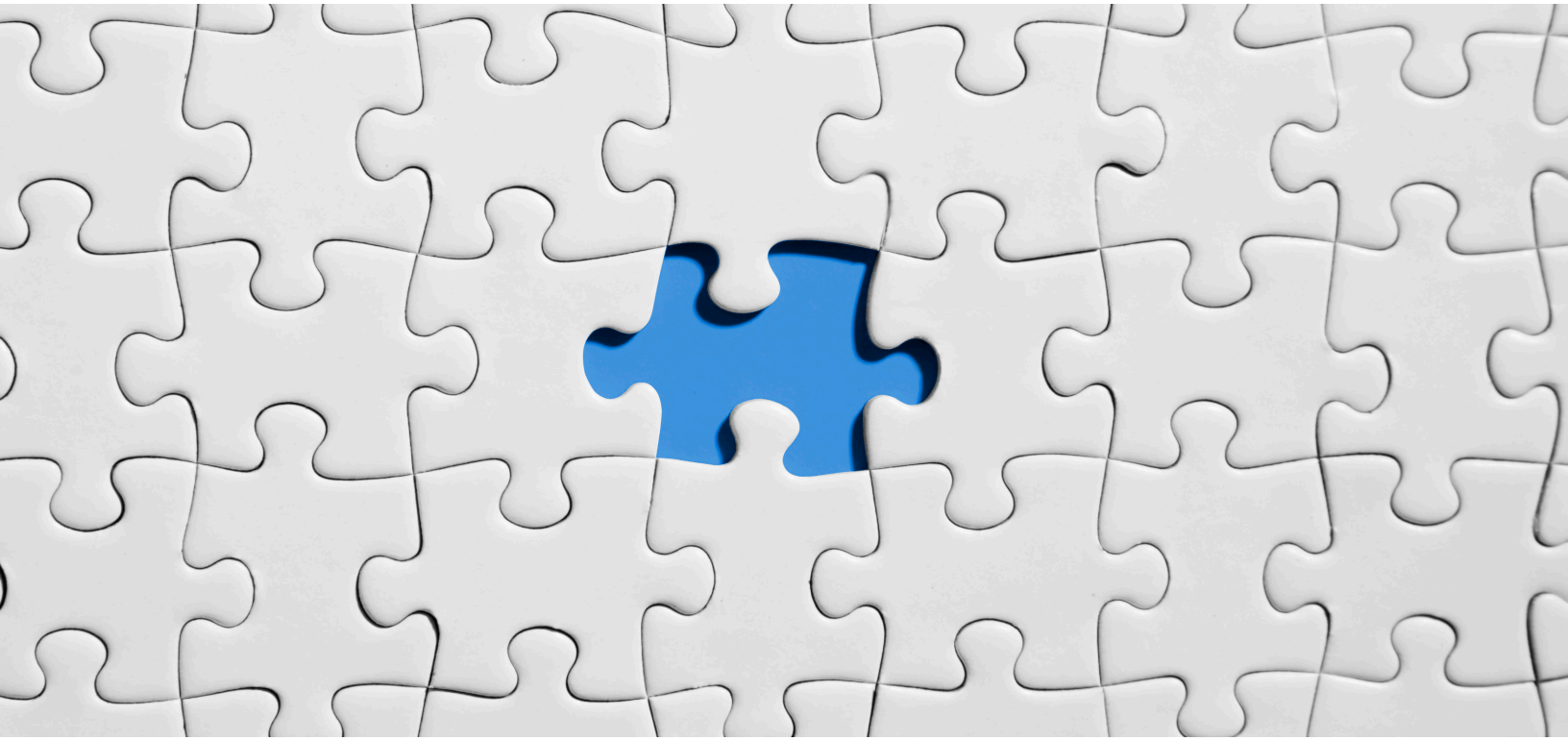
DELL Technologies

FUTURE OF CONTAINER-NATIVE STORAGE



Sarthak Sharma

Associate Sales Engineer Analyst

Dell

Sarthak.sharma@dell.com

Abhisek Sharma

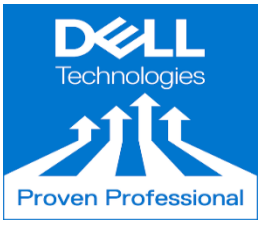
Senior Sales Engineer Analyst

Dell

Abhisek.Sharma@dell.com

Proven Professional

Knowledge Sharing Article



The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across multiple technologies and products.

