EPISODE 750

[INTRODUCTION]

[00:00:00] JM: When a developer provisions a cloud server, that server is called an instance. These instances can be used for running whatever workload a developer has, whether it is a web application, a database or a set of containers. The cloud is cheap to get started on. New applications with few users can often be hosted on infrastructure that cost less than $10 per month. But as an application grows in popularity and in complexity, there is more demand for CPUs and storage. A company will start to buy more and more servers to scale up to the requirements of their growing user base, and the costs of running infrastructure in the cloud will increase. The company will start to look for ways to save money.

One common method of saving money is to buy spot instances. A spot instance is an instance that is cheaper than reserved instances, or on-demand instances, and the reasons that there are different instance types is because a giant cloud provider has a highly variable amount of work that is being demanded from that cloud provider and it's not necessarily that these are different computers where the spot instances versus the reserved instances versus the on-demand instances are available is different. It's just that at different times, these instances are priced differently.

Think about it this way, if you're in charge of AWS, you have to make sure that at any given time, you can give server resources to anyone that asks for it. So your data centers need to have physical machines that are ready to go at any time, and that means that much of the time you have server resources that are going unused. If you are a cloud provider, how can you get people to use your compute resources? Well, you can make them cheaper.

So a user can come along and buy your compute at the discounted spot price, but if you start to make your compute resources cheaper and price them at a spot rate, this presents a problem for you as the cloud provider, because if you start to give away your compute at cheaper prices, then the overall demand for your cloud resources goes up once again and you're going to miss out on profits.

As the cloud provider, you need to kick people off of your spot instances so that you can take those same instances and then sell them to people at the higher market prices as the market fluctuates, and this presents a problem for the user. If you buy a cheap spot instance, that instance is only available until the cloud provider decides to kick you off. You have a tradeoff between cost and availability of your instances, and because of this, spot instances are typically used only for workloads that are not mission critical. Workloads that can afford to fail.

Spotinst is a company that allows developers to deploy their workloads reliably on to spot instances. Spotinst works by detecting when a spot instance is going to be reclaimed by a cloud provider and rescheduling the workload from that cloud provider on to a new spot instance.

Amiram Shachar is today's guest. He's the CEO of Spotinst and he joins the show to talk about the different types of instances across these cloud providers. He gives us an overview of the engineering behind Spotinst and how the usage of containers and the rise of Kubernetes is changing the business landscape of the cloud.

If you listen to the show regularly, please fill out our listener survey. We're gathering feedback and we want to know how to improve. Go to softwareengineeringdaily.com/survey. Tell us how to improve. We read all the feedback. We read all the results. Also, I'm looking to figure out what to cover in 2019. Send me an email, jeff@softwareengineeringdaily.com if you have suggestions, and I really want to have a cohesive narrative throughout this year and touch on the subjects that people are truly interested in.

We're also looking for sponsors for 2019. If you're interested in reaching the 50,000 developers that listen to Software Engineering Daily, check out softwareengineeringdaily.com/sponsor to learn more, and you can help us out if you send an email to your marketing director, your CMO or your CEO. These people may not listen to Software Engineering Daily and they may be surprised that podcast advertising actually can get their message out quite effectively. It's often difficult to convince advertisers to buy podcast ads and your recommendation can go a long way. This money gets invested heavily back into Software Engineering Daily. We are working on products and ideas and editorial content that will expand what we can deliver to you as the listener.

As always, send me an email, jeff@softwareengineeringdaily.com if you're ever having a question or a criticism or a feedback. I want to know. I'd love to hear from you.

With that, let's get on with the show.

[SPONSOR MESSAGE]

[00:05:15] JM: Today's episode is sponsored by Datadog, a cloud scale monitoring service that provides comprehensive visibility into cloud, hybrid and multi-cloud environment with over 250 integrations. Datadog unifies your metrics, your logs and your distributed request traces in one platform so that you can investigate and troubleshoot issues across every layer of your stack. Use Datadog's reach customizable dashboards and algorithmic alerts to ensure redundancy across multi-cloud deployments and monitor cloud migrations in real-time.

Start a free trial today and Datadog will send you a t-shirt. You can visit softwareengineeringdaily.com/datadog for more details. That's softwareengineeringdaily.com/datadog and you will get a free t-shirt for trying out Datadog.

Thanks to Datadog for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:06:18] JM: Amiram Shachar, you are the CEO of Spotinst. Welcome to Software Engineering Daily.

[00:06:24] AS: Yeah. Hey, how are you doing?

[00:06:25] JM: I'm doing great, and I'm looking forward to talking to you about spot instances and the different products you've built at Spotinst. When a developer provisions a cloud server, that server is called an instance. Can you define the term instance?

[00:06:43] AS: Yeah. So you're basically right. When a developer launches a server in the cloud called instance, an instance basically is a virtual machine that the cloud provides you. The cloud is using different virtualization mechanisms. For example, AWS, using Xen; and Google are using KVM. Basically when you launch a server, they give you a virtual piece of that server, that physical server called instance.

[00:07:12] JM: Cloud providers offer multiple types of instances. Can you describe some of these different instance types?

[00:07:19] AS: Sure. As you know from like – And based from like – As we know servers from like the 90s and the 2000, basically you have like a bunch of hardware that you can build servers with. You can use hardware, which is more compute-oriented or memory-oriented of I/O intensive. So the cloud providers allows you to basically get different types of servers. They call it family types, and these family types are actually implying the names or implying on their usage.

