-master capability, you get a single-digit-millisecond write latency guarantee at the 99th percentile all around the world.

Now, databases that offer masterless replication can give you low-latency writes, and they can give you high write availability, typically trading it off for consistency.

What is truly unique about Cosmos DB multi-master replication is that it composes incredibly well with the multiple, well-defined consistency models that the service offers you. So developers can make precise, intuitive, and well-defined tradeoffs with respect to latency, consistency, high availability in Cosmos DB.

With databases that support multi-master replication for active-active workloads, write-write conflicts are rampant. What that typically means, multiple users could be drawing to exactly the same pixel at the same time. This forces developers to add that logic to their application and participate in the multi-master replication protocol.

In Cosmos DB, we wanted to remove all of that burden away from the developers. So Cosmos DB, in addition to its anti-entropy protocols, it tries to reduce the number of record-level, write-write conflicts down to the minimum, but indicates when the conflict occurs, developers can pick among multiple flexible conflict resolution policies.

Cosmos DB supports automatic policy with the nature CRDT support, you can also write your logic in the form of a stored procedure as a conflicts resolution policy or use last write wins.

Now, in conclusion, with its highly decentralized, multi-master protocol, multiple well-defined consistency model, and the capability where the developers can add and remove any number of regions in a scale-independent manner, Cosmos DB becomes a ubiquitous choice for the modern application development in the intelligent cloud, intelligent edge era.

Every single app that you're building could transform the world that we're living in, and if your app happens to need a database, we sure hope it's going to be Cosmos DB. We're very excited to see what you're going to build with Cosmos DB. Thank you so much. Back to you, Scott.

(Applause.)

**SCOTT GUTHRIE:** Thanks, Rimma. So let's now drill into how you can take all of your data and use it together with AI to add true intelligence into your applications.

Now, our goal with Azure is to enable every developer to easily add AI to their applications. With our Azure Cognitive Services, we enable you to easily call pre-built AI models that we expose through API Services.

As you heard from Satya this morning, you know, we provide great AI models to do things like speech-to-text translation, image detection, object detection, video translation, language services and more. And you can invoke all of these using either our REST APIs or by using our pre-built .NET, Java, Python and Node SDKs.

Now, in addition to calling these Cognitive Services programmatically, one of the cool new capabilities that we're making available today is native integration of Cognitive Services with our Azure Search Service.

Azure Search enables you to index any type of content and build rich experiences from it. And with this Cognitive Services support, you can now, for example, just automatically index images, videos and PDFs, extract all the -- use AI to extract all that insight from it -- things like face and object detection, sentiment analysis and video transcription, and just expose it directly as part of your search experience.

Now, we've worked with some great customers using our Azure Cognitive Services. One of those is the NBA, who are using it to deliver some really great experiences. And please join me in welcoming Garth Case from the NBA to talk about the work they're doing with us and what this experience enables. Here's Garth.

(Applause.)

**GARTH CASE:** Thanks a lot, Scott.

**SCOTT GUTHRIE:** Good to see you.

**GARTH CASE:** So I'm going to start out with a question: Show of hands. How many NBA fans do we have in the room? OK. That's a reasonable amount.

So my name is Garth Case. I'm in my 20th year at the NBA. There are two questions I get asked most frequently. The very first one is: Have I met Michael Jordan, Shaq, Kobe or LeBron? I'm lucky enough to say yes to all of that, I have.

The second question, though: Why have I stayed so long at the NBA? The answer might seem cliché, but it truly is because of the people. Working for the NBA, I've had the opportunity to meet and learn from the best and brightest in sports.

Now, for those of you that don't know, during the NBA regular season, 30 teams play 82 games. Each game generates terabytes of content. As you can imagine, this much information makes it difficult to manage and curate and get insights delivered consistently across all our internal and external channels.

Now, let's talk about the NBA's journey to finding a solution to our growing data problem. Eighteen months ago, when we first started to organize and understand our data, we started like most -- thinking that we could build a data lake and that would solve all our problems. But our data lake quickly became a "data swamp," and it was difficult to hear important signals in the data through all the noise.

Even though we had designed the perfect analytics layer, we needed something more. We needed a learning layer.

