Company Participants

- Al Gillen, Group Vice President, Software Development and Open Source
- Gary Chen, Research Manager, Cloud and Virtualization System Software
- Mandeep Singh, Equity Research Analyst - Software

Presentation

Mandeep Singh {BIO 15014535 <GO>}

Good morning, and welcome to the BI Analyst Briefing with IDC on Containers - A Possible Growth Area for Infrastructure Software. My name is Mandeep Sing, and I'm the Software Analyst at Bloomberg Intelligence. With me today are Al Gillen, Group VP, IDC; and Gary Chen, Research Manager at IDC.

A couple of housekeeping notes, today's presentation will be recorded and available for playback. Feel free to ask a question by submitting one to the right of the slides. We will address questions at the conclusion of this presentation.

What is BI? Bloomberg Intelligence provides in-depth analysis and datasets on industries, companies and government, ESG, credit, economic and litigation factors that can impact business decisions. Bloomberg Intelligence research is backed by 300 plus third-party data providers and is delivered with no buy or sell recommendations. Our analysts team averages 20 plus years of buy-side and sell-side experience. BI is exclusively provided to clients who subscribe to the Bloomberg Professional Service and can be accessed at BI GO.

I'll hand it over to Al and Gary to kick off the presentation. Al?

Al Gillen {BIO 3529173 <GO>}

Thanks, Mandeep. So, get us to the right slide here. So what we want to do at -- sorry, I'm looking from a -- slide doesn't exist, okay. so what we're doing today is, we want to talk about containers and I'm joined by Gary Chen. This is Al Gillen speaking right now. And just to give you a little bit of background; Gary and I have been involved with infrastructure-oriented research for a long time at IDC. Gary is our lead on virtualization and software defined compute and containers very much roll up in his space. I've worked collaboratively with Gary in that area for a number of years.

More recently, I took over our developer research and now I'm looking at the container technology from more of the developer angle. So what we're going to do is, I'm going to kick off the presentation in just basically four areas that we want to talk about. We want to start by explaining what IDC sees as the -- really the driver, which is changing the industry, and we refer to it as digital transformation or DX, so we'll talk a little bit about that, what it is and what impact that has on customers.

We'll talk about cloud native applications specifically and why they're so important and why they change some of the norms for application development that we've had in the past, and then we'll dive into container specifically and give you a little bit of background on what they are and why they essentially change or they bring new construction techniques and new management and lifecycle paradigms to the mix that customers are going to be deploying as part of their digital transformation. And then we'll talk about orchestration, because containers alone are only half of the story, you really need to have an orchestration engine in conjunction with containers to get the full benefit out of them, so we'll talk about that. And then, of course, we're expecting to have a very healthy and robust Q&A discussion after that point.

So moving forward to the next slide, focusing on digital transformation. I apologize, I'm getting some -- Mandeep, I'm getting a collection of error messages on my screen. I'm presumably --

Mandeep Singh {BIO 15014535 <GO>}

We have the right slide in front, so you can go ahead.

Al Gillen {BIO 3529173 <GO>}

Okay. Thank you, yeah. So anyway, the slide titled Digital Transformation or DX, so when we think about that from IDC's perspective, where we see the industry moving? We like to categorize it in three broad periods of time. We had the first platform which was mainframes, the connectivity typically was direct-to-cache and the access device were fixed function terminals.

The second platform is of distributed computing and really we're just coming out of the second platform each today. In the distributed computing or the second platform era, we used UNIX servers, Linux servers, Windows servers, x86, and risk-based system very typically. We use local area networks and wide area networks as a connection medium and primarily PCs as our point of access.

And, of course, as we transition into the third platform, we're moving into a scenario where much of the backend infrastructure moves over time to cloud, the front-end devices are a whole collection of things are not just simply PCs and that may be one of the devices we use, but of course we use phones, and we've got things that are going to be accessing this data as well. Intelligent things or not so intelligent things.

So what happens here is, as we start to move forward to this third platform era, we have to do this digital transformation to really take advantage of it. And a number of things that happen as we move through that conversion. One is that, we start to look at a scenario where our infrastructure becomes more standardized. And the rationale or the reasoning for having standardized infrastructure is simply because you get more benefits of efficiency if you can have a very, very common set of baseline infrastructure servers and services that you're consuming.

And the reality is, as you start to think about a movement to a cloud-based infrastructure, you get a standardized infrastructure when you're in a cloud-based environment, you can call up Microsoft or Amazon and say, hey, I want to have a special server in my cloud. I mean you get what they make for their base infrastructure and since their all customers get the same basic infrastructure.

Today, many customers who are consuming Infrastructure as a Service or IaaS typically may have the opportunity to do some specialization at the base infrastructure layer. But frankly we think that's not necessarily in our best interest long term, simply because as they move from an IaaS to a Platforms as a Service model, as they get to a Platform as a Service model, customers will not have any say of what that infrastructure looks like.

So again, the benefit of having a standardized infrastructure in your datacenter today is that you wind up with something which looks and acts a lot more like the cloud environment that you will be using over time.

The second piece about this is, you've got to modernize your applications. So the applications themselves no longer are they monolithic nature as they were during the second platform era. Today, we have customers that are starting to build applications in a much more modular fashion and that really ties back into container, the use of containers, and use of cloud native type development techniques. And we'll drill into that with a little bit more depth in a couple of minutes.

