White Paper

Modernizing Applications with Containers in the Public Cloud

Sponsored by: Amazon Web Services

Gary Chen
June 2019

IDC OPINION

Containers are the heir apparent next generation of compute. However, containers don't just bring a new style of infrastructure. Containers are intimately tied to a transformation in application architecture, software development methodology, and operational principles. Ultimately, the combination of these changes enables the faster delivery of software and more modern, scalable, and agile applications. IDC forecasts a five-year CAGR of 79% for enterprise container instances, with over 1.8 billion enterprise containers by 2021.

While web hyperscalers have built containerized applications for many years and have proven the concepts behind a new generation of infrastructure and applications, enterprises face a different situation as most aren't building greenfield and are juggling older generations of applications. While containers can still be used to containerize traditional applications with benefits, larger returns come from refactoring the application over time. The majority of enterprise applications lifted and shifted into a container will be eventually refactored. Enterprises are beginning a transition phase where they are learning new container skills, methodologies, and processes as they begin to both build new cloud-native applications and refactor and modernize existing applications.

The public cloud has played a major role in the development of containers. Container technology and the resulting transformational effects were essentially born in the cloud. Cloud providers developed much of this technology and the early adoption of containers in the enterprise was predominately in the public cloud. Containers in the public cloud were a natural fit because these developers were targeting cutting-edge cloud-native applications. Public clouds provided scalable infrastructure as well as application and data services that these containerized applications consumed. Today, the growth of containers in the public cloud continues to grow. Core container orchestration services are already a given, and the focus today is on building a full container application platform, integrating elements such as persistent storage, networking, service mesh, and continuous integration/continuous deployment (CI/CD) and monitoring and extending to new models like serverless computing. By 2021, nearly 60% of all enterprise containers will run in the public cloud.

SITUATION OVERVIEW

Containers are a type of operating system (OS)–level virtualization that provides an isolated, resource-controlled environment to run applications. A container is basically a type of sandbox around a normal application OS process and is generally considered to be much more isolated than an uncontainerized process, but not as strong as a virtual machine (VM). Container images define how applications are packaged and only contain the application and its dependencies such as libraries, configurations, runtimes, and tools, making a container more lightweight than a VM. The container image and runtime are standardized through the Open Container Initiative (OCI), which makes containers highly portable and universal.

Containers are particularly relevant when considering cloud-native applications. Cloud native is defined as a new approach to building and running applications that takes advantage of modern cloud computing. This isn't exclusive to public clouds, cloud-native applications can also be run on on-premises private clouds. Cloud-native approaches can use several techniques, with the following being the most common:

- One technique is **microservices architecture.** Instead of a monolithic architecture, microservices break down an app into multiple logical services. Each service can be developed independently, leading to greater developer efficiency by allowing more parallel development — allowing teams to maintain independence — while at the same time allowing the services to work together reliably with stable APIs in spite of fast-changing code. Running microservices is also much improved, as each service can be deployed, upgraded, and scaled independently.
- **DevOps** brings software developers and IT operations in closer collaboration to release software faster. DevOps engineers are tasked with creating services and process automation to allow developers to focus on creating code and then getting that code into testing and deployment as quickly as possible in an automated way without additional human overhead.
- **Continuous integration/continuous deployment** is a common DevOps technique where a software build and deployment pipeline automate the release of software updates, constantly and in small changes.

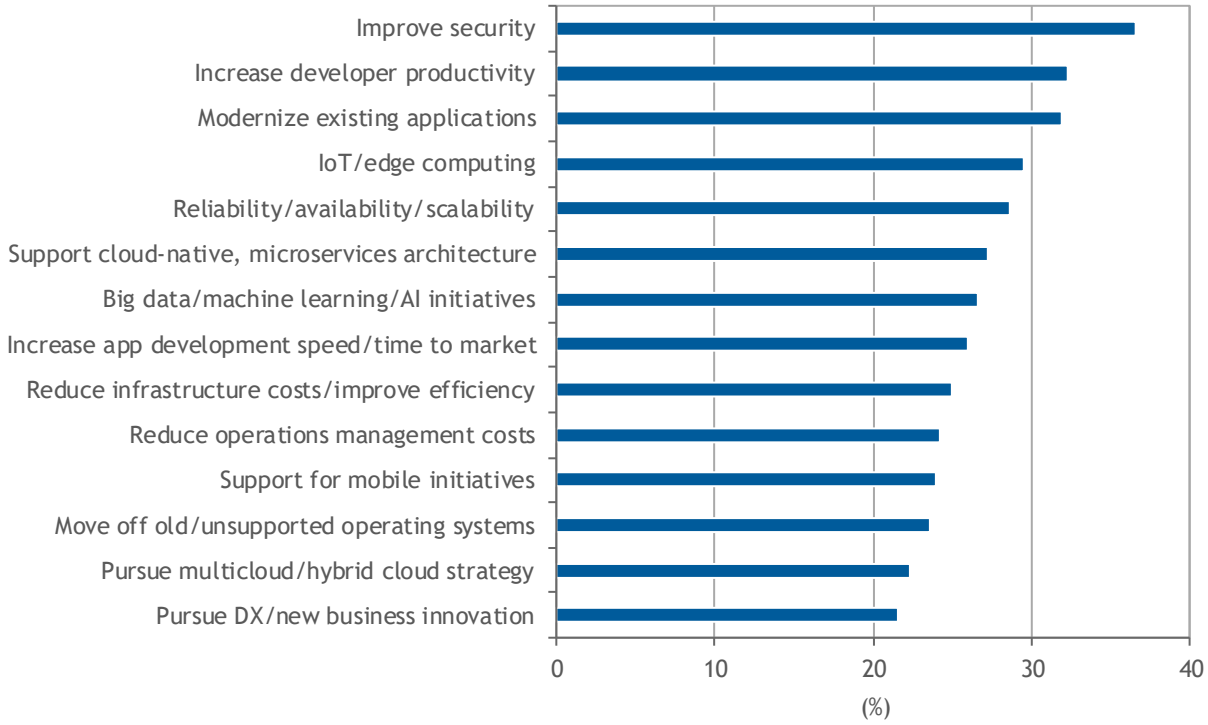
Containers, being highly efficient at packaging applications, portable, and fast start-up and teardown, became a perfect fit to encapsulate these new types of applications. Coupled with advanced automation, the agility of containers helps automate operations that typically have been done manually or required human intervention such as optimized provisioning, auto scaling, and fault recovery. Container orchestration systems are also able to automate more complex patterns such as A/B testing and rolling upgrades of new software versions.

