Amazon.com Inc at At AWS re:Invent 2019

Company Participants

- Clare Liguori, Unknown
- Werner Vogels, Unknown

Other Participants

- Jeffrey Dowds, Analyst, Unknown
- Martin Hofmann, Analyst, Unknown
- Sebastien de Halleux, COO, Saildrone, Inc.

Presentation

Operator

Please welcome, Chief Technology Officer of amazon.com, Dr. Werner Vogels.

Werner Vogels {BIO 16365903 <GO>}

Nice t-shirt there, (Shifty). Look at that. Welcome, everyone, to the first day event. I hope you had a really good re:Invent up until now. As always, it's a learning. It's an educational experience. And I hope that you guys learned a lot in the past days.

I thought I'll do something different this year and really focus on a bit more behind the scenes of AWS and Amazon, than on sort of new features and services. So the one thing that we've been very fortunate at AWS is that we've been doing this for, let's say, 13 years now. And we've gotten a lot of experience in that. And that experience has allowed us to innovate a range behind the coverage that maybe if you've seen all the new services and features that have been launched in the past days. But you sort of forget that there's a lot of things that we're doing behind the scenes. And this is a quote from Jeff Bezos, he says that sometimes innovation, if you want to focus on innovation, you have to focus on the things that will never change for your customers. Because if we do that, we create flywheels that will benefit them forever.

And it's easy to see how that works in retail, yes? A larger catalog means that you have a higher likelihood that you can find what you're looking for, lower pricing, better guarantees on delivery. But then how does that work for AWS? What are the things that never change for our customers? Security, performance, scale, reliability, cost efficiency and operational excellence, all those things never change. And anything we do over those parameters will benefit you forever. And most of our sales things are not things where we put out a press release about, it's really things sort of

innovation that we do behind the covers, to, for example, always make sure that we improve the performance of all the components in the system that we have.

And so I thought that today, we look at some of those things that we've done at AWS, behind the scenes. And then doing a little bit of a deep dive on them and see what kind of less as we learned from those. Then maybe you'd like those investments as well. And maybe you could take them home and do something with that.

So the first thing where I would like to talk about is virtualization as virtualization has been sort of the bread and butter of the computer parts of any cloud environment from day 1. And it's one of those major technical underpinnings that have really allowed cloud computing to become as big as it is.

And if you look at classical virtualization, that has been actually around for quite a long time. In the '60s, the major mainframes worked already with virtualization. But the way we see virtualization today, basically, the x86 virtualization, this really came to life by major research at Stanford and at Cambridge. The one at Stanford resulted in VMware. And they made use of some binary rewriting to trap the privileged instructions into the hypervisor. Xen took a different approach. And Xen actually modified the operating system to make sure that we trapped into the hypervisor to execute these privileged instructions.

So if you look at sort of what is the basis of virtualization, yes, we've really pushed the boundaries of virtualization over time. There is something in today's virtualization called the root I/O virtualization text; it's basically, if you have all these guests OSs, they are all fighting for the same I/O resources. And especially, you see that in the network. If you look at sort of when this virtualization really scaled up, you would see that most of the customers -- or most of the guest OSs would see significant data variations in latency on their network, mostly because they were all fighting for the same network device.

So with that, we started to think about how can we radically change this and think about sort of rethinking virtualization such that we can actually create a base for innovation for our customers. It also had to do with the fact that the newer kind of architectures, really were being hampered by the old style virtualization. Basically, we really wanted to give our customers performance at almost bare metal, if that would be possible. And we saw that with additional virtualization, there was significant overhead in all of that.

So we really wanted to build a modular system, at least that's the things we do at Amazon. We're really thinking about sort of taking some of the lessons that we've learned in software and actually apply them to the hardware world and the virtualization world as well. Basically, you should see the traditional virtualization world as a monolith, yes, where all the hardware components actually are managed by the same hypervisor.

And so well, if we actually take the lessons from micro services, where you have small building blocks, where you can really quickly innovate on and if we can then supply those to the hardware world as well, maybe we can chase the world of virtualization as well.

So just think about sort of, can we create a world where we have all these devices. And they have an API. The API may be a hardware API like a PCI bus. But it's still a hardware -- it's still an API to reach our hardware. So let's take a look at sort of all the different steps that we've taken in sort of the evolution of the Nitro System.

Let me start it off with really traditional virtualization. I'm really thinking the first problem we tried to solve was that of the network. Because basically, just transferring, let's say, 8 gigabit file from SV, would result in literally hundreds of thousands of kernel traps. And so we really wanted to see whether we could actually solve that particular problem.

So what we did, we actually moved the network component onto a separate card. And so that was really the first version. And that was actually what you saw in the C3 instance that we launched in 2013. That's -- we really learnt a lot from actually offloading the I/O onto a separate card. And it took us actually another two years to really become much more familiar with what that would take to actually offload processing onto separate cards that sit on the same server.

Step 2, was actually in the C4 architecture, when we started working with Annapurna Labs to actually move EBS processing into a separate card. So no longer is the -- sort of the volume processing and the network there happening on the main CPU.

That was such a success of the Nitro cards in the C4 that Annapurna Labs actually joined AWS. And we started working on the C5, which was a major jump because you could also do all I/O that we were doing on the servers, we would just, say, offload all the I/O onto separate cards.

And the next step was then to really start thinking about can we actually remove all the other pieces of the hypervisor and move all of those into a control plane on a separate card. And that became the complete Nitro System. So there's a nitro controller that does the management and then we also built a new hypervisor that is absolutely really minimized. And so you can do all of this. And so the first ones were the X1s and they actually really became first much more faster, much better performance and much more secure than we could ever have done before.

So let's take a look at sort of how this works, how you interact with it. And basically, if you would create a volume, an EBS volume, you would do a volume attach. That would actually talk to the EBS control plane, who talks to the EBS cards on the Nitro -- in the Nitro System who then actually makes that as an NVMe device, notifies the PCI bus and then the hypervisor traps that and actually mounts that volume then in the guest arrest.

And we've been quite successful with this. If you look at sort of the pre Nitro hypervisor. And this is sort of a typical jitter that you would be seeing if you interact with an EBS device. This is after we've offloaded it onto Nitro. Basically, all jitter has disappeared. This has allowed us to double the IOPs to EBS, increase the throughput tremendously and then you should hear that earlier in re:Invent that year, the EBS optimized bandwidth, that's actually increased from 40 gigabits per second to 19 gigabits per second. Now this is networking and EBS, which is really important to actually all of our customers.

But if we look at processing, what has happened to the processors? And if you -- David Brown earlier in the week talked about a customer who had a requirement of a 150 microseconds processing time. If you look at the C4, which is before the Nitro hypervisor was introduced, you can see that the customers had trouble actually meeting the processing capabilities. Then we asked the customer to actually try the same thing on a C5. And as you can see, the C5 has performance that is almost close to bare metal. This is purely because the hypervisor is so thin that it is no longer in the way of the guest OS to get the performance that they want.

