TECH DOSSIER

CONTAINERS AND MICROSERVICES ARE REINVENTING SOFTWARE APPLICATIONS

Modular, building-block approaches to application development and deployment are gaining traction.

Only a small percentage of organizations think of themselves as software companies. On a fundamental level, however, almost every organization in our modern world depends on an extensive software foundation to power their operations. Without an underpinning of flexible, scalable, and highly reliable software, most organizations would have a tough time surviving, much less thriving.

INSIDE

TODAY, with all the talk of digital and business transformation, the infusion of software into almost all business activities is continuing to accelerate. But it isn't just business models that are changing. Software itself has had to transform to keep pace with the increasing demands it faces, and the expanding capabilities it must deliver.

One of the central elements of this software transformation is to move away from monolithic applications that consist of hundreds of thousands of lines of interdependent and—as a result—inflexible code, to service-oriented models. In a service-oriented model, developers create discrete, modular "microservices," each of which performs a specific function or small group of functions. These building-block pieces of code can be combined and recombined in innumerable ways to create full-blown applications that replace—and improve upon—their monolithic-code predecessors. This new application model provides greater speed and agility in meeting user needs and helps organizations quickly respond to market changes.

While conceptually compelling, the microservices software model is far from simple to achieve. Beyond the challenges of creating and integrating hundreds, if not thousands, of individual services, the surrounding environments in which these modules must operate are also in a state of flux.

The past decade has seen the proliferation of virtualized environments that provide a layer of separation between the software code and the hardware platforms on which it runs. Another pervasive trend, cloud computing, is providing flexible, shared, and scalable environments that exploit virtualization, automation, resource pooling, and other cutting-edge architectures and technologies, including containers.

Modern containers serve as powerful vehicles that organizations can leverage to more easily exploit cloud infrastructure and adopt microservices. In essence, containers are standards-based wrappers that can be used to envelop microservices, making them easier to distribute, deploy, and orchestrate on any platform. Thanks to containers, the long-standing software development objective of "write once, run anywhere," is becoming more readily achievable.

Container technology has emerged largely because of open source software community efforts, primarily via an initiative (now a company) called Docker®. Docker took a complicated Linux technology that had been around for over a decade and made it understandable and usable by a broad range of application developers and operations professionals. Since Docker's initial open source release in 2013, the company's containers and tools have become de facto standards on Linux® platforms, with Windows platform support in the works. In June 2015, in order to help stabilize and accelerate the adoption of containers, 21 companies, including Docker, formed the Open Container Initiative (OCI) to establish industry standards around container formats and runtimes. The OCI, working under the auspices of the Linux Foundation, is basing its efforts on the Docker container format and runC runtime.

Containers are well suited to cloud-native apps and cloud computing as evidenced by the support provided for containers in the leading platform-as-a-service (PaaS) environments such as Cloud Foundry and Red Hat® OpenShift, and in the leading cloud platforms such as Microsoft® Azure, Amazon® Web Services, and OpenStack®, the popular open source cloud infrastructure and deployment framework. Driven by growing interest in containers among OpenStack users, the OpenStack community has launched several initiatives to help organizations create and use containers within that cloud environment.

In this Tech Dossier, we briefly explain the functionality and the interrelationships of microservices, containers, and the complementary technology of virtual machines. We also describe some of the main orchestration tools and supporting technologies for containers available to developers, as well as outline key OpenStack projects that support containers. The dossier concludes with a brief assessment of the importance of containers and microservices for the future of IT.

Microservices, virtual machines, and containers

Before exploring the current activity related to containers and microservices, it's important to understand how these technologies relate to one another and to the well-established technology of virtual machines (VMs). In practice, microservices, containers, and VMs are largely complementary, and can beneficially interact with one another in a variety of ways.

VMs run on top of a hypervisor layer that separates them from the host hardware and operating system. (Hypervisors can also run directly on bare metal.) A virtual machine contains its own operating system instance (which can be different from the host platform's OS, if desired), applications, and support libraries. The hypervisor acts as a translation layer between the underlying OS and the OS(s) in the virtual machines, meaning that a single hardware platform can support multiple operating system instances, and a variety of software applications.