Why Are Enterprises Adopting Containers?

Enterprises are adopting containers for a number of key reasons (see Figure 1).

FIGURE 1

Top Drivers for Containerization



Source: IDC, 2019

Accelerate Software Development

- Modern applications are shifting to cloud-native and microservices architectures, and containers are a perfect way to encapsulate these pieces into portable units.
- Development methodologies are changing, with developers shifting to agile methods and DevOps and leveraging continuous integration/continuous deployment systems. Containers, being lightweight and portable, make great vessels to wrap code with in order to push them through these new software pipelines.

Modernize Applications

Containers can also be used for existing applications without any refactoring. Use of containers in this way allows the application to be more portable for migration to the public cloud and can also retrofit the application into newer developer workflows and software pipelines. However, refactoring the application will bring further benefits, encouraging customers to first containerize, then refactor parts of the application or build modern extensions to the application over time.

Automate Operations at Web Scale

Container orchestration and management systems are designed to deploy and manage distributed, cloud-native applications. To do this at any scale, full automation is a core mantra of these systems. These systems are also designed to operate at web scale, providing scalability and resiliency capabilities to modern applications. This entails monitoring specific performance metrics and health checks and automatically scaling or reacting to failure when thresholds are crossed.

Areas That Containers Do Not Address

Containers are catching on very quickly in the industry because it has applications to many of the current industry trends. But as a hot technology, containers often get overhyped and get spun as a cure-all for a variety of problems, and a number of myths and misconceptions can be commonly heard today.

The following are some areas that containers do not address:

- **OS and kernel portability:** Remember that a running container is just a sandboxed process. That process or application is still bound by the normal rules of operating system/application limitations. For example, containers don't let a Linux app run on Windows and vice versa. A Linux app inside a container is still a Linux app and still needs a Linux kernel to run. A container may make a Linux app more portable across different Linux distributions, provided that the underlying container host provides a suitable kernel.

  The OS kernel on a container host is shared across all the containers running on it and the kernel version still matters to some applications, so portability is still bound by those requirements if changing Linux distributions or versions.
- **Infrastructure:** Containers are an operating system–level technology, and while they do have major impacts on how compute, storage, and networking work, there is still a layer of real infrastructure underneath that containers do not manage or provision. Underneath containers still exists a physical infrastructure and usually a software-defined compute, storage, or networking virtualization layer as well. Containers can provide an abstraction layer on top of infrastructure, but containers are not a solution for poorly managed or poorly architected infrastructure.

Containers and Virtualization

One common question many enterprise customers have is whether containers replace virtualization. For the most part, containers aren't replacing virtualization today because containers and virtualization do different things and work together very well. Virtualization operates at the hardware level, virtualizing server hardware and carving it up into smaller pieces. Containers operate at the operating system level and application packaging level.

As a result, virtualization is still very useful for functions such as:

- **Secure separation.** The boundary between containers, while very good compared with an uncontainerized process, is not as secure as a VM. In scenarios where strict separation is needed, VMs are used today — for instance, to separate multiple tenants in a public cloud.
- **More flexible OS and kernel choices.** If a container host OS was installed on bare metal, all containers would share that same kernel. Using VMs to run multiple container host OSs allows mixing of OSs. This could be a mix of Windows and Linux, different Linux distributions, different Windows versions, and different patch levels that might be needed by applications.

- **Hardware provisioning.** Remember that containers don't manage the underlying infrastructure; so that still needs to be managed and provisioned by something. While some bare metal provisioning tools exist, the IT industry has spent the past 15 years building virtual infrastructure tools, which are much more widespread, flexible, and agile. Virtualization is still the best way to carve up a physical server in a secure and granular way, to be used for multiple tenants and different operating systems.