The same goes for storage latency. Yes. The ICs were already beasts in terms of storage. But you can see that after the introduction of Nitro, both at the P50 as well as the end of the performance spectrum, there's a rock solid performance. And it's increased almost as much as 4x.

The same as if you look at the network optimized instances that we've created, the ones with the n at the back of the instance name. If you went to Peter Desantis' talk on Monday night, you saw the additional boards that they've created to give you a 100-gig network. And again, performance increased 4x. And it's not just that we were looking for improving performance. We've actually offloading a lot of these onto separate cards, you also could improve security, trust no one: kill Dom0, yes? We have to remember that Dom0 in the old style virtualization is a complete Linux instance. Basically, you could log into it and do a memory dump. That has a very big security risk, yes? And as such, by actually removing Dom0, you also remove any SSH or other access to the devices and, as such, you create a much more secure environment.

Oh, thank you. And I would give you some detail and actually, I like -- this is a very important part of the whole Nitro security design. Basically, you have to control the communication flow. What can happen is that the Nitro controller will talk to the hypervisor. That's allowed communication. The hypervisor is not allowed to communicate back. If the hypervisor take actions to control -- to actually access the Nitro Controller, you know that the system is compromised and you can isolate it and you can start investigating it.

The same goes -- the external control planes, whether they be EC2 or EBS, they are allowed to communicate it in Nitro Controller, like we just saw earlier on in the EBS example. The Nitro Controller, however, is not allowed to take any other actions. If the Nitro Controller actually starts actively going out on the network, you again know that the system is compromised and you can isolate it.

This is a unique communication design that allows us to build extremely secure systems, yes? So you really need to be able to do, in my eyes, trusted computing in an untrusted world. And as such, we've created 2 trusted parts, with the Nitro system. But the Nitro system also sits on a complete trusted network by itself. So all the components can actually talk to each other.

Another thing that we were able to do is actually add encryption to it. And you know that I've been always a big fan of, actually. So encrypt everything. And with Nitro, we are able to encrypt everything. We encrypt your communication out of the Nitro cards by default. Everything is encrypted by default in Nitro, yes. And not just the things that go out over the network. Also, your local disks are encrypted by default with no performance implications at all. And as you can see, in that way, there's actually improved security significantly.

And we can't even trust the hosts or the guests, yes? You have to make sure that the guest can no longer, at no moment, actually do anything to the hardware of the machine that we do not allow them to do. And one thing, for example, that we absolutely do not allow them to do is to modify any of the nonvolatile memory in the machine.

And what we also do when the machine boots up to actually cryptographically check all the components of the machine to ensure that they are never compromised in any sense and otherwise, we will isolate the machine and actually start to investigate.

The cool thing was that Nitro now became a base for innovation. Now we could do all these other things that we could never do before. For example, we could do live updates. We could start patching the operating system. We could start patching the hypervisor from the Nitro Card, without anything taking down. We could add new hypervisors to it. We could also start running bare metal. And we could also create outposts. All of these things have been enabled by the fact that which created this platform called Nitro that controls our computer environment.

But also another innovation that we have been asked, can we actually protect more sensitive data? We have a number of customers -- and, of course, EC2 is used for a lot of highly-sensitive processing. The customer has asked us, can you do something more for us, yes? Can you -- are there any innovations that you can do? And given that we're now at Nitro, we could do very unique things. And earlier this week, we announced Nitro enclaves, which allows you to cordon off a little piece of your memory and that memory is in golden enclave. And in that enclave, you can run your code and the code is continuously checked to be cryptographically correct, making sure that that code is never compromised. The enclave also has no access to network or to disk. And you, as the parent instance, you can actually communicate it over free (Sox) with a secure channel. This allows quite a few of our customers to have even higher control over sensitive data.

Now if you look at sort of Nitro-related innovation, can you figure out where that started, yes? We've been able to increase the release of 4x number of instances and

after that, we induced Nitro. And it really has been a game changer. It has really changed the way that we think about our compute environment and the control of our compute environment.

It also allowed us to go up stack and think about sort of, can we find support for -- not just for VMs. But can we find support for containers and for servers. With that, we introduced a micro VM called Firecracker. And here, to tell us more about Firecracker and Fargate is Clare Liguori, Principal Software Engineer at the AWS Container team. Clare?

Clare Liguori

Hi. I'm Clare Liguori. And I am so excited to be here today to take you under the hood on Fargate. Fargate is serverless compute for your containers. On Fargate, we run tens of millions of containers for our customers every week. And security is our #1 priority that is serverless container workloads.

Virtualization provides a strong isolation boundary between workloads and in Fargate, we use virtualization to isolate customers from each other and even to isolate each copy of an application. An application in Fargate is made up of one or more containers. And each copy of that application runs in its own virtual machine under the hood on Fargate, isolating it from all other containers running in Fargate. Inside that virtual machine is a dedicated kernel, network interface, data volume and credentials. So we ensure application isolation at multiple levels.

I'd like to show you an example of what a typical application looks like running under the hood, running in these isolated virtual machines. This is re:Invent trivia, an online trivia game about re:Invent and it's running on Fargate. On Tuesday, we announced that you can now use Fargate with the Elastic Kubernetes Service, EKS. And I'm running this application using EKS and Fargate. So I'm going to show you what happens under the hood on Fargate when a large traffic spike comes into this website.

Say, for example, I showed it in a re:Invent keynote to thousands of people. So let's first look at what happens when I run this application with EKS without Fargate, running on my own EC2 instances. To handle changes in traffic, I've configured Kubernetes to auto scale both the number of containers, called pods in Kubernetes. And the number of EC2 instances. So with normal traffic, like we're seeing here, I'm running a few pods on a few EC2 instances. Now I've just shown the game in the keynote. So of course, the traffic is going to flood in with people trying out the game. But it takes a little while for Kubernetes to spin up enough EC2 instances to run all the pods needed to handle that traffic. So while we're waiting for those instances to start up, we're under-provisioned, causing that big latency spike. Then, after a while, the traffic is going to start to drop off. Kubernetes is going to spin down pods and eventually terminate those EC2 instances. But that instant scale-down tends to lag behind the pod scale-down. So during this time, we're actually over-provisioned.

Let's now look at what happens in the same situation. But running under the hood on Fargate. With Fargate, I don't want to worry about having enough EC2 instances to run all of these pods, it's serverless compute. Fargate takes care of isolating each pod in a virtual machine and allocating the right amount of compute per pod. So now that traffic spike comes in, again, new pods are spinning up and Fargate is quickly allocating a new virtual machine per pod under the hood. There's a small latency spike there as pods start starting up. But it's quickly resolved once they're up and running.

So using Fargate, the number of pods can react really quickly to changes in traffic to the site. Kubernetes spins down these pods as traffic is dropping off and I don't have to worry about being over-provisioned because I'm using Fargate serverless compute.

Side-by-side here, we can really see the difference between running on my own EC2 instances and running on Fargate. With EC2, we saw both under; and over-provisioning. But with Fargate, I was able to quickly react to changes in traffic, Fargate isolated each of my pods with a virtual machine and it allocated the right amount of compute per pod.