Analytics layers are usually active post-ingest of data. But a learning layer could be active through all stages -- from ingest all the way to archive. It would help us organize and graph our data, surface patterns that connect dots and allow us to focus on the high-value information. We would now be able to hear the signals loud and clear.

This pivot in our journey meant that we were no longer building just a data lake, we're building a smart data platform.

This revelation gave birth to many use cases and scenarios, most of them related to data mining, process automation and decision intelligence.

As I searched for technologies that we could use to build our smart data platform, I saw a demo where Microsoft used an AI-driven approach to understand the recently released JFK files. They were able to run thousands of unstructured documents through their system and build a knowledge graph using a set of AI models and services. Their use of new cognitive capabilities in Azure Search, as well as leveraging custom and pre-built AI, appeared to be the perfect recipe for a cocktail that would enable us to understand our content and add the layer of learning and intelligence we needed to build our smart data platform.

Now, as we built the platform, we also developed plenty of value-driven use cases and scenarios. One of these use cases in production today is the application called NBA Photo Sorter. The name is not terribly creative, I know that, but please feel free to ping me if you have any suggestions.

Before the Photo Sorter was created, the effort of tagging photos took a very, very long time, was extremely manual, resource intensive, and sometimes inaccurate.

Now with the help of Azure, photos and other types of content can be automatically tagged and related quickly with increased accuracy. The Photo Sorter has already aided us in quickly tagging VIP photos from our events. These photos are turned into printed and digital albums which are given to VIPs as a personalized gift.

But this example is just the beginning. As we get more insight from our data, I see a future where we can not only further automate our business, but use the system to connect our channels and drive personalized, value-based engagement with our fans across the world.

Together with Microsoft, we've embarked on a journey to bring AI into the world of the NBA, across all types of data and content. My intent is to use AI to power the next generation of experiences for employees, our teams, fans, everyone around the world.

I have one plug before I turn it over so you can check out the Photo Sorter. At this very moment, we are in the semifinals of the NBA Playoffs, where you can experience the excitement of watching the best athletes in the world compete for the title of NBA champion. I hope that you all will tune in and share your excitement with me.

Now, I'll turn it over to Paige Bailey to do our demo, thank you.

**PAIGE BAILEY:** Awesome. Thanks so much, Garth. (Applause.)

So as we just learned from Garth, the NBA truly is a data-driven organization. And here we've ingested all of the NBA's content -- so videos, images, text documents, player telemetry -- into blob storage on Azure, and we've applied Cognitive Skills to enhance and annotate that data.

So I'm from Houston, which means that the only James I care about is James Harden, but I understand a lot of folks on this coast like LeBron. So let's see what our Cognitive Skills can tell us about LeBron.

Here, our face API has automatically detected LeBron in this image, and it's been trained on all of the players, owners and coaches in the NBA. So not only do we have his face, we also have lots of other additional metadata such as emotion and age.

So how does "The King" feel as he's about to score an epic shot in front of tens of thousands of people? Neutral. (Laughter.) That was completely in the zone, and this guy looks like he's about to be sad in about five seconds.

But it's not just about LeBron; how does he relate to other entities in and around the NBA? Let's take a look at the Knowledge Graph that Cognitive Search has automatically created for us.

So this is interesting. It looks like our AI Skills have detected that there's some sort of relationship between LeBron and Nike, so let's explore that a little bit further.

Here, our object detection API, which has been trained on all sorts of items that you would expect to see in the NBA, so basketball, players, shoes, and even specific brands of shoes, have determined that LeBron, apparently, loves wearing Nike. So now we know a little bit more about his fashion sense.

Let's try and take a look at his performance. So how is he as a defensive player? We get back a short video that our video indexer API has automatically annotated and transcribed showing LeBron make a defensive play. So he's good at offense, he's good at defense, how is he as a player all-up?

That's usually done with something called Game Notes. These are PDF documents created after every single NBA game, and they contain a wealth of information. So player stats, scores, play-by-plays, and we could look through every single one of these PDFs to try to build this player top-performers column here on the right.

But since we've already extracted all of the text from the tables in the documents themselves, we're able to build a customer algorithm with Azure machine learning to extend Cognitive Search.

So this all sounds kind of like magic, right? Like being able to apply Cognitive Skills to the data that you've already got in Azure. So how easy would it be to go about creating this solution for yourself? Let's step into the portal and see.

