Simon (00:00):
Hello everyone. Welcome back to the AWS podcast. Simon here with you. Great to have you back, and I'm joined by both of my co availability zones. Firstly, Jillian Ford, welcome to the podcast. Jillian.
Jillian (00:10):
Good day. How can you not be excited when we're going to talk about Gen ai?
Simon (00:14):
Exactly. And Shruti Koparkar. Good day, Shruti. How you doing?
Shruti (00:18):
Good. Good. And again, excited to be here.
Simon (00:21):
It's going to be fun. So we are talking about an interesting topic, which seems to be on the lips of most people, but we're going to probably take a different tack here. So we're going to talk about generative ai. We're going to talk a bit about the landscape, and we're not going to take the typical approach. We'll define some terms upfront. So if you're brand new to this whole space, this is an episode for you. But also if you've been using it for a while and you want to understand what's the Amazon take on things, how can I use it on AWS, et cetera, what are some of your fellow practitioners seeing? This is also the episode for you. So it's kind of going to cover a lot of things. And it was interesting before we even started recording, we're just having a chat between the three of us.
(00:58):
So we all have different approaches to this technology and different strengths, the different layers of this technology, which will be good. None of us is an expert. So firstly, some disclaimers or statements of fact this space is moving unbelievably quickly. So we're recording this episode in, I want to say March, 2024 and has to check what date it is. This is the life we lead these days and stuff is moving incredibly fast. This is a Cambrian explosion of development. And it's interesting. I obviously lived through the cloud explosion and it was really quick, but this is orders of magnitude faster. So everything we say trust but verify. And as much as their stuff could have changed, just because we say it doesn't remain mean, it's true the whole time, but firstly, it is hard to keep across everything. Jillian, how are you sort of tracking things, how you trying to stay a little up to date with what's going on, accepting that no one can be across everything, and if they tell you they're across everything, they're being disingenuous.
Jillian (02:03):
That's a nice way to put it, Simon.
(02:06):
I mean, of course I'm very lucky that being a host here of the AWS podcast, I get to learn from also the experts that come on. It's also learning with customers. And so for the people who are listening of like, what does that mean for me? I would say for you it's, there's probably a use case within the company that you work at or the company that you're starting that uses generative ai. So I would start really from working backwards from some of those use cases. And then from there you can really figure out, okay, what are the different options that can be able to help you to be more productive, maybe save time, save money. Those are different ways that you can really start to look at how generative AI can be able to help.
Simon (02:53):
And Trudy, what about you? I know you've been playing a lot in the guts of it all, but all this stuff runs on computers. How are you staying across the state of the art of what's going on at that

really building layer of things?
Shruti (03:06):
Well, I have to echo what Jillian said, that it's really privileged to be the host of this podcast where we get to talk with different teams, but a few other ways I try to stay updated. One is the AWS Machine learning LinkedIn page. We have a social presence there, and that team does a really great job of sharing very useful content up-to-date product news from AWS. So for all AWS generative AI news, I've subscribed to that page and that I find really helpful. The other thing I did was I joined Twitter or what's now called X many, many years ago and hardly ever used the platform. And then about three or four months ago, this idea just dawned on me that I could just use that as my learning tool for Gen ai. And so what I did was I went back in and I unsubscribed from all the previous subscriptions I had and then basically found all thought leaders in this space, like researchers like Jan Koon from Meta or others, our very own swami or our very own Emily Weber, and just followed all of them.
(04:15):
And then through them I found other people. So now my Twitter feed is basically a curated list of AI ML developments. And some of them is really in the weeds, very, very high research, but some others is super interesting. I think something like that can help a lot. And then one final thing, find discord groups, or for example, I found this newsletter that I find really interesting, T ldr R ai, and they send me an email every week and there's interesting announcements and an interesting technology that they share. So those are the few ways I have found useful to stay updated.
Simon (04:53):
That's a good way to get some signals. So let's dive in and we're going to cover things from a variety of different angles. It'll kind of be like dipping, dip out, et cetera. But just to start with a quick mental model before we get into the guts, one of the ways to think about generative AI and the infrastructure and how you use it is really at the bottom layer. You've got the hardware and the tools and the infrastructure for training the models. How do I build these models? How do I run these models? Spoiler alert, it takes a lot of iron to run this stuff and a lot of time and effort and memory and storage. And shrudy will get us into some of the guts of that a little later because she's been doing a fair bit in that world. Then you've got this sort of middle layer, which is kind of, I want to access these large language models, I need to integrate them into my applications.
(05:38):
I need to do fine tuning, I need to do stuff. How do I get into action with that sort of stuff? And that's really where stuff like Amazon Bedrock comes into play. And then there's what I'd call the top layer, which is really where you've got the capability of gen AI already baked in for you or baked inable for you. So things like Code Whisperer and Amazon Queue where you don't have to do anything, you just plug it into your existing data sources and hey, Presto, you've got the thing, which is always the easiest way to go. Taking something off the rack is nice and easy. So you've got this three layer approach that you can take, but let's get into some of the guts of it. And we're going to take probably an unashamedly developer focused approach here in terms of some of the, how do I actually plug some of this stuff in?
(06:19):
And the first thing to talk about is really the models themselves. This is the big topic of conversation is which model should I use, when should I use it, et cetera. They've all got cool funky names like Mytral and Falcon and Llama and GP GPT, and it's one of those interesting