When you start to get to that model where you've got these modern applications, which are largely componentized and are essentially small -- broking down into smaller pieces which individually can go through their life cycle. What happens is, we've got to move through different model of actually deploying and managing those applications, and that's where the notion of using a DevOps methodology or DevOps process to manage in life cycle of those applications.

So those are some of the things that really are very important. As I mentioned before, I mean the benefits are -- having this type of a DevOps process together with a modern cloud native type application portfolio. It allows you to iterate rapidly, allows you to deploy, on-prem off-prem and frankly move between the two and it also allows you to have a great deal of flexibility and agility, one of things that customers are looking for today.

Okay, so moving on, I think I hit a lot of these points already, but this is just a summary list of some of the digital transformation, digital transformation benefits and activities that customers are going through. So again that's standardization of the infrastructure is one piece of it, you will see customers that are starting to make sure that all of their systems are interconnected.

There's more automation, things are managed autonomically or automatically and things tend to go through their life cycle in a much more automated fashion.

And the other thing is, we've got data. A big part of the digital transformation includes consuming data from a wide variety of sources. We certainly have social media and social data that's pulled in and utilized for applications. We've got things like weather data and census data, which is not necessarily changing as quickly but consist a fairly large datasets. And, of course, I skipped over the first one, the Internet of Things, there's going to be a tremendous amount of data coming in from the Internet of Things as we go forward.

So that's important to be able to consume those sets of data and leverage them into your applications, and that's part of what happens as customers go through a digital transformation, so that data gets organized and stored. We've got to make real-time decisions based on that data and if you can't act on that data in a real-time manner and your competitors do, you're going to find yourself at a significant disadvantage when your -- in a competitive situation.

And the bottom line here, the final thing I think about is software intellectual property for the application functionality that really becomes a differentiator and it becomes something which was delivered a lot of the value, that we believe that a lot of the value software is going to be monetized not as subscription or as license sales as we've had historically, but rather a lot of the software intellectual property becomes part of a set of resources that devices and things have accessible to them and as these things are able to do -- make smart decisions and function using the software. The software gets monetized, because the things themselves become more valuable. So just very, very important how that all tied together.

Okay. One more slide, then I'm going to get Gary to start jumping in on the next couple of slides after that. Cloud native applications, we talked a little bit about that and I think it's important to recognize that cloud native applications are one of the cornerstones of what is going to make a digital transformation possible, as one of the things that are going to empower organizations. And, by the way, the relationship between a cloud native application and containerized infrastructure is going to be pretty close. We don't think many organizations are going to go and build cloud native application portfolios without using container technology as the underlying packaging format that they use.

So just a few bullet points here and I don't want to drag you through too much detail, but basically the design principles in -- is, for example, what it means that the application components tend to be a stateless as possible and there is a notion of scaling them out. You need more capacity, you spin up more instances, you time back together through an orchestration engine.

These assumptions here that the infrastructure itself may or may not be reliable, which means that the applications themselves are able to move around and find the infrastructure they need and they're built for an environment where they find a way to continue to work. If a particular site or geography becomes unavailable, the applications can reconstitute themselves in other areas and continue to function as a collection of services.

I mentioned about the notion of having things broken down into small components, we refer to that typically as microservices. And typically what you do is, you build an application component, if you will call the microservice and you front-end that with fairly stable -- highly stable APIs or application programming interfaces that will allow you to write to that application component and take advantage of the service that it provides back to you. And then the presumption is that the APIs remain stable and the way the APIs produce results and return the data if you remain stable. The underlying code may change and probably will change on a fairly regular basis, but that's not visible necessarily to people consuming the APIs.

In terms of the updating and life cycling, so you can start to refresh these components by essentially doing a rolling refresh, where you start to replace individual instances over time. In some cases, you would wait till new instances or spun up and allow the new ones to be spun up using a refreshed version of the code versus the older version of the code, and as they age out and get removed from the system the older code goes away.

In many cases, we're not going to fix or patch/repair microservices once they're put into use, you'll just basically replace them with new instances. And, by the way, it's important to note that one of things that's functionally different here is, in the past, we wrote an application and it was an ERP application or it was an HR application or it was a sales force automation application, and that was the function of the application itself.

Going forward, as you think about this notion of a service, the service can be used for any type -- any number of different types of applications. So, for example, a scientific function could be used for an HPC applications or could be used to front-end or to do some calculations on data that is coming from an IoT data stream. So, if the services is useful you can use it.

I mentioned about the agility notion that these applications -- these components tend to be upgraded on a very continuous basis. And, by the way, one of the things that we see is a high alignment between open-source tools effect. In fact, when you look at the development world today, the vast majority of the components and the applications and the languages and so forth that developers use tend to be open-source. And going forward we think that that's all going to continue to be more and more the case. So the open-source software really becomes the basis for much of the content that developers will use going forward.

And finally the last bullet point here, the notion of the PaaS layer. So the functionality in terms of how we're going to consume this, we are going to see the industry, over the longer term, move away from an Infrastructure as a Service model and move to more of a Platform as a Service model or quite probably a Container as a Service model. And the rationale for that is that as a developer you don't necessarily need to care about the server and that base level infrastructure software.