From Associate, entry-level courses to Expert-level, experience-based exams, all professionals in or looking to begin a career in IT benefit from industry-leading training and certification paths from one of the world's most trusted technology partners.

Proven Professional certifications include:

- Cloud
- Converged/Hyperconverged Infrastructure
- Data Protection
- Data Science
- Networking
- Security
- Servers
- Storage
- Enterprise Architect

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

Learn more at www.dell.com/certification

Table of Contents

Disclaimer: The views, processes or methodologies published in this article are those of the authors. They do not necessarily reflect Dell Technologies' views, processes or methodologies.

What's Wrong with Traditional Storage?

Storage appliances weren't designed for the agility, speed, and scalability that enterprises demand today. While you could simply plug a traditional storage appliance into a container platform, an antiquated storage platform can hold you back from reaping the benefits of containerizing your applications.

Software-defined storage (SDS) separates storage hardware from storage controller software, enabling seamless portability across multiple forms of storage hardware. using appliances or typical SAN/NAS, you can't slice and dice storage as easily, readily, or quickly as you can with SDS.

What is Container as A Service?

According to searchitoperations.techtarget.com, within the spectrum of cloud computing services, CaaS falls somewhere between Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). However, CaaS is most commonly positioned as a subset of IaaS.

We often talk about vSphere container and its application in VMware. Basically, a container is a sandbox for a process. All the process that reside within the sandbox are visible to each other. We can run multiple versions of the same applications which require different versions of libraries. Container lifecycle lasts if the process which resides within it starts/lasts. Container as a Service could be explained as a cloud-level service that allows you to manage such containers as mentioned above. You can deploy, manage, run, scale or stop containers using API calls or Web interface.

Running a container allows us to focus on specific tasks. Multiple synchronized containers can be used to build a *microservice*, a crucial software development strategy. Independence of containers and the ability to run in isolation with minimal or zero associativity significantly lowers the risks and costs involved in upgrades/updates. Since all those happen at an image level, the chances of a container being confused with the VM is quite high. When virtual machines are abstracting the physical hardware, turning one server into many servers, a container is used to package code independencies. The virtual machines would be the size of somewhere close to Gigabytes in storage and memory as each instance of it would contain the entire OS with all its libraries and application binaries, while a container would be in Megabytes.

One thing that challenges the existence of containers more often is the security features. If the host kernel is affected by a flaw, the containers deployed over the kernel will face security concerns as well. Containers allow us to move applications seamlessly across the servers. Orchestration comes into place when handling multiple container. They allow provisioning hosts, rescheduling failed containers, linking containers through agreeable sources and scaling out the cluster by removing the containers. Management of containers is done by orchestration platforms like Kubernetes by Google or ECS by Amazon (EC2 as a Container Service).

Application-specific requirements

Applications have specific requirements from an architectural, security, and performance perspective. Some of these requirements will have an impact on the level of effort required to migrate the application into a container or break it apart into multiple containers.

Architecture

From an architectural perspective, moving applications into containers is not unlike a Unix to Linux migration, or an operating system upgrade. Often, an application will have been running for years. The documentation will often be nonexistent or out of date. As with most migrations, the technical person performing it will have to do the work necessary to understand how the application works structurally. They will have to reverse engineer how it was set up. At a minimum, they must be able to answer questions like:

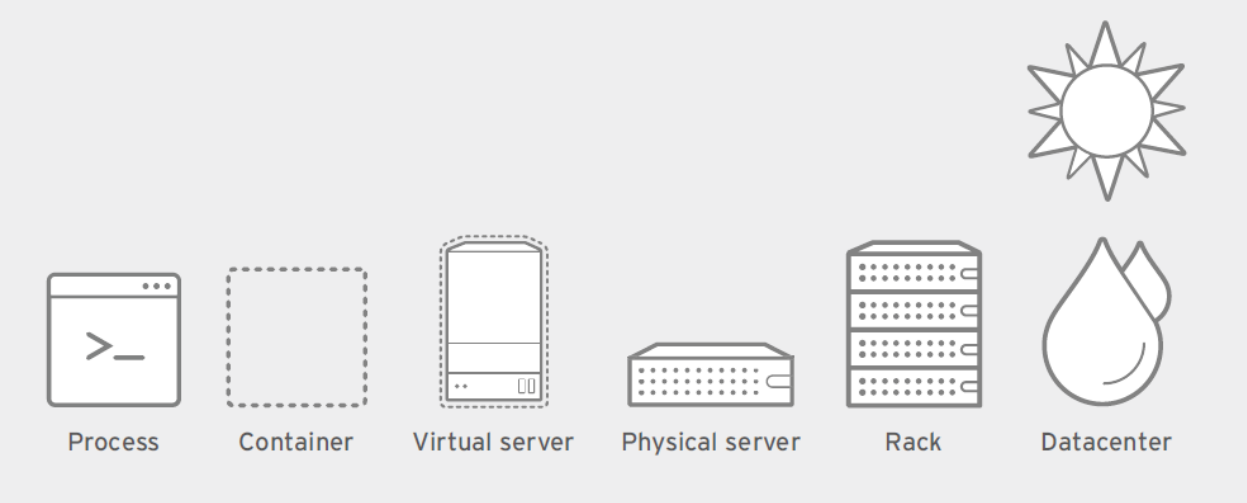
1. Where are the binaries for this application? Are they installed through an installer that puts them in a single place, or are they spread throughout the file system? Is there a single binary that is easy to start, or does it have a simple system unit file that can be used?
2. Where does the data for this application reside? Is it read-only or read-write? Can it safely be written to by two concurrent processes?
3. Where does all the configuration data reside? Is it in a single directory, single file, or multiple places throughout the file system?
4. What kind of secret data does this application have? Can the location of secrets be configured in the application? Can they be moved into separate directories, or can they be accessed using key through an identity or certificate server?
5. What kind of network access does this application need? Is it simple HTTP? Is it name server, which will require user datagram protocol (UDP)? Or, is it a complex application that needs point-to-point encryption between containers using something like internet protocol security (IPsec)?
6. Is the installer a shell script that can be reverse engineered for more information about application setup? Are the binaries installed through RPMs or some other kind of package manager?
7. Does the application's licensing allow you to easily distribute the application inside of a container image? Sometimes licensing is very restrictive; other times you may have a site license.
8. Does the application restart easily? Apache can restart thousands of times without failing, but a database may corrupt the tables. Could this make it harder to orchestrate and recover?

Answering these questions determines if your application is even a good fit for container migration – if the difficulty level is too high, it will not provide a return on investment.

Security

In many ways, security decisions for containerized applications are no different from regular applications that run in processes. You must decide what level of isolation is enough for the given application. When evaluating a workload for migration, and deciding whether containers offer enough isolation, it is important to review how much isolation the workload has where it is currently running.

Going from left to right in the Figure below, each technology decision offers increasing isolation. For some applications, regular Linux processes offer enough isolation. It is common to run MySQL and a web server on the same Linux operating system instance. At the other end of the spectrum, there are times when two copies of an application need to live in two different data centers, which are affected by different weather and earthquake patterns – this is common for disaster recovery.



As an example, high-performance computing (HPC) workloads are commonly run today in large clusters with only regular Linux process isolation. This is not perfect – researchers in a large cluster could attempt to hack each other's processes, but this is commonly determined to be an acceptable level of risk.

Performance

Workloads must also be analyzed for performance. Containers and virtualization are additive technologies that can be combined with bare-metal hardware to provide features and capabilities. Table 2 offers a quick guide to help determine important capabilities that should be considered when migrating applications into containers.

Containers are Linux processes that use technologies such as control groups (cgroups), Security-Enhanced Linux (SELinux), and namespaces to provide a higher level of isolation to applications. This allows them to run at native or near-native speed. There is no layer of abstraction as found with virtualization, so all the containerized applications in a cluster must be based on the same hardware architecture and operating system.