things where that's the thing now. And just to really come to some very fundamental things, the first thing we need to understand is the concept of tokens because you'll hear tokens mentioned all the time, and a token is really the smallest unit into which any text data can be broken down for an AI model to process. So it's kind of like how we break sentences into words or characters. Well, the token is the measure that these models use. And tokens are important because models are often defined on the number of tokens it can take in into its context window.
(07:10):
So it's attention and also the number output as well. These are all tunable factors because remember, you pay for the amount of tokens that go in and they go out, et cetera. So it's important to understand that. Now I'll share my own quick learning on tokens is I used to take a very naive approach to token length, which is I just mentally thought, well, it's kind of this rough amount of character, so I'll just let the code figure out a character count and then figure it out from there. Naive approach is what I would say. There are some great libraries out there that let you pump text into the library and it tells you the token count based upon the model you're going to use, and that's the way to do it. So then you actually using an accurate measure of tokens. But again, starting naively is never a bad thing.
(07:53):
The other topics are temperature and top P and top K and temperature is probably the most important one, I think because kind of how zany the model can get. So you can have a temperature that's very strict and restricted and doesn't give much creativity, and you can have a temperature that's really super creative, but maybe less accurate, and you've got to play between the two. I guess Jillian, some of the things that catch us out in terms of the fact that they're these dials and knobs you play with, and it means that the answers we get are non-deterministic, which is really great when you're a coder
Jillian (08:28):
For sure. If you're looking at the different models or even looking at the hardware level, it really depends on, okay, what is your skillset and what's the problem that is that you're trying to solve and how is it that you want to be able to solve it? And I think looking at it across all those different dimensions can help you decide, okay, do you want to look at it from the infrastructure layer? Do you want to look at it from being able to optimize on the actual EC2 instance type as well as how it is that you're actually the parameters that you're putting in or maybe using a higher level service like Amazon Redrock that I know you maybe kind of started to allude to a bit. But I think all of these factors can really help you coming up with what's the right strategy for you.
Simon (09:11):
And I think it's also, what am I trying to get done? Because each of the models have different strengths and also aspects to them. So the models will tend to have different licensing rules so we can use them and how you can use them. That's a thing that's a factor. They'll also often have different philosophies behind how they're built. So for example, Andro focuses very much on fairness and clarity in terms of how decisions are made by the model and the types of outputs it gives you. So again, you can pick different models that have different aspects, models that are just trained in different countries. So mistri for example, is a more European model. And so that may affect, because these models are in many ways functions of the type of training they've done as well. And I think Shruti, when we see models being trained, these huge corpus of data have a material effect on what the model's good at doing.
Shruti (10:03):
Yeah, absolutely. I mean, it's internet scale data that these models are being trained on and what

type of data was used, what quality of data was used really impacts the output that you get. And so that is something you need to think through. And actually maybe this talks about a little bit about the next topic we might want to talk about, which is, okay, you select a model, but then how do you decide what technique to use with it? Do you just use prompt tuning or prompt engineering if the model works out of the box for you or do you get into more customization? And so this is where do you have data that can actually compliment or even add on to what the model was already trained on? Do you have domain specific data? And those kind of questions start to come in to allow you to decide what technique to use once you've sort of thought about what model to use.