So if I wanted to create a search service, all I would need to do is go to create resource, Web, and then click Azure Search, or you could just search for search, I guess.

But since I've already created one, let's take a look at the NBA content search service that's powering our application.

If I wanted to import data, I would connect to my data source, and I can use a variety of things. So SQL databases, SQL Data Warehouse, Cosmos, Azure Blob Storage.

And if we take a look at our existing data source, this NBA content, we get pulled up the container with all of the items that we've seen before. So our videos are automatically being annotated, being transcribed. Our PDF documents are having the text extracted and the images automatically documented and described. And we can even apply Cognitive Skills out of the box to extract people names, locations and languages. And as mentioned before, we can also extend this Cognitive Search with custom machine learning logic.

Once this has all been created, we can go and look in the search explorer and search against all of the returned JSON.

So I type in LeBron, we get back every single instance. I can even do more complex queries, so search equals LeBron, and facet equals organizations -- one of those things that we just clicked before, the little checkboxes.

And here we get back all of the organizations that LeBron is associated with -- the Cavs, NBA and Nike.

So these cognitive capabilities, being able to apply AI to the data that you already have stored on Azure, the capability is only available with us on Microsoft Azure Platform. And they're available for public preview starting today.

I can't wait to see what you create with this AI-first approach to cognitive understanding.

Thanks so much, guys.

(Applause.)

**SCOTT GUTHRIE:** The great thing about Azure Cognitive Services is that you don't have to be an AI expert in order to take advantage of AI within your applications.

We also know that a lot of you want to be able to build your own AI models and tailor them to your precise business needs. And we provide a great way to do that on Azure.

So when you typically want to build an AI model, you typically walk through a workflow where you do three things: Step one is where you prepare your data. Step two is where you build and train an AI model from it. And step three is when you deploy the model and start using it within an application.

Let's walk through how we're trying to streamline all of these things using Azure.

I'm going to start with a simple example to illustrate these three steps. And that is trying to build an AI model that can help answer the question of how much a particular car is worth.

So step one of the process, again, is to prepare my data. Now, data scientists building AI algorithms spend, on average, about 80 percent of their time doing data preparation steps. Working from multiple data sources and merging the data together, finding and fixing anomalies and outliers, and standardizing on formats. It's a really arduous task, and a lot of people often refer to it as "data wrangling."

You know, Azure Databricks Service is an Apache-Spark-based analytics service that's optimized for Azure. It enables you to quickly launch and scale up Spark clusters on demand. It includes a rich, interactive workspace that makes it easy to build Spark-based data workflows and also includes built-in data adapters that allow you to work with all the different data that you store within Azure.

And it makes it really easy to prepare data for AI-based workloads, whether you're just working with a few gigabytes of data, all the way up to hundreds of petabytes of data.

Now, once we've got our data assembled for our car model, the next step is to build and train an AI model using it.

Now, for example, for our car scenario, we might want to create and train a model that predicts the value of the car using a data set of historical car prices. And this process of building, training and testing the model is going to be iterative as I evaluate various different attributes and hyper parameters to more accurately fit the model as part of my solution.

And our Azure Machine Learning Service helps dramatically with this process. It enables you to train and evaluate models on any number of servers within Azure. You can immediately scale up from using just one server to build and train your model to immediately running across hundreds of thousands of servers with just a single parameter change in your application.

Azure Machine Learning enables you to use every popular data science and AI framework out there, including TensorFlow, Caffe2, CNDK, Keras, PyTorch and more.

In addition to every popular AI framework, we also include pre-built AI models with Azure, specifically designed for computer vision, text and forecasting scenarios. This can dramatically improve both the performance of your AI models, as well as the speed with which you can go ahead and build solutions with them.

Now, once you have built an AI model with our Azure Machine Learning, you can then package it up into your Docker container and deploy it to run anywhere.

With our Azure Machine Learning Service, you can build and deploy AI models that literally run anywhere. This includes our new Azure Kubernetes service you saw Scott Hanselman demo earlier, which is great for hosting an AI model that you used for online API scenarios, or Azure Batch Service if you want to iteratively batch base process lots of data. Or you can deploy it, as you saw Jeff show, on IoT Edge for edge-based computing scenarios.

And because these models are container based, you can also run them anywhere else as well, including on-premises environments. This gives you maximum flexibility to use AI literally everywhere.