So now that we've seen how Fargate scales out and isolates the containers for the serverless compute, let's now look at the virtualization technology we use to provide that strong isolation boundary between applications. Since Fargate's launch, we've used EC2 instances to isolate applications in virtual machines. In this model, each application is allocated a fresh EC2 instance running the Fargate data plane. But traditional virtual machines are pretty heavy weight to use for isolating containers. Traditional virtual machines tend to present a lot of interfaces and devices that containers simply don't care about. As a small example, a traditional VM will typically present a video card and reserve at least 4 megabytes of memory for graphics. The containers almost never have a graphical environment. And it can be really small, as small as 256 megabytes on Fargate. So these are wasted resources when using traditional VMs to isolate containers.

Fargate provides better efficiency for isolating containers. Fargate is purpose-built for isolating containers and functions in virtual machines called microVMS. microVMs provide the same level of isolation as a traditional VM. But they're fast and lightweight. They don't have any of the devices they don't need; the device model only implements the devices that are actually needed by containers and functions. So there is no video card inside of a microVM reserving 4 megabytes of memory like in a traditional VM. In fact, a microVM requires less than 5 megabytes of overhead total. So it's highly efficient to use for isolating containers. And we've been really excited to see open-source projects like Weaveworks/Ignite and Kata Containers leveraging Firecracker to provide fast, efficient isolation in their projects.

So with Firecracker-based isolation, each Fargate application can be allocated a fresh Firecracker microVM instead of a fresh EC2 instance. These microVMs look pretty similar to the EC2 instance I showed you already. They look the same on the inside. There's no difference to the application container running inside them. And they

have all the same components like the Fargate data plane. But with Firecracker, we're able to achieve better efficiency compared to traditional VMs. We can pack many microVMs onto a single nitro bare metal instance with each microVM running its own isolated Fargate application. As we run more and more of Fargate on Firecracker, that high density means better efficiency in Fargate.

I want to share a bit about what we're working on now under the hood on Fargate. We're optimizing how we fit Firecracker into Fargate. We originally ran the same Fargate data plane inside each microVM just like we did on each EC2 instance. But that's no longer optimal at our high scale and high density on Nitro bare metal. So we've redesigned the Fargate data plane from the ground up for the unique needs of serverless container compute. This new data plane that's under development now is designed to run directly on the Nitro bare metal instance, where it manages all of the microVMs and their container workloads inside them.

The containers running inside the microVMs can actually start up faster in this model because they don't have to wait for any other components to start up in the microVM like that Fargate data plane.

We're developing a core component of this new Fargate data plane in the open on GitHub, the Firecracker containerd project. Firecracker containerd makes it simpler to use Firecracker to isolate containers. It enables using the open source project containerd to manage my Firecracker microVMs. It minimizes the overhead required for isolating containers in microVMs. And it exposes container images as block devices to the microVM.

So check out Firecracker-containerd on GitHub for a little sneak peek at the future of part of Fargate under the hood. Thank you. So much.

Werner Vogels {BIO 16365903 <GO>}

Thanks, Clare. So it's -- we are, of course, fortunate that microVMs also allow us to run Lambda. And I think if you looked at the -- watch. So if you've seen Lambda improve over the past years, what we've done in the past, was it weeks, months and then releasing, we did some really cool stuff. I think given many of you rely on VPC boundaries, we're able to actually really reduce the start-up latency in the VPC boundaries -- in VPC boundaries. And if you look at sort of the new concurrency scaling that we've released and all the other components, we have proficient concurrency, we can get really good control of the start-up times and quite a few other pieces.

The thing is that with Lambda as well as with Fargate, we also want that the serverless technologies will be first adopted by, let's say, young technology companies. And it turns out, that's not the case. The rapid adoption of serverless is happening in the enterprise, mostly because you really only have to pay for your execution times. And the management is so much simpler. And as such, enterprises are adopting serverless at tremendous speed.

And we'd like to introduce to you Jeffrey Dowds, the IT executive of Vanguard to share about how they are making use of these technologies to completely revamp Vanguard. Jeffrey?

Jeffrey Dowds {BIO 19999183 <GO>}

Good morning. I'm thrilled to represent Vanguard today. And I'm excited to share with you Vanguard's journey to the cloud.

At Vanguard, our core purpose is to take a stand for all investors to treat them fairly and to give them the best chance of investment success. Let me introduce you to our firm. We're a global asset manager. We have 30 million investors. They entrust us with $5.7 trillion of their assets. We offer 450 investment products. We have 17,000 crew, that's how we refer to our employees. We have no physical branches. We're a digital firm; 90% of our client interactions come through digital channels. We have 40 years of lowering the cost to invest. Most importantly, we have industry best client satisfaction results.

From an IT perspective, we're big and we're complicated. We have global data centers, mainframes, thousands of servers, lots of storage, thousands of apps, 50,000 endpoints, 5,000 technical staff. And in our business, downtime is not tolerable.

60 years ago, Vanguard's senior IT leaders set out on a transformation. We knew that if Vanguard was going to stay competitive in the digital age, we needed to be better at the business of IT. We wanted to accelerate the pace of innovation. We wanted to deliver business value with start-up speed, continuous delivery, DevOps, microservices, cloud, new ways of working, CI/CD, all of these concepts were in play. But we knew cloud was the cornerstone to going fast. We knew it was the linchpin to our success.

So we set out on a private cloud journey since we had some concerns about public cloud security. One year into our journey, back in 2015, we sent 3 of our cloud architects to re:Invent. Upon their return, we knew we could not compete with the cloud-based services being delivered by AWS. We also knew that building a private cloud was going to take too long and be too expensive. A quick huddle with our CISO and other senior leaders, we pivoted to public cloud. And we selected AWS as our cloud provider.

With public cloud as our destination, we quickly formed the cloud construction team, many of whom are in the audience today. Thanks, guys. They are full stack in their structure. They are outcome-oriented in their mission. Most importantly, they have aligned goals.

So how does a big firm like Vanguard, with big data centers, get to the cloud? What was our starting point? We had a traditional tech stack heavily virtualized. We had big data platforms, monolithic applications. I'm not talking about monolith. There are 1 million lines of code. We had monoliths that were 30 million, 40 million, 50 million

lines of code. And we had an aPaaS running on-prem for our emerging portfolio of microservices.

Following a design guideline of security first, commensurate with a heavily regulated asset manager, we built out our accounts, DPCs and the security apparatus that entailed over 150 security controls. With security in place, we wanted to start moving some of our workloads to the cloud. We started with some of our web apps, our microservices. We moved our aPaaS, we thought this was the fastest way to start getting some workloads into the cloud. At the same time, we established secure Internet connectivity using Route 53 for DNS, AWS's web application Firewall and CloudFront for CDN. We also migrated from VPN access to Direct Connect for improved resiliency and bandwidth between our facilities.

Then we wanted to shut down our rapidly growing on-prem big data platforms. We became heavy users of S3 and EMR. Other machine learning capabilities such as Comprehend, Lex, Sagemaker, Transcribe and Glue were introduced. More AWS security services were implemented. We use Secret Managers for authentication credentials. We use Macie for discovering and protecting sensitive information. And we use Shield for DDoS protection.