Simon (10:55):

Let's dive into that and let me maybe step back first and then sure, we'll come back into the fine tuning. I think it's very important. So is firstly the question of, well, how do I know which model to use? Well, it's a big fat, it depends. Our favorite answer in the IT industry, and one of the approaches people take is they use different challenge or evaluation sets that exist out there. There's a bunch of different ones. I'm not going to run through all the names of, there's lots of different tests you can run your model through to say, well, here's the answer we would expect to give. Here's the answer it gave. Is it good? Is it bad at this particular task? And different models are good at different tasks at different times. So for example, a lot of the most famous LLMs were really bad at maths early on, couldn't do maths because that's not what they were designed to and be, oh, look, it can't do maths.

(11:35):

That's so stupid. But really it was just, well, they hadn't been trained on that, and so they trained it on that. Now can do maths. Running a set of tests that indicate the key requirements of your use case is really important. And having that as a separate thing is important because then as you're evaluating models and expecting models to come in the future, that will become really useful. So for example, recently the Anthropic Claude three came out. Now I was using Anthropic Claw 2.1 for a particular use case I'd written, I was accessing it through Bedrock. Now, if you haven't come across Bedrock, bedrock lets you access all these different models through the same API, you don't have to change a whole bunch of code. It manages the running of the models, et cetera, the load management, all that good stuff. So it takes away a lot of undifferentiated heavy lifting.

(12:17):

And so for me, when I had to change my model, I simply just changed the model I specified in my code and I could use the new model and I could test that it was useful. So having something like Bedrock provides that layer of indirection for you to be able to access these things really easily. But what was interesting is I couldn't assume the model was better. I could only assume it brought something to the table that I wanted and I had to then evaluate myself to see if it actually gave the right result. And I think this is one of the challenges that, again, it's not predictable if it's going to be useful. Now these language models are trained to be generalists, but you can make them specific. And Shrudy, you touched on fine tuning, et cetera. Talk us through what that is. It can be challenging, say, well, this model is great except it knows nothing about this particular domain that my business runs on. Do I use retrieval augmented generation, or how do I tackle that?

Shruti (13:06):

One of the things that Emily, who is a principal specialist here always likes to say, is that you start out with the easiest of techniques that are fast and could just be effective. And so say you

tried out prompt tuning, prompt engineering, which is just basically playing with the prompt and trying to see if the model is giving you the output that you desire and it just works for your use case. But in many situations, for example, if you have very domain specific data like financial data, for example, Bloomberg created Bloomberg, GPT, because finance has a lot of complicated terms in very specific domain knowledge that can benefit from teaching the model about that. So if you have a use case like that, what can be helpful is are these two techniques rag, which is retrieval, augmented generation or fine tuning. And sometimes they can also work together with each other.
(14:00):
So what fine tuning does is if you have good quality data, that is typically fine tuning is supervised, so you need label data that is of high quality. And if you do have that and a domain specific use case, you can fine tune your model, which is basically only a few layers of the model or only additional sort of the last layers of the model are fine tuned on that data. So it's a step that is very similar to training or pre-training, except that is much less compute intensive, takes much lesser time and money and effort. And then what you have is a model that is literally fine tuned to the specific domain, to the specific use case. You can also do instruction tuning, which is just another form, which is you can teach the model to respond to specific type of, with a specific format of answers.
(14:54):
And again, this needs labeled dataset. And the other technique is rag, which is retrieval augmented generation, which is you don't really touch the model, you don't change the weights, you leave it alone. But instead of just asking the model whatever questions you might have with that prompt, you attach better context. And this is basically if you have a giant database of knowledge through documents or whatever, they are vectorized translated into a vector database, and then depending on your prompt, that is a vector search that happens that grabs the right context. So something like, oh, this user seems to be asking about our policy A, B, C, and then it'll go into your documents, pull out whatever information it has on policy A, B, C, attach it to the prompt and then pass it to the model so that you're telling the model, answer this question based on this extra context that I just provided you. And that typically increases the accuracy of responses as well. So those are sort of the two techniques you can use short of pre-training, which would be to train a model from scratch, which is you should consider that as the very last step. But yeah, those are some customization techniques,
Simon (16:09):
But it's doable. And we do have customers who have used a WS to train their LLM from absolute scratch, but I just want to come back to the fine tuning. You raised a really interesting point is that often for a lot of customers like, well, I want to fine tune the model, but I don't want to give my data away and I don't want others to use my data in their model because it's a competitive advantage. That's where something, again, bedrock really fits in because you can do all your fine tuning in Bedrock, you can create your own version of the LLM that only runs in your account that only you have access to that you haven't shared the data with others. So again, bedrock becomes a really nice way to just securely and easily get running and get the things you need without exposing your data, et cetera, because it's really important to keep the things that are important to you.
(16:52):
And as you said, even the way you query the LLM at this point in the generations of Gen AI is a differentiator. Now, personal opinion alert, I think that'll change over time. Much like at the start