And the great thing is you can take these three-step flows and use it with any data stored inside Azure and make any application you work on much more intelligent.

What I'd like to do is invite Starbucks -- who is another great customer of Azure -- on stage to talk about how they're leveraging both our Azure Data Services and AI Services to transform their customer experiences.

So please join me in welcoming Jeff Wile on stage here to talk about Starbucks and the great work they're doing. Here's Jeff. (Applause.)

**JEFF WILE:** Good morning. Can I borrow that?

**SCOTT GUTHRIE:** Sure.

**JEFF WILE:** Well good morning, everyone. It's a pleasure for me to be here, hope you're having a great morning so far. This is great.

And it is my pleasure to have the opportunity to spend a little bit of time sharing about how an iconic brand like Starbucks is being transformed by many of the same technologies that we've heard about this morning.

Many of you know us, but let me share just a little bit more about what Starbucks looks like today.

We have 28,000 stores around the globe, over 300,000 partners who proudly wear the green apron in 77 countries around the globe.

All of this equates to over 100 million occasions each week where our customers visit a Starbucks store, and that equates to about $22.5 billion of revenue last year. We like to say at Starbucks that we earn that $22 billion $5 at a time.

So how does all that happen? Well, our company has been built over the last 40 years on three really important pillars. No. 1, hire great partners who care passionately about the craft of coffee and who care about the customers that walk into our stores.

No. 2, build innovative and exciting handcrafted beverages and unique food offerings, such as our brand new Nitro Cold Brew. If you haven't experienced it yet, it's life changing, I promise you'll love it. And, finally, stores that are welcoming and a great place to refresh and relax, which many of you have come to call the "third place."

But we've added a fourth pillar recently, and that is technology, because we believe that technology is the enabler at Starbucks to spread that experience everywhere. Let's take a little bit of a closer look on what that technology is doing to transform our systems.

We're looking to use technologies like blockchain to track our coffee around the world from bean to cup. We're using data and modeling to share best practices with our coffee farmers around the world. We're developing systems to optimize inventory so we are delivering the right products to every store at the right time, while at the same time delivering -- reducing waste.

And, finally, we're creating new capabilities that optimize scheduling for partners, those 300,000 partners in the stores, to make sure that they're there when they're needed.

And let's not forget our industry-leading mobile application, where we're leveraging scalable infrastructure that adjusts automatically as demand changes throughout the day.

But probably the biggest transformation for Starbucks as a whole is in our partners, who have been able to innovate faster and build solutions more quickly for our business than ever before.

So what do all these have in common? Well, every one of these projects leverages cloud platforms like Microsoft Azure, which enables us to move faster and build better solutions.

We're also partnering closely with great companies like Microsoft, who not only help us along this journey, but are teaching us as they go.

Finally, another benefit we see from the cloud is our ability to deploy these solutions globally. We can build it once and leverage it anywhere around the world that Azure may be. It really is helping transform our business.

So let me give you a specific example that we're super excited about. Over here, you can see a typical store, and we have orders coming into that store, maybe a mobile order, maybe a drive-through, or maybe someone walking up to our register.

Today, our partners do an amazing job to fulfill and craft those orders, but we do it in the order that the order -- the sequence that the order came into the store. Things get made, things get picked up, and you can see we do our very best to satisfy our customers.

But what if we could use big data and machine learning and AI on something as simple as creating coffee in our stores? And we could us those algorithms to help us optimize how we produce those orders. We still have the same number of orders coming in, but maybe that drive-through order is just a cup of coffee, so we're going to fulfill that first, instead of first-in, first-out like it is today.

And once that's complete, we've now handcrafted that large order for our mobile customer, and it's there ready right when they need it.

We're super excited about this capability because it enables our partners in the stores to focus on our customers and not spend time trying to figure out how to manage that queue. And we're leveraging things like Azure Cloud and the capabilities like Service Fabric, IoT Hub for our connected devices in stores, and back-end data stores like Cosmos DB to enable all of this and make it happen.

Who knew there was so much technology behind a simple cup of coffee? Our mission at Starbucks is to inspire and nurture the human spirit, one person, one cup and one neighborhood at a time. And together with partners like Microsoft and technologies like Azure Cloud, we strive every day to fulfill that mission.