What you really want is somewhere to place applications and have that function be made available by basically having -- you can create it to almost like, I have the car but you're going to supply the roads and I will just use the roads, I don't need to own the roads. That's really what we're talking about when we talk about the PaaS layer where all that basic infrastructure is there for you to use and you don't necessarily have to create it yourself, and it's expected -- it's got a known format, it's got a known way of consumption, so the application can be deployed and just spun up and used.

Okay. With that, I'm going to switch over and start talking about containers. So this is a diagram, which works a little better if you build it, but the long and short of it is, what we're trying to illustrate is really the notion of what applications looks like before and what they look like today.

So on the left side of this chart, this notion of classic virtualization or classic virtualized deployment, essentially from the bottom up you've got the basic infrastructure, you've got the desk CPU, all the components that make up the server. And generally between the server and the operating system, that's where the hypervisor lives. In some cases, the hypervisor may be part of the operating system, but in many cases its actually below the operating system.

The applications themselves function as individual components that live on top of the operating system and in and of themselves, they're fairly dependent on that operating system and they don't necessarily -- they're depended upon what version the operating system is and what updates have been applied and so forth.

And where that becomes a problem is, they start to have the applications, they have different requirements you make under a scenario where one application may have different runtime needs than the next application. And that's where the notion of containers really starts to become very interesting.

Looking at the right side of the chart, so this notion of -- what we do is, we take the application and we take the things that the application is most depended upon. So, GLIBC [ph] or the DLL's that the application needs maybe very closely related to that application.

And what we do is, we essentially tie them together in a logical fashion and refer to that as a container. And then what happens is, beneath that stack of the application plus the applications' dependencies, there is this notion of container hosting engine and essentially that allows that -- those components to be logically reconstructed in a way that they were intended to be.

And below that is what remains of the operating system which will be common to all the applications. Generally that's much more of primarily the kernel, the operating system itself, whereas all of the add-ons to the kernel, meaning the runtime libraries and services that are added on, those are the things that actually get packaged up into the container itself.

So the notion of having a big heavy operating system goes away, and we start to have this notion of having a thin operating system and it can potentially have a hypervisor involved or might not have a hypervisor involved with it. The most important thing to think about is, each application is going to have its own container and the dependencies that individual application has tend to live with that application in that container.

So, I'm going to pause for a second and let Gary jump in and see if he has any comments he wants to add on to this particular slide.

Gary Chen {BIO 16178892 <GO>}

Yeah. I mean, I think, so containers are basically not really that new of a technology. It existed for many decades in various forms, in UNIX. And basically if you think of really containers as really an evolution of what the OS, the operating system, was originally designed to do, which is to keep things separate, right to the application one and application two, they should really be isolated from another and it's just like when your computer -- when you're running Chrome, you should be able to see what you're running, an outlook or some other program.

And so operating systems over the years have been working on that to make that better and better and better. And there certainly have been newer technologies in Linux and things like that to enable that. But that's kind of the underlying foundation for containers, that's been around for a very long time. So containers is just basically better isolation between things that you are already running on an operating system. And I think what's kind of different is that, when this stuff existed before it was pretty esoteric, I mean it was basically something that system administrator would try to do. It's like, oh, well, this application is really sensitive, I'm going to try and build a wall around it with the container.

And I think what Docker -- they really did to kick off kind of the next generation of containers is like -- you know what, let's take this kind of kernel technology that's like really kind of out there and not many people know about, and let's find a way to apply it to developers and that was sort of the magic. They really took it and turned into a packaging technology, integrated to develop a workflow that hated to use their shippings around, package things together. And so I think that's what kind of exploded in kind of this new era of containers that was different. But, yeah.

Al Gillen {BIO 3529173 <GO>}

Okay. Thanks, Gary. So for just another slide, I want to talk a little bit more about containers and virtualization. So just before I move off of the slide, one second we can move to the next slide.

So if you notice on the right chart, we're not showing a hypervisor in place, we've got a thinner OS that's sitting literally on the server physically. So moving to the next slide, we've taken that same diagram and we've expanded and we've split the hypervisor back into the mix, and this was one of the things I think has been somewhat misunderstood or maybe people jump to the conclusion fairly early that container technology was going to effectively replace virtualization.

There are some similarities in the benefits that you get from containerization and virtualization in that you can allow multiple applications to coexist on the same physical server without having conflicts. But these things are accomplished in very different ways, whereas a container scenario typically has a single -- a container scenario with a bare metal deployment has a single shared thin operating system, which supports all of the individual containers that are located on the system.

If you put a hypervisor down, you can still have multiple applications and, frankly, you can have multiple containerized applications with multiple thin OS's sitting above the hypervisor as this particular chart indicates. So in many sense, in many aspects, you can combine these two technologies and get benefits from both of them.

However, that's not to say that long-term there won't be some settling out of these two different technologies. And what we think will happen is, over the longer term, we will see container technology run on a bare metal scenario as being a more efficient and more cost-effective way to support containers. For customers that are willing to run in that type of multi-tenant environment, they're probably going to experience better performance and better operational costs or lower subscription fees from their hosting provider if they're doing this in the cloud.

What they don't necessarily get is, they don't get the isolation in a virtualization engine we give to them and that's a decision that some customers will make the deal that they want to do that or they don't want to do that.