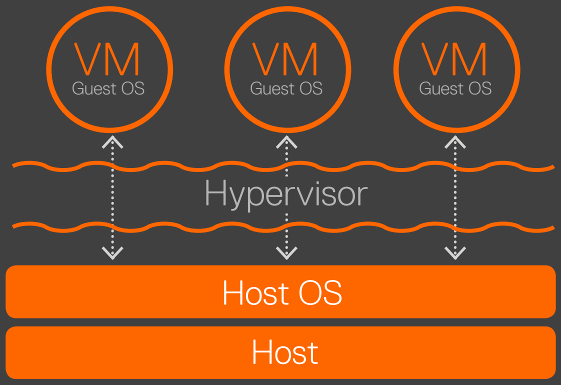
Application containers, by comparison, don't require their own OS instances or hypervisors. Instead, they utilize features of the host's operating system to wrap resources—application code, runtime, system tools, system libraries in compact packages—and to isolate them from other containers. Like VMs, application containers share the host system's compute, networking, and storage resources.

To effectively operationalize containers, a supporting infrastructure is required. This infrastructure comprises several toolsets and technologies, including a container-optimized operating system, networking and storage infrastructure, runtime components, and image creation and management tools. As with containers themselves, these toolsets are most often open source, which facilitates rapid innovation but can also pose challenges as there are many different approaches to solving problems.

VIRTUAL MACHINES AND CONTAINERS: KEY DIFFERENCES

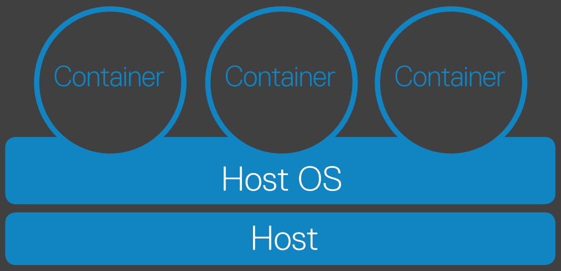
Virtual Machine Environment

Virtual machines run on their own operating system instances and are managed by a hypervisor that functions as translator between them and the host OS.



Container Environment

Containers do not require a hypervisor and share resources of the host operating system, making them much smaller and more agile than virtual machines.



Because application containers don't include a full OS, containers can be orders of magnitude smaller than VMs. Making containers even more efficient was Docker's "invention" of container images, which are essentially snapshots of container file systems that can be stored on disks. Because the container image only needs to store the bits that are different from its parent image, each image can be quite small.

The smaller size of container images—megabytes vs. gigabytes, potentially—also means containers can be created faster and started up in milliseconds rather than the minutes it may take to launch a VM. In addition, because containers have direct access to device drivers through the host OS kernel, their I/O operations can be much faster than their VM equivalents.

Containers and microservices come with challenges

While containers provide benefits for service-oriented applications, this technology has some notable challenges, security in particular. As mentioned above, containers share much of the underlying OS kernel along with many libraries and binaries. This interplay introduces dependences that, if not properly managed, can negatively impact container portability.

This sharing of the OS kernel also means that containers have a deep level of authorization (usually root access in Linux environments), so are less isolated than VMs. A bug in the kernel affects every container, and flaws and attacks have a much greater potential to carry down into an underlying OS and over into other containers.

To address security concerns, containers can be run inside a VM, which provides a greater degree of isolation for the application from the underlying platform. A key trade-off of this approach, however, is the loss of the lightweight execution environment provided by containers.

Is the reverse—running a VM in a container—advised? Theoretically, containerizing VMs with their associated apps could be a way to increase application portability between various clouds. While possible, because VMs have specific process and functional requirements for the underlying operating system, these requirements may not be effectively supported in containers. As a result, the VMs-in-containers scenario introduces significant complexities and challenges.

The fact that containers are small, lightweight, and can be created quickly can also cause problems. These characteristics can easily lead to container sprawl and create situations where more data center resources are consumed than needed. Container management and orchestration toolsets are evolving rapidly to address challenges related to container sprawl.

As for microservices, they can be instantiated as either individual services or functionally integrated collections of several services. Developers, however, must consider a number of architecture styles and design principles when creating microservices. A microservices architecture is an approach to developing software applications composed from a collection of small services, each running in its own process and communicating with lightweight mechanisms. These services are small, fine-grained, and designed to perform a single function. Services are built around business capabilities and independently deployable by fully automated deployment machinery; they are designed to embrace failure and faults, and each service is elastic, resilient, composable, and complete.

Microservices architectures greatly increase the ability to iterate, scale, and make continuous delivery (CD) and continuous integration (CI) of software applications possible. CD/CI methods accelerate software deployment and operations by enabling software to be updated dynamically and continuously. Similarly, software development processes and tooling have evolved to more agile methods that can refine and iterate software code rapidly.