We knew we had to get our data closer to the microservices. They were still reaching back to our on-prem data center for their data. Using CDC technology along with Aurora, it allowed us to move our data in a similar schema from our on-prem relational databases. Some of our microservices solution delivery teams wanted access to data and more of a key value structure. So we introduced Dynamo.

Using in Kinesis for data streaming and Lambda for event-driven data transformation, we began moving to DynamoDB. And this put us in a position to eliminate our cloud-based data cache.

Our next huge design decision focused on our aPaaS. We pivoted the ECS on Fargate. As mentioned earlier, in Werner's remarks, we got stronger container isolation, we got security out of the box, we got integration with other key services, especially identity and access management, most importantly to me, the guy that was paying the AWS bill, we got into a pure consumption model. And we hooked ECS on Fargate up to Dynamo and Aurora.

We are now starting to drain our microservices from our aPaaS. We are accelerating the pace of our monolithic composition and this should allow us to retire our aPaaS in the near future.

Finally, we started to move the gold copy of bounded context of function and data to the cloud. Recently, we have strategically decided to host our emerging advisory platform on ECS on Fargate. This platform supports our advice services that are increasingly in demand from our clients.

So here's our end state, just about a 100% cloud native architecture. So what does Vanguard get out of this? We know we can reduce the cost of compute by at least 30%. We know we can build software 30% faster. We know we can deploy our capabilities 20x faster. And all this leads to a better ability to innovate. And along the way, we get improved resiliency.

Since 1973, Vanguard has been disrupting how investors pursue financial security. Today marks the 17th time Vanguard has been on stage somewhere at re:Invent. I'd like to thank AWS for these opportunities. I'd like to thank the Vanguard cloud construction team for making all this possible. And I'd like to thank our investors who entrust us with their assets. So that they can enjoy financial security. Thank you.

Werner Vogels {BIO 16365903 <GO>}

Thanks, Jeffrey. So it's really cool to hear from customers as faithful as Vanguard, especially seeing all the benefits they're getting out of the platform.

One thing, if I think back about when we just talked about Nitro, I think about sort of a more general concept. It is the fact that at AWS and Amazon as well. But AWS, especially, we always think about evolvable architectures. And what do I mean by that is that often when you start building an architecture, you have to be keeping in mind that, that might not be the same software you will be running a year or two years from now. And especially when you have to scale up, like we have to have to do within AWS, we have to make sure that, for example, with each order of magnitude, you can almost revisit the architecture that you have built. And I think probably, there's no better example for that than S3, the -- sort of the first real service that we delivered to everyone.

And I remember that when we were designing S3, we had a number of objects on the board that we thought we would be storing in the first six months. And just for the heck of it, we added 2 orders of magnitude to it. We blew through that in the first month, yes? And that meant -- certainly meant that we had to keep a good eye on our architecture.

Now last year, Mai-Lan was on stage to talk about sort of how S3 evolved over time. And it started off with 8 microservices. And last year when Mai-Lan was on stage, they were holding a 53 microservices in S3. Now there are 262, yes? But for example, all the new capabilities that you've heard about, for example, S3 Access Point, we first launched in Andy's keynote, that's a new microservice. S3 Replication Time Control is 8 microservices. The Access Analyzer for S3 is another 4 to 5 microservices. What it shows is that we're able to evolve the systems because we've taken really good care in thinking about that this would be an evolvable architecture.

Now one of the things that I'm always proud of with S3 and which is one of the core principle with S3 is reliability, mostly because, as I always like to say, everything fails all the time, yes? And mostly, if you think about hardware, if you think about disks, they really have high failure rates. But network controllers do weird things. You have bit flips in memory. There's all these things that will happen to you at scale that you

need to be able to be prepared for. And that's even the hardware, not even thinking about sort of like Black Swan events like a bug catching its -- in one of the components.

So we're always thinking about how can we reduce the impact -- if things fail, how can we reduce the impact on our customers. And we call that blast radius. So we're always thinking about how to reduce the blast radius. And I've talked last year -- and last year, also, Peter Vosshall gave a great talk on cell-based architectures. And so let me just quickly revisit that, yes?

If you would have a regional architecture, something that spends multiple AZs, a cell-based version of that would sort of this big -- the smaller components into that, such that the blast radius of a potential failure is just limited to the cell itself.

But also, if you have a zonal architecture, something that just lives in 1 zone at a time or at different zones, you can also actually make use of cell-based architectures for the same approach. It is always the case that we want to reduce the blast radius. How exactly to pick the size of the cells, if you have smaller cells. And you can really reduce the blast radius, it's easy to test and easy to operate. But if you have larger cells, they're more cost-efficient and reduce splits. And so you can also see the whole system is easier to operate. And the question is always how to really nicely partition your system, such that you can actually make use of cell-based architecture.

Now, if you think about the zonal-based service in AWS, probably EBS is a really good example. So let's take a look at the kind of things that we've done in EBS to really reduce the blast radius in that route, yes? So you have to think about EBS as a block store service. We don't think about it as something that has just disks attached, right? It is really the case that there are multiple shards that actually make up the volume. And of course, we replicate that, yes? So addresses another set of shards that actually is the replication for all of this.

Now to control any type of failures of any of these shards, we have a configuration master, yes? And the configuration master, sits actually on a second network. And the second network is just a sort of an overflow network, it is not as big provisioned as the front end network between EC2 and EBS. But the configuration master sits there. And the configuration master does nothing. Actually, the only thing it does is when any of these shards would fail or any of the nodes would fail, it sort of restarts -- triggers the re-replication. So it's actually a pretty simple one, yes? So something fails, it fails it over to your backup and then starts re-replicating to a new set of shards.

And so, the configuration master doesn't have to do that terribly much. But if multiple things fail at the same time, this thing can get easily overloaded and especially because we're not talking about 1 disk. We're not talking about 1 volume. We are talking about millions and millions and millions of volumes.

And so if you only have 1 configuration master, that configuration master actually becomes a single point of failure because it can easily to get overwhelmed. So thinking about sort of what can we do to improve that? Now as always, there is always this tension between consistency and availability. Now the CAP theorem says that in a world where you have partitions, yes, you cannot have end consistency and availability. But consistency in EBS is nonnegotiable. So we need to make sure that we have an environment where we can actually ensure both high availability and consistency. And make sure that the consistency never get traded off. Again, cell-based architectures come back into this, yes? So if I apply that to EBS, the EBS master, what you would do is you can maybe start off reforming the zone and then you split the zone into 2 and maybe split them into 4. And every time you reduce the blast radius of the impact of a failure or of an overload of the configuration master.

But you're thinking about this, we're thinking about what is actually the smallest unit of cell that we can actually achieve with EBS. Then, especially for the configuration master. Then it dawned on us that actually, the EBS is a very unique case and the configuration master there as well, because not all data needs to be available for all clients.

Now remember that, in essence, in the old style database world, that meant that you have this 1 database, it needs to be accessible by all your clients. That's not the case here because it's only the client, only the EC2 instance and the EBS volumes that actually need to have access to this particular configuration data.

