CHAPTER

MANAGING CONTAINERIZED APPLICATIONS 13

In light of today's competitive realities, as outlined in our industry vision, enterprises must utilize leading-edge technologies like private PaaS and containerized apps to remain agile.
Bart Copeland

Virtualization will always have a segment, but generally people don't want to virtualize the whole operating system. They just want to run their app in a container.
Jevgeni Kabanov

INTRODUCTION

The introduction of containerized **applications** marks a pivotal moment in the trajectory of information technology (IT). The container concept was adapted from the freight industry, where the term "**containerization**" is used to describe a technique to simplify and speed up the transportation of freight in a secure and efficient manner. Different types of freight are placed in separate uniform size containers and transported using any number of different transportation methods, such as planes, trains, trucks, and ships to specified locations. Once packed, the freight is not unpacked until it reaches its final destination. In IT, the term "container" is applied to a technique whereby an application and data (freight) are securely and efficiently transported over local area and virtual networks to specific locations, such as desktop or laptop computers, servers, mobile devices, remote data centers, and public and private clouds, and are not unpacked until they reach their final destination (Fig. 13.1).

Containerization is sometimes referred to as lightweight virtualization or **container-based virtualization**. It is a Linux-based technology where a common operating system kernel runs multiple instances on a single operating system, eliminating the need for each instance to run on its own operating system (Fig. 13.2).

Although it has been heralded as the latest major technology shift in the IT industry, containers are not a new technology. Application containers have been used for more than a decade by virtual private server providers. However, as these companies switched to **virtual machines (VMs)** to get better performance, interest in container technology declined. Recently, IT organizations began to call for tools that provide more rapid deployment of applications. This led **platform as a service (PaaS)** vendors to adopt and standardize container technology to provide isolation and resource control, which consequently led to a renewed enthusiastic interest in containers. In fact, some analysts have speculated that containers are the next logical step in server consolidation.

FIGURE 13.1

Freight containers vis-a-vis application containers.

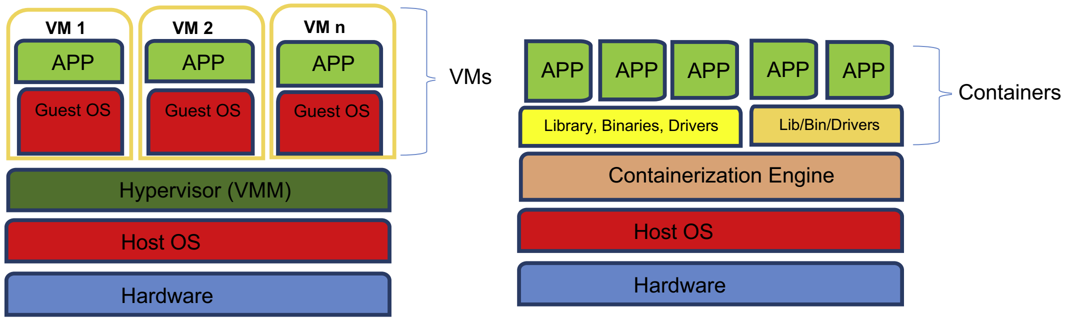

FIGURE 13.2

Hypervisor-based versus container-based virtualized architecture.

WHY CONTAINERIZE?

The most valuable benefit containers provide is a portable, reliable format that runs applications on a variety of hosts and reduces memory requirements, increases ease of migration and use of applications, enables faster deployment and backup, and frees up server capacity. In contrast to hypervisor-based virtualization, containers use the same host operating system repeatedly and the role of the hypervisor is handled by a **containerization engine** that is installed on top of the operating system. In this way, the container is notably smaller than a VM. The underlying concept of containerization is that virtual instances share a *single* host operating system and relevant libraries and drivers. As a consequence, significantly more containers can be hosted by a host server. It was estimated that 10–100 times the number of container instances can be hosted, compared with the number of VM instances on the same server. However, one of the reasons that containers lost some of their appeal was that applications all relied on a common OS kernel, so this approach could only work for applications that shared the same OS version.