You know, we've got some hybrid scenarios coming along, certainly where -- including now what Microsoft is doing, what Canonical are doing in particular and that to trying to build ways of essentially merging the best of hypervisor virtualization together with the best of container technology. And I think that the book has yet to be fully written on how this all sorts itself out. But suffice to say that I think there is going to be a lot of different options open to customers that want to do deployments. So, anything you want to add here, Gary?

Gary Chen {BIO 16178892 <GO>}

Yeah. I mean, I think a lot of this is driven kind of how the cognitive apps are being designed. So when the apps were large, I think virtualization brought in kind of a one-app to one-OS model, right. So there are some problems with operating systems, you know, kind of running multiple applications, so people started segmenting that up. And where the applications were large, that kind of largely -- but it was, okay, but when you think of microservices, app being decomposed now into tens or maybe hundreds, thousands of little pieces, putting one of those into each VM becomes kind of impractical, duplicating the OS, which is probably a lot larger than the application.

So, I think, containers is really kind of more about moving kind of back to kind of where we started, where people are using the OS to consolidate, where you're running multiple services, multiple apps on the OS. And yeah, it's just -- they really kind of work at different level, so depending on what you want to accomplish in terms of app architecture, in terms of security, you may choose to put container and VM boundaries at different points. And those boundaries are fairly different. I think the VM boundaries are still considered much more secure than the container boundary, but the container boundary is much more secure than an un-containerized boundary.

And I think the VM known as -- outside here as, people are definitely trying more efficient ways to combine hypervisors and the containers, but the evolution of how much containers will involve on bare metal, a lot of that may depend on how much container technology itself may develop. Can they eventually grow to match the isolation and security of a virtual machine and a lot of it will even depend on things like hardware.

Will Intel find some way to help that in hardware in the silicon, which really gave them big boost to virtualization when that happened, something similar there may happen. So there is a lot of, I think, things that could drive for that line-wise between virtualized or bare metal.

Mandeep Singh {BIO 15014535 <GO>}

Gary?

Al Gillen {BIO 3529173 <GO>}

So just briefly, I think we touched on lot of words in this particular slide already, but the notion of Containers as a Service and Platform as a Service. I mean, Container as a Service is effectively a Platform as a Service offering. We'll -- I think in the industry we're going to see people referring to CaaS or Container as a Service on an ongoing basis, and that's fine. I mean, the way to think about it is, Container as a Service is effectively a Platform as a Service offering.

So the notion of whether CaaS or PaaS, the benefits are largely the same in that you don't worry about the underlying infrastructure, somebody else manages that and makes that available to you. Obviously, if you're running at in-house to your IT deployment folks, your operations people are going to have to make sure they deploy that infrastructure, but the concept of running that in cloud is very much where you bring your application and everything else is just supplied for you. So you wind up focusing on those things, you worry about the life cycle of application, you monitor the application, you keep the application running efficiently, but you don't necessarily worry about the infrastructure under it.

So that notion of multi-tenancy, that's -- frankly that's one of the things you get as a benefit in terms of cloud economics are better and you get better density.

And as we just talked about in previous slide, this notion of density versus isolation, that's a trade-off. If you want to go with PaaS, you can get higher density and whether you go with a containerized solution or if you go with a pure PaaS solution, the level of isolation is different. If you want true -- truthful isolation, you wind up having the step away from PaaS and potentially looking at more of a virtualized and, frankly, you can probably get a virtualized Platform as a Service offering from some of the cloud vendors as well.

So -- and if I -- let me talk about containers and software development, specifically. So from a developers perspective, the notion of going to a containers is -- conceptually, it's not a new thought. When I was in school, back well into the last century, the notion of writing reusable code was something which was very much front of mind and we talked about it all the time. And as a practitioner after I left school we were striving to right reusable code, but was never very easy to do and many times we had to have some specialization of the code, which meant that it wasn't really reusable at all or we had multiple versions of the same thing.

Today, the industry overall is moving more and more to being able to truly embrace this notion of reusable code. And again, that containers are one of the things that makes it possible, simply because you've got a relatively small component of code with a known API access point and you can use it as a service. And, by the way, the cloud vendors are very much moving to deliver services, whether they are IoT services, Big Data services, management services, social data and analytic services, things like that are very much being moved the same model.

So going to a containerized developer model allows you to move very much to this notion of immutable -- infrastructure immutable deployment where you don't patch your update, you just simply replace the image when there is time to do it.

These things wind up getting stored in a more common resources, such as GitHub as a -- Docker Hub being a GitHub, if you will, of container images and for things that are publicly available, that's where you go and get them, frankly that notion of using public software is becoming more agreeable to a lot of organizations.

If they are able to get these things and make sure that they're using things that have known capabilities, it becomes really a way of speeding your way to bringing the application to the market. So when we think about the developer tools today, Docker being one of the most popular engines out there. We see a lot of the development tools are integrating/update Docker support directly into them.

And then finally container orchestration. As I mentioned, we are actually going to dive into this towards the end of the presentation. So I think I'm going to hold off diving too much on the orchestration piece at the moment. But suffice to say, for the moment, that this is an area in the market where we haven't necessarily seen the winners and the losers emerge here. We think that there's a certain amount of Darwinian Evolution that has to take place in the orchestration end of the business.

So, I'm going to turn it over to Gary. Let him lead on the next couple of slides.