So Amazon gives you family type of like C3, C4, C5, which is compute-intensive machines. Each one of them is a different generation. C3 obviously is the older one, and then C4 and C5 are the newest one where they're using a new CPU technology, like better Intel chips or different types of faster CPUs. You can also find types of servers, which are memory-intensive, like the M3, M4, M5.

Also servers with specific reasons that they are out there, for example like I series instances, which are I/O intensive; or Z instances, which are more for very heavy compute NMemory intensive workloads. So that's what when you say compute types, which basically what is the hardware that you're using.

[00:08:45] JM: How do the prices vary across these different instance types?

[00:08:50] AS: It's a very good question. When you look at – There's a general name that's called general purpose instances that are usually the C families and the M families that are compute-intensive and memory-intensive, and the prices are basically almost the same, maybe it's like a penny difference between one instance to another. For example, let's take M3. That's

a family of instances, and let's take a size from this family, which can be large or extra large, but let's a large for example. So that will cost you about anywhere from 10 to 30 cents. So the M3 will cost you maybe 11 cents, and the C4, which is on the same family of general purpose, it will cost you maybe 12 cents. So it's like very similar. However, when we go to very specific instance type for I/O or different purposes, we can go up to maybe 20, 25 cents for that same amount of CPU and RAM that you would pay for general purpose.

[00:09:59] JM: There is something called a spot instance that we should define. Can you explain what a spot instance is?

[00:10:06] AS: Spot instance, if you think about cloud providers, they are processing – They have a large amount of compute in their data centers, and all their idle compute, which is the compute that they don't sell, they want to somehow actually do sell it. They want to actually sell it. So what they do, they provide you with an access to the idle compute that they're not actively selling in a single moment, and you can get it on a first come first serve basis, and that's what spot is all about. You can get access to the idle compute in the cloud provider, but you're subject to get kicked off whenever the cloud needs that instance back.

[00:10:46] JM: Where does that term spot come from?

[00:10:50] AS: That's a great question. So spot is it's a historical term that people use for loading things to ships or like shipping oil. Whenever that you were filling a big ship with a lot of load from like a place to another. So always you had like a spare place left on the truck or a spare place left on the ship. So people would sell these spot for a really cheap pricing and like really few minutes before the ship will leave the port. Basically, it's all like real-time right now first come first serve just get capacity on demand.

[00:11:30] JM: And the idea here is that you've got these instances that are just available right now. They're ephemeral. They are in some ways lower in terms of reliability or quality or maybe you want to define why their cost is lower. But for all of these different types of servers that you defined, like on the AWS market, you have these C instances and the Z instances and the M instances and all these different types of machines that you can get. For all of these different types, there are spot instances. There are reserved instances. There are on-demand instances,

and the spot instances are the cheaper of each of these classes of machines. Can you explain why there are these different price categories for each of these server types?

[00:12:23] AS: Absolutely. As you mentioned, there are three main pricing models when you purchase compute in the cloud. The first one, which is the most common one, called on-demand, which basically go you have like a price list in the cloud provider website. Then if you provision an M3 large instance, you know that you're going to pay 11 or 10 cents for the hour build by the second or the minute, depends on the cloud that you use. Then it's yours. Nobody's going to take it from you when you shut it down. So the charge is full stop, but it's yours. Server is yours and you have SLA for this server if it goes down. So the cloud provider is liable for that. So that's on-demand.

The second one called reserved, whereas reserved, it's basically when you purchase like [inaudible 00:13:14] amount of compute and you're paying that on-demand price, you probably can commit for a year or two years or three years and exchange for this commitment that you're going to use this amount of servers. The cloud offers you a discount.

So with reserved pricing, you can reserve your amount of capacity that you desire and you can get anywhere from 30 to even 40, 45% discount if you commit to the cloud provider that you're going to use it for the next year or the next three years.

We have the last pricing model, which is getting more popular and popular overtime, which is spot. As we defined spot earlier, so the cloud provider has a lot of capacity. They don't sell all of their capacity in any single moment. That's why they're incentivizing their customers to purchase more capacity in a pricing model called spot.

With spot basically, let's say that Amazon have 100 pieces of hardware of the M3 large instances and are only selling right now 80 of them. So they have 20 that they can basically sell in a lower price. They can give it to people to bid on them in a lower price. When I say a lower price, I mean like a really lower price. They can get up to 80%, even 90% of the price. So think about it. Instead of paying 10 cents per hour, they'll pay around 2, maybe 2 cents, 3 cents for the hour. That's significant savings.

However, the caveat here is that with spot pricing model, you don't have SLA. So you have access to traditional compute. However, Amazon or the cloud that you use can reclaim that server from you at any single moment within a 2 minute notice. That's spot and that's why it's cheaper than on-demand and reserve.

[00:15:09] JM: The example you gave, let's say Amazon has 100 different instances of their M3 machines and 80 of them are occupied. So they've got 20 available. It seems like all 20 of these could either be sold as on-demand instances or as spot instances, or maybe they're all scheduled for reserved instances, but they're not being used quite yet. They've got an hour of availability. So are we talking about the same machines here? They're just available under different pricing models? Under different conditions?