THE REVITALIZATION OF CONTAINERS

One of the leaders driving the renewed interest in containers is **Docker**, an open-source project developed by dotCloud, a PaaS vendor. Docker was initially launched as an interactive tutorial in 2013 to address the issues of the lack of container portability caused by reliance on a common OS and the high level of Linux kernel experience required to develop containers. Docker offers a containerization engine that has an easy-to-use packaging format that envelops an application with its dependencies and container management tools, which empower a wider range of individual developers. The Docker containerization engine enables an application to run across a different Linux OS and is portable. In addition, developers can use any language to write an application and easily move the application across different devices. Software developers flocked to try this new approach, and within a year, Amazon and Red Hat had provided commercial support for Docker. Since then, a number of other vendors followed suit with products that rival or improve on the Docker concept of container-based systems. These include CoreOS Rocket, Cloud Foundry Garden, Google Omega, and Linux Containers (LXC).

To address the complex coordination issues associated with deploying multiple containers, that is, a **container cluster** or **container pool**, Google developed Kubernetes to manage a cluster of Linux containers as a single system. In the same way that containers empower individual developers by reducing much of the overhead associated with deploying a single container, the Kubernetes containerization cluster manager empowers teams of developers to create services involving multiple containers and directs these container clusters to follow a set of deployment rules to ensure correct operation. The ensuing **containers as a service (CaaS)** infrastructure is as reliable, fast, scalable, and flexible as any **infrastructure as a service (IaaS)** coupled with the ease of use of PaaS solutions (Fig. 13.3).

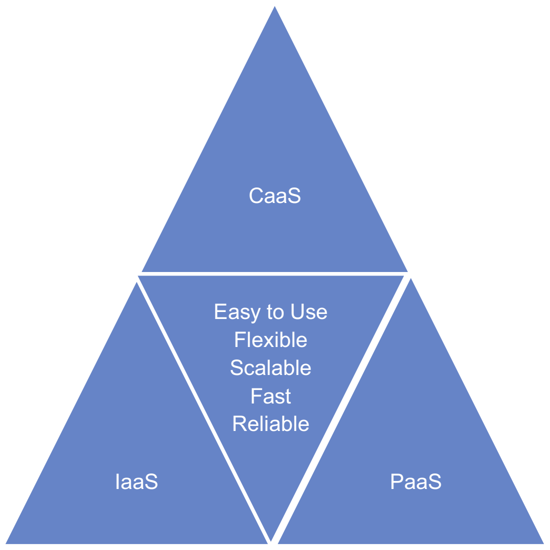

FIGURE 13.3

Evolution of the CaaS infrastructure.

At this point, it is important to note that the renewed and enthusiastic support for container-based virtualization does not mean that hypervisor-based virtualization is no longer a viable option. On the contrary, container-based virtualization and hypervisor-based virtualization should be viewed as complementary, not competing, technologies.

In different situations, they are both good choices for an IT organization, each providing a different solution to a different use case and even working together to provide the best solution. For example, if it is important that resources are distributed equally and performance is task independent, then hypervisor-based virtualization is the best choice.

Hypervisor-based virtualization is also the best fit for complex enterprise applications that depend on large data sets. On the other hand, container-based virtualization addresses scalability challenges, making it a good match for microservices-based architectures. For example, for an organization that executes many small isolated processes, container-based virtualization is the best option because it uses less overhead and performs better in those situations.

BENEFITS OF MANAGING CONTAINERIZED APPLICATIONS

From a management point of view, container-based virtualization offers new possibilities and flexibility for specific workloads because it enables applications to be available and quick to deploy. Unlike virtualization, which runs separate operating systems on each VM and thus can lead to high resource costs and more performance overhead, containerization generally simplifies application lifecycle management tasks by using a singular host OS and improving performance. In addition, the implementation of containers reduces memory requirements and consequently increases the ease of migrating applications. It also frees up server capacity for greater container storage for VMs to use, and the use of layers in containerized engines, such as Docker, improves security in application deployment.

In addition, while hypervisor-based virtualization uses resources more effectively than physical machines, container-based virtualization has shown a potential to further reduce overhead and improve the performance and utilization of data centers because everything runs on top of the same kernel. LXC is a kernel technology used in container-based virtualization that can run multiple processes in its own isolated environment. This isolation of processes can result in reduced overhead on major software deployments by decreasing deployment time, increasing application portability, and consuming fewer physical resources.