Gary Chen {BIO 16178892 <GO>}

So we've seen that we kind of talked about some of the differences between containers and VMs. And ultimately I think, for the short term, we don't really see containers necessarily replacing VMs. Certainly, we're going to change the role of VMs, but we feel that -- I think the bigger transformations here is that -- and just like what happened with virtualization, when you brought in kind of a new underlying, kind of, computing paradigm, the rest of the stack above it changed and even under it as well, right.

I mean there are changes in the silicon in the way, kind of, how servers were configured, as well as all the management above it. So I think containers are bringing that same kind of change. There still going to be management functions and monitoring functions, but they're going to have a function in a very different way. What worked in the old world isn't going to work in the new world. So certainly a lot of players in that existing business are either going to adapt and you will see new people with new technologies kind of arise from another [ph] world.

And for container leadership, I think, right now it's -- unlike virtualization, where there was never really an open common standard, right. Everyone, there are several different hypervisors, different formats, source formats were necessarily open. And so, VMs ultimately, kind of, weren't really portable in some instances, right, you can't go from VMR [ph] to Amazon AWS, because these are different hypervisor.

Fortunately, in kind of a new world that open-source has really become standard. So, the first step is called the de facto standard and now that's moving to something called the open (inaudible) initiative, which is kind of -- will give the industry a standard container format and a run time.

And that will bring a lot more portability, but what's happening is that, since that base format is essentially kind of commoditized and available to anyone, most of the battles really being fought at this moment, at the kind of that clustering orchestration level, that's really been kind of Kubernetes, the Docker's format and Mesos.

And I think as over time goes on, that's going to move to the next higher level into management, into PaaS and things like that. So it's going to evolve very quickly, it's moving up -- higher up the stack in order to differentiate in the commercial market.

And so, when we look at container enterprise adoption today, it's drawing huge interest. There is no doubt that there's a lot of hype and there is a lot of interest, but when you're actually talking about production usage to data enterprise, it's extremely small. Most of it -- of the users, the early adopters, they're really using a lot for Text Tab and software development and a very, very small in development -- in production.

So, it's the experimental, it's largely unmanaged. They're not buying into that clustering orchestration solution today, but that's kind of around the corner as in the next step. And lot of the applications are not kind of net new cloud native applications, certainly a lot of that is happening, but in enterprise they always come with a lot of baggage and lot of legacy stuff. And so we do see a lot of lift and shift. So I think there is going to be a market around that. Certain people will be driving that and selling -- and working on those kind of solutions for people to find new technology to kind of existing thing.

Of course, the public cloud is kind of a huge part of containers as well. But anything you talk about cloud, there is a lot of containers that just start off in the public cloud and don't settle off on-prem. And I think one of things is that, with containers, I think with virtualization there was new and everyone was kind of following that evolution.

I think with containers and cloud native applications, a lot of that was -- it's already been proven to work by these web scale pioneers like Google and Facebook and Amazon, and these others that are using these types of technologies. So, a lot of kind of the things have already been solved, right, and those guys have solved that at scale, this sort of methodology and tools to use. And now the enterprises are starting to just try to replicate that. But it's not sort of like they are forging into the unknown. Someone else has already done that for them, and now it's just like, can we do it in our environment and in our scale.

So when you look at kind of the -- some of the observations, so we've seen that, as I mentioned, some of those web scales have been using them for years and that's really where it's kind of already been proven. And we see that today most of the enterprise containers are not managed and we really see it less than 10% are using any kind of orchestration, whether it's paid or unpaid to manage their containers. But we think that's really going to grow, so when you look kind of four years from now, we really think that, at least, half of containers are really going to use commercially supported orchestration technology and should be (inaudible) another percentage that is unsupported kind of open-source unsupported.

So ultimately we expect that those Kubernetic solutions are going to be orchestrating all of these containers and those solutions will expand to do more functions over time and vendors will overlay their own functionalities to fill in some of those white spaces and development.

I'm going to skip this for now. I think we talk a lot about our containers versus VMs. But I think, we said that the containers when -- I think the important part is, containers won't necessarily replace VMs but they're certainly going to have a dramatic impact on the virtualization market. So just because it doesn't replace the hypervisor, it doesn't mean that the hypervisor doesn't become more commoditized and the container take over a lot of the functions that VMs already took. So a lot of people today use VMs to manage their applications as essentially a packaging technology. So when that shifts over to containers, then the VM really just becomes kind of the plumbing underneath, it doesn't really do anything with -- in terms of application management. So that's certainly going to affect how people perceive the value of what a hypervisor does, and so -- there are certainly some splitting of functions that can happen there.

So these are some key vendors to discuss. Al, I don't know if you wanted to go through any of these or just open it up to questions. But we just put a list of some of the key vendors in the space, there is certainly a lot more when we talk about the ecosystem, even the storage, and I think there is even more than that, but these are kind of key players in the space of kind of core container infrastructure or virtualization, public cloud as well.

Al Gillen {BIO 3529173 <GO>}

Yeah. I think that we could probably spend an hour on the slide alone, but suffice to say that since we've got three tiers of vendors here, we've got vendors that are involved with the basic technology it allows you to package and orchestrate the applications of the company. Companies like CoreOS, products and technologies like Kubernetes and Docker, you know they are at that level of the market.

And Red Hat, of course, plays in that space as well, but Red Hat plays at a much broader level and only supply the orchestration and packaging solutions, which are typically based on Docker and Kubernetes technologies anyhow. But they also supply the virtualization and the other infrastructure that goes around.