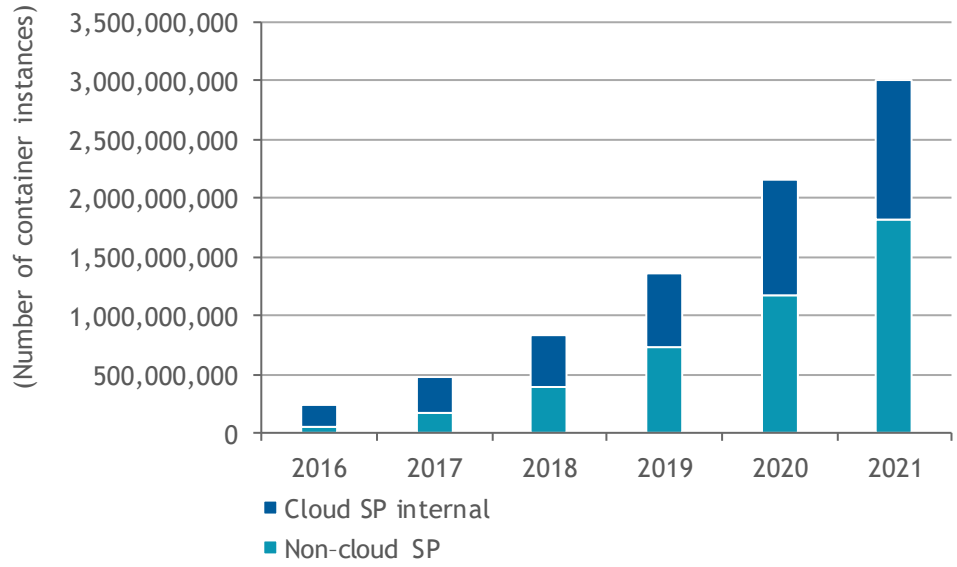
However, just because containers don't necessarily replace virtualization doesn't mean that hypervisors aren't evolving to accommodate containerization use cases. See the "Optimizing VMs for Containers" section for more details on microVM technology.

TRENDS IN ENTERPRISE CONTAINER ADOPTION

IDC forecasts that by 2021, there will be approximately 3 billion containers deployed, more than six times growth from 2017 (see Figure 2). Some of this growth is from hyperscale web services providers that have long been using containers in their own datacenters to host their applications. But the fastest-growing segment is enterprises that are adopting containers both on-premises and in the public cloud, accounting for 1.8 billion containers, which is about 60% of the total containers in 2021. As the industry transitions to cloud-native applications, containers are serving as the next-generation compute primitive and package for those applications.

FIGURE 2

Worldwide Container Instances Installed Base, 2016-2021



Source: IDC, 2019

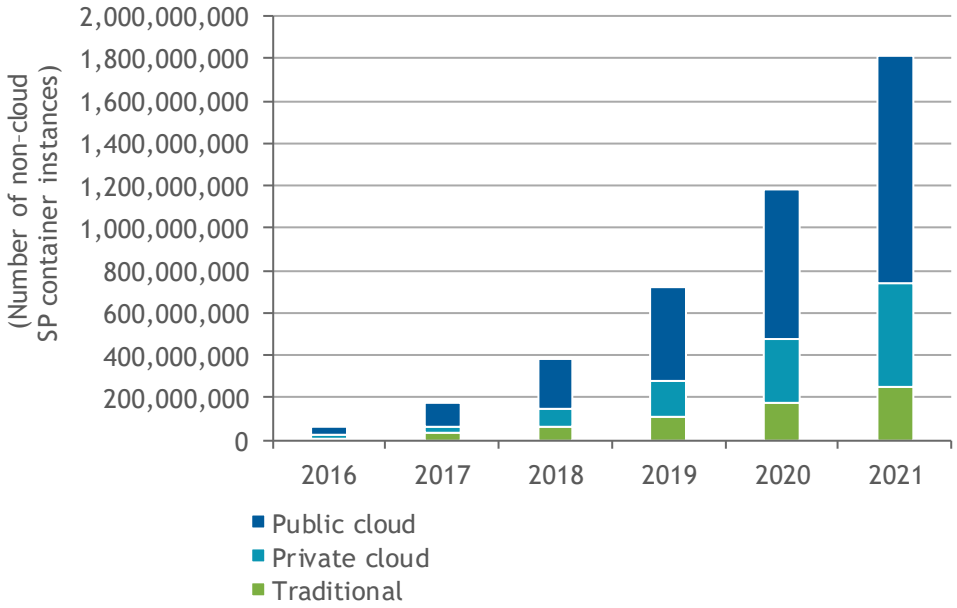
Where Are Containers Deployed?

Containers initially began to be deployed on the public cloud, which is no surprise given the affinity containers have for cloud-native application development. As containers have grown in popularity and beyond just cloud-native applications, we are now seeing container infrastructure buildout in

on-premises environments. By 2021, containers on public cloud will make up nearly 60% of total container deployments (see Figure 3). With the standardization of containers with OCI and the wide adoption of Kubernetes, container infrastructure is far more standardized than VMs are, opening up a range of hybrid cloud use cases for containers. Containerizing an application will also make it easier to migrate that app to the cloud in the future.