Another advantage of container-based virtualization that appeals to application development managers is increased developer productivity. Containers allow developers to write code in a standard way in any language. The number of applications inside a container can vary. Some organizations prefer the one application per container approach. This enables new stacks to be created without difficulty and applications can be built or rebuilt and managed easily but results in a large number of containers that must interact and more complex container management. Others favor multiple applications in one container, making them easier to manage but adding complexity to the building or rebuilding of applications. In either case, the increased level of efficiency introduced by containerization lends itself particularly well to a number of specific application management situations.

BALANCING MICROSERVICES IN THE CLOUD

Containerization really excels in the **microservices** architecture that are increasingly identified with cloud-native web applications. The introduction of containerization engines such as Docker, Cloud Foundry's Garden, and LXC motivated organizations to utilize the service as a means for cloud scalability. As a result, containerization is becoming increasingly popular among cloud providers. One of the largest users of containers in the cloud is Google. It is estimated that Google starts 3300 containers per second, 24 hours a day, 7 days a week. That is more than 2 billion containers per week. Google runs its container stack on Ubuntu (an open-source, Debian-based Linux operating system) with an emphasis on performance, rather than on ease of use.

To capitalize on this approach, Google developed Let Me Contain That For You (LMCTFY), its own LXC variation. LMCTFY isolates resources used by multiple applications running on a single machine and manages the workload by balancing needs of overcommitted machines with batch workloads. The fact that containers can be more densely packed on servers is a big advantage in the cloud where success or failure is measured by overall efficiency. As a result, Amazon Web Service and Microsoft are also enthusiastically embracing containers on their cloud hosts.

ENHANCING THE BRING YOUR OWN DEVICE EXPERIENCE

Containerizing applications is the predominant approach recommended by industry analysts for enhancing the bring your own device (BYOD) experience of organizations and employees. In the age of BYOD, **mobile device management (MDM)** is no longer simple. Gone are the days when corporate mobile devices were loaned to employees for business use. In the past, if a device was lost or an employee left the company, maintaining data and application security was as simple as remotely wiping all data and applications from the device. Now, when organizations set BYOD policies, administrators must consider the rights of its employees as device owners, as well as the security of organizational data and applications. In an environment where corporate and personal data coexist on personal devices, containerized enterprise mobile applications allow corporate applications and data to safely reside on personal devices under the control of the enterprise without infringing on the employee device owners' rights.

Application containerization provides each application and its data with its own secure runtime container that uses strong encryption, separate from native device encryption, and is controlled by a strong password policy. This enables IT managers to set and implement broader security policies confined to enterprise applications and implement features, such as sign-on authentication, data manipulation restrictions, and application-level encryption to protect sensitive corporate data and applications. The isolation provided by containerization reduces malware infection or privilege escalation from malicious applications on employee-owned devices. To assist in the adoption of these types of BYOD policies, it is important to preserve the look and feel of an employee's device to ensure a positive user experience. Containerization does this by securing enterprise applications without the need to cut over to a separate environment to use them. Caution still needs to be exercised, however, because containers are an effective layer in a security strategy, but the reality is that while creating a container raises the barrier of entry for data access, it will not necessarily prevent it.

INCREASING ADMINISTRATIVE CONTROL

To increase the level of administrative control, IT managers are using **app wrapping**, a useful containerization management technique. App wrapping also enhances the mobile application experience for the user. App wrapping essentially allows an administrator to take an application, implement extra security and management features on it, and redeploy it as a single containerized program. To achieve this, the mobile application management administrator must set management policies to create a set of dynamic libraries that are implemented on top of the application's native binary. For example, the policies might direct developers to add dynamic libraries to an application to make it behave differently when it is opened and determine whether authentication is required for a specific application when a certain type of communication occurs. Another management policy might be to establish whether data associated with an application can be stored on a device and/or whether file sharing is allowed. For example, app wrapping can be used to implement the BYOD security policies discussed in the previous section, thus saving time and reducing the complexity of developing mobile applications.

MANAGING THE ENTERPRISE APP STORE

Growing numbers of organizations are adopting **enterprise app stores**, which are driven by the increased use of enterprise mobile devices and the introduction of MDM. Applications that are downloaded from public app stores can disrupt IT security as well as application and procurement strategies. To overcome these issues, MDM providers are beginning to launch sophisticated app stores to enterprise and third-party applications for access by mobiles devices such as smartphones, tablets, and PCs.