Companies like VMware, of course, have a hypervisor solution, but also are trying to be relevant in the container space and have their own container initiatives. They've got a couple of One-Net that allows containers that run natively on VMware's existing virtualized infrastructure and then they've got a container native solution.

Then the third tier are companies like Microsoft and Amazon and Google, of course, which are going to be the hosters for containers and all these service providers are moving very quickly to support Containers as a Service. So, I think -- anything you want to add there Gary, but otherwise we can turn it over to Mandeep to start fielding some of the questions.

Gary Chen {BIO 16178892 <GO>}

Yeah. Thanks. Yeah. We've [ph] got questions. I think we've taken enough time with the slide, so let's hear what people want to talk about.

Mandeep Singh {BIO 15014535 <GO>}

Great. Thank you, Al and Gary, that was a terrific overview on containers. And just a reminder for everyone, you can send in your questions, there is a Q&A tab, you can type in your questions there and send it to us and we can take it up in real-time.

I'll start it off. And basically I'm very clear on what the value proposition of containers is and why we need to care about it. I'm just wondering at what stage are we in terms of monetization? So when I look at what VMware did with vSphere and how quickly it was able to monetize it, is this something on the horizon or it's going to take a life before we can talk about monetization of containers technology. (Multiple Speakers), Gary. Yeah. Go ahead.

Al Gillen {BIO 3529173 <GO>}

In terms of monetization, I think, it's going to be very different from what happened with VMware. So VMware was able to monetize the core hypervisor very strongly and they still continue to do that today. And eventually they started moving up the stack in selling more management tools.

But there is a ton of money to make just in the hypervisor itself and I think with Docker, that's not really the case. There is -- most people today don't buy a container engine just by itself and it's not something, I think, that's going to generate anywhere near the kind of money that something like vSphere or ESX did. And the reason for that is that, it's not really kind of a stand-alone thing, right.

Most people today get the container engine through their OS vendor, because it's essentially a feature of the operating system. So the operating system is already a product that they buy, so it's an extra feature. So in some cases they will just get it for free and becomes a free feature with the operating system. It's not sort of a free, because they have to pay for that operating system, so the operating system is paid. So that money is already there, but incrementally it's not going to add a lot.

So in some cases, there are some stand-alone container engines and OS's that you can buy, but I think for the most part you are either getting it from the operating system, you already have possibly a small incremental to add that on or you're getting it from a hypervisor vendor or you're getting it from a public cloud service and you're not even buying the software. So I think by itself I think that's going to be related [ph] different. Not to say that there isn't money to be made in monetizing containers.

But I think from the get-go, it's going to be monetizing a much larger stack, more of a mature virtualization stack that you see being sold today with a stack of management and all these other infrastructure functions and I think containers are headed there kind of immediately. There is not going to be a period where people are -- containers, I think, the core engine by itself.

Mandeep Singh {BIO 15014535 <GO>}

Okay. And I want to piggyback on the comments you made around the public cloud vendors, that a fact that they are supporting Containers as a Service and they are trying to embed containers as a technology in their infrastructure. Now Amazon, in particular, offers containers, but it also has this vendor service which it offers to basically run smaller workloads, temporary workloads, what's the difference? And are these competing technologies?

Al Gillen {BIO 3529173 <GO>}

So, I mean, obviously the full story is yet to be written, because these technologies are all evolving, but the way I think of it is, Lambda is intended to be essentially code on demand. So you have an application, you stand it up using Lambda and some of them comes along and causes that code to go into play and be used. And presumably that can be packaged in the container or potentially not packaged in the container.

But when you think about container specifically, the notion of, if I want to stand on my container and have it be available for me and for my customers on an ongoing basis, you may not necessarily want to use Lambda, you may want to use just a standard Container as a Service. I think, you'll see a lot of -- I think, you'll see a lot of mixing between those two approaches and who knows over time maybe something that using a Lambda-like approach is the better way to do it long term. But in either case, you're looking at a container or an application runtime environment, which is delivered as a service.

Yeah. I think Lambda is basically just one type of container architecture or container deployment model, but it's only good for certain things, right. So it's not going to be something that, I think, it's going to be universally applicable. So it just -- I think the way to think of it, it's just one form or one possible way that you could use containers and it's certainly something kind of newer, but it's not going to be something that takes over an entire world, it'd be only model.

And I think with Amazon and other players, I think, they have a variety of services that they can offer. They're going to offer more Platform as a Service that will be underlined with the containers, that will be more -- kind of, infrastructure-oriented where you can play around with clusters and load balancers and things like that, as well as Lambda. So I think they're coming out, kind of, the fullest, longest word of different options for people.

Mandeep Singh {BIO 15014535 <GO>}

That's very helpful.

Al Gillen {BIO 3529173 <GO>}

Just -- you'll hear the notion of server list computing and that is essentially Platform as a Service where you don't worry about the server or the server infrastructure software, you worry about the application code.

Mandeep Singh {BIO 15014535 <GO>}

Terrific. We have audience question here. What are the cyber warfare challenges with the containers technology? Will it make it easier for attackers to use the applications not normally housed in VMware?