So instead of doing sort of a splitting up into 4, we went to millions of tiny databases to ensure that the blast radius in EBS is as small as possible. Now Physalia, this actually comes out of the -- from a lesson in the world, is the Portuguese man-of-war. And it looks like a jelly fish but, in essence, it just consists out of hundreds or thousands of really small microorganisms that have a colony together and present this 1 thing to you. So that's where the name comes from.

If you look at sort of what the data model is in it, each volume that gets created, gets a partition key and each database in Physalia manages only 1 partition key. Then what you do, you create this colony of micro cells, where each of these cells support only 1 partition key. And so it really means we end up with millions and millions of these cells. But that's okay. They're small and simple to manage. And the cell lives in the same environment as the nodes. And so actually, inside the cell, there are 7 nodes, runs Paxos and actually is able to have a distinguished proposal as the master. So in 7 cells with Paxos to have this state machine be reliable.

One of the things that we've been able to do with Physalia as well is to make sure that these cells are always as close to the client as possible because these databases are really tied. And so what often happens is that clients move throughout the AZ and this gets -- volumes get attached to different servers, things like that. So with Physalia, we're able to make sure that the configuration master lives as close to the cells as possible, to the clients as possible and as such, again, reducing the blast radius.

So if you look at sort of what the impact has been of Physalia, this is before Physalia. This is sort of the error rate in -- this is an aggregate error rate of accessing configuration master in a pre-Physalia world. And always, as you can see, what happens after that, it's actually pretty spectacular.

So again, cell-based architectures play a very important role here. And in this particular case, we've been able to go to a cell to cells that are as small as 1 single key and as, as such, significantly reducing the blast radius of any potential impact of a failure.

Now if you think about sort of cell-based architectures is one thing, there's been some other techniques that we've learned over time at Amazon that I think is very cool. It's because, especially if your application is either stateless or has soft state, what are the kind of things that you can do, right? And so this is where clients actually are involved in the story as well. Think about sort of a regional architecture. And I just used a card -- deck of cards as representing the different customers and the dice, given that we're here in Vegas. If the diamond one actually starts -- fails, that's introducing enormous workloads or he pushes something so hard that a bug is being triggered. If you have a regional architecture. And it might take out the first node and what they will do, of course, it will immediately retry. And with that, it will take out everything in your whole region.

Now if you have a cell-based architecture, yes, basically, you map them on to -- you map your customers onto a particular cell. So if again, the same scenario happens, yes, what happens is that the 2 nodes in the cell get taken out. Okay, good. In this particular case, only 25% of your customers are affected. And this would take the clubs -- no, what is that? Yes. That's clubs. And so -- but what now, if we go to something completely different. And it's called Shuffle Sharding. Basically, what you do is a number of nodes. And you have -- you actually take each client twice and actually take a random hash or just randomly spread them over the different shards, if now diamond is actually introducing failures, what you can see is that they are actually not another customer that -- in this shares the same cell is affected, which means basically, that if we -- if you look at the math behind this, it's basically combinatorial. Because in this particular case, with 8 nodes and a shard size of 2, yes, the failure rate of the impact of a particular failure is only 3.6% of your customers.

Now that's if you have a very small set. But imagine if you improve this to 100 and have a shard size of 5 and you can see that any 1 client can almost not impact any of the other clients. And so as end grows, the combinations grow. And you can really make use of the math to actually really build a highly reliable system and really minimize the blast radius of any failure into the system. And so it does mean that you have to have a smart client that actually knows how to do retries and things like that. But for the rest, I think the math really builds this real.

Now with all of this, building distributed systems is hard, yes? And we've always done that. And we've done this at Amazon for the past 20, 25 years at scale that is unparalleled. And so -- but many of you have always been asking us, how do you guys do this? I mean, after 25 years, you must have a lot of experience in building

these kind of systems. So how does Amazon do this sort of build resilience into that? How does Amazon engineer this scale? What are the kind of lessons that you have learned about managing operations.

And so we've been thinking about what we can do there for you. And we've gotten our best engineers and architects together, who've been given talks over the past years. And I'm happy to announce today The Amazon Builders' Library that will actually bring all of this information for you so that you can build highly reliable systems, just like Amazon is doing.

And so we're launching this with 15 articles and in all sorts of different areas. And for example, how to implement health checks. What are the best practices around that. What are -- what is the history of Amazon there, what are the kind of things that we've learned there. And so I hope that this all helps you build distributed systems at the same scale and the same reliability as what Amazon and AWS is doing.

So if I think about that we're all charting into unchartered territories, yes? Then -- and next customer is one that is really charting new territories. So Saildrone is a very exciting start-up that creates wind and solar powered autonomous surface vehicles. And they're making use of all sorts of technologies to charge everything in the seas.

So I'd like to invite the CEO of Saildrone, Sebastien de Halleux to actually come and talk to you about that. Sebastien?

Sebastien de Halleux {BIO 17721598 <GO>}

Our oceans are unbelievably vast, covering 70% of the planet. They act as a powerful engine driving complex planetary system that affect all of humanity. And yet, ocean data is scarce by any standards. And that's because oceans are not just incredibly vast, they are also unforgiving, dangerous environments.

Now on land, we have grown accustomed to billion of connected sensors. But at sea, we only have hundreds, principally for moorings. Imagine a large steel buoy tethered to the ocean floor with a 4-mile long cable weighted down by a set of train wheels, which is both dangerous to deploy and expensive to maintain. Now of course, satellites have been providing the big picture over the past 25 years. But they can only measure a few variables with low resolution and they cannot see through water. So we know we can do better. After all, we've been using robots to study distant world in our solar system for a while. So it's about time we start quantifying our own planet because we cannot fix what we cannot measure and what -- we cannot prepare for what we don't know.

So this is what we set out to do at Saildrone. Our breakthrough started 10 years ago, pursuing the land-sailing speed record. On March 26, 2009, our founder, Richard Jenkins, broke the record in the Mojave Desert, 126.2 miles per hour in a wind-powered car called the Greenbird Mark V. That record is still standing. And -- thank you. And this was the birth of the Saildrone.

At the core of the record was an innovative wing and tail arrangement similar to what provides lift to an airplane. But tilted 90 degrees. This is a solution that's capable of producing immense forward propulsion for very long periods of time. But it only consumes less than 3 watts of electrical power, less than a refrigerator light bulb.

And each Saildrone carries a suite of sensors to measure atmospheric and oceanographic variables with incredible precision, things like wind speed and direction, air temperature, pressure, barometric pressure, humidity, solar radiation; but also in the water, water temperature, salinity, dissolved oxygen, dissolved carbon dioxide, atmospheric carbon dioxide, underwater sounds, current profile, biomass, bathymetry down to 1,000 meters. I'm talking about a powerhouse of data collection. And of course, not just 1 Saildrone. But a global fleet of Saildrone. And this incredible planetary infrastructure, naturally, is powered by AWS.