of Google search, there were people whose job was just to write Google searches. That was your job because it was so specialized. At the moment, prompt engineering is kind of in that category, but over time it'll become easier. But at the moment, that's an advantage. So you can train it yourself, but what if you don't want to train stuff? What if you just want to use stuff? Jillian, you've got your hands on some of the developer focused tooling and that sort of stuff. So the top of the layer, in fact, as we've been talking, I've been using that Shruti spent a lot of the time at the hardware layer and the infrastructure layer of the three layers we spoke about. I've spent probably a lot of time in the bedrock middle bit, and you're at the top of the tree. Jillian, you're talking about the useful stuff where people are actually using things. Tell us about your experiences with things like Code Whisper, et cetera.
Jillian (17:50):
Yeah, there's a couple of things I want to answer about this question. So first I actually want to go back to what Trudy was saying. I thought it was so good, and I think it really applies to everyone who's listening here because I feel like a lot of people, they see this generative AI and they're, wow, this is really an opportunity for me. And as someone who works with startups, for a lot of people, I've seen a lot of companies get formed where they had expertise in a certain area and they fine tuned a model to be able to adapt to a certain industry. For example, being able to create a large language model that's specific to healthcare or something where it's a specific type of art off of stability. That is a very specific genre I would say, of art. We'll keep it as that.
(18:37):
Or maybe there's other people who are not necessarily looking to start a business, but they want to be able to evolve their career into a new direction. And so I'm seeing other people, this could be an opportunity for the listeners here where they can learn more about retrieval augmented generation. And if you work at a company that's got mountains of data and they're like, I don't know where anything is, how do we find things? I want an easy way for people who don't have SQL skills to be able to go and ask questions and get more information about their data. This becomes a huge opportunity. I mean, I really think that the data lake is really going to evolve into a gen AI type of data lake where you, yes, you can still be able to write SQL queries to get information about your data, but now there's going to be other people who are going to be involved, who are going to want to ask natural questions and be able to look at charts and be able to ask questions of those charts and all these things are going to come together.
(19:33):
And then now back I guess to your question about Q. So that's really I would say where Q and some of these higher level services come in as part of this overall, how do I understand my data and be able to do so in a really simple way. So I like that Q just simply, you just connect to your data source and you can just literally start asking questions. I like that. And then code whisper. I mean it's just something where productivity, right? I think anything really with productivity of being able to just write comments. And I always think that the code that you don't have to write is the best code. So that's certain my opinion about Code Whisperer,
Simon (20:11):
It also generates error checking code, which for some reason I never can write myself. I dunno what's wrong with my fingers, but I never write good error checking code. But it actually does that. And the other nice thing is you can train Code Whisperer on your own code base, and again, keeping it all internal, keeping it in your own VPC in your own account, et cetera. But I guess similar to that fine tuning, you kind of need to fine tune your coding assistant on the APIs and SDKs and sets that you use internally, which the world doesn't and shouldn't know about. So

