05252022 Build Kevin Scott

Microsoft Build 2022
Kevin Scott

KEVIN SCOTT: The act of human creativity often starts with a blank canvas. Because of our curiosity and ambition, we embrace complexity and create tools to manage it. These tools rapidly get more complicated, and we layer and compose them to solve more complex problems. Over time, we're able to do more and more useful things.

In every era, we need special people to focus and harness our great creative ambition, to make and use the tools that extract the possible from the impossible. Today, those people are you, developers, and these tools are called software. Developers have a special creative magic, and your work allows all of us to turn ideas into action at massive scale.

Soon, software becomes maybe the single most useful tool we've ever had and powers almost everything on the planet. But the mounting complexity means that developer's work starts to become more than you can handle. Your creativity gets weighed down by the parts of your job that are more drudgery than imagination, and the magic that happens as you do your work gets harder to conjure.

Luckily, we have powerful new tools in our arsenal to help us tackle problems of unprecedented complexity. One of these is called AI. AI tools can assist you with your work and give you new ways to unlock your creativity and to get more done. And the creative energy that made software so rewarding to create, starts to return, maybe even more so than before.

But the very best part is that AI makes software so easy to use and to create that everybody, not just developers, gets to build things and unlock their creativity because of it, and the world becomes generally a more magical place to live in.

Hi. I'm Kevin Scott. Although some of it is just beginning to happen, the story you just heard is true. It's a story of software, developers and how AI is going to make all of our lives easier in the very near future.

Today, we're going to meet some people who are already using software and AI to make the world more helpful and magical. We're going to talk about how Microsoft and our partners are working to create the most powerful software and AI tools we've ever seen. And we're going to give you access to some of those tools so you can try them out yourselves.

I'm here at Microsoft's beautiful new Silicon Valley campus. After the past couple of years, it's super exciting to see people back working together in person, sharing ideas, and getting inspired to do great things.

Over the past couple of decades, software has increasingly become one of our most important tools to solve the world's problems. But that software means nothing without the human beings who create it.

The people who make software, developers, are one of our most important resources. And even just over the past 12 months, the advancements we've made in AI and large AI models have exceeded what we once thought was possible. It was probably the most exciting year in AI we've ever seen.

Today, I'm going to show you how we're using AI to help developers accomplish more and share how the latest breakthroughs in AI can empower not just developers, but anybody who wants to turn their creative dreams into reality.

In every field and every industry, we're entering a new era where AI is poised to conquer the impossible. Large AI models continue to evolve as platforms that can help us solve our most challenging problems like never before.

Over the last year, a tool called GitHub Copilot has shown us how these platform AI models will transform the way software is created, allowing every developer to do more, and lowering the barriers to creating your own software.

Available for Visual Studio and Visual Studio Code, Copilot is an AI pair programmer that draws context from code and comments you've written, and then suggests new lines or whole functions. It uses AI to transform ideas directly into functioning code using natural language, freeing up your time so you can be more productive and creative.

Github released Copilot into technical preview last year and the response has been incredible. Tens of thousands of developers are already using Copilot every day. In some programing languages, we're seeing over a third of newly written code being suggested by Copilot, and it's already delivering significant impact to developers' workflows.

As Satya shared yesterday, we're excited to bring Copilot out of preview and make it available later this summer. We want all of you to get started with Copilot today, so we're giving free access to all Build attendees until it launches in the summer.

We believe we've only seen the beginning of what tools like Copilot are capable of. Earlier this year, GitHub announced Copilot Labs, a Visual Studio Code extension for experimental applications of Copilot.

Every developer learns that writing new code is only a part of software development. Often the hardest job the developers face is reading and understanding code. Whether it's because it's an unfamiliar codebase or whether the code in question is using a library you might not know, the first step is getting a rough idea of what's going on, so you can jump in and use the code or make some changes.

Let’s take a look at one of these projects right now. Copilot Explain. Basically, it’s like Copilot in reverse. Just select some code and you can ask Copilot to explain it to you in plain language.

The possibilities for exploration and creativity with Copilot are practically endless, but the best part is that tools like Copilot won’t just make developers more productive. They will increasingly make coding more accessible to everybody.

Let’s visit a developer who is already using Copilot to create their own software.

JULIAN BAUER: Hi. My name is Julian. I’m 16 years old and I live in San Francisco. I’m part of like the Computer Science Club. We prep for competitions together and then also for the ACSL test together.