FIGURE 3

Worldwide Non-Cloud SP Container Instances Installed Base by Deployment Model, 2016-2021



Source: IDC, 2019

Containers in Production

Containers have moved into production faster than any other recent compute technology. For example, it took nearly a decade from when server virtualization was introduced until the industry saw a significant number of production applications deployed in VMs. Server virtualization took time to work out stability and performance issues. With containers, 85% of container deployers today are already using containers for production workloads and 75% are even using it for tier 1 applications – according to a recent IDC container customer survey. Containers are particularly prevalent in DevOps teams, with 95% of DevOps users already running some number of applications in containers, though most footprints are still small today. So why have containers become stable and trusted so quickly? The key reasons are:

- **The core Linux kernel functions that containers use for execution are actually not new but quite mature.** These functions are also used for other functionality outside of containers. So from a fundamental execution standpoint, containers are using very stable and mature technology.
- **With containers, there was never any performance hump to get over as with server virtualization.** Server virtualization has to emulate server hardware and there is overhead with that. Containers work completely differently and there is no inherent performance overhead with containers.

- **While containers are new to the enterprise, they aren't new to hyperscale web companies.** These companies did a lot of the pioneering work in developing the previously mentioned Linux kernel functions and also perfecting the architecture, methodology, and processes behind container infrastructure and containerized software development.
- **Most of the key container technologies are open source projects with broad industry participation and high development velocity.** This allows software to be developed much faster than any single closed source company could do.

What's Being Containerized?

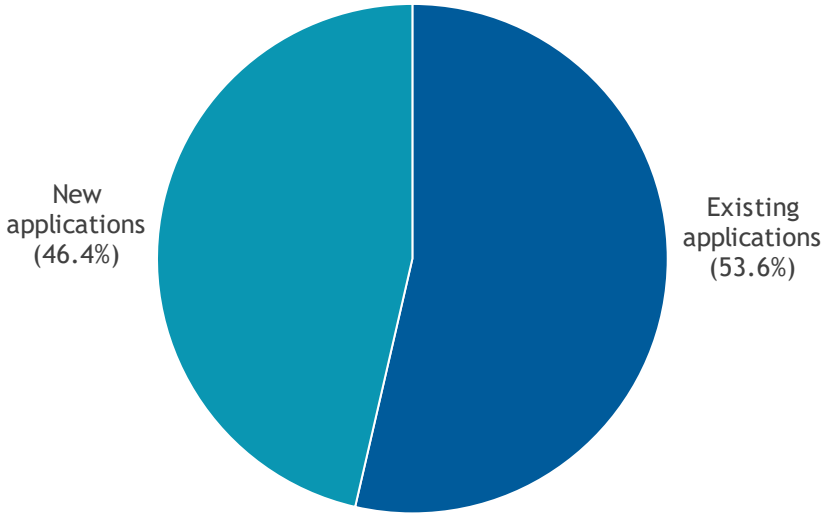
With most enterprises, juggling multiple generations of technology and legacy assets is standard. Thus for new technology to be deployed broadly in enterprises, it has to be applicable to existing assets and not just greenfield buildouts. While many enterprises are beginning cloud-native software development, this is still a work in progress in most organizations, and it will take time to make this transition. In IDC's latest container customer study, 54% of containers are running existing applications (see Figure 4). However, containers aren't transparently compatible with existing applications like VMs were. While 42% of existing apps that were containerized required little to no modifications to containerize, 25% required significant modifications. One of the major use cases for containers in the enterprise is to modernize applications, and users reported that 58% of their existing apps that were containerized will undergo further refactoring over time.

Even without refactoring, containers can bring benefits to existing applications. Setting up complex multipart applications with containers is much faster and easier, and this can greatly assist developer productivity on laptops as well as in automated testing environments. Containers also allow developers and operators more control over the environment (library versions, etc.), also leading to increased code quality and fewer issues with moving apps from desktop to production. Implementing immutable infrastructure principles can also assist with faster and more automated testing and patching of applications. Even with a tier 3 application, container orchestrators can offer more agile, reactive, and automated scaling and availability than VM counterparts, a benefit to apps that are more dynamic.

However, a great level of benefit is attained when the application is refactored into smaller, loosely coupled parts, allowing a higher degree of parallel development and agile operations. This doesn't have to necessarily be a full microservices implementation for users to see returns. Most customers initially only make changes to the app during the initial containerization that are absolutely necessary, typically the changes that are needed to solve compatibility issues with containers. Over time, often years, the application will then be slowly refactored. Monoliths can be hard to break apart, and users don't generally attack the entire code base. A common approach is to identify the high-change, high-dependency portions of the app and focus on those areas for maximum return, leaving the more static parts of the monolith alone. Extensions and new functionality to the app are generally created as a separate, linked service as much as they can, rather than adding it to the monolith.