With these advances, however, comes another new challenge: How can the historically separate software operations and software development systems and processes be coupled without creating bottlenecks or choke points? Creating this seamless bond between operations and development is where DevOps comes in. DevOps is a methodology that gets developers and operations to work together and decrease friction while increasing the velocity of software environments.

Defining and describing DevOps is beyond the scope of this paper, but it is important to note that DevOps provides key operational

SIDEBAR

DELL EMC ENGINEERED SYSTEMS

IT groups are providing acceleration to their organizations with engineered systems. Examples of these solutions include:

- **Dell EMC Native Hybrid Cloud**
  A turnkey Pivotal Cloud Foundry developer platform that accelerates and simplifies cloud-native application development and delivery. The solution integrates a marketplace of Developer and IT Ops Services with PaaS and a choice of infrastructure-as-a-service (IaaS) technologies to provide a complete cloud-native solution that is supported and sustained as one product.
- **Dell EMC Red Hat OpenStack Cloud**
  An integrated, modular, open platform for flexible, agile, and scalable cloud environments. Customize your IaaS environment and prepare for the future with a validated core architecture and certified options, including options for PaaS and software-defined networking from the OpenStack community—all supported by the industry's leading cloud solution providers.

When evaluating any provider to be your technology partner, it's important to weigh that provider's ability to meet your needs both today and in the future. Dell EMC's ability to help you pick your best path to cloud and new technologies like containers is founded in a deep technical understanding and management of complexity and costs along the way. Providing a choice of integrated solution stacks and the underlying technologies used in them is what makes Dell EMC a leader in the marketplace, and the chosen technology partner for organizations across the globe—from small startups to large hosting providers.

Providing a choice of integrated solution stacks and the underlying technologies used in them is what makes Dell EMC a leader in the marketplace.

aspects for modern cloud applications in areas such as automated build and release, configuration management, and performance management. In short, containers, microservices, DevOps, and cloud go together to deliver applications that can autoscale, be rapidly updated and improved, and be reconfigured dynamically to meet the changing needs of businesses and users.

The ultimate configuration and deployment approach selected for microservices, containers, and VMs will vary greatly depending on several variables—performance, storage capacity, security, interoperability, etc.—that must be considered and assessed for any given use case. Similarly, the roster of toolsets and technologies deployed to create a container environment will vary greatly from organization to organization, making the identification and transfer of best practices difficult.

Managing and orchestrating containers

Some of the biggest challenges associated with containers aren't involved with creating individual modules. Rather, they are the challenges related to managing and orchestrating what may be thousands of individual containers (container sprawl)—some potentially located on different platforms and in different geographic locations. If you can't rapidly and dependably aggregate containers and the microservices they carry to deliver the end-to-end functionality required for any application, the entire container environment can quickly crumble.

Fortunately, there are several container orchestration engines and platforms available, offering a range of capabilities and functionality. Note: Container orchestration and management tools are evolving very rapidly and no one tool or platform can address all of the challenges associated with managing containers. Some of the best-established orchestration tools include the following:

- **Docker Swarm**—As noted, Docker spearheaded the concept and delivery of container images, and Docker Swarm supports native clustering for Docker containers. This lets you turn a group of Docker engines into a single, virtual Docker engine.
- **Kubernetes**—An open source orchestration system for containers that handles scheduling and manages workloads based on user-defined parameters.
- **Apache Mesos**—Can be used to deploy and manage application containers in large-scale clustered environments.
- **Cloud Foundry**—An open source PaaS for developing, deploying and running apps in private or public clouds.
- **CoreOS Fleet**—Deploys Docker containers across a cluster.
- **RancherOS**—Deploys containers across a cluster and simplifies the building of container images.
- **CloudSlang**—An open source project within Hewlett Packard Enterprise intended to orchestrate container-based solutions such as Docker and CoreOS using ready-made workflows.
- **Mesosphere Marathon**—Launches long-running applications, and is application aware and intelligent.