[00:15:46] AS: Yes, that's absolutely true.

[00:15:49] JM: Okay. So obviously, there's some penalty to working with these spot instances, because they're so much cheaper. There's no free lunch. What are the difficulties of working with Spot instances?

[00:16:02] AS: So obviously, if you think about it, when you buy a server with an on-demand pricing, so you know that you have the server. You can put your data on that server. You can run your application. You can sleep with a peace of mind. The caveat with Spot is you basically need to realize that you're in a different situation. Like the server can go down at any single moment and you need to be ready for that.

In order to use that penalty to your favor, the server to go down, you need to actually architect to your application or make sure that you run specific type of application that can handle that. Otherwise you'll be in a situation where you pay less but you're compromising on availability, which usually you're not going to find companies that do that.

[00:16:50] JM: What are some good applications for these raw spot instances? In a second we'll get into what you do with Spotinst. But if we're just talking about the raw spot instances, how did people – I think Netflix was probably an early example of how spot instances were used. How have spot instances been used in their raw form?

[00:17:11] AS: Historically, organizations that were super cost sensitive, they were using spot instances for environment called batch, or offline job processing, which are things that are not customer spacing. Think about a university that needs to run a big computation around Monte Carlo algorithms, or you want to run something like DNA sequencing. So these things, it's not like a customer-facing or online jobs. For example, when you go to Facebook and you post a status or whatever, so you're expecting to get a response back. So that's like online-facing.

But when somebody hands over his DNA to a DNA sequencing company and he can wait for like three days until he gets an answer, but these are like these offline job processing that people historically were using spot instances. They got very popular in this area of batch computing.

[00:18:10] JM: Right. So the reason for that is because if a batch job gets scheduled on to a spot instance, and then the spot instance disappears in the middle of the batch job, well, you can just schedule it on to another spot instance and rerun it and it's going to take a bit longer, because you wasted some time on a spot instance that disappeared because the cloud provider needed to reclaim it. But it's not going to affect any users immediately, because this is an offline job.

[00:18:39] AS: That's 100% right.

[00:18:41] JM: Okay. So you are the CEO of Spotinst. What does Spotinst do?

[00:18:47] AS: So Spotinst, we had a pretty interesting story that started about four years ago. I was a student for a computer science in a university in Israel. Basically my thesis – Or I tried to do my thesis around cloud computing and how to optimize resources and kind of like save money in the cloud with some clever algorithms. When I first read about spot instances, it really got me, and I tried to do my thesis around how can you reliably use spot instances not only for these batch job processing, but for online user-facing environments. What I tried to do is basically how can you anticipate that as a customer? How can you anticipate when servers will terminate so you can take actions ahead of time and make sure that even that your app cannot tolerate interruptions or cannot tolerate, its service will service will disappear on it. You'll be able

to take actions ahead of time and migrate the necessary data or the necessary network and storage configurations from one instance to another so you can basically run some user-facing production applications on spot.

[00:20:01] JM: When you first started tinkering around with the spot instance market and you were trying to think of ways to turn these spot instances into more durable compute resources, what did you learn?

[00:20:19] AS: I learned that it was pretty interesting to see that there are very constant trends that I was able to learn kind of like very easily at the beginning, and then things got very complicated overtime. However, I learned that it's all about demand and capacity. So if you know to look at the full cycle of the day and understand in which hour you would like to gain capacity, and if you know what happened yesterday and what happened a week ago on the same day, you're probably going to be taking way better decisions than you would do if you just like provision instances on your own.

[00:21:00] JM: So if I am using Spotinst, how does my experience compare to managing spot instances myself? Explain what I would do as a Spotinst user.

[00:21:16] AS: So as a Spotinst user, you will just ask capacity in the same that you're asking capacity from your cloud provider, for example, Amazon. You would just go and use the Spotinst interface or the Spotinst APIs. What we'll do, we'll just go to your cloud account, your AWS account and ask for these resources. The experience of getting the spot instances is 100% the same experience. However, the experience of like after you're getting the spot instances is a different experience, because as the user, when you do it yourself and you get spot instances, so great. Everybody can get a spot instance, but then what's next? What happens if out of 10 instances you're losing 9, or out of 10 instances, you're losing 5? Then your application is in trouble.

While with Spotinst, we provide a software that actually predicts when the next interruption will happen. We're always 24/7 and in real-time also cross-referencing historical data. We're anticipating when instances will disappear. We're preemptively and proactively launching new capacity and adding it to the application clusters and then we are actually migrating the data or

the configuration that its needed for by the customers and [inaudible 00:22:37] data experience where the customer is like 100 to 50X less interruption rates in terms of like observing interruptions and problems in the applications, and our goal, our mission in the company is when you run your application through Spotinst, you'll just feel that you run it on on-demand instances.

[SPONSOR MESSAGE]

[00:23:07] JM: Clubhouse is a project management platform built for software development. Clubhouse has a beautiful intuitive interface that's made for simple workflows or complex projects. Whether you are a software engineer or a project manager or the CEO, Clubhouse lets you collaborate with the rest of the product team.