I started coding when I was eight. We had this thing called Hour of Code at school and that was fun. I like made a bunch of games, like platformers where you like jump around for platformers.

I learned about Copilot actually through YouTube. I was just watching a coding YouTube channel and it talked about Copilot. So when I saw the examples they gave on the screen, I was very amazed.

The project I was working on at that time was basically a note-taking app, but for debate, where you have to take specific types of notes. At the start of my debate, I didn’t use Copilot. I kind of found out about it midway. And then when I got it, things got a lot faster.

So this is my debate flow app.

The thing I love about Copilot is how it makes you write a lot less code. It makes you write code a lot faster. It basically can sometimes read your mind. It almost feels like the final puzzle piece in a puzzle being put in.

I’m not really sure what I want to do in the future, but I probably want to do something related to coding. So it’s very interesting what will happen next.

[Music.]

KEVIN SCOTT: I’m here at the Microsoft Garage, a unique makerspace with locations around the world that allows our employees to work on the things they’re most passionate about.

Copilot is incredibly cool, and we’ve seen that it’s already transforming how developers of all ages write code. But the large AI models that make things like Copilot possible can not only change how we create software; they can fundamentally shift how we interact with it. And the applications of doing so are nearly endless.

Copilot is built on a breakthrough model called Codex from our partners at OpenAI, which can translate natural language inputs into more than a dozen programing languages. Along with GPT3, it's one of the first models we're delivering to developers through our preview of the Azure OpenAI Service.

Historically, computer programing has been all about translation. Humans have to learn the language of machines in order to communicate with them. But now Codex lets us use natural language to express our intentions, and the machine takes on the responsibility of translating those intentions into code. It's basically a translator between human imagination and any piece of software with an API.

One example where this can help that we all use every day is the web browser. Let's look at a proof of concept demo our friends at OpenAI developed using Codex and GPT.

JACOB HILTON: In the near future, OpenAI's assistive models will be able to translate natural language expressions into action. To give you a sense of this, we've created a demo of an AI assistant that can help users with a wide variety of tasks.

So let's say you're about to start coding. Your first priority, of course, is to find some music. What kind of music is good to listen to while coding? It suggests music without lyrics. I'm in the mood for some romantic era music. Could you suggest some classical composers from the romantic era?

That's a nice list. Let's go with Brahms. Could you send me a link to some Brahms that I can listen to right now? Now, the assistant is browsing the web for us. It's going to use the sources it finds to compose an answer. Looks like it's found us a link to a Brahms playlist.

All right, let's get to work. We need to read in some files from Azure Blob Storage. The assistant can help us with that, too. What's a good Python library for reading from Azure Storage? It recommends Azure Storage Blob, but I think I've heard good things about a library called BlobFile. I heard there's another library called BlobFile. Could you tell me about it?

The assistant is browsing the web again. This time it's looking for the PyPI page for this library. It's summarizing some key information from the PyPI page.

The assistant can even write code for us. How can I use it to read and pass a JSON file? And that's some code we can try.

Let's finish by saying what the assistant knows about Build. When's the Microsoft Build conference in 2022. The assistant's not sure. So let's let it browse again. Looks like it found the right answer.

What would you say if I told you this conversation would be shown at Build? Excellent. How polite.

KEVIN SCOTT: To see more ways that Codex could transform and simplify not just apps and software development, but things like visual design, gaming and education in the future, we're going to visit some of our colleagues at Microsoft Garage locations around the globe, with one quick stop from orbit a couple hundred thousand miles up.

[Music.]

FIONA O'GRADY: Hi, my name is Fiona O'Grady, and I'm joining you from our beautiful Garage space at Microsoft in Dublin. I am so excited to be sharing this Codex demo with you today.

As a model that understands language and code, we can use Codex to build more fluent user interfaces. In this case, we are using the model to power a language-based command line interface.

So let's start in PowerShell. Let's say, "What's my IP?" We hit a keyboard shortcut for the model to suggest a script. In this case, it suggests a web request against a service that returns our IP address.

Next, I'll say, "What's running on Port 1018?" Developers often want to know what's running on a port so they could free the port for something else. And it looks like there's a node process running on here. I actually know what the process is and I want to stop it. Let's see if Codex can do that. And look, it wrote the correct script to go ahead and free up that port.