that's kind of important too. It's almost like it's got to have a personality.
Jillian (20:45):
It needs to augment my personality.
Simon (20:49):
Well, when you think developers, you think personality, let's face it shrudy talk about the running of the thing. As we mentioned, it's all fun and games to talk about, oh, we can L LM this and that and the other. But if you're weaving it into your value chain, if you're weaving it into the software you're using, if you're weaving into your products, then it means that you're going to constantly be getting calls to these LLMs to do things, which means they need to run quickly, they need to be cost effective, they need to be good with power consumption, et cetera. What are some of the things you are seeing at that hardware and infrastructure layer to make it easier for customers?
Shruti (21:21):
Yeah, absolutely. At the very foundation of course is our EC2 accelerated computing instances. And these have both the Nvidia GPU powered instances as well as our purpose-built accelerators, a W straining, which is actually for the training or the pre-training or the fine tuning aspects, and then AWS nfr, which is the accelerator for deployment or for inference. And we've had a 13 plus year partnership with nvidia. So we continue to sort of innovate on that end partnering with them, we just launched H 100 powered P five instances last year, and we have the H 200 GPU powered instances coming in 2024. Also some new ones in the G series of instances. But the one thing to remember is that it's not just about the compute. Often the conversation is about GPUs and GPUs utilization or these new accelerators like Tanium and infr, but we also have to remember that you need to really optimize the full stack because for example, data is such a huge piece.
(22:26):
Data is what the models are trained on, and you need services such as Amazon, FSX for luster, which has hundreds of gigabytes per second of throughput to make sure that these models are kept fed with data, whether it is when you're training them or when you are deploying them. Same thing with networking. These distributed workloads for training or in some cases for inference, mean that you need really high networking bandwidth between nodes and we have up to 3,200 gigabits per second of EFA networking. Those are some of the innovations at the very core infrastructure layer. But then of course, we want to make it really easy for customers to get started with this. And for that, we have Amazon SageMaker, which offers a fully managed end-to-end machine learning experience. One important service that just launched last year is the Amazon SageMaker hyper pod. When you are training these really large models, there are so many challenges like maintaining a giant cluster of nodes, making sure if a node goes down, trying to recover quickly because otherwise you have all these GPUs just idling sitting there doing nothing, and that's not a very good and very cost efficient way of doing things.
(23:44):
And then of course, scaling is hard. Distributed training is not that easy. SageMaker HyperCard really simplifies it. It distributes your model across different nodes. It makes sure that if a node goes down, it can recover gracefully. It saves checkpoints and recovers from the last step checkpoint. So I highly recommend our listeners to, if you are training your own models, please, please check out SageMaker Hyper Bot. But then again, if you want to build your own ML pipeline, we have EKS, which is a really popular way because a lot of people use Kubernetes. So it's a very good way to orchestrate your training or inference workloads. And the beauty of this is