Saildrone is all about endurance and resiliency. Our robots are working around the clock, achieving mission duration of 12 months or more. So Werner's talk about resiliency is the exact reason we chose AWS as a cloud provider, a partner that can not only massively scale. But that can also provide industrial-strength resiliency. And AWS delivered, enabling us to provide mission-critical data, real-time, around-the-clock to a global customer base without skipping a beat.

Here's an example of why this resiliency is so critical. Better fish stock data is very important to manage our fisheries. This is information that affects millions of jobs globally and after millennia of growing catches, 1996 was peak fisheries, catches have been flat, indicating severe overfishing of our oceans. But how do you manage a resource that you cannot measure?

So the mission is estimate the biomass of various fish stock, such as pollock, sardines, mackerel. And Saildrone do this using sonars, a device that emits sounds, a sound wave and listens for the echo from the back of fish and from the seabed. And by painting the ocean with sound, you can develop large statistical model of biomass over very, very significant areas.

So we do this with fleets of Saildrones because the fish migrate. So the faster you count them, the better the estimate. So here's a fleet of Saildrone at work over the Bering and Chukchi Sea in the U.S. Arctic, which is home to about 1/3 of our commercial catches. And closer to home, we do this work every year along the West Coast of the United States from Vancouver Island in Canada, all the way down to the Mexican border. This is truly groundbreaking work by autonomous vehicle.

Above the surface, our onboard cameras are busy accruing a very large data set of tag images, which we have come to think as the image net of the ocean and this data set now powers the very first machine learning algorithm optimized for maritime domain awareness, which is a place where everything is always moving in every frame, a very complex problem. This is, of course, still work-in-progress because the ocean keeps on surprising us. And the algorithm encounters novel scenes like a drone-riding seal, which is very hard to tag.

Earlier this year, Saildrone successfully completed the first ever unmanned autonomous circumnavigation of Antarctica. You've seen from the deep race how hard is to go around the track. We went around Antarctica, 196 days nonstop from New Zealand to New Zealand. And data from this mission quantified the key role, thank you, that Southern Ocean plays in regulating our planet's carbon dioxide, a key driver of climate change.

So to achieve this speed, we relied on some serious data crunching on AWS to navigate from waypoint to waypoint. First, we had to ingest numerical models describing changes in pressure systems, which in turn drive various wind patterns from different directions, something that influences the trajectory of the drones in pretty dramatic ways.

Below the surface, we had to track currents, which are adding around and swirling, can slower progress and, of course, waves 60-foot large waves, different wave height, different periods, which act as so many obstacle that can slow progress or literally destroy the vehicles. So as you can see, as we zoom in onto the drone, you can see that the navigation logic is actually quite complex, navigating these multiple fields in the most optimal manner. And this is a task we accomplished seamlessly with clusters of AWS compute and then sending the resulting instructions to the vehicles via satellite.

Our longer-term vision is one we called the quantified planet. It's not about just 1 Saildrone. We're dividing the entire global ocean into 1,000 sub domains, each 6x6 degree in size. And we're working to deploy a drone in each of those boxes. The goal is to achieve planetary coverage and thereby help deliver better insight into those planetary systems that affect humanity. And we can do this because these systems, like weather, can now be modeled numerically on AWS using new instance types like P3 and C5n instead of the old supercomputers.

So combining the AWS compute and Saildrone's unprecedented institute data, we can deliver new insights into global precipitation, for example, or into global winds monitoring hurricanes, storms and typhoons. And of course, for long-term monitoring of heat fluxes in the Tropical Pacific, the famous El NiÃ±o. And we do this delivery in -- all in near real time. We compute, package and deliver these insights in the hands of our users globally by our Saildrone forecast app.

So what's next? Our next frontier is mapping the global seabed. As hard as it may be to believe, 85% of the global seabed remains unmapped and unexplored. And we are hard at work to solve this data collection problem. But big problem need big solutions. So we have built the ultimate machine for this task. And I'm very proud to present the Saildrone surveyor, which is a 72-foot long wind-powered vehicle carrying a 1 ton sonar system capable of imaging the bottom of the sea down to 8,000 meters. That's 24,000 feet.

So together with our partners, we aim to complete this mission within the next 10 years, creating the very first complete map of Earth for the benefit of humanity. It's a

true planetary endeavor. And it involves unprecedented scope and data sets. And like so much of our work, it's really made possible by our collaboration with AWS to help store and process this amazing set of critical data.

So we're excited to see what the future holds as we explore new realms and new possibilities with Saildrone. And we look forward to continuing strengthening the amazing partnership we have with AWS. Thank you.

Werner Vogels {BIO 16365903 <GO>}

Thank you. A big part, of course, of weather prediction is fluid dynamics. If you want to see how you can treat AWS as a supercomputer, you should rewatch Peter Desantis' presentation from Monday night, where he really explains how you can make use of a EC2 instances to build supercomputers.

Now I'm going to take a little bit of a different track now looking a bit at -- more at the big industry kind of stuff that we've seen doing -- seen happening. And especially in the world of manufacturing. And so Industry 1.0 was -- of course, was at the end of the 1800s when steam machines came into action. And we saw the whole Industrial Revolution started to take place and the gin carts starting to be invented. And so if you have a look at sort of from Industry 1.0, moving over to 2.0 is where electricity gets introduced, it becomes a big source of power. You see -- you start to see major shifts in the manufacturing process as well. Because the first -- this is the first assembly lines are being built in those states. Whereas 3.0 is when all the electronics start to the arrive and all factory floors start to be sort of controlled for PCLs and things like that. Then in the next phase in what we know these days called Industry 4.0 is where automation starts taking place. Now everything becomes automated. And especially, also things like logistics and everything else around.

But is this -- are we really at Industry 4.0? Because I don't really believe that so much. Because if we look in 2015, the average age of the equipment in factories is 22 years. And they have never been so old as since 1935. And factory equipment really is too old to be able to produce the data that we want to get out of this to create insights. And so I don't think we are at 4.0 yet because factories and manufacturing sites need to change significantly if we really want to start creating insights into this world, yes?

And if you look at all the different pieces that are still really a big deal in manufacturing sites, there is no data that is helping them create that insights. And the data needs to start flowing out of these manufacturing sites into places where you can actually do the analytics.

And if you want to create insights, there's a lot of insights to be had. I mean, if I look, for example, at predictive maintenance or autonomous transportation, wearables, collaborative robots, all of these kind of things is, in my eyes, what Industry 4.0 is about. But there's hardly any manufacturing that is already at that pace.

And today, then if these factories are already producing about 1,800 petabytes of data a year. But it's only a small fraction of what is needed to really create the insights into this world. And whether that is in manufacturers or whether it is in smart cities or -- if you actually start moving this data into the cloud, you can actually have multilayers approach and whether you want to do things that this is actually where we are, here. Then or if you want to go out and actually start thinking about transportation, how companies can match -- make smarter decisions about computation. Or, for example, move out into the level of cities or go even further. Yes, go up and think about sort of what are the kind of things that we can and have to do around agriculture.