FIGURE 4

Workload Types in Containers



Source: IDC, 2019

The Rise of Kubernetes

Much of the attention today in container technology is around Kubernetes, a massive open source project that is governed by the Cloud Native Computing Foundation (CNCF). Kubernetes is the most widely used and supported container orchestrator, with nearly every vendor offering a Kubernetes-based product or service. The CNCF also maintains a Kubernetes conformance test, to ensure that Kubernetes products that pass this certification will interoperate with each other. This brings another level of standardization to the container stack, which is a benefit to end users — who will reap portability and interoperability benefits that VMs were never able to establish. Kubernetes already accounts for the majority of container orchestration usage, with 87% of on-premises and 82% of public cloud container deployments using it. However, there are several key facts to keep in mind about Kubernetes:

- Kubernetes is the center of a container management system but doesn't address every area of container infrastructure or container management, focusing primarily on scheduling and orchestration. To build a full container platform or solution, many other components besides Kubernetes are required.
- Kubernetes is a complex technology and it is not trivial to deploy, manage, and make highly available. Skills will be needed to manage it and a learning curve should be expected.
- Kubernetes focuses on managing containers, and it still requires users to provide and manage the infrastructure underneath. The quality of this infrastructure will affect the quality of the Kubernetes and containers running on top of it.
- Kubernetes is focused on core container infrastructure management and doesn't address the software build process or application architecture. Also keep in mind that most customers find changing their organizational structure, processes, skills, and attitudes is even more challenging than deploying a piece of software like Kubernetes.

- Other container orchestration systems besides Kubernetes still have their place. These orchestrators often target specific uses cases that Kubernetes doesn't address well. For instance, some systems are designed to be much simpler than Kubernetes, a good choice for beginning users and certain application types. Others target legacy application patterns, allowing these applications to be containerized without the retrofitting work sometimes mandated by Kubernetes for easier migrations.

FUTURE OUTLOOK

From Point Technology to Full Platform

Containers have evolved in a very short time from a small computing primitive into a full enterprise platform. This is very similar to how virtualization evolved, with the initial focus on the hypervisor but then quickly evolved to virtualization management and then other areas such as storage and networking. Containers are rapidly integrating into every area of the datacenter and having an impact on every layer of the stack. Some of these areas are:

- **Storage.** The developing Container Storage Interface (CSI) will define how storage systems interface with containers and how they surface storage features to containers. Persistent storage wasn't something container platforms initially offered, but now they are an essential requirement for most enterprises. Persistent storage is needed for application compatibility and to run stateful apps in containers such as databases.
- **Networking.** The Container Network Interface (CNI) defines how networking systems plug into containers. Containers are also spurring the development of new networking technologies such as service mesh, which is networking designed for microservices applications.
- **Security.** IDC research has found that security is a top driver/benefit of containers, while at the same time the top challenge. Moving to containers can improve security, such as how fast users can patch and update software. But it also introduces a new layer that needs to be secured. Some of these technologies are embedded deep within container platforms, such as the isolation between containers. Others are external, such as network security or image scanning.
- **Management.** While Kubernetes handles a lot of the fundamental management capabilities, the domain of system management is very large, and no single product can address everything. Application monitoring and application performance management are some examples of areas outside the scope of Kubernetes. Containers have the potential to operate on a much larger scale in terms of the number of instances and are also extremely ephemeral, which creates new management problems to be solved.

Optimizing VMs for Containers

As container adoption ramps in the industry, infrastructure in many areas, such as virtualization, will evolve to become more optimized for containerized workloads. As previously discussed, virtualization serves a very different purpose than containers and most containers today are deployed in VMs. However, there is a new technology that seeks to optimize the hypervisor and container host OS inside the VM. By slimming down the entire footprint, start-up time and resource utilization of the VM can be dramatically reduced. This type of VM is often referred to as a microVM and the optimized OS a microOS. A microVM/OS strips out any unnecessary feature and code, leaving only what is absolutely necessary to run a container. This has several benefits:

- Reduced attack surface for improved isolation and security

- Reduced resource utilization, with microVMs consuming as little as a handful of megabytes of RAM
- Fast start-up time (Traditional VMs and operating systems normally take several minutes to boot. A microVM/OS can boot in several hundred milliseconds.)

MicroVMs in the public cloud have several important applications. As the unit of provisioned compute increasingly moves toward containers, such as with serverless and functions, public cloud providers need a way to efficiently provision at a per-container level while providing secure separation between tenants. Optimized virtualization and container technology can allow public cloud providers to provide new container services at a more cost-effective price, with better performance and security.