that all of this infrastructure integrates seamlessly with our developer workflows in frameworks like PyTorch or TensorFlow or JAKs, as well as with open source repositories like hugging phase. So if you are sort of more on the fine tuning training your own model and then building your own ML pipelines and deploying them, and a few of these other services are a good point to start. But then again, I'll bring it back to Bedrock once again, many of you may not need that. Many of you will do just fine accessing through a single A p. I mean the beauty of that single API is it can be overstated, choose well, but AWS of course has options for all of it.
Simon (25:08):
Exactly. And I think the other element there as well is the ongoing performance of the thing and the capacity to run. And one of the things I like about if you go the bedrock routers, you've got provision throughput. So again, you can plan out if you've got base load that you're going to, you can just get access to stuff because these GPUs are kind of rare as hence teeth at the moment around the place. They're quite prized. But there's also a lot of development also going on in terms of AWS training and in as well to do both training and inference at scale and at low cost too. So again, this is being attacked from multiple ways, and this is why I say that it's so exciting but challenging to stay across everything. And so I think building components where things are going to change becomes really important, and that's why abstracting where necessary and making sure that your approach to LLMs is quite pluggable I think is really good because the one that's cool and useful changes.
(26:01):
So for example, I was using a lot of LAMA two when it came out, that was my go-to I think folks end up finding their quote favorite one as in it suits their particular use case the most. And so I was using that a lot until Claude 2.1 came out, and now I'm on Claude three. And so I'm not closely wedded to a particular model. I just want the model that does the job that I needed to do. And what's interesting though is how good a job things can do with very little work. So really simple, trivial example that I did for myself, I said, I want to analyze where I spend my time. So I downloaded the contents of my calendar for the last 12 months and said, Hey, LLM, tell me where I'm spending my time. And it said, the context is not big enough for the data you're putting in there.
(26:44):
So then I said, well chunk it up, chunk it up, process it separately, and then bring it together. And it could do that for me. In fact, it wrote the code for me to do that for me. That's the nice thing, and it gave me some really interesting insights and it didn't take long to do that. And it's like, well, you start to think about how you can apply that to different things you do every day and different things your business does every day, and you can start to see how this will become a feature of pretty much any piece of software. And I think that's why getting onto it and using different SDKs and seeing what works for you in terms of a style is really important. So for example, Lang Chain is a library that a lot of folks use is available natively in Bedrock as well, and it does some things really effectively. But for me, I was fighting it all the time, so I just went directly to the Bodo three bedrock call. But everyone's different. Jillian, you'll probably tackle it completely differently to me in terms of how you think about this in terms of your software development.
Jillian (27:39):
Yeah, definitely. I think one thing that, as you were just saying, it made me think of that a lot of people might not realize is that, so Amazon Queue as an example, is being embedded in other AWS services. So I say that because yes, we've been talking a lot about all the different options

of SageMaker. You can use EC2, you can use Bedrock, but Q is already in other AWS services. And so as, yes, it's March right now, but I would say my suggest is keep a lookout because let's say you're a DevOps engineer and maybe you don't have any intention of going into SageMaker and fine tuning a model, but for all, maybe within the context of your world, there's going to be a service that you use within AWS that then uses generative AI on top of it to be able to help you be more productive. So I think those are just some things to be on the lookout for this year as you're really looking at how you can be able to be more productive with what it is that you want to be able to do and grow in your career.

Simon (28:44):

Yeah, I think it's important to, as you say, to see it filtering in and Q start to become a much bigger part of my use of the console. For example, if I want to write some cloud formation or a script or what have you, I'm just asking it to do it for me, and it just does it all. I think I was in Athena the other day and I had to write a query and I didn't have to write the query, and CloudWatch was another one. Actually, CloudWatch has its own way of doing querying, and I couldn't remember what it was, and I didn't have to remember. I just said, Hey, do the thing, and it wrote the thing to do the thing for me, which is kind of nice. Let's also talk about, I guess ethical considerations and the way we can use this technology responsibly, because always with great power comes great responsibility, and I think one of the important things is tracking back to how the decisions are made. What are some of your thoughts on how this sort of plays out and how the different model types and the explainability will come into play?

Shruti (29:40):

That is a really interesting question, Simon and top of mind for many people and a hard one to solve. Honestly, when it came to traditional machine learning, it was much easier to evaluate the model output and check if there are any sort of model drifts, if there's any bias. But because with generative ai, the content that is being created is long form in many cases is images text. It's so subjective that this is really where the human in the loop really starts with become very important. And of course at AWS, we have services such as SageMaker Ground Truth, where you can tap into that resource and have humans in the loop to help you evaluate your models and make sure that they're doing the right thing. But then also on the training side, it's going to be really important that we are careful about the type of data that we are training on, but also about what models you're choosing.

(30:43):

Going back to your earlier point about all models have a personality in a way, and they've been trained on a certain type of data. And then again, so that's where service like SageMaker clarify for example, can really help in helping you identify what is the right model and if it is in fact following the guidelines your organization has set in terms of compliance standards they need to meet or whatever guardrails they may have set internally. I would say the most unsolved problem in this space is how to get this right. The way to solve it and get it right is to keep talking about it, keep experimenting with the tools that are out there, maybe building some of your own and going from there.