Now the world actually grows more and more. The expectation is that by 2050, where we have 35% more people in this world that need to be fed. All the way to transportation, right? We can have all these multiple layers of how sort of we should be collecting data and how this could -- data need to grow together to create new insights. And so if I think about that and think about sort of bring that back to how Amazon operates, I think back on sort of the physical parts of Amazon.com. And that's the fulfillment centers. So in the early days, we sold books. And there were about 3 million books when they were available when Amazon started. And of course, fulfillment was challenging. But they have their own logistical challenges. But it's actually relatively simple.

However, if you need to start selling TVs and toothpaste and pillows and shoes and coffee, things become a lot more complex, yes? And our plans was really to do that. So in the early days, this looked like and worked like a normal warehouse would look like. But that definitely is no longer the case.

We aimed to really sort of create better deliveries. We moved at 1 moment from $25 free shipping and then to 2-day shipping. And 1-day shipping now. And if you use Prime Now, you get it within an hour. So how do all of these things change, while you still need to improve worker safety and make sure that you can do the guarantees at the speed that you would like to do?

Of course, in all of that, in a typical warehouse, there's about 4 million bins and about 10 million items. We use computer vision throughout the systems. We set it out in what's called a Manhattan grid style, where the parts are where the bitsy pods. The Kiva Systems can actually follow. They bring him towards where the workers are such that you can put things in the box together. And make sure that actually, if there are 2 items that they've ordered, that they do go into the same box together.

And as such, we use machine learning to actually predict how to do these things, yes? And it's definitely in 4 different areas, yes? There's forecasting, what products should we be buying? Then there is who should we buy it from? From -- is this from the large manufacturers, is it from a mom or pop shop? Then the question is, where do we place them? Then if -- once we place them, what kind of promises can we give to our customers in terms of delivery? And this is actually pretty challenging, yes? Because actually, if it's a very popular product, not something seasonal like warm socks or sun hats in the summer, that's easy here to do. Or if it's just coffee and

things like that. Those are all easy things to predict because they actually sold across the world.

But what now if you need to -- if you have a Nicolas Cage reversible sequin pillow cover. Well if you have that one, that's harder to predict where you would actually have to place that one. And so -- but what you can do, you can make use of machine learning and look at thousands and thousands of similar items and start making predictions where you should buy them. So it's that if you are in Vegas, you want this pillow, you probably can get it in the coming two days. And that's where machine learning plays an increasingly important role.

If you look at many of the things that we've done at Amazon in the past 20, 25 years, machine learning is probably at the basis of all of that. And this is small stuff like (fender) lead detection or fault detection of all of these kind of things that are all happening behind the covers. But also major innovations, like Alexa, yes, or the drones or scouts, all of these are driven by machine learning, or probably the most extreme one that you all know about is the Amazon Go store. You could just walk in, take things off the shelf, put it in your bag and walk out and you get charged. There's enough challenges with all of that, yes? You have to first figure out what the account is and how the account is moving through the store. Here you see an unmarked CDV presentation of that and some of this you see turning yellow. And that means that we actually know a certain who do you count, what are you counting is that it's actually moving up.

With all challenges are if computer vision is actually products. Some products are the same. But they're crumbled, yes, or other products are actually really almost identical. But they're different products. Then also how are people actually interacting with the store?

And so we've invested a lot in generating synthetic information such that we can actually start making these algorithms more efficient and more accurate. And with all the results in, you can just scan in where you walk into the store and walk out and get charged.

But if you look at sort of outside of AWS, outside of Amazon, there's so many cool things happening in the physical world that are actually powered by AWS. So let's take a look at a few of those.

Yes, first of all, about sort of workplace safety. This is one of my favorite examples of customers that went on the path to improve the quality of their operations and their safety. This is Woodside. They are a liquid natural gas producer in Australia and have actually a significant operation. One of the parts of that producing liquid nitrogen -- natural LPG, whatever, is actually to freeze it down to 160 degrees under 0. And so that happens in this massive refrigerator over here. And one of the things there in the old world, if you would have sensors on them, the only thing that these sensors would do is alarm you if something went wrong. They didn't have any ability to produce data.

Woodside moved to a system where they actually were producing data and actually -- so there's a process part of this, there's failure part that's called foaming and that would -- and that was sort of an alarm go off. And then the whole factory would have to shut down for maybe even weeks. And it's a pretty hazardous environment.

And so what happens now that these sensors and there's 10,000 sensors in this refrigerator, if actually -- they now can actually start to predict foaming and no longer create these hazardous conditions for their customers. But they -- so after that -- this is the first experiment which they did. And after that, they flipped the switch and brought 200,000 sensors online and actually started to create autonomous environment for their workers.

All these environments are extremely hazardous. And so on one hand, they have autonomous platforms that are out in the sea. And there's no workers on there, there's only robots. And in this particular case, you can see the robots that are actually moving throughout their plant, which is an extremely hazardous plant, without any danger to the workers. Because the workers sit in a centralized environment, where they can actually see what these robots are also seeing. These boxes that you see popping up are IoT sensors and they run AWS IoT core there. And they're able to actually take actions if sort of controlled things need to just -- they happen, instead of just having to send people in big hazardous suits out there to do the work for them.

Modjoul is another interesting company. They make wearable devices to improve workplace safety, like safety belts and gloves and things like that, just so they can track how workers move through hazardous environments. And there's one interesting story that they told me is that one of the major U.S. airlines actually use these safety belts to measure how far people are actually bending over, how far their workers are bending over, like whether they actually use the right techniques to pick things up from the ground to really improve worker health.

United Rentals is moving to create these massive machineries that are completely autonomous so that nobody longer needs to work with these massive machineries in hazardous environments.

Another area that is really interesting for -- in this digital sort of physical world combination is the way that cities are being made smarter. And there's a lot of work going on in that particular world. And these are just a few things, a few interesting examples. ShotSpotter measures gun activity in the city. And apparently, for every gun-related homicide, there are 100 gunfight -- 100 gunshot accidents that are never reported. They make use of multiple sensors to really be able to accurately and within 60 seconds, be able to actually identify where the location was of a gunshot and send -- and have law enforcement actually take care for that.

The city of Miami has taken this on and -- in 2014. And since then, homicides have dropped 35% in Miami. The city of Virginia Beach has put sensors out in the ocean to start measuring and predicting when floods are going to happen. The same is for

Miovision, they actually put sensors in and around the streets so they can optimize transportation.

A lot of very interesting sort of physical digital world is that of agriculture. And there's a lot of work going on in developing and finding new ways to feed the future. And there's 2 of our interesting customers in that space. One is The Climate Corporation. They actually make use of digital -- whole digital architecture, it's called the Climate FieldView that make use for all the sensoring of their tractors and other equipment on-site as well as satellite imagery to help farmers optimize yields.

The Farmers Business Network is an interesting one because here is farmers that are sharing anonymously information about the yields of their fields and the kind of crops that they are growing. So it's that they can actually have collective bargaining with seed providers as well as setting the prices for the crops that they are selling.

And the same goes for the world of transportation, whether it is Siemens that are putting sensors all around the world on train tracks because the biggest cause of delay is often bending of the rails and things like that. So they can all measure that.