Better management of the enterprise app store can be realized by effectively using containerized applications. To accomplish this, the IT organization must follow the maxim of "contain, control, and deliver." This is achieved by containerizing apps, data, and devices, controlling access based on identity and policies, and delivering a personalized app store to each user. In all cases, it is imperative that the IT organization views its enterprise app store as part of its application strategy and not its infrastructure strategy. Users must have a reason to use the enterprise app store. Therefore, the primary determinant of its success is supply and demand. To determine this, application managers must work collaboratively with **DevOps** teams to monitor downloads and usage to determine the value of its applications to users and regularly update enterprise app store offerings to reflect user needs.

TRANSFORMING THE DATA CENTER

The **containerized and modular data center facility (CMDF)** is an increasingly popular method for constructing and deploying data centers. With the introduction of this approach, the speed with which data can be handled has increased by orders of magnitude, allowing the data center to use the infrastructure more efficiently. With this new approach, IT developers can achieve faster deployments, reduced I/O latency, and lower operating and capital costs for IT managers through a set of replicable, preengineered, preassembled, and reliable components that produce the required capacity. An additional advantage is that, with a CMDF, organizations can increase or decrease their data centers' capacity in smaller, prescribed steps. As a result, for many organizations, containerized data centers were viewed as a quick fix to address an existing capacity shortfall. However, caution should be used when determining a CMDF's suitability as a strategic asset. It is important for CIOs and CTOs to assess not only facility needs but also the organization's overall IT architecture and its need for large-scale deployment of computing capacity before proceeding.

AUTOMATING DEVOPS

The introduction of containerized engine tools led to the automation of DevOps principles. By automating the software release pipeline, containers allow individual components and services to be viewed as discrete components that facilitate debugging or analyzing an application's architecture. Now integration-and-test means testing the relationship and interactions of the containerized modules, and management of the pipelines now becomes management of the containers. Instead of focusing on coding, IT operations personnel are freed up to work on business problems such as load balancing, capacity planning, disaster recovery, and distribution.

CHALLENGES OF MANAGING CONTAINERIZED APPLICATIONS

Despite the advantages that the container-based virtualization approach offers, it has its drawbacks. Some of these challenges and ways to manage them are discussed next:

Image Sprawl—The ease with which Docker-like images can be created can lead to image sprawl, which creates the same problems with managing security and compliance as VM sprawl, which was discussed in the previous chapter. Thus, the same management principles that apply to VM sprawl apply to image sprawl in a containerized environment.

Security and Data Confidentiality—When using containers in a mobile environment, data security can be compromised. Therefore, it is important to place applications in a secure container, that is, a third-party mobile application that acts as a storage area, authenticated and encrypted by software and governed by IT policies. This requires organizations to ask users to consent to full-device encryption and remote wipe before allowing them to use their own devices at work. Unfortunately, this can undermine the user experience.

Other considerations include cost and capability; security data containers must be purchased, deployed, and maintained. Organizations that lack established mobile application management practices may not have the necessary infrastructure to support large-scale implementation of secure containers. For these reasons, many organizations opt to use secure containers only for high-risk devices and for employees who need access to sensitive business data. Another approach, highlighted in recent research conducted by Enterprise Management Associates, is to deploy containers on top of VMs to leverage existing management tools and security practices related to VMs.

Network Management—Containers in distributed systems need advanced network support. However, the container platform was designed to assist developers in writing and testing code on a single host. This led to network management difficulties in production settings where interhost communication between containers is required. Because containers are not exposed directly to the layer-2 network, external systems that need to directly communicate with a container have to communicate with the IP of the container's host and a TCP socket. The larger the number of containers that are frequently accessed externally from the host, the greater is the complexity of network management. Fortunately, providers have acknowledged this management challenger and began to develop products to address this issue. For example, to extend network capabilities, Docker developed SocketPlane, a software-defined network technology that provides a plugin architecture for its product. SocketPlane puts control back into the hands of developers by creating an overlay network that allow containers to connect directly. Docker also created Libnetwork, an