If you're an engineering-heavy organization, you will love the integrations with GitHub and Slack and Sentry and the other tools that you're feeding into your issue tracking and project management. To try out Clubhouse free for two months, go to clubhouse.io/sedaily, and it's easy for people on any team to focus in in on their work on a specific task or a project in Clubhouse while also being able to zoom out and see how that work is contributing towards the bigger picture. This encourages cross-functional collaboration. That's why companies like Elastic and Splice and Nubank all use Clubhouse. Those companies have all been on Software Engineering Daily and we know they are competent engineering organizations.

Stay focused on your project and stay in touch with your team. Try out Clubhouse by going to clubhouse.io/sedaily and get two months free.

Thank you Clubhouse for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:24:37] JM: So you're giving people the ability to run these long-lived interactive user-based applications on top of spot instances, because you are able to predict when a spot instance is going to disappear. So when you can detect that a spot instance is going to disappear or when it might be about to disappear, because like you said, you don't have this SLA from the cloud

provider. The cloud provider can grab it from you at any time if somebody's willing to pay a higher price than you for that instance, they're going to take that away from you. But you can preemptively schedule another spot instance and put the application on to that instance. When the cloud provider is ready to take it away from you, you can just route the traffic to that fresh instance that gets spun up.

So when you are trying to detect when that's going to happen, when the spot instance is going to be taken away from you, what are some signals that suggest that the instance might disappear?

[00:25:47] AS: That's a very good question. Basically, our web algorithms that runs in the background and always creating a forecast based on historical disappearance of instances, we're kind of predicting how the next – When the next interruption will happen. So just something which is very important to mention is that when you're using different compute types on AWS, so it's called uncorrelated. What do I mean by uncorrelated markets? For example, if you purchase M3 medium and M4 medium, that's a different type of hardware.

So the fact that Amazon is running out of capacity of M3 medium doesn't mean that Amazon is running out of capacity on M4 medium. What our platform does in the background, we're looking at M3 medium and we're looking at trends of M3 mediums and we see, "Hey, this is right now Monday 8 AM in the morning, and usually at 8 AM, 8:10 AM in the morning, something usually happens in terms like disappearing of instances."

Where like about 7:15 or even before, we're scheduling new capacity to launch into that cluster and we'll schedule that from an uncorrelated market from a market that we understand that there will be capacity at this time of the day based on historical data, and then we can safely go through that interruption of eviction of instances during the time that we were sure that it's going to happen. So there is a machine learning in the background that always fixes that, but that's like on the surface just to explain what we do behind the scenes.

[00:27:26] JM: Are you continuing to iterate on those machine learning models? How rapidly is the market changing? Are your models as good as they need to be today?

[00:27:38] AS: It's always changing, because Amazon is always adding new compute hardware. So you need to learn it just like from scratch. They're changing capacity. We see it all the time. Obviously as a cloud provider, they need to do a lot of reallocation of capacity between data centers within that data center. So it's a system that always learns itself and learns from like things that you predicted and didn't really happen. So you always know how to fix it. We actually have like a team of people of data science that are always looking at the data and doing the relevant corrections to that.

[00:28:16] JM: What do you need to do to shift the workload from one machine to another when that instance is going to disappear?

[00:28:27] AS: That's where it depends on the workload type. So to give more context, when you launch an application in the cloud, you would usually either use a single server or you're going to use something called like auto scaling groups. Basically like a horizontal way to your scale your servers when you need more capacity in an automated way. Basically, then you can run containers on top of that. You can run a microservice. This microservice can be Java, can be Python, can be Node.js. It depends on you.

When we as customers to provision capacity through us or their clusters through us, we basically want to know what is running on top of it. Is it Java? Is it Node.js? Is it a container? Because all these information is going to help us when we're actually going to deal with the inevitable. We just replace the spot instances, because in every different type of app, we'll take a different top of decisions.

For example, if we understand that you have a microservice based on Node.js that is running behind an nginx load balancer or running behind an Amazon Elastic load balancer. So we know that we have like certain steps that we need to make in order to launch a new Node.js app, reroute the traffic to the Node.js app, but we need to make sure to deal with the existing connections, existing HTTP connection, TCP connection, database sessions that were already open on that application server. So we let the server drain. We realize, "Hey, whenever the server ended the draining period," and then we can safely terminate that.

So it really depends on the app. We have many different type of apps that we support. The most common ones are microservices, web applications, containers and there are several other more, but that's basically what we do.

[00:30:27] JM: Are there any consistency issues that can develop when you're trying to do this handoff from one spot instance to another? You're trying to shift of copy a service from one instance to another. I can imagine if one request is in the process of going through the instance of that service that's about to get decommissioned and you're about to reroute traffic to another instance, there could be some potential for consistency issues or availability issues. What are some cases that can rise there?

[00:31:07] AS: Yes. That's a good point. Usually in traditional web applications, as long as you know to manage the draining time properly, you can be safe, because usually it's a short-lived HTTP connection that are basically know the default timeout might be anywhere from 60 seconds to 120. So if you give it enough time, all the HTTP connections will close and then you can just monitor the server and there's like zero HTTP connections that are open and then you're safe to go and decommission the server.

However, in a more long-running applications, they're doing like transcribing a video or opening like really long session with databases and access to memory, this is where you really need to be sophisticated with a training timeout and the way you replace servers. This is where we introduce to customers different features that it can use in order to tell us, "Hey, if you decommission the server here, I need to wait at least one hour, or I need to make that three steps of like closing the connections with predefined scripts and only when you get an okay from the script, you're good to go." So we've seen some pretty complex stuff over the years.