Next, I'll say, "Make a .gitignore and add node_modules to it." This requires the model to understand the semantics of what we're doing, that a .gitignore has a period in it, that node_modules has an underscore.

And I'll say, "Open it in Notepad" and the model successfully opened this .gitignore that it created in Notepad. And you can see right there it has the node_module folder in it.

Building this prototype didn't require us to retrain the model. It was enabled through a discipline called prompt engineering. We gave Codex a few examples of the kind of code it should write, and it can generalize from those.

If I switch to giving it more conversational examples, then we can coax the model to produce conversational responses. Now when I ask, "Who made the last change to this repo," Codex should respond conversationally and use our Speech Cognitive Service to respond.

CODEX: The last person to change the repo was Dhruv Singh.

FIONA O'GRADY: And see, it responded with voice.

We can also ask something pretty chatty, like what's the meaning of life?

CODEX: The meaning of life is 42.

FIONA O'GRADY: And the model cleverly referenced "The Hitchhiker's Guide to the Galaxy."

Now this prototype isn't limited to PowerShell. Codex can also produce Bash and Z shell scripts. I'm using WSL and Ubuntu here, so I can run Bash on Windows. And now I'll ask, "What's the weather in Dublin?"

The model knew of a Curl friendly API that it could call. And it's a lovely, partly cloudy day. Not a huge surprise for the weather in Dublin.

Now, this is just one example how Codex can make developers more productive. But more generally, modules that understand human language and code have the potential to enable much more natural interactions with our computers.

One area we're really excited about as the potential for this kind of natural user experience is in gaming. My colleague Martha is going to share just some of what we've been working on.

[Music.]

MARTHA GITHUI: Jambo. Which is hello, in Swahili. I'm Martha Githui and I'm joining you from our newest garage space here in Nairobi, Kenya.

In this demo, I'm going to show how Codex can create new natural language experiences in games using existing tools and APIs. So let's get started.

We're playing Minecraft, accompanied by an avatar that uses Codex to produce code and language with the Minecraft game test API. The Codex model has never seen this API before. At runtime it's given the API spec, along with a few usage examples, and it's able to extrapolate from there.

We'll start by having a simple conversation. The model is tuned to produce code, but it can also produce dialog. First I'll say hello, and the bot responds with hello back. What game are we playing today? Indeed, we're playing Minecraft.

Night is coming soon, so let's see if we can get some help from our friend. I'll say, "What would be a good item to have at night?" And it suggests a torch, which is a smart idea.

Codex understands when a message is conversational and can answer with dialog and keep track of the context of the conversation. It's been answering us with information it already knows, which is not in our bot API code or examples.

Now let's use natural language prompts to have it produce some basic code to move around the world.

I'll start by saying, "Come here." And "look at me." The model is able to use API primitives to navigate the scene and to look at the player. It can also produce more complex code to handle more complex behaviors by extending primitives with its own code and logic.

Since night will be coming soon and now we know that we could use a torch, let's see if our bot can do that for us. "What items do I need to make a torch?" And it correctly responds that you need one stick and one coal. We already have the coal in the chest over here. So let's go ask our bot to get that first and then craft the rest.

"Go to the chest and open it." The model is writing code on its own to deliver the actions we've asked it to do. "Get the coal from the chest." All right. "And get some logs and make some planks." It's even able to take imprecise language and turn it into precise actions.

Finally I'll say, "make a stick and make a torch and equip the torch." And as you can see, the model was able to take the conversation and turn it into executable code in real time and deliver the actions in context of what it was already doing.

We think Codex's ability to write code for games can open up a whole new world for gamers and models to create complex behaviors using natural language instructions, and can enable game-makers to create interactive, natural language experiences for their players.

Next, we're going to the Garage at our headquarters in Redmond, where my colleague Ryan is going to show us how AI can help creators and designers do some amazing new things.

[Music.]

RYAN VOLUM: I'm Ryan Volum, a principal software engineer here at Microsoft. I'm joining you from the Redmond Garage on Microsoft's campus.

Beyond helping developers, we also believe that models like Codex can empower creators. In this prototype, I'm using Codex to turn natural language commands into Babylon.js Code. Babylon.js is a 3D renderer that runs in the browser and can be used to build 3D scenes.