Thanks for listening, and enjoy the rest of the conference. (Applause.)

**SCOTT GUTHRIE:** Thanks, Jeff. So you heard about how Jeff and Starbucks is using Azure to better improve their customer experience. Let's walk through an example of the steps involved to build an AI model for a scenario like this. And to do that, I'd like to invite Paige back on stage to show us how. Here's Paige.

**PAIGE BAILEY:** Hi, again. Awesome. So if you've ever worked in a restaurant, you know it's an art and a science to prepare a complex order at precisely the right moment for a customer to arrive.

And what we have here is a restaurant owner's dream. Orders come in on the right, I click on them to confirm them, and they're automatically placed in the correct order in our order queue. So I'll complete Harvey's order, then Francis's, then Barbara's, and I'm sure that all of them will be ready at precisely the right time as my customer arrives.

So how would we go about creating this solution backed by a deep neural network? Let's step into Azure and see.

So here I am in my Azure portal using something called the Databricks Notebook. And this is an interactive, collaborative programming space for data scientists and software engineers to work together.

Using Databricks, you can pull in data from a variety of sources. So here, we're grabbing stuff from SQL Data Warehouse, Cosmos DB, and also some static CSV files that we have hosted on blob storage.

Once our data has been pulled in, we can prepare it, remove outliers, remove null values, get it into a nice rectangular data format, and then use that to build and train our model using any framework, as Scott mentioned.

So if you like TensorFlow, use TensorFlow, if you like MXNet or Cognitive Toolkit, you can use those, too.

And then once your model's been trained, and once our hyper parameters have been tuned, you can package it up into a container and then deploy it anywhere you choose. So locally on an IoT Edge device, or even at scale on a smart cluster in the cloud. And any developer in your organization can then call that model just as easily as they would a Cognitive Service.

So let's take a look at what that would entail. So, first, we bring in our data. And if I click "shift enter," automatically, we're sending a request to the smart cluster to pull in data from our SQL Data Warehouse. I even have data streaming in live from Cosmos DB. So let's take a look at that here.

Awesome. I can also pull in orders just from blob storage. So here we're grabbing a CSV file.

Now that all the data's been ingested, it's time to prepare it. So we probably don't want to keep these outliers, that would probably skew the model that we're building. So if I want to remove them, I would do orders that total order amount, less than say $200, that seems reasonable. I click "shift enter" again, and automatically my graph changes. So that looks like a much more representative sample.

Now that we've got our data in a great state, it's time to join it and prepare our model. Here, we have a neural network with two hidden layers, and we have 200 nodes per layer. As I mentioned before, this is backed with TensorFlow, and we're using Keras as the developer-facing front end.

So what we can do with Azure Machine Learning is we can train and then change minutely some of our hyper parameters, make small modifications to our model to see how that impacts our performance over time. We're aiming for high accuracy and to reduce loss as much as possible.

So here you see a whole bunch of runs of our model with those minute hyper parameter changes. And lower is better, I know that's kind of counterintuitive for a performance graph, but you want to minimize your loss over time.

So here it looks like that orange model is the best performing one. So that's the one that we'll package up in a Docker container and then deploy to use as the back end for our application.

So we'll create a schema.JSON file to define inputs and outputs, we'll create a Python script to initialize and run the model, and then we'll deploy it using Azure Kubernetes Service.

So now that it's been created and deployed, I click "shift enter" with some prospective restaurant orders, and we get back an output with score categories and confidence levels. So any developer in my organization can call this REST service exactly as they would a Cognitive Service.

As you can see, on Azure, you're capable of using any machine learning framework you choose. You can pull in data from a variety of sources, and you can use Databricks Notebooks as a first-party solution. And we're the only cloud service provider that offers that functionality.

I can't wait to see what you build with AI on Azure. Thanks so much.

(Applause.)

**SCOTT GUTHRIE:** We've covered a lot of new services and capabilities that we're releasing with Azure today, and walked through some of the amazing opportunities that are out there for all of us to build great new applications with it for this intelligent cloud and intelligent edge world.

I hope you enjoy the rest of Build. We're really looking forward to getting your feedback, and I can't wait to see the great applications that you build with Azure and across the Microsoft stack.

Thanks so much, and have a great rest of the conference.

(Applause.)

END