So the last thing that I would mention here is that we have one rule of thumb of things that like don't ever use on spot instances, which is a single point of failure. So we have to have a single failure, maybe a single database or an application that it can run only on a single instance. It cannot be moved to another one at all. So that's probably a place where you want to – You would not want to leverage spot from the first place.

[00:32:56] JM: Today you have a lot of customers, because obviously this saves people money. There's a lot of volume going through your system, and this is good for you because you have some economies of scale, but it also comes with the challenges of scale. How has managing Spotinst and the Spotinst infrastructure, how has it changed as you've gotten bigger?

[00:33:22] AS: I think as you defined it as the economy of scale. I think that the more customers we have and the more instances that we manage on behalf of them, the smarter we become, the more accurate we become, the more easier it becomes for us to predict stuff. When we started, we just had like data points from less – I don't know, 100, 500 machines, and today we manage several hundreds of thousands of machines in any single moment. So we can see what's going on in any single place of the cloud provider. It helps us to be way more accurate than before.

Regarding the challenges of scale – So obviously as people want more access to do spot, so you need to realize that there is limited amount of compute that you can possibly get, because you're limited by the cloud provider. However, if you know to distribute and scale it across different compute types and hardware types, this is where you can get the economy of scale, because a cloud provider has a lot of different machine types and a lot of data centers. If you know how to utilize them correctly, you can both support the scale that people want and also not to exhaust the compute in a single machine type.

[00:34:44] JM: In the original Spotinst product, you needed to be able to schedule these VMs. You needed to be able to request new spot instance VMs as your customers are requesting more resources. I'd like to know about the process of building that first product and some of the code that you wrote for that first use case.

[00:35:09] AS: So that's a very exciting question, because always when I go back and starting to remember the first day of the company, it's really exciting. So without knowing like how Amazon is working in terms of like culture, so we always had this in mind, which is start with a customer and then work backwards. We were meeting with customers who wanted to leverage spot and we were asking them how would you like it to be? What is the experience that you would like it to be? How would you like to provision the compute and how would you love to use the service? We interviewed a lot of customers, and then eventually we worked backwards and

developed the things that they actually needed and asked for. We ended up with developing a cluster software that is basically targeted for developers and operations or dev ops people or IT persons that is basically the same as the cloud APIs for launching a VM.

You can just choose. It's very dynamic. You can choose if you want to get one, if you want to get 100, if you want to get a dynamic number of servers. You can start with one and then scale to 100. That's how it all started, and the standard was that if you looked today about a software that we provide, which is called Elastigroup. So it's very similar to the cloud APIs as people know them today.

[00:36:35] JM: What programming languages did you use for the first version?

[00:36:38] AS: We used Node.js and Java.

[00:36:42] JM: And where does the code first Spotinst itself run? Do you run in a cloud provider?

[00:36:48] AS: Yeah, absolutely. We leverage mostly AWS. We run our code, our software on AWS. We leverage multiple regions for high-availability, but 90% of our stuff is running on Amazon.

[00:37:01] JM: It's interesting to talk to companies like yourself who have built a cloud provider on top of another cloud provider. Like I just did a show with Guillermo from ZEIT, and that's another cloud provider built on cloud providers. What are some of the fundamental difficulties of building, I guess you would call it second layer cloud provider?

[00:37:25] AS: Yeah. The customers who are using the cloud, they already get like an excellent service. They get amazing SLA. They already get a very good – They already have like really high expectations from you, because your competitors or like your alternatives are the cloud providers. So building a layer of cloud providers on top of cloud providers, customers had like a really, really high expectations and are very demanding. So that was – I can tell that it is a challenge for sure. You cannot be like not good as them, because people are not going to use you.

The second thing is that the cloud providers are just acting like startups and they're moving as fast as startups. So keeping up their feature sets, keeping up with their offering, and also run your own business, that's also very challenging.

[00:38:25] JM: You started the business in 2015 and since then the world has moved more towards thinking about containers rather than VMs. How has the shift to containers changed your product strategy?

[00:38:40] AS: You are correct. Basically, when we first started the company, so mostly were VMs and applications on top of VMs in a very traditional way, then just because we're working with so many customers and talking to so many customers every week, we just started to hear the word containers and containers again and again and we also heard about Kubernetes.

People starting to leverage Kubernetes, thinking about Kubernetes, and at the beginning you're not sure of like these signals that you're hearing from the market they are real, or just another tool that people will just like – It's another hype or something like that. But then when we're looking in our statistics, so we're actually looking about the workload types that we're managing. For example, we want to know any single moment how many web applications versus containers, versus like Node.js, Python, Java stuff we're managing. So we can get better decisions on our product roadmap.

Then we saw like – Since 2017, we saw like a sharp increase and people asking for containers are actually starting to use containers in production. That had a lot to do with our roadmap, because then we saw that containers, when you run containers on top of VMs, so it introduces another level of complexity that we realize that we need to solve even bigger problems right now as people are using containers.

[00:40:05] JM: How did that shift what you had built? Let's take an example, if somebody schedules a bunch of containers on top of a VM that they've requested from Spotinst, was there any inherent difficulty in that or did your platform continue to work when people were started breaking up these VMs into containers?