I'll start simple. I'll ask Codex to make a red cube. And you can see that it wrote a line of code to render this cube, another to add a material, and a third to make that material red.

Next, I'll say, add teal spheres above and below the cube. And sure enough, Codex created those spheres and positioned them relative to the cube.

Next I'll say, make the cube spin. And Codex chose to use JavaScript setInterval function to make the cube rotate once every ten milliseconds, giving the impression of spinning.

And finally I'll say, stop it and see if the model can clear that interval. And sure enough, the cube is no longer spinning.

So we were able to create and manipulate simple Babylon shapes, but beyond that, Codex can compose Babylon primitives into more abstract objects.

Next, I'll say, make a chessboard using black and white cubes. And this will require Codex to understand the semantics of a chessboard, that it's an eight by eight grid. It will also have to know where those black and white cubes should go. And sure enough, it used some elegant code here to go ahead and create and render this chessboard into a scene.

Let's try one more complex example. I'll start by saying, make the background black. I misspelled background, but the model should read through it. Sure enough, the background is black.

And then I'll say, create a model of the solar system. And this will require Codex to know the different planets, to know their relative colors, their relative sizes, and position them in order. It will also have to write quite a bit more code to do this. And sure enough, Codex wrote a ton of code here to create a model of the Sun and of nine planets, which is incredibly generous to Pluto.

Let's see if we can add rings to Saturn. And the model chose to use a Babylon.js Toros primitive to wrap Saturn in rings.

And then we'll say, scatter a few hundred stars around the scene and see if the model can approximate something that looks like stars. And sure enough, Codex tries to use white spheres scattered randomly around the scene to approximate the look and feel of stars. And we were able to create a model of the solar system using nothing but natural language.

All of these capabilities were enabled out of box with Codex. We didn't have to retrain or tune the model. And with capabilities like these, we envision the models of Codex will empower not just developers, but also creators.

It's also easy to envision how real-time 3D scene generation could revolutionize gaming and metaverse experiences. To give you a picture of what we think is possible, we asked

for some help from our friends at FrameVR. Frame is a web-based metaverse platform and part of our team's ecosystem, alongside Microsoft Mesh. Let's see what the team at Frame came up with up. Off to you, Gabe.

[Music.]

GABE BAKER: Hey Ryan, this is Gabe from Frame with my Frame teammate, Jure. I'm in California and Jure is in Slovenia. Hey, Jure.

JURE TRIGLAV: Hey.

GABE BAKER: So we're in Frame right now, which is our web-based metaverse platform built with Babylon.js, which we love. And we're inside the Frame app that we built for Microsoft Teams that we can access it right within Teams.

A lot of our users use Frame for online classes, so we thought we would use Codex to create an engaging classroom together. You can go anywhere in the metaverse, so we thought we'd set up shop on the moon.

So let's wake this thing up. Want to kick it off, Jure?

JURE TRIGLAV: Sure. OK, Frame. Bring in the model of the earth. Make it spin and move it up 20 meters and backwards 40 meters.

GABE BAKER: This uses Azure Cognitive Services for speech recognition. And look, we can actually see the Babylon code being generated by Codex as we speak.

Bring in a model of the command module and the model of an astronaut and move them a bit apart. These models are actually from a cool Smithsonian 3D library.

JURE TRIGLAV: Students might also want to share their webcam or share their screens.

Create a rounded box and make it so that when I click on it, it creates two streaming screens and moves them five meters apart from each other. And now I click on it.

GABE BAKER: Add a table and a whiteboard next to each other.

Ryan, can you move that around with the mouse?

It's great that we can speak things into the scene, but for some things it's still easier to use your mouse or keyboard. You can use either. We think these experiences should be multimodal.

JURE TRIGLAV: Gabe, what are you doing?

GABE BAKER: Oh, I thought the students might like some basketball.

JURE TRIGLAV: Looks too easy. Set the gravity to match Moon's gravity.

The model already knows the gravity on the moon and can apply it to the scene. Whoa.

GABE BAKER: Well, there you have it, in just about a minute, we were able to make a really cool classroom in Frame, and this was only scratching the surface of what you can do with the Codex model. We'll get back to building.

KEVIN SCOTT: Thank you, Fiona, Martha, Ryan and Gabe. That was awesome.

We want everyone at Build to start imagining what you can build with models like Codex right now. So we're making the code from these demos available to all Microsoft Build attendees today.