The Transition to Containers

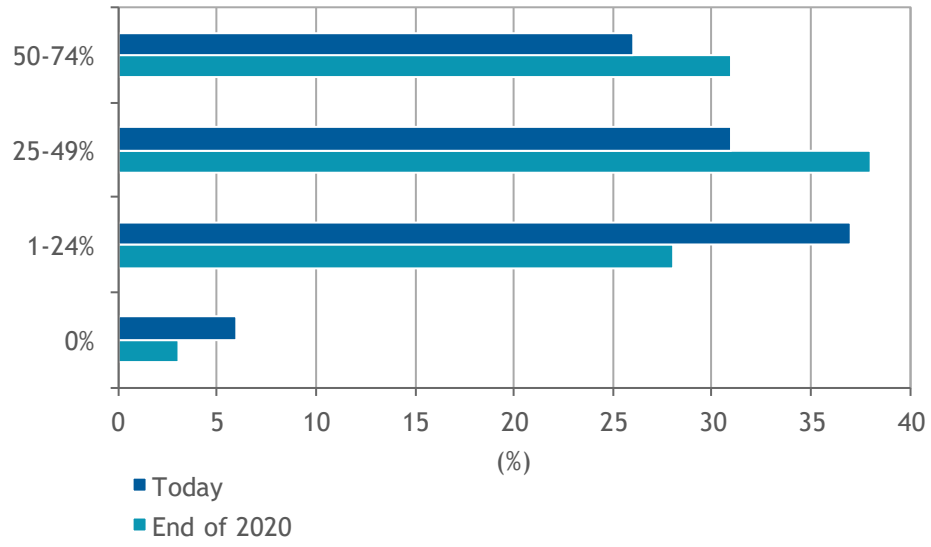
The industry will be in a transition to containers and cloud native for many years. It is a gradual evolution in many steps. For the foreseeable future, cloud native and traditional apps will coexist and integrate with each other. While 95% of DevOps teams today are using containers in some capacity, scaling this to the entire organization is a long-term task.

Simply containerizing an application doesn't imply that it is now cloud native. An IDC container survey shows that 54% of containers today run existing applications that haven't been refactored in any way. However, nearly 60% of these existing apps will eventually be refactored over time. Figure 5 shows the transition of apps to a cloud-native architecture over the next several years.

FIGURE 5

Cloud-Native Apps

Q. What percentage of your applications are cloud native today? What percentage are expected to be cloud native by 2020?



Source: IDC, 2019

Choosing a Public Cloud Container Service

Containers are available in many different services, depending on what the customer needs. Container infrastructure, or what many call containers as a service (CaaS), provides just the infrastructure and operations layer, letting customers choose and integrate their own developer and build tools. Many platform-as-a-service (PaaS) systems now use containers underneath and offer opinionated ways as to how applications are built, tested, and deployed. There are newer services emerging as well, such as serverless computing and functions as a service (FaaS). Serverless lets users to simply request and get a container, without having to manage or know about the container hosts underneath. Functions as a service lets users to write code that gets executed when a certain event is triggered. FaaS uses containers behind the scenes, but users only manage their code functions and never have to know about or touch a container.

While it is possible to bring your own software to the public cloud and deploy and manage it yourself, there is a general trend toward fully managed cloud services. In the early days of containers, container services weren't yet available, so users were forced to manage a container themselves. But now, service providers typically offer a wide range of container services and are rapidly iterating on them with new innovation. Using a managed service lets the cloud provider do the dirty work of managing the infrastructure, freeing up customers to focus more on their application. However, in some cases, customers may need to manage the full stack for reasons such as compliance; so clouds need to offer the full range of DIY to meet enterprise requirements.

When considering a public cloud container service, don't forget that infrastructure still matters. Containers ultimately still run on real servers, use real storage, and so forth. Containers build on the existing core cloud infrastructure of VMs, storage, and networking, so customers still need to evaluate these areas of the cloud. The scale, location, features, stability/availability, and cost of this infrastructure will play a key role in how well container services operate. Other points to consider include:

- **Decide what you want to manage and what the cloud provider should manage.** Customers generally choose to manage it themselves if those parts either serve as market differentiators, have special requirements that cloud can't fulfill, or for compliance and security reasons. For many though, infrastructure is becoming less strategic, and customers are choosing to offload that layer and focus on the application. Cloud providers now offer managed services that span the range of infrastructure, platform/middleware, and app development; so customers have a lot of choices of what they can have managed. Keep in mind that the mix of managed may vary from application to application and that this may change over time.
- **Serverless.** Most container services today offer a managed container control plane such as Kubernetes. The cloud provider fully manages the deployment, scaling, and upgrades of this plane. Customers typically manage the data plane, the servers where containers run. New serverless offerings now also offer fully managed data planes at the container level, where the cloud provider will manage the provisioning, scaling, and updating of the data plane. This serverless can further reduce the infrastructure burden on customers, but customers need to evaluate the operating system options, isolation models, and provisioning models to see whether it can meet their needs.
- **Cost and billing models.** There's a large variety of container services out there and more on the way, and the cost and billing models vary greatly. Sometimes container orchestration is a free service if you pay for the data plane compute and sometimes it is an extra charge. Data plane cost models are also changing. Commonly, most clouds charge for the underlying VMs at the standard rate. But with some newer serverless models, some may offer more granular metering that may have advantages for certain use cases. Customers need to evaluate the behavior of their applications and determine which models may be beneficial to them.