[00:40:28] AS: Think about it. When we used to manage traditional applications, like an application that actually runs on VM, so they are coupled. So you know whenever the application needs more capacity or like CPU is going up. So you can just provision out of VM. It's very logic. With containers, what happen is that right now you have two layers. You have the layers of VMs and then you have another layer, which is the containers, and then the apps running on top of the containers. So like there is another level of abstraction from the VM to the app, which is the containers in the middle.

So whenever customers want to run one container on a single instance, that easy. But when they run, as you mentioned, schedule more containers on to VMs, then it becomes tricky, because there are many ways how to do that. When you're requesting additional one container, additional 10 containers, like the infrastructure or the service provider needs to respond differently.

We've seen customers like telling us, "Hey, guys. We're trying to schedule these containers and we're getting these errors. What can you do for us?" Then we realize that there is like really complex mathematic issue, which think about an example, you run two servers. Two of them are like medium servers and each one of them can run two containers in a specific size. Then a customer needs 10 containers or 20 containers, additional 20 containers. So you need to have like a very smart system in the background that not only introduces new compute whenever you need more containers, but actually looking at the containers, understanding what are the requirements of the containers, because it can very dynamic. You can request a container with a single gig or RAM or 10 gig of RAM. It's really dynamic according the developer that's asking for the container. So we started to build a layer on top of our software that basically listens and understands what are the containers needs, and according to that, introduces the capacity that these containers need.

[00:42:35] JM: Right. So is it also an issue where let's say somebody provisions a VM and they run let's say service A and service B on that VM and then service A is getting hammered and hammered and hammered with traffic, and so they need to scale up the container of service A, but they don't really need to scale up the container of service B. But the way that Spotinst was originally architected, I'm imagining that the scalability would be, "Okay, let's spin up an entire new VM with both service A and service B," and you don't actually need additional instances of

service B, the container in service B, but you're going to get it anyway if your system is built in terms of VMs.

[00:43:25] AS: Correct. That's a big problem that was happening at the beginning. Even further than this, think about it when if service A, which service A is requesting 2 gig of RAM and 2 vCPUs and then something changes on service A, and service A right now is asking for 6 gig of RAM and 4 vCPUs per container, and then like the traditional way of deploying machines is that, "Okay, I need a medium, and if I want to horizontally scale, I'll get more 10 medium or 100 mediums."

But then you'll get another medium or another 10 mediums, but the new service A size, they're going to fit this new medium. It's like there is an alignment between the container size that you need and the VM that you're getting. These two problems, where like the main problems that we saw, we said, "Okay, we need to solve that.

[SPONSOR MESSAGE]

[00:44:35] JM: Software engineers can build their applications faster with the use higher-level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin up cost-efficient, short-lived infrastructure. Container platforms let you scale your long-lived infrastructure for low cost, but how do you get started with all of these amazing tools? What are the design patterns for building these new types of application backends?

IBM Developer is a hub for open source code, design patterns, articles and tutorials about how to build modern applications. IBM developer is a community of developers learning how to build entire applications with AI, containers, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job. Get help with side projects, or think about how new technology can be used to build a business. A softwareengineeringdaily.com/ibm, you will find the resources and community who can help you level up and build whatever you imagine.

Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications, and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless, and containers, and artificial intelligence APIs.

So thanks again to IBM Developer.

[INTERVIEW CONTINUED]

[00:46:34] JM: You eventually built a product called Ocean, and this is a platform on top of Spotinst for deploying containers specifically. Explain what Ocean is.

[00:46:46] AS: Yeah. Ocean is first of all our recent product which we launched about a few weeks ago, and Ocean, based on what I've just said, Ocean is technology that we realized that we have to build when people are leveraging containers on top of VMs. Because we've listened to the market and we saw two main signals. Like one, we want to unlock compute for a reasonable price, and that's what we were solving from day one.

The second is that people want like a more hands free, and we call it serverless way of managing their apps. They want to deal with less compute. They want to deal more with their applications. About a year ago or maybe 18 months ago I think the word serverless started to resonate a lot with people, like Amazon was pushing a new product four years ago called Lambda and people started to understand what are the benefits of running application in a serverless way, and a serverless way that means that you provision your app, and then that's it. You don't need to manage the compute. You don't need to scale that compute. You're paying only when you've used this compute, and whenever you don't use it, you don't pay. So it's like it's a utility billing. It scales horizontally if this service has hammered. So you'll get more copies of compute and it scales down to zero when you don't use it.

We saw these two signals, like people want to manage things in a serverless way and people wants to get better compute offering and pricing, and when we looked at Ocean at the

technology, it's doing exactly these two things, but for containers. So Ocean is basically introducing an interface to developers or dev ops that they can launch and provision their containers through their preferred scheduling systems. It can be Kubernetes. It can be Amazon ECS.

Very soon it's going to supporting Docker Swarm or HashiCorp Nomad, any scheduling platform for containers that you wish. Then once this container reschedule, the compute is basically abstracted away from you. It will scale. It will scale down to zero. It will scale to whatever copies of compute that you need automatically. You don't have to do anything. You don't need to manage the VMs in terms of like security patches or OS patches. It's already done in the background by Ocean. Basically gives the developers serverless way to manage their containers and also very, very reasonable pricing of compute they can unlock as they scale.