We're super excited to bring Codex, GPT3 and future OpenAI models to developers through the Azure OpenAI Service, with enterprise grade capabilities like security, compliance and regional availability. And the examples you just saw are only scratching the surface of what's possible.

To start exploring the possibilities of Codex even more, we're partnering with OpenAI to present a Codex Innovation Challenge for all Build attendees. Just submit your idea for an amazing Codex scenario, and we'll select a group of finalists to build and present their ideas to our teams. The winner will get resources and mentorship from Microsoft and OpenAI, along with a very cool prize package. We can't wait to see what you come up with.

Beyond scenarios like the ones we've just seen, the ability for anyone to access the tools to create new AI-powered applications that can help us solve the world's hardest problems is going to be incredibly important.

Here's a technologist who is using AI tools today to reduce pollution and energy use and help combat global warming.

[Music.]

NICHOLAS BECKER: My name is Nicholas Becker and I work for Microsoft AI for Good.

Growing up, I wanted to be an astronaut, and I still do. I hope to go to space someday. But that led me to science and math, and my interest just bloomed.

So I didn't get any access to computer science or coding until I was in college, and that gave me the background to bring several different disciplines of technology together.

The airline industry is one of the world's biggest environmental impactors. Globally, it makes up 2.5% of total global emissions. A parked plane waiting at a gate burns 60 to 80 gallons of jet fuel per hour. But there are other sources of power for an aircraft while they're on the ground.

We started working with the Port of Seattle and Sea-Tac to try to tackle this issue. We used off the shelf components, Raspberry Pi sensors, 3D printed parts to try to collect the necessary information to train AI models to detect this needless emission. And as we went along, we iterated to more and more mature devices.

We created a distributed sensor network to monitor these emissions in real time. This network can then tell the airport which airplanes need to have their engines shut down to mitigate these emissions.

All right, we are as prepared as possible. It was so much fun to work on it, you know, climbing around on top of jet bridges and around and underneath the airplanes and installing sensors all around the airport to try to collect the necessary information to solve this problem. It was a pretty incredible experience to see like, we can do this, we can stop these emissions.

This is a great solution because everybody benefits. The airlines benefit from fuel savings and hitting their environmental targets. The airports hit their environmental targets and have a healthier work environment for their personnel on the ground. And the consumers benefit because their individual impact is now lower while flying.

I don't come from a traditional computer science background, but today, all of the science and engineering disciplines require some levels of computer science. So now everybody's a developer, whether they had that background or not. If you want to solve a really big problem, the tools available today have made that easier than ever.

[Music.]

KEVIN SCOTT: The technological achievements that have brought us things like Copilot, Codex and other large AI platform models are going to allow us to take on global challenges bigger than anything we've ever imagined.

But keeping those models running is going to require more computing power than we've ever seen. Two years ago at Build, we announced our first Azure AI supercomputer, which our partners at OpenAI used to train breakthrough models like the ones you've seen today. By our measure, it was one of the top five supercomputers in the world at the time.

Since then, we've continued to push the limits of what's possible in AI supercomputing. We now have multiple Azure supercomputers up and running, which we believe are the largest and most powerful AI supercomputing systems in the world today.

These systems are being used right now to train OpenAI's next generation of increasingly ambitious models, which together we'll continue to put into the hands of developers around the world through the Azure OpenAI Service, just like we've done with Codex and GPT3. While there are only a handful of companies like us that can construct systems at this scale, we believe what's most important is that their outputs benefit everyone, not just us.

Building clusters at this scale is an incredible engineering challenge, but the real breakthrough is the layer of software that optimizes how models and data are distributed across them.

System software like DeepSpeed, our MSCL library, and the ONNX Runtime allow us to train and serve significantly larger models faster and using fewer resources. These tools have been optimized for Azure's AI infrastructure and open sourced so that the community can keep improving on them. And because their systems are hosted in Azure, we're deploying the same innovations into our public cloud, available for developers at any level to use for their training needs.

It's this feedback loop from research to deployment that has made Azure far and away the most powerful cloud for AI training, based on public benchmarks like MLPerf, and it's why other leading AI research organizations are choosing Azure for their most demanding workloads.

Yesterday, we announced a strategic partnership with Meta to use Azure's compute power for their large scale AI research, accelerating their ability to train cutting-edge large models.