Beyond these focused container orchestration engines, there are several cloud and application environments that include container management and orchestration capabilities as part of their broader functionality. Among them:

- **Pivotal Cloud Foundry**—A complete, developer-focused cloud-native platform that delivers sophisticated infrastructure-focused management controls, including one of the most complete container management capabilities available.
- **Red Hat OpenShift**—A Kubernetes-based PaaS solution that goes beyond native Kubernetes capabilities by offering application staging and testing integrations and a rich web user interface into common role-based access control (RBAC) engines.
- **Red Hat CloudForms**—Initially a cloud/OpenStack-focused management platform, the PaaS environment is expanding to support management of Microsoft Azure-based infrastructure services as well as Kubernetes-based container platforms.

4 FUNDAMENTAL USE CASES FOR OPENSTACK AND CONTAINERS

Use OpenStack and containers to:

1. ACCELERATE AND STABILIZE AN OPENSTACK ENVIRONMENT
2. DEPLOY APPICATIONS ON BARE METAL
3. CREATE CLOUD-NATIVE APPS ON VIRTUAL MACHINES
4. DELIVER HIGHLY AGILE MICROSERVICES WITH OPENSTACK

- **RightScale**—A well-established cloud/multicloud management platform; it was recently announced that RightScale will support management of Docker clusters across bare metal and in VMs.
- **Scalr**—A cloud management platform, available in open source, enterprise (on-site), and hosted PaaS versions. The platform can help organizations build automated workflows that leverage Docker containers and other container orchestrators such as Kubernetes and Mesos.
- **Platform9**—An OpenStack "source as a service" distribution that supports Kubernetes orchestration.

Containers and OpenStack

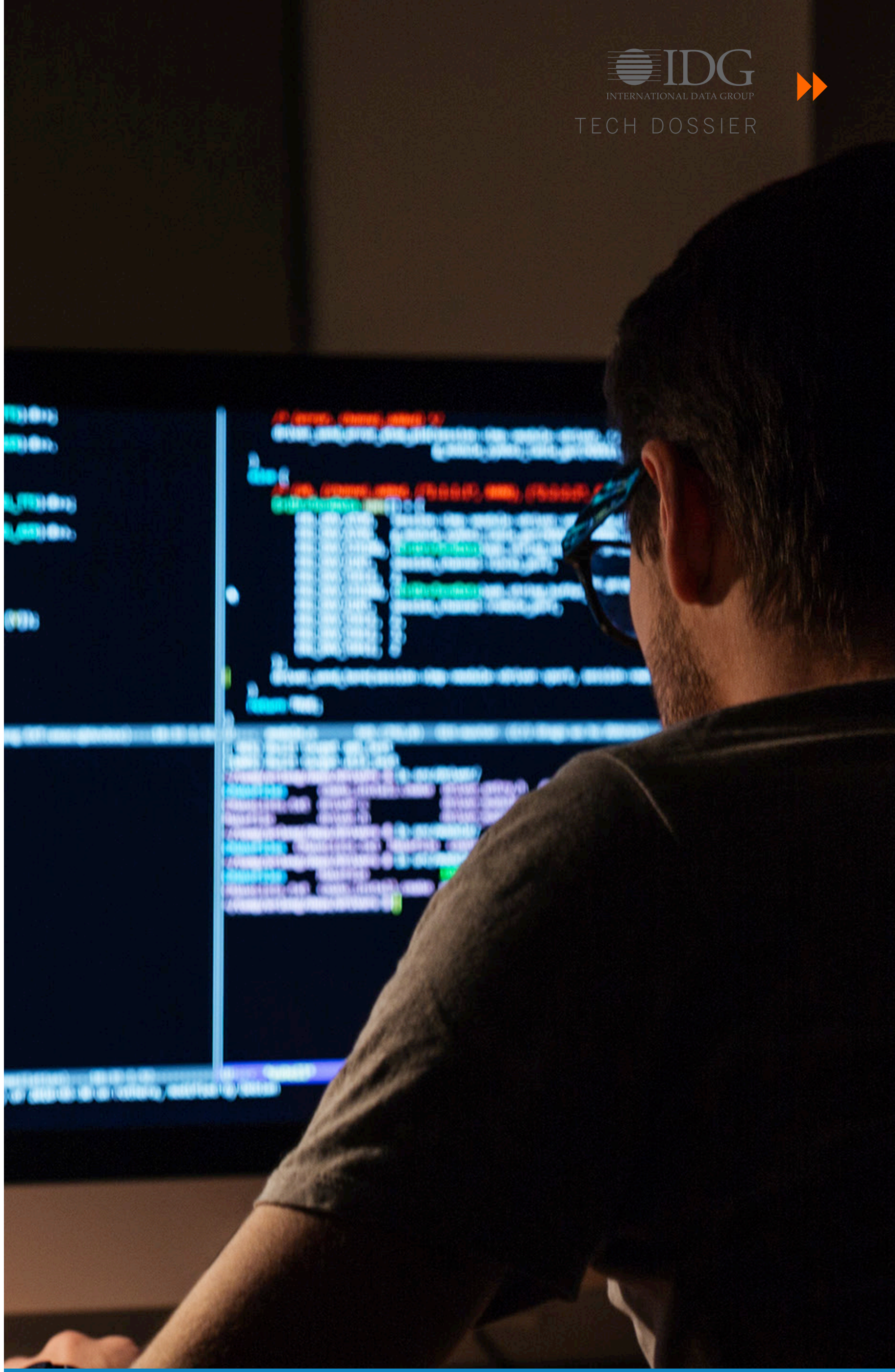
As mentioned previously, all leading PaaS and cloud platforms support containers, including the popular OpenStack initiative. Given the power and attractiveness of containers, it was only natural for the OpenStack community to embrace this development and deployment technology. There are four fundamental use cases in which OpenStack and containers can now interact:

1. OpenStack services can be implemented on containers to accelerate and stabilize the deployment of a highly efficient OpenStack environment.
2. Containers can be used to deploy applications on bare metal.
3. Containers can be deployed on VMs to create cloud-native applications.
4. Containers can be used to deliver highly agile microservices with OpenStack.

Two of these four container use cases are likely to prove particularly valuable to a wide range of organizations: deploying applications on bare metal and delivering microservices. The OpenStack community has distinct projects for each of these scenarios.

OpenStack Magnum

OpenStack Magnum is designed to offer container-specific APIs for multitenant "containers as a service." It provides container-specific features that go beyond the scope of OpenStack's core compute API— known as Nova. Containers started by Magnum run on top of resources called bays (essentially, collections of Nova instances).



Given the power and attractiveness of containers, it was only natural for the OpenStack community to embrace this development and deployment technology.

The containers can either run within VMs or, to speed performance, run directly on bare metal.

Magnum's central function is to manage so-called container orchestration engines, which are more fully discussed below. (You can learn more about Magnum here.)

OpenStack Zun

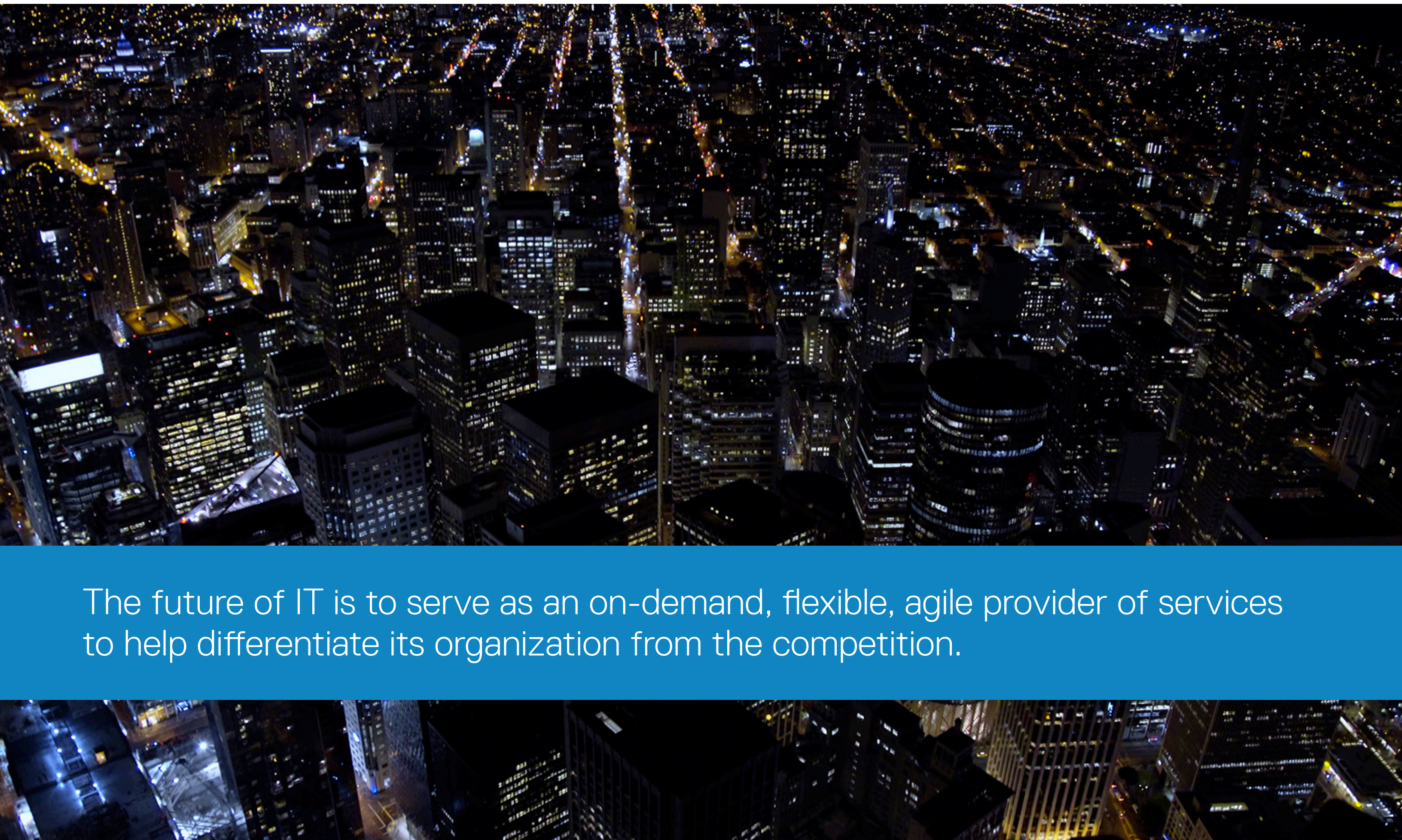
For the management of individual containers—including those functioning as microservices—OpenStack has launched a new project called Zun. OpenStack Zun is a container management service that aims to provide an OpenStack API for launching and managing containers created with different technologies. As envisioned, Zun will allow organizations to gain the many benefits associated with a microservices architecture. (You can learn more about Zun here.)