[00:49:40] JM: The experience of deploying a container on your product, Ocean, it seems quite similar to deploying a container on an AWS Fargate or Azure Container Instance. These are these long-lived container instances, the serverless containers. How do the containers that you're spinning up for people on Ocean compare to these container instances that the cloud providers offer?

[00:50:15] AS: So I think we're hitting a spot where we basically offer the same service, like Azure Container Instances, the ACI or Firegate, and it's just a matter of like how efficient we are in acquiring the underlying compute, and our goal is to be way more efficient at any other provider in acquiring the compute underneath of the container.

[00:50:42] JM: Right. Now I find these container instances so interesting and so useful, because I've been a longtime user of Heroku, and Heroku is as far as I know, just giving you a container. It's been giving people containers for a very longtime since before containers were trendy, and they're called dynos on Heroku, and Heroku has made a great business out of selling Dynos to people, because that's all you need often.

Many people who are hobbyist developers, or even at a large company and they wanted to use Heroku. They can spin up their entire application as just some monolithic application that just runs in a long-lived container, and that's great. That's what a lot of people need. So that's what

AWS and Azure have been offering with this Fargate and container instances products respectively, and I was talking to you about this when we were at KubeCon. But it surprises me that there's not more excitement around the idea of these low-cost cloud provider container instances where you can just spin up your monolith in a standalone container. People see more excited about the idea of managing a Kubernetes cluster, when in fact like why would you want to be a cluster operator. You could just be an operator of your application abstraction. Why is that? Why do people want to manage their Kubernetes when they could just manage their application in terms of these container instances?

[00:52:22] AS: Jeffrey, I couldn't agree more. I think that you're absolutely right in terms of – Also, it's great to see what Heroku did, like becoming so trendy now. Also, if you go like even 10 years back and look at Google app engine, which is basically a very same thing, but just people weren't ready for that yet.

[00:52:41] JM: That's true.

[00:52:43] AS: I think this is like the strength of AWS, which they're actually meeting the market whenever the market needs that. They will not introduce some spaceship when you just need a regular car, and I think that's a big thing for Amazon. Right now it's becoming popular, because people did that runway of like, "Hey, [inaudible 00:53:03] what is container. Okay, we're sick and tired to manage Kubernetes. So we want like EKS now. We want manage control plane of the master." I think that people are still not yet that excited about the container instances and these serverless containers, because right now a lot of people are in the exploration mode. They want a lot of control. So they want to deploy. They want to see what happened, and I think that the – We're going to see pretty soon how the excitement is going to sharply increase around this. We're already seeing it from our customers when we pitch them and tell them, "This is what Ocean does."

I see the aha moment. I see like how people like their eyes are shining. They're saying, "Oh, are you serious? Can you really deliver that? If yes, that's like the holy grail. We just deploy containers and then the infrastructure self-manages itself," and I think we're hitting that sweet spot right now.

[00:54:02] JM: So what's the contrast between the experience of somebody who distributes their microservices application across a self-managed Kubernetes cluster versus distributing their microservices across container instance? Whether their container instances are on Fargate, or on ACI, or on Spotinst Ocean? Is there something you get out of distributing it across a Kubernetes cluster you're managing yourself?

[00:54:32] AS: Of course. It's just like something very manual versus something so automatic. So if you think about it, if you run your containers on a static infrastructure, so you're probably paying anywhere from like 30 to 40% more on the underlying compute, because it's going to be static. Maybe it will scale up and down from time to time, but it will not fit itself to the actual application need. That's the easiest part.

The worst part is that your application will suffer from performance issues. You'll have to use different tools and application performance monitoring to realize if you have a problem and then fix it yourself. Whereas with type of like container instances, serverless container, the container says, "Okay, I just need more compute, because somebody is watching the metrics of the containers," the containers increases itself. Then a new infrastructure just pops up with like a better compute, better memory and then things just run faster and better, and that's one thing.

The second thing is operational agility. Think about how much you'll need to patch your instances, deal with security stuff, provision it yourself, take care of your operation system maintenance and so on. Whereas with container instances, you're really doing less in operations stuff.

[00:56:02] JM: I want to zoom out, and I've talked to so many different companies about I feel like in the past, which is this vision of having a cloud provider that sits across all the different cloud providers and is able to arbitrage between the different cloud providers to offer the lowest cost for any given moment. For example, maybe if you want to spin up a container, you just want to spin up an application container, your ideal pricing would take into account whatever the price of AWS instances are, whatever the price of Azure instances are, whatever the price of – I don't know, DigitalOcean instances are.

It seems like Spotinst could potentially do that. Now, I know that a lot of people have tried to do this in the past and it hasn't really worked, this arbitrage model, for one reason or another, but is that something that you can do arbitrage between these different cloud providers and seamlessly move workloads between cloud providers?

[00:57:12] AS: It's a question that I'm dealing with a lot, because when people are looking at our business and seeing in fact if we support multiple clouds. So arbitrage is the first thing that ops in everybody's mind. To be honest, if you see the way that cloud providers are operating, they are trying to make sure that your application is so deeply coupled with the cloud toolings and services and managed services that it will be super hard for you to pick an application and just run it somewhere else.