Using the same example, in an HPC environment, containers may be added to bare metal to provide a similar level of performance, while increasing the level of isolation. On the other hand, if the workload is a corporate application that requires components that live in Windows and Linux, a combination of containers and virtualization may be a better choice. Running containers inside of virtual machines offers the combination of hardware freedom, better isolation, and increased manageability using container images

Figure 2 - Better understanding of the trade-offs of combining different technologies

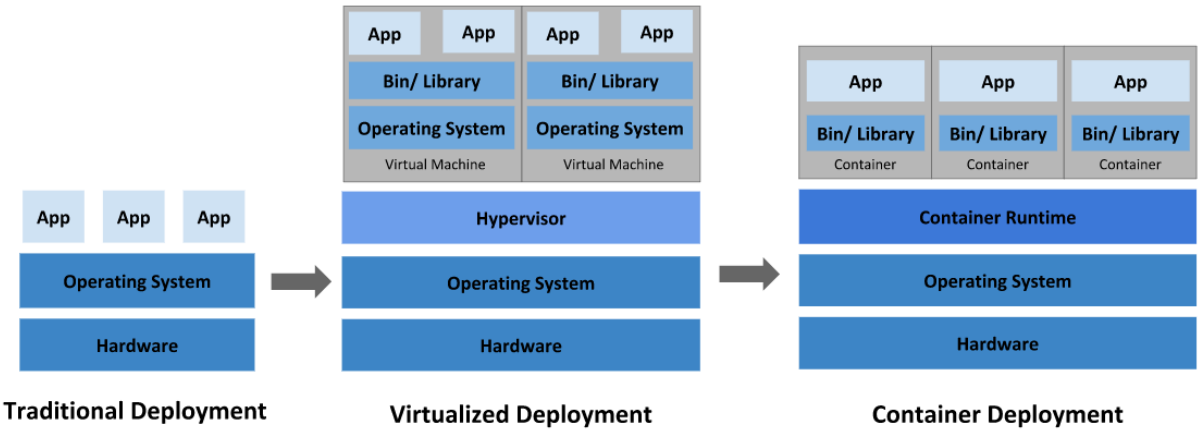
	BARE METAL	+CONTAINERS	+VIRTUALIZATION
CPU intensive	Fast	Fast	Fast
Memory intensive	Fast	Fast	Fast
Disk I/O latency	Fast	Fast	Medium
Disk I/O throughput	Fast	Fast	Fast
Network latency	Fast	Fast	Medium
Network throughput	Fast	Fast	Fast
Deployment speed	Slow	Fast	Medium
Uptime (live migration)	No	No	Yes
Alternative OS	Yes	Some	Yes

Google Kubernetes

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. Google open-sourced the Kubernetes project in 2014. Kubernetes builds upon Google's decade and a half of experience running production workloads at scale, combined with best-of-breed ideas and practices from the community.

Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, CPU, memory, process space, and more. Since containers are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions. Figure 3 shows different deployment models.



Why you need Kubernetes and what it can do

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by a system?

That's how Kubernetes comes to the rescue! Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. For example, Kubernetes can easily manage a canary deployment for your system.

Kubernetes provides:

- **Service discovery and load balancing**
  Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes can load balance and distribute network traffic so that the deployment is stable.
- **Storage orchestration**
  Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks**
  You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- **Automatic bin packing**
  You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

- **Self-healing**
  Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management**
  Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

  Kubernetes is not a traditional, all-inclusive Platform as a Service (PaaS) system. Since Kubernetes operates at the container level rather than at the hardware level, it provides some generally applicable features common to PaaS offerings, such as deployment, scaling, load balancing, logging, and monitoring. However, Kubernetes is not monolithic, and these default solutions are optional and pluggable. Kubernetes provides the building blocks for building developer platforms but preserves user choice and flexibility where it is important.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates the need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Amazon EC2 provides a web-based user interface, the Amazon EC2 console. If you've signed up for an AWS account, you can access the Amazon EC2 console by signing into the AWS Management Console and selecting EC2 from the console home page.

When you sign up for AWS, you can get started with Amazon EC2 for free using the AWS Free Tier.

Conclusion

Successfully migrating an existing application into a container, or containers, requires that you understand the application and develop a comprehensive plan. Almost any application can be containerized, but it is important to understand the amount of effort that will be required and ensure that the transition to containers preserves performance and maintains or improves security.