- **Integration with the rest of the cloud.** To fully operationalize containers, many supporting services are needed, such as persistent storage, networking, and identity and access management (IAM). Cloud providers generally have offered these services for a long time. But customers should investigate how integrated these services are to the container services. This should also include operational tools such as monitoring, log management, and analytics. Most cloud providers offer some existing capability in this area and these will still be needed for containers. However, cloud providers also need to extend this capability deeply into the container layer to give customers insight into what's happening at that level. There's a wide range of functionality required to build a container platform; so customers need to look beyond the basic container execution and cluster management for their future needs. Containers will touch every part of the cloud; so customers must evaluate not just a single service but all the other services that it may rely on or extend to.
- **Operating system support.** Enterprises typically manage multiple operating systems spanning from free Linux to paid enterprise Linux and Windows. Some applications are more reliant on specific versions of the operating system than others. The availability and support for the required operating systems is a fundamental issue for enterprises and will affect the usage of some services like serverless that may be less flexible.
- **Upgrade cadence and disruption.** Some projects like Kubernetes are evolving very fast, and users today generally will want a faster upgrade cadence for new technologies as new releases bring significant new features and fixes. As the technology matures, the cadence generally slows down and the upgrade benefits are less from version to version. Customers need to determine how often the cloud provider can provide upgrades and what is the disruption in service for the upgrades.
- **Strength of other application and data platform services.** Many containerized applications don't just need a solid infrastructure to run on but will want to take advantage of other cloud services to build their application with, such as databases, analytics, AI, and machine learning. These services may be more valuable and differentiating than infrastructure, depending on what the customer is looking to achieve with the application. The depth, cost, and innovation around these types of services can be critical to the code running inside of the containers.
- **Ecosystem, marketplace, and partners.** Ecosystems can make or break a platform. There is huge value in having a wide range of complementary solutions available from technology partners and integration partners to deliver it. No cloud provider or service can fill every need and segment within the market, and the strength and richness of the ecosystem surrounding it adds very real value to customers.
- **Pace of innovation.** A huge value proposition of the cloud is being able to take advantage of new innovation quickly and without a huge up-front investment. Clouds are not static things; they continually refresh and innovate across all areas of the stack. Containers are a new area, but core infrastructure and developer/application/data services are also tied deeply into what containers can do. Customers need to consider who they want to bet on to stay at the forefront of innovation in order to future-proof their investments.

AMAZON WEB SERVICES PROFILE

Amazon Web Services (AWS) is a pioneer and a leading provider of public cloud services. As one of the first and one of the largest public clouds, AWS has matured its technology over time, earning the trust of customers and operating with a large global reach and scale.

AWS is building container services with the goal of allowing its customers to run their containers at scale. Containers are just one part of AWS' initiative to support and drive modern application development as part of its customers digital transformation journey. AWS sees CI/CD, microservices, infrastructure as code, automation, and security/compliance as the key pillars to supporting application modernization, with containers as an underlying technology to support these.

However, AWS realizes that this journey for most is a stepwise path, with customers starting with a lift and shift migration to cloud, replatforming applications for cloud services, and finally, refactoring applications for microservices. Enterprises start in different ways and adopt different mixes of cloud-native technologies over time as the paths to application modernization are extremely varied. AWS seeks to supply customers with a wide range of cloud services, with container services just being part of the total solution, that can help with every step of the journey.

AWS also realizes that customers will have very different needs for managed services. Some customers, whether for differentiation or other reasons such as compliance, will want to manage more of the stack, including infrastructure. Others will want a managed approach, offloading the infrastructure part to focus on their application. AWS offers a broad range of services to accommodate both models and any mix the customer chooses.

One emerging way to approach computing is with serverless computing. AWS defines serverless as an operational model. There is no infrastructure to provision or manage (no servers to provision, operate, patch, etc.).

Key characteristics of serverless are:

- Automatically scales by unit of consumption (scales by unit of work/consumption rather than by server unit)
- Pay for value billing model (if you value consistent throughput or execution duration you only pay for that unit rather than by server unit)
- Built-in availability and fault tolerance (no need to architect for availability because it is built into the service)

AWS' primary container offerings include:

- **Amazon Elastic Container Service (Amazon ECS)** is a highly scalable, high-performance container orchestration service that supports Docker containers and allows customers to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for customers to install and operate their own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

- **Amazon Elastic Kubernetes Service (Amazon EKS)** makes it easy to deploy, manage, and scale containerized applications using Kubernetes on AWS. Amazon EKS runs the open source Kubernetes management infrastructure as a managed service across multiple AWS availability zones to eliminate a single point of failure. Amazon EKS is certified Kubernetes conformant so that customers can use existing tooling and plug-ins from partners and the Kubernetes community. Applications running on any standard Kubernetes environment are fully compatible and can be easily migrated to Amazon EKS.
- **Amazon Elastic Container Registry (ECR)** is a fully managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon Elastic Container Service, simplifying development to production workflow. Amazon ECR eliminates the need to operate container repositories or worry about scaling the underlying infrastructure. Amazon ECR hosts your images in a highly available and scalable architecture, allowing customers to reliably deploy containers for applications. Integration with AWS identity and access management provides resource-level control of each repository. With Amazon ECR, there are no up-front fees or commitments. You pay only for the amount of data you store in your repositories and data transferred to the internet.
- **AWS Fargate** is a compute engine for Amazon ECS and EKS that allows customers to run containers without having to manage servers or clusters. This removes the need to choose server types, decide when to scale clusters, or optimize cluster packing. AWS Fargate removes the need for customers to interact with or think about servers or clusters. Fargate lets customers focus on designing and building your applications instead of managing the infrastructure that runs them.
- The **AWS Container Marketplace** extends the existing Marketplace to containers, allowing customers easy access to a wide range of container ecosystem solutions. The Marketplace offers prebuilt Docker images, ECS task definitions, and Helm charts. They can be offered by sellers in a variety of pricing models, including free, bring your own license, and usage-based pricing.