Reinvent was like a really good proof for that, where when you saw all the different tools and services that Amazon is offering, it's just like making applications so deeply and tightly coupled with Amazon. So it will be almost like impossible to take an application and run it somewhere else, unless you're so advanced and all of your stuff and services are cloud native and you're running Kubernetes and Prometheus for monitoring and all the different natives, open source projects that are installed on your infrastructure, and then it becomes super easy. Okay, it's either you're getting oil from AWSs compute or getting oil on a different cloud provider. I think that Amazon is doing a really good job on looking at the future and making sure that services are tightly coupled with their compute and infrastructure.

I see our job moving forward just meeting the customers wherever they are. If you're using Amazon in a very tightly coupled, we'll meet you there. If not, we can help you even further. If you're using Google in a certain way, we'll meet you there and we'll help you that way so that we'll – That's how we see them market. That's how we see the market moving, and I think we need to be able to align ourselves to that.

[00:59:10] JM: Out of curiosity, what are those very tight couplings that AWS introduces?

[00:59:17] AS: So things like Azure Container Instances is a really good example, but think about like queuing as a service and databases as a service. I think that at Reinvent, you saw that like Amazon is aiming really, really big on becoming like the one stop shop database cloud

vendor. So they have all these [inaudible 00:59:39], PostgreS, SQL and MySQL managed by Amazon and then they also give you NoSQL databases, graph databases. Once your data is already in the cloud in the managed cloud service, it's really hard to go backwards. Really, really hard to go back, as well shipping your messaging bus across services. If you use like a managed service, like SNS, SCS, SQS for that. It's becoming almost impossible to go out.

[01:00:11] JM: I understand with the queuing services, because the APIs for the queuing services are very Amazon-specific, but if you're writing to a managed database by Amazon, aren't the APIs the same from one database provider to another, just PostgreS or –

[01:00:27] AS: That's true, but data has a lot of gravity. It's like migrating right now, like X amount of gigs of data or terabyte of data is not as easy and it can take time and –

[01:00:43] JM: It's scary.

[01:00:44] AS: It's scary. Yeah, that's the right word.

[01:00:48] JM: Yeah. As we begin to wrap up, what are the other opportunities that you're looking at? What's in the future for Spotinst?

[01:00:58] AS: So future for Spotinst is we look at this in like two ways. We look at like east and west, and north and south. What do I mean by that? So east and west support as much as we can across different cloud providers, like from Amazon, to Google, to Azure, to IBM. Maybe future Oracle. So we want to meet customers as they are meeting with more cloud providers and provide our software to the personas that we're already selling them today.

Second thing is north and south, which is it's also something which is increasingly getting more popular as cloud and on-prem. So we see a lot of people that are using AWS and using on-prem. Especially with containers, especially with Kubernetes, we see people that have Kubernetes on-premise and they're bursting to the cloud or have Kubernetes on-prem and Kubernetes in the cloud and they just need a better way to manage the ratio between what they're managing on on-prem and what they're managing in the cloud. But we're investing in these areas a lot. That's like one thing for us.

The other thing is how do we become the perfect automation platform for dev ops. Very soon, we're going to be introducing services like in the areas of CI and CD. Today we're managing the infrastructure, managing the containers, but we want to take like even one step back in the chain and, okay, if people need better continuous deployment, better continuous integrations, there are so many tools out there and we just want to help people with standardizations. I think that's what's next for us, and it's very exciting times.

[01:02:44] JM: Very cool. Well, Amiram, it's been great talking to you. I enjoyed our conversation at KubeCon, and I think you're in a really interesting spot and you've clearly got a lot of traction and I'll be curious to see what else you build in the future.

[01:02:58] AS: Absolutely, Jeff. Yeah, it's great to be here and also great conversation. So thank you for having me and we'll definitely stay in close touch.

[END OF INTERVIEW]

[01:03:09] JM: Kubernetes can be difficult. Container networking, storage, disaster recovery, these are issues that you would rather not have to figure out alone. Mesosphere's Kubernetes-as-a-service provides single click Kubernetes deployment with simple management, security features and high availability to make your Kubernetes deployments easy. You can find out more about Mesosphere's Kubernetes-as-a-service by going to softwareengineeringdaily.com/mesosphere.

Mesosphere's Kubernetes-as-a-service heals itself when it detects a problem with the state of the cluster. So you don't have to worry about your cluster going down, and they make it easy to install monitoring and logging and other tooling alongside your Kubernetes cluster. With one click install, there's additional tooling like Prometheus, Linkerd, Jenkins and any of the services in the service catalog. Mesosphere is built to make multi-cloud, hybrid-cloud and edge computing easier.

To find out how Mesosphere's Kubernetes-as-a-service can help you easily deploy Kubernetes, you can check out softwareengineeringdaily.com/mesosphere, and it would support Software Engineering Daily as well.

One reason I am a big fan of Mesosphere is that one of the founders, Ben Hindman, is one of the first people I interviewed about software engineering back when I was a host on Software Engineering Radio, and he was so good and so generous with his explanations of various distributed systems concepts, and this was back four or five years ago when some of the applied distributed systems material was a little more scant in the marketplace. It was harder to find information about distributed systems in production, and he was one of the people that was evangelizing it and talking about it and obviously building it in Apache Mesos.

So I'm really happy to have Mesosphere as a sponsor, and if you want to check out Mesosphere and support Software Engineering Daily, go to softwareengineeringdaily.com/mesosphere.

[END]