Simon (31:28):

And I think that, as you say, the development in that space is really moving on very, very quickly. So again, as you can probably tell, I really like Amazon Bedrock I've been using a lot is a new capability in that. One is something called guardrails for Amazon Bedrock. It's in preview at the moment, but basically it will allow you to bring a consistent level of AI safety across your applications so you can block the undesirable topics from your applications. You can filter

content, you can redact PI as well. That's another thing that's going to come into place, but this will continue to evolve. So as I said, we're in this sort of berberian explosion and everything's going to change very fast. And when someone says to you, well, we can't use that because it can't do X or because they haven't solved why, just remember all of those are just a function of time.
(32:13):
And in this case, time has tended to be weeks, weeks or months to solve some of these problems, which has surprised me. I shouldn't be surprised anymore. I'm too old for that. But it's surprised me how quickly these things are getting solved for and how quickly new models are emerging. If though listeners are wondering, well, who's actually using this stuff? Who's investing? Is this just startups, et cetera? Well, it's everyone. So for example, some of the customers that are working with us using Bedrock, again include organizations like Aidas and Merck and Ashi Group. So we've got sportswear manufacturer, we've got a brewer, we've got medical places like Content Stack who focus on digital experience or virus who are an energy dedicated SaaS provider or Genesis who do customer engagement or GoDaddy. Everyone's heard of GoDaddy, Intuit, Lexi Nexus, who do legal documents. All these spaces are being dramatically revolutionized by this technology, and it's because it actually does stuff.
(33:16):
I come back to my fundamental thing of transformations and changes happen when it actually helps people and actually does something different. And again, the cloud made it a lot easier to use it way easier than if you're my age. You know how bad it was. It's way better than it was, and that's why it became popular. I think this falls into the same category. It solves for a lot of issues pretty straightforwardly. Yes, you've got to think about your data strategy. Yes, you've got to think about usage guidelines. Yes, you've got to think about the models and the performance and the future and the cost and all that sort of stuff. You've got to do all those things. There's no getting away from eating your vegetables, but if you do those things, the results are just unbelievable. I think that's what's really transformed things. I think we're going to see a lot of change. Jillian, what are you most excited about putting your future hat on? What's looking like an area that you're excited about and where you think things are going to draw your attention to?
Jillian (34:08):
I'll go back to what I was saying earlier because I really think that this is going to happen really soon, especially as soon as this year is really the evolution of the data lake. I think a lot of companies, whether you are late stage startup enterprise, any company that has a lot of data and it's all over the place, you've got a lot of just messy data. There's a lot of these companies that just want to be able to better make sense of it. And I think now that there is a very easy way to be able to just ask questions of the data. I think it's going to allow for just a lot of people to be able to come up with new revenue streams, to be able to ask better questions, to be able to open up other opportunities. I think that maybe people just weren't available before because maybe it was just too daunting of a task to actually go and figure out how do we actually make sense of it? And so people procrastinate and now it's like, no, now
Simon (35:06):
You totally, it's like, oh, I got to get my data. Got to train and model. It's all too hard.
Jillian (35:10):
Yeah, totally. Yeah, so that's one thing that comes to mind for me, and I'd love to know Simon, the same for you, and also just from your experience of whether that's experimenting with it yourself or just from what you've seen working with the customers that you have, not only maybe the future or are there other things that maybe you're seeing of how people can start to use