AWS' core infrastructure services also power the company's container services, making them an essential foundation that containers are built on. These services are highly performant, scalable, reliable, secure, and available in many global locations. For compute, EC2 provides the VMs that run container hosts. Storage is available in several different formats (S3 for object storage, EBS for block, and EFS for file), which are all available for containers to access. For customers that want to manage their own container infrastructure, they will be accessing these services more directly.

Amazon's container offerings integrate well not only with each other but also within the rest of the familiar AWS ecosystem such as VPC networking, Elastic Load Balancer, IAM roles, CloudWatch events, CloudTrail logs, and CloudFormation templates to create a full end-to-end container infrastructure and operations platform.

AWS Fargate and Lambda use an optimized microVM technology developed by AWS, called Firecracker. Firecracker has been open sourced by AWS and is available on GitHub. Firecracker is a virtual machine monitor that works with KVM to operate like a lightweight hypervisor and is designed to run serverless applications, such as ephemeral-like functions or Fargate-style containers, which cycle very rapidly and are lightweight. Firecracker allows Fargate and Lambda workloads to be run more efficiently, allowing them to be offered at competitive price points. The performance is also increased by having ultra-fast boot times, which can reduce the cold start latency for containers. And, of course, the primary benefit is security, allowing containers to be securely separated with high granularity. Firecracker allows users to get not only the benefit of containers but also the isolation and security model of virtual machines.

Complementing container orchestration is AWS App Mesh, a fully managed service mesh offering. A service mesh is a networking technology specifically designed for the needs of microservices applications. Traditionally, networks in the cloud are at the lower layers of the network for those familiar with the OSI model. In AWS, it is best exemplified by VPC, the virtual private cloud. Users set up their CIDRs and their virtual private clouds into which clusters, servers, and containers are deployed. All networking happens at that layer. Users deploy an elastic network interface, and then attach security groups to it.

But as users migrate to smaller and smaller microservices-based applications, networking presents new challenges. AWS App Mesh, a type of service mesh, is a Layer 7 communications mesh that services register into. The mesh applies all the appropriate policies and traffic management so that users can build reliable and resilient applications with all the governance and compliance that enterprises need.

AWS App Mesh is available today and works with ECS, EKS, Fargate, CloudWatch, and X-Ray.

CHALLENGES/OPPORTUNITIES

Challenges

- Containers don't fully solve the portability problem. While the container stack is much more standardized than with VMs, mobility is only at the infrastructure level. If the code within the container is tied to a particular cloud service, API, framework, and so forth, then the container is still only as portable as those dependencies. Data that the application depends on can also limit portability, depending on the size and cost to move that data (data gravity).
- Containerizing Windows workloads remains a relatively untapped and important segment of customers. Amazon must identify the best candidates to build traction. It's still very early for these customers and Amazon needs to hone its value proposition for Windows containers.
- Containers as a technology is still very new to enterprises. Even newer is expertise in building cloud-native and microservices applications. There is going to be a lot of research, experimentation, and learning from enterprises in the coming years in order to adopt containers. Cloud providers can provide a lot of the technology as managed services, but skills, processes, and people are much slower to change.

Opportunities

- Amazon has a number of developer tools such as CodeBuild that address the development side of containers and these are being integrated into its container services. While not everyone wants an opinionated PaaS, Amazon does have the opportunity to build a more end-to-end system for those that want a guided PaaS experience.
- Containers are clearly the compute vehicle for the next generation of applications and public cloud plays a very large role in that development. Amazon, which has established itself as a leading builder of robust cloud infrastructure, can leverage the strength of that infrastructure for containers and capture share as the market transitions. However, as critical as infrastructure is, it may be the other services developers' access with their code inside the containers, such as data services, AI, or ML, that ultimately is the differentiator for Amazon. Differentiation will require constant innovation and Amazon must continue to push the envelope.
- With the announcement of App Mesh, AWS is focusing a lot of its investment into application-level networking and application-level communications across AWS, a key requirement as users increasingly transition to microservices.

CONCLUSION

Containers are the foundation for cloud-native applications and fueling a transformational change in how enterprises build and run applications. Enterprises will increasingly leverage containers in the public cloud to build modern, cloud-native applications and iterate on these applications extremely quickly. Enterprises evaluating container services in the cloud need to take a broad view. Containers still run on real physical and virtual infrastructure, so basic IaaS robustness and physical presence still matter. Users also need to consider containers as a complete and holistic application platform instead of a single service. A complete container platform needs to integrate multiple areas of infrastructure such as storage, networking, and monitoring, as well as developer-facing tools such as CI/CD pipelines and application data services. Containers also serve as the technology foundation for serverless and functions as a service, a higher level of abstraction, and a new compute model that can benefit certain types of applications. The change that containers are ushering in can't be understated. Containers and the public will enable enterprises to operate more like cloud hyperscalers, building and iterating apps quickly and operating them at large scale.

About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world's leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter: @IDC
idc-community.com
www.idc.com