Al Gillen {BIO 3529173 <GO>}

Yeah. It's a really interesting question. So I mean, I guess the security challenges around containers are kind of at a whole bunch of different levels. I mean, one, it's just kind of the isolation itself, right. So we talked about kind of differences between a VM boundary and a container boundary. So I think it's accepted that virtualization was very, very close to physical separation, right, having two physical servers.

It was very, very difficult for one person to break into a VM, it either compromise a hypervisor or compromise a VM. So with containers, it's easier, right. So today we do not see tenants being separated by only a container boundary, so there is always a VM boundary. So if you look at any of the major public clouds, Google, Amazon or AGORA, when they stand up a container for a customer number one, that is in a separate VM the customer download too. They never co-host among the same OS kernel and we're on the same OS and I'm in container one and you're in container two, that would never happen.

So they are not really comfortable with that model yet, things may change, that boundary could become stronger, but it's quite dangerous right now and I think a lot of it will just be really how aware people are of there and how they architect their applications. You just have to put certain boundaries in certain places.

So that is one potential thing that is evolving. So yes, I think in an all container world, yeah, it's easier to break into another container. It will be easier to compromise the host, but that could be somewhat mitigated by putting VM boundaries around some of those things.

But there is other implications as well. I think, the way containers are built often using -- it's kind of the Docker ways to use a repository to start off with an image. So those public images that are available, public repos and people are pulling in code from somewhere to start off their application, getting a server here and there, getting a framework here and there, and I think kind of a -- a lot of challenge will be to make sure those things are clean before they get injected into an application and into your enterprise.

So a lot of it is around where did this code come from? Who made it? Who pulled it? Were there any changes? Is it signed? So a lot of those things too is like, you don't want people to insert kind of bad things and try to poison the whole bunch of people that may be pulling from it.

Gary Chen {BIO 16178892 <GO>}

Just one thing I'll add. Just remember that with containers that notion of having them be relatively mutable, meaning you don't update them, you can't essentially spin up brand new instances from a gold master each day or each hour to protect the integrity of a particular application. So that if you feel it, is that at risk of being compromised, you can just simply replace the code with fresh code as known to be secure. But that doesn't stop it from being compromised later.

Mandeep Singh {BIO 15014535 <GO>}

Great. I have another question on cloud. So, obviously most of the new workloads are being deployed on cloud and cloud infrastructure is likely to grow much faster than on-premise. Does container technology help IT guys and you know, moving their workloads easily from one public cloud vendor to another cloud vendor. And is it just make your infrastructure agnostic of -- where you're deploying it?

Gary Chen {BIO 16178892 <GO>}

Yeah, I think I mean -- I think from a core infrastructure level yeah, that's certainly a promise. And then that's something that never happened with VMs right? So and I think people found that VMs weren't quite as portable as -- if they're in different environments, yeah, the convert and things like that, so I think containers -- yeah I mean, they didn't make it much of portable. I mean, there are certain portability things that it can't solve, right? I mean, you can't move a Linux application to Windows or anything like that, you have to assume that it's still the Linux, a similar Linux kernel.

But yeah, I think a lot of people that we talk to, at least from infrastructure point of view, they see that as a compatibility layer, that they input [ph], so as well if I just use containers, at least, at some level, I can move these things very cynically, but there are still a lot of challenges, I don't think that at one level, yes, containers will make a fundamental unit of compute [ph] portable, but if your application has dependencies on some cloud service or cloud API that only resides in location A and it's not in location B, then you're going to have other problems besides just kind of infrastructure portability movements.

If you have -- I don't know if you have something there, (inaudible).

Mandeep Singh {BIO 15014535 <GO>}

I guess it will be --

Al Gillen {BIO 3529173 <GO>}

Yeah, if you consume cloud services as part of your infrastructure, there may be a preference for one cloud or another based on better or lower latency by being in Microsoft's cloud or Amazon's cloud, but that's specific to things that are external to containers you build.

Mandeep Singh {BIO 15014535 <GO>}

And a couple of things that come to our mind even Microsoft, it's supporting Red Hat technology, now they are supporting containers and I guess it just goes to show that they want, I guess companies to have the options to move their workloads as well from Windows to Red Hat and yeah, so that's I guess more and more we are seeing portability is a big criteria when it comes to deploying your workloads.

Questions And Answers

A - Mandeep Singh {BIO 15014535 <GO>}

Okay, moving on and I think we have time for a couple of questions, so if there are any questions from the audience, I would encourage you to use the Q&A tab and send us your questions.

There is one more question that I had so obviously Dockers has taken a lead in terms of just developing this technology and they are using this open-source model for their -- it's collaborative, it's using the combined power of all the developers out there. Do you think that is the future of containers or do you think some -- a large vendor is going to take a lead and really wrap their own services around it and kind of try to monetize it?

A - Al Gillen {BIO 3529173 <GO>}

I'm of the opinion that technologies that the Docker is offering are largely going to be consumed and packaged as a service by Amazon and Microsoft and already it's being done by Red Hat and so forth.

In some respects, I mean, there will be some benefits presumably associated with using the Docker branded products in some types of deployments. But in many cases, it's going to be just a generic service, it's going to be incorporated into the base infrastructure and maybe you feel differently, Gary?