generative AI today that maybe they're just overlooking?
Simon (35:35):
I think we're seeing an unconstrained starting to come into place because you don't have to do a lot to get up and running, and because you get really useful answers, particularly if you know a little bit about prompt engineering, a little bit about how to use the tools. It's kind of like an accelerant of good ideas. I think we'll see it pervade the customer experience side of things. I think that's going to make that whole side of things a lot better. Like call centers, business intelligence, analytics, all that sort of stuff, game changer. I mean, you're already seeing it. I mean, when you go to amazon.com, it's already in there. You get summaries of reviews through LLMs. So it's like, okay, that's really interesting. I think though, that we're going to take, and there'll be another radical step change once a whole lot of really smart developers get their hands on this stuff and showcase to their business stakeholders, Hey, we can do this now.
(36:25):
Like, Ooh, that's interesting. Now what does that mean from a business strategy standpoint? How do I apply it? How do I use it? Has it changed my decision making? How does it help me move faster, et cetera? It can really change the dimensions from that side of things. So I don't think we've seen the business transformation happening yet. I think we've seen a lot of business leaders go, oh, there's something here. I better pay attention to it. But they haven't necessarily figured out exactly what that is, but time will tell. Trudy, what about you? What is your forward looking view look like?
Shruti (36:53):
Very aligned with what both of you have said, but I'll answer a slightly different question that you also asked is what am I most excited about? And I think what I'm most excited about is democratization of this technology because as you alluded to earlier today, in many ways people are either capacity constrained or budget constrained, and options like Tanium and Inia, which allow people to do it as soon as they have capacity, as soon as they need it, do it at a much lower cost, but also newer innovations from Nvidia such as their next generation GPUs and what that will allow folks to do. I'm just really excited about that. I feel like in this particular field, it's been a flywheel of innovation and investment in infrastructure that has then led to the creation of these models. It started back in, I think 20 13, 20 12, like with AlexNet on GPUs, and it really is a field where the top layers of the stack, like the models, the applications are still very inherently linked to the infrastructure and the innovation at that infrastructure layer is driving more innovation and more efficiencies further up.
(38:05):
So I'm really interested to see how that flywheel pays out, where this sort of innovation in the infrastructure will lead to better models, to more adoption of use cases, because there's easy availability of capacity at a fairly lower cost without sacrificing performance, and then how that sort of flywheels into further investment in infrastructure and further innovation. So for the first time, I have a lot of background in the hardware in sort of that layer. I used to work at ARM a long ago, and in many years it hasn't felt like hardware was cool, and suddenly generative AI has made hardware and system design very cool. Again, they're inherently linked and it's exciting.
Simon (38:49):
Hardware is back again people. That's really cool, really cool. Well, let me take our listeners again through those three layers because I think it provides a really useful mental model of how you apply this technology. So the top layer, you've got the, what I would call off the rack stuff, code whisperer, Amazon queue. You have to do almost nothing to get the amazing benefits of it.

You're activating a service or it's already integrated into a service or you're connecting into your data sources securely, and suddenly you've got all the coolness of generative AI at your fingertips and away you go, very low friction. Then in the middle layer, you've got Amazon Bedrock, which lets you access all different kinds of leading LLM models, allows you to customize, allows you now to put in front those guardrails explainability agents we haven't even spoken about. So the ability to go do stuff on behalf and for the agent tying into knowledge bases, et cetera, lots of stuff.
(39:42):
But as I said, that's that nice middle ground. I reckon. If you're a developer and you're building stuff, that's the one I would gravitate to because it gives you the most options and the most choice and the most ability to swap and change and move things around. And then at the base layer, there's of course the best hardware, tools and infrastructure you can get to train your models, the best access to efficient training technologies and the latest generation of GPUs and the train chips and the inference chips and the SageMaker Hyper Potters. As Shrudy mentioned, these things take hours and hours and hours, thousands of hours, compute hours to train. It's a lot of money. It's expensive, it has to run well. So for example, the Technology Innovation Institute trained their Falcon, LLM. A lot of folks around listening would use the Falcon LLM model.
(40:27):
They used Amazon SageMaker to train it. It was a great example of moving really quickly. They wanted to build their own model. They wanted to create something very specific for the UAE to lead in this place, and they're able to just grab the infrastructure and go, and they're able to build really useful models that are now used globally. So you can see the different layers will come into play depending on your experience level, and you'll probably bounce between layers as you go through your journey. So it's going to be interesting, and I think this is something we'll revisit in a few months time and talk about where things are at and how things are going. But until then, how do folks get in touch with you?
Shruti (41:02):
Well, I'm on X now, so Truthy coworker. Yeah,
Simon (41:06):
There you go. As you said, yeah.
Shruti (41:08):
Yes, but I keep my interactions there limited to the generative AI or AI ML space, but that's what we are talking about. Or I'm also on LinkedIn so people can find me there.
Simon (41:18):
And Jillian, where do you receive your input?
Jillian (41:21):
And on X, miss Jill Ford,
Simon (41:24):
That's the way. And I'm old school, A WS podcast@amazon.com is the place to do it as well. And until next time, using all your generative AI skills, keep on building. I.