Realizing the microservices vision

As outlined in this paper, containerizing applications is a rapidly developing and evolving technology, and containers will play a significant role in the future of IT. Containers, microservices, and virtual machines will co-exist, as each provides a combination of benefits and trade-offs.

For example, VMs offer better application runtime isolation than containers can deliver. On the other hand, containers are lighter weight than VMs, so can be deployed more dynamically. Container management tool providers have developed toolsets that help in this process. Some organizations use PaaS toolsets such as Red Hat OpenShift or Pivotal Cloud Foundry that inherently produce containerized applications. It is also possible, albeit somewhat complicated, to take an existing application and simply "wrap" it in a container to realize some of the IT operational benefits containers provide.

Successful businesses are now engaging customers through innovative applications delivered from the cloud and accessed by mobile devices as well as traditional PCs. Containers and microservices will be the foundation for



The future of IT is to serve as an on-demand, flexible, agile provider of services to help differentiate its organization from the competition.

these cloud-based applications. The future of IT is to serve as an on-demand, flexible, agile provider of services to help differentiate its organization from the competition. IT must embrace the modern data center, break down traditional silos of complexity, and streamline operations to keep pace with today's on-demand business culture. Support for containers is a key priority for all leading cloud and PaaS platforms, and we can expect to see cloud, containers, and microservices all playing significant roles in the future of IT.

This "IT as a service" trend raises some tough questions for IT departments. Should they focus on integrating all the technologies and components required to build modern infrastructure and new cloud-native application delivery systems? Or should they focus their effort, time, and resources on aiding business innovation? If the latter, IT may decide to buy and deploy engineered solutions that provide the speed, agility, and simplicity required to compete in today's digital economy. Successful IT organizations will target investments in areas where they can derive differentiation against their competitors.

Clearly, containers and microservices have the potential to radically transform how applications are built, deployed, and managed. Such a fundamental shift—from monolithic, inflexible applications to new, granular microservices architectures—will not happen overnight. With the maturation of container and microservices technology, the modernization of software code is certain to accelerate rapidly in the coming years.

IT MUST EMBRACE THE MODERN DATA CENTER,

break down traditional silos of complexity, and streamline operations to keep pace with today's on-demand business culture.

For additional information, visit:
www.dellemc.com

For details on Cloud Foundry, visit:
www.cloudfoundry.org

For details on OpenStack, visit:
https://www.openstack.org

For details on the Open Container Initiative, visit:
https://www.opencontainers.org

DELL EMC