A - Gary Chen {BIO 16178892 <GO>}

Yeah, no, I mean, I think that -- I think when you look at Docker, the company, it's -- so this Docker, the community and the project and there is Docker, the company. So Docker actually makes a very large stack of software. Now the thing that everyone uses is the Docker engine in that format. So that's been pretty universal, but Docker had its own suite of orchestration and management tools and they compete head-to-head with Kubernetes and Mesos and a variety of others. So yeah, I think when you look at open source and I think you can't really put open source into just one big bucket.

I mean, there are very different levels of open source and just having the source code open is one thing, but having an open development process and community is something else. So I think we're going to see different types of open source that are out there, some are that are more open than others, some of them maybe open core with some proprietary ads on and some of them are -- will be very tightly controlled in terms of who controls the development.

But yeah, I think basically that people will take the core Docker engine. They may take higher parts of the Docker stack, what they call Docker data center and add things on to that or modify it or compete with their own completely separate stacks with Docker. So yeah, it's going to be quite a varied market with lots of different pieces and Docker is both the kind of driver of some of the core, very foundation plumbing parts of that, but they're also completing a lot with those people in the community.

A - Mandeep Singh {BIO 15014535 <GO>}

Okay. We have one more audience question. They're asking container appears to be good for portability and reuse. What's the cost of containers, especially in terms of latency or any other costs?

A - Al Gillen {BIO 3529173 <GO>}

So containers --

A - Gary Chen {BIO 16178892 <GO>}

So just --

A - Al Gillen {BIO 3529173 <GO>}

Oh, yeah, I was just going to say that, typically containers are able to be started very, very quickly in terms of small number of milliseconds. If you start up a virtualized infrastructure with -- using a VM, that can take 20, 30 seconds or even longer depending on when it has to be loaded. So in some respects, I think that the container technology, just the way it's constructed and the way it's used, can deliver lower latency for startup.

Now in terms of the actual functional latency once it's running, there is arguably a thinner less complex infrastructure stack between the containerized application and the physical server underneath it. I don't know that there is that much of a difference between a -- virtualization may put a couple of percent of overhead in that mix, but I don't know if that makes that much of a difference in the grand scheme of things. So I don't know, Gary what your thoughts are there.

A - Gary Chen {BIO 16178892 <GO>}

Yeah, I mean, I guess that's sort of a tricky question. So when you talk about latency, I mean -- so we're talking about the cost of energy, really talking about the performance cost. So it really can depend whether the container is for -- on a hypervisor or not. So when you look at bare metal container, the cost of the container is really kind of in the startups, so -- right?

So you are -- it's going to start up slightly slower than an uncontainerized process. It's still pretty small, I mean, we're talking in the order of milliseconds. So depending on really what your start-up requirement times, that could be a lot or a little. Once it's actually running, there's really no latency there. Most things that -- I think most of you will consider once the container is running, it's essentially you get bare metal performance if the house is on bare metal.

Now when your virtualization is different, when you compare it against a virtualized application, yeah, there's going to be a lot more latency there. So containers improve on latency versus a virtual -- something in a VM. So people that didn't virtualize things very high-performance applications, because of that performance overhead are very much interested in containers because that reduces it all the way and still bring some of the -- those similar benefits. So yeah, I guess, I mean there's a lot of kind of angles to that, but I think for the most parts, I think if you're comparing to virtualization, containers are much more efficient, if you're kind of (inaudible) this bare metal adds a little bit just on the startup, but not an excuse.

A - Mandeep Singh {BIO 15014535 <GO>}

Okay and we'll end with this question. So is there any connection between containers technology and the trend of mobility? Is it more suited for an app based environment?

A - Al Gillen {BIO 3529173 <GO>}

And then, do we know if the context of the question is with regards to mobility? So is that mobility of the allocation?

A - Mandeep Singh {BIO 15014535 <GO>}

Yeah, so I mean I think the idea is that, we are going to have more and more functionality wrapped in apps, which are delivered on mobile devices. Is there any connection over there that you think that kind of an architecture is going to facilitate the adoption of containers or it's kind of agnostic of that?

A - Al Gillen {BIO 3529173 <GO>}

I don't think mobile devices, at least the mobile end of the application, I don't think is much talk of containerizing that part of the code. The stuff in the back-end, it supports the mobile apps, yeah, that's going to be containerized.

A - Gary Chen {BIO 16178892 <GO>}

Yeah, I mean, I think the mobile back-ends are a perfect candidate, right? I mean a lot of them are kind of looking at, kind of these cloud native use cases of online and digital. I think a lot of the way that mobile apps interact with the back-end is very smoother for containers, it's a very small kind of micro interactions rather than some large monolithic app or client that you may have on a PC. So yeah, I think containers are great and -- I think when you talk about mobility too, I mean, a lot that brings in the scalability, right?

So a lot of -- kind of in the mobile world, it's about how fast you grow or I mean, you have these mobile apps that take off or you're using them for events. So there is a lot of -- kind of -- you have to deal with click storms or your app gets super populace, and when you talk about the scalability or agility that has a lot to do with it.

And mobile apps generally move faster too. I mean, apps get updated all the time, they really kind of follow that kind of a CICD kind of methodology. So now containers as the back-end is great to support that when you have an app that's rapidly changing and new versions are coming out often on a daily basis or something like that.

A - Mandeep Singh {BIO 15014535 <GO>}

Great. That's all the time we have. Thank you so much, Al and Gary. I really enjoyed the discussion. And yeah, let us know if any of you have any follow-up questions, they will be happy to take those. Thank you for joining us.