open-source project that offers an API for third-party solutions to provide network services.
In contrast, Weaveworks takes an "under the hood" approach in Weave, their open-source network management offering. Weave simplifies application development and eliminates the need for developers to change their practices, code, and tools. This is accomplished through the use of Weave routers that automatically track the state of the network. These routers perform low-level networking tasks including tracking MAC addresses, encapsulating data, and routing packets.
Tooling—Containers do not share many of the same configuration attributes at the OS layer that physical and virtual servers do, nor do they have a well-developed set of tools for managing them. This means that application dependencies are no longer managed on a server or at the VM level; instead, they are managed *inside* the container. Consequently, IT personnel are required to manage the container software instead of the applications and their associated dependencies, and currently, container management tools are scarce.
One management tool that begins to address this challenge was developed by CA Technologies. Their Smart Containerization technology associates a policy describing security, performance, and support requirements with individual content, applications, or devices, thus protecting the target by enforcing policies that are appropriate to the type of content being managed. For example, a document may have a policy associated with it to prevent it from being stored locally on a device, or a mobile application may have a policy determining where it can execute. While other providers are also beginning to develop these much-needed management tools for container deployment, most of the tools are somewhat immature and open source with a resultant lack of a dedicated support structure present in proprietary software packages. In the meantime, until these emerging management tools become more mature, IT managers in large enterprises need to maintain a dedicated development staff to support containers. Unfortunately, this approach is not feasible in smaller companies that lack the necessary human resources.
Establishing Standards—Although some consider Docker as the de facto standard for container technology, at this point it is uncertain what the official standard for containers will be. However, a move is currently afoot to address this conundrum.
On June 22, 2015, the **Open Container Initiative (OCI)** was launched under the auspices of the Linux Foundation. OCI is an open-governance structure formed to expressly focus on open industry standards around container formats and runtime to ensure interoperability of different container technologies. The list of high-powered OCI sponsors is indicative of an increasingly high level of interest in and use of container-based technology. The sponsors include Apcera, AT&T, AWS, Cisco, ClusterHQ, COEOS, Datera, Docker, EMC, Fujitsu, Google, Goldman Sachs, HP, Huawei, IBM, Intel, Joyent, Kismatic, Kyup, Mesosphere, Microsoft, Midokura, Nutanix, Oracle, Pivotal, Polyverse, Rancher, Red Hat, Resin.io, SUSE, Sysdig, Twitter, Verizon, and VMware (the OCI will be addressed in greater detail in Chapter 16, "The Case for Standards").

SUMMARY

Containerization is a solution to allow software to run reliably when moved from one computing environment to another, such as a data center on a physical machine to a **virtual machine (VM)** in the cloud. Containers are much more lightweight and use far fewer resources than VMs, but are generally accepted to be somewhat less secure than VMs. As a result, it is likely that containers and virtualization will evolve into complementary, rather than competing, technologies. Although given that more and more microservices are being deployed in the cloud and containers enable faster deployment, more reliable services, and more efficient scalability and use fewer resources, it can been speculated that a greater number of next generation applications will be built using container-based virtualization rather than hypervisor-based virtualization.

Containers excel in the microservices architecture of the cloud and offer great potential for enhancing the BYOD user experience, increasing administrative control, managing the enterprise app store, transforming the data center, and automating DevOps.

However, it is important to note that containerization is not a panacea. Management challenges include network management, tooling, security issues, data confidentiality, and image sprawl. Added to these is the lack of robust management tools necessary to orchestrate a large number of containers or container clusters. This deficit is expected to be short-lived as more vendors step up to develop new tools or improve existing ones to create and manage containerized applications.

A move toward establishing common standards for containerization is under way, led by the Linux Foundation and a coalition of vendors, users, start-ups, and industry leaders. The project, known as the Open Container Initiative, expects to deliver final container standards by the end of 2017.

KEY TAKEAWAYS

- Containerization was initially understood by a select few and was unavailable to the vast majority of organizations because of the lack of Linux expertise and portability.
- The newly developed availability of containerization technologies has exponentially increased the economic advantage of microservices-based application architectures.
- Containerization and virtualization are both good choices but in different use cases.
- More management tools are needed to help manage containerized applications.
- Standards to oversee containerization are being developed by the Open Container Initiative.