Deutsche Bahn, the big German transportation provider, is putting this out in -- also in sensors in each of their trains so they can accurately measure delays and things like that. Vantage Power is an interesting customer because they have sensors in all the electric buses. So they can do preventive maintenance of the batteries of these buses.

Then if you think about sort of coming back to where we talked about earlier, is really modernization of manufacturing. This is all about creating data streams out of manufacturing to actually really optimize insights.

And to talk more about this, we have a really interesting next speaker. He has won the CEO Innovation Award of CEO Magazine this year. And to talk more about this modernization of manufacturing is the group CIO of Volkswagen, Dr. Martin Hofmann.

Martin Hofmann {BIO 17725670 <GO>}

Hi. I am Martin Hofmann from Volkswagen. Volkswagen started with the Beetle. Today, we are a group of 12 iconic automotive brands. And the brands with our portfolio are VW, Audi, Porsche, Bentley, Bugatti, Lamborghini, SEAT, Skoda, Ducati motorcycles, MAN and Scania Trucks. We have 365 models in our portfolio. So if you want to drive a different model every day, it's going to take you a year of pure fun and joy, I promise.

We produce 11 million vehicles per year, which is about 44,000 a day. Why do I mention this number? This is a massive scale and a requirement for any supply chain on a global basis to run in a very efficient and effective way.

So our supply chain is highly complex and global. We have more than 1,500 suppliers globally. And they produce and manufacture every day more than 200 million components and parts. And these components and parts have to be shipped into our global factories. So there are 18,000 trucks per day shipping components into our factory, 7,700 ships on the oceans, crossing the oceans with produced vehicles. And in a year, it's about 75 million cubic meters of material that we process.

And we do this in 122 factories worldwide: 5 in North America; 9 in Brazil and Argentina; 71 Europe; 4 in South Africa; 33 China and India. All these factories are running on different grown technology that have different age. And so it's very hard to scale from an IT standpoint.

So what we are doing together with AWS, we're lifting 122 factories into the cloud. And we do this with one common global architecture that we are building with AWS. So this entire project is what we call the Volkswagen Industrial Cloud, right now, probably the biggest IoT project in the manufacturing world.

And we chose AWS not only because of technology. But because of their ability to scale, to provide us with standards for our factories. Implementation speed that we are getting from AWS methodologies, the flexibility and the culture is helping us to really accelerate this project. The way we build it, in the center, we built on AWS Cloud, what we call the digital production platform.

This is an AWS Cloud platform, in which we connect all the machines, the robots, press shop, print shop, the body shop, assembly machines, the logistics system, they all get connected into the cloud on which we run heavy lifting AI and machine learning algorithms, IoT services and security controls to constantly optimize and compute situation of the factory.

Then what we do is, on top of the AWS Cloud, we put what we call the Volkswagen Group App Store. This is where we develop use case-based applications for the factory. We develop it once and copy paste in all our factories. They can download it from the cloud, applications like predictive maintenance, algorithms that manage and optimize factory functions. So this is our digital production platform.

Now, if you take it and extend it to the outside world, which means we are integrating and connecting all our suppliers into that cloud, we're integrating our logistics companies, equipment manufacturers. And all our business partners will be connected to the Industrial Cloud. We're even that open that we will open it up for other automotive manufacturers to use applications and technology in the Industrial Cloud. They are also invited to contribute and to load software into the cloud, like an open source ecosystem. So this is going to be one of the biggest ecosystems that will be built in the industry.

The architecture behind, I can't go into all the details and you know them better than I do. But there are 4 major building blocks that we're using the architecture. One is the OT, operation technology, IT gateway, connecting machines, robots into the

cloud. It's the Edge gateway because a lot of the functions and workloads have to run close to the machines. We're using outposts, 2 outposts per factory because there are several applications that have to run on-prem and this is why we're using outposts for our plant cloud. And the fourth element is our DPP enterprise cloud. This is the application framework, highly standardized, where we'll be able to quickly, at higher speed, develop applications for our production system and keep them in the cloud.

Now let me show you something.

(presentation)

This is where it all happens. This is the shop floor. This is where the AWS Cloud gets married with our machines, with robots, with welding systems and all you can imagine running a factory. So thousands of sensors are constantly sending data into the AWS Cloud, into our digital production platform. So that the algorithms, machine learning algorithms can constantly optimize the setting of the factories and parameters of the machines. And will also run applications for our employees based on the cloud to better manage our factory system.

As Werner mentioned earlier in his talk, transformationally, the manufacturing goes way beyond automation. It's about full integration of all data points in the factory.

So Volkswagen Industrial Cloud is the foundation for our new production strategy. In the future, we will have autonomous factories, dark room factories and this is the start in that direction. But we'll also reduce factory cost by doing that. We define IT standards between the plants. We increase production program fulfillment. That means delivering vehicles on time to our customers. And our product launches will be much faster. So we can come to market earlier with new vehicles.

In numbers, it's a 30% increase in productivity. We can decrease our factory cost by 30%. And we are targeting EUR 1 billion of savings in our supply chain. It has been an incredible journey with AWS as a partner. Thank you very much.

Werner Vogels {BIO 16365903 <GO>}

So we have been fortunate in the past years to travel the world and meet with many of our customers. And it's always interesting to see that there's a certain group of customers, younger businesses that are trying to solve really hard human problems. And whether that is around education, whether it is around health, all of them are working hard in tackling the hardest human problems that we have at this moment.

For example, Aquabyte is a company out of Bergen -- it's in Bergen in Norway. And they're working on how to create protein. Now, the protein is likely to become the major food source in the future. And so what they do, they grow salmons. And in a bin like that, it easily has 200,000 salmons. And so it's interesting to see that they allowed us to come on-site and actually video this.

So we started a TV series called Now Go Build. And there's been 8 episodes by now. And they all deal with some of the world's hardest problems, where young businesses are really tackling hard problems in collaboration with AWS. So this is around food. This is one of my favorite. This was the first episode we did. It is in a company called HARA Token in Jakarta. And the problem in Indonesia is that many of the farmers, small farm holders have no identity. And as such, they cannot get loans to buy seeds for their farms. They have to go to a loan shark, who there in Indonesia often charge 60%. And so by giving these farmers an identity but not only an identity, also measure the size of their plot of land and the yield of their plot of land, they're able to get these farmers an identity and data that they can take to the banks. And the banks are eager to actually give these farmers loans because the repaying rate is almost 100%.

So again, these are companies that are really tackling really hard human problems. There was one episode that was a bit more fun in all of this. And that's the episode we released today, the episode that we did in Amsterdam.

(presentation)

So the other episode that we'll -- that we've released last week was in Rio de Janeiro. It's all about health care. How to find new ways of providing health care to the poorest people in Brazil. Even though Brazil has a national health service, they are not capable of actually keeping up with the needs of the Brazilians and especially not the poor Brazilian. So I urge you to go check it out. These are very inspirational stories of very young businesses that are running on AWS, solving some of the world's hardest problems.

Well with all of that, I hope to see you guys tonight. So have fun at the party and go build.