We're also deepening our investments in the open source PyTorch framework, working with the PyTorch core team and AMD both to optimize the performance and developer experience for customers running PyTorch on Azure, and to ensure that developers' PyTorch projects work great on AMD hardware.

And as part of our long-term partnership with AMD, I'm excited to share that Azure will be the first public cloud to deploy clusters of AMD's flagship MI200 GPUs for large scale AI training. We've already started testing these clusters using some of our own AI workloads with great performance.

And all of this is happening in Azure. Not only is it the fastest cloud, but it provides developers with the most choice and flexibility from end to end throughout the entire stack.

But ultimately, the point of having all this power to train all of these large models is to make sure that they operate as platforms that we can put into your hands to do great things with. Massive scale in research papers and benchmark results are important, but the true test of these systems we're building is whether you, who are building the software and apps that will power the future, can do cool stuff.

To make these systems truly valuable, it must be easy and cost effective to deploy large models using computing resources wherever they exist. As you heard from Satya yesterday, we're ushering in a new era of hybrid AI development where large scale training happens in the Azure Cloud and inference can be optimized for your local device in silicon, the cloud, or both.

We call this concept the hybrid loop, and we believe it's a breakthrough that will make platform models more accessible than ever before. You can imagine how it can enable a Copilot-like experience on every device and for every developer, no matter what kind of compute resources you have.

We've just seen some pretty incredible AI tools and experiences, and nearly all of it is available for you to try out today. But there's still a phenomenal amount of creativity and possibility ahead of us that AI can unlock in the future.

Last month, OpenAI introduced a successor to DALL·E, an AI model it introduced last year. DALL·E creates original, realistic images and art from simple, natural language descriptions. You just describe the scene you're imagining, and it brings it to life. It's still a research project, while OpenAI, Microsoft and other partners study its capabilities to ensure it can be deployed safely and responsibly.

DALL·E and other AI systems like it can create incredible things, but we have to remember that these are our tools. Only if they're paired with human ingenuity can we get something truly special. And when that happens, these tools have extraordinary potential to empower people to express themselves in new ways and unlock the creativity that's always been within all of us.

(Begin video segment.)

[Music.]

TEACHER: Who here has a big imagination?

STUDENTS: Me!

TEACHER: Today in class, we're going to get to think about how we use our imagination to solve some of the biggest problems in the world. Grab your marker. Grab something to sketch with. Think about what matters to you and something that's going to do good for the world.

STUDENT: What matters to me is like climate change and rising sea levels. So what I would design is like a house that's like on stilts. Like when the sea level is like rising, you could like pull a lever that brings the house up and down.

STUDENT: My idea is an electric table that you can heat up food and give it to homeless.

STUDENT: I would probably create a soccer ball that has lights so people who don't have power, can play with it in the dark.

STUDENT: We can take plastic that would have gone into the ocean and make it into a new shoe.

TEACHER: These ideas are amazing. I cannot wait to see your ideas come to life. And we have some friends that are working with some technology that's going to make that happen. Should we go check it out a lot?

STUDENTS: Yeah!

TEACHER: All right, come on! This is so exciting. Who's ready?

STUDENT: My idea is a club that helps clean up the creek. What we do with all the trash is make it into other cool stuff. The world could be better with no trash than more trash.

STUDENT: A computer drew that?

STUDENT: This idea, there's like a garden on the moon. We can use some of those gardens for biofuel.

STUDENT: My idea is the underwater school to help the ocean.

STUDENT: I thought, OK, let's make an efficient sprinkler, a sprinkler that knew exactly what your plant needed.

STUDENT: A robot could go in space and pick up trash.

TEACHER: Keep fighting for the world that you want to see. Keep thinking about big ideas that can make a difference for us and the planet. Your imagination has never been more needed than this moment right now.

(End video segment.)

KEVIN SCOTT: I really hope that everything you saw today inspired you to build, create and develop with AI. Over the coming years, AI is going to make more and more incredible things possible for everyone. But there's also a ton of stuff that's ready for you to try out right now. We want all of you to have access to the tools and the computing power to create AI that's safe, responsible and that makes the world feel a little more magical.

So my challenge to everyone watching is to find something that you can do with AI today, big or small, and just go make it happen. The only limitation is your imagination.

Thank you all so much for joining us and enjoy the rest of Build.

[Music.]

END