CONTAINERS & CONTAINERIZATION EXPLAINED

Containerization is a modern virtualization method that accesses a single OS kernel to power multiple distributed applications that are each developed and run in their own container.

A container takes its meaning from the logistics term, *packaging container*. When we refer to application containers we are describing a container for packaging software that deploys together, or in a single virtual shipment.

This is different from traditional approaches to virtualization where each application requires an operating system or an entire virtual machine to run on a server. Containers are versatile because they can function on bare-metal servers, cloud servers or a single virtual machine on a server. They transcend Linux by functioning on some Mac or Windows-based operating systems.

Containerization: How it Works

In containerization, containers hold all of the figurative nuts and bolts needed to run a program. This runtime environment could include files, libraries and environment variables that allow a program to be executable.

Because containers don't require their own operating systems to execute applications, they consume fewer resources in a server, virtual or cloud environment.

Apart from understanding the framework, it is also important to note the outcomes. An *image* is what results from the complete set of runtime components in a single container. And images are

deployed by a container onto a chosen host.

App containerization technology can be used on a number of engines by enterprise organizations. These include:

Docker

The most widely used platform is Docker, an open source container system based on runC. Docker images work on a number of "as-a-Service" platforms, making them more versatile than some of its competitors.

CoreOS'rkt

CoreOS'rkt, pronounced CoreOS "*rocket*", is a low-level framework that uses systems to create foundational applications. It's designed as the container engine that powers Google Kubernetes.

Google Kubernetes

Google Kubernetes has its own container engine, rkt, but it is also a community where users can run other popular engines. It's an open source host environment for creating libraries of applications to share or develop. Kubernetes is a good portable option, since it offers a full cloud server that can be accessed anywhere.

Amazon AWS

Amazon AWS offers Backend-as-a-Service (BaaS) that includes a Containers-as-a-Service (CaaS) offering to its customers. It also happens to be a widely used host for those looking to deploy Docker images. Like Kubernetes, AWS offers the portability of cloud computing.

Cloud Foundry

Cloud Foundry's Garden offers containerization as part of its Platform-as-a-Service options (PaaS). Garden is the container engine and Cloud Foundry is the host for developing, testing and deploying portable applications on a cloud server.

Benefits of Containerization to Enterprise Companies

As engines like Docker grow, the benefits of using containerization continue to increase for enterprise businesses. In the following section, we will examine the benefits of deploying software on a container engine.

Multi-Cloud Platform Technology

One of the most desirable benefits of using containerization as a virtualization method is that it can operate on the cloud. Many engines support multi-cloud platforms so they can be run inside platforms like Amazon EC2 instance, Google Compute Engine instance, Rackspace server and VirtualBox.

Testing and Continuous Deployment

Containerization gives enterprise businesses the flexibility to to build, test and release images to deploy on multiple servers. While consistency across environments sometimes fluctuates when it comes to development and release cycles, container providers like Docker are making it easier to ensure consistency no matter which environment a developer chooses to deploy on. This makes containerization a good option for those organizations using DevOps to accelerate application delivery.

Version Control

As this technology develops, one thing that container platforms must do to remain competitive is to ensure version control. Docker set the bar high, by offering simplified version control that makes it easy to roll back to a previous image if your environment breaks. Other competitors have followed suit, making this method of virtualization good for developers who need version control to be available at their fingertips.

Isolation

A key component of virtualization is isolation or the act of segregating resources for each application. A recent report by Gartner stated that some container engines now perform as well as virtual machines when it comes to isolation.

Security

Providers of containerization offer different ways to ensure security, but one thing is consistent across the board. If one of your containers gets hacked, applications running on other containers are not susceptible.

Versatile and Resource-Friendly Approach

Most developers like containers because they can benefit from a versatile, resource-friendly approach to software development. With a number of container engines to choose from, enterprise businesses can ensure all of their development and deployment needs are met while conserving server space, physical, virtual or cloud.

Portability

Because container applications can run on cloud servers, they are generally more accessible than other applications. Programming in containers offers a portable approach to software development.

Reproducibility

Containerization is a DevOps friendly approach to production because it offers the benefit of reproducibility. Each container's components remain static and unchanging from code to deployment. They create one single image that can be reproduced in other containers over and over again.

Containerization Versus Server Virtualization: What's the Difference?

When people talk about server virtualization, they are usually speaking of a virtual machine or VM. A VM creates a hypervisor layer between operation system, applications and services and memory, storage and the like.

This layer acts as its own VM creating a self-contained environment to run a single application. Each application that is virtualized consumes its own version of an OS. This requires enterprise businesses to have several versions of operating systems available if they want to develop in a virtual environment, and also requires the purchase of multiple licenses.

On the other hand, containers allow multiple applications to run on a single VM. This limits the number of software licenses an enterprise company must invest in to develop in a container environment. They do this by sharing OS kernels instead of requiring their own. For this reason, developing in a container is a more resourceful approach to enterprise software development.

DevOps Culture and Containerization

In an enterprise environment marked by the need for portability and versatility, both DevOps and containerization are growing in demand. This is fueled by the desire of businesses to adopt cloud technology that allows them to run their operations from the palm of their hands, as well as the continued growth in adoption of "*as-a-Service*" providers that package cloud services to simplify business operations and reduce cost.

While more businesses navigate the tricky waters of making sure all of their cloud resources are customized, integrated, automated and functioning as they should, the need for qualified DevOps professionals has skyrocketed. With more DevOps professionals usually comes a culture shift, one where applications are accessible and companies seek innovative solutions to mitigate rapid growth.

Containerization offers exactly what they are looking for -- a package for software contents that ship together. However, without DevOps, the luster of using containers wears off quickly as it is DevOps that automates and innovates the virtualization process.

Final Thoughts

Containers are perfect for any enterprise company that is looking to enhance digital enterprise management via solutions that offer reliability, portability, versatility and reproducibility in a virtual environment.

Companies making a digital transformation into DevOps culture should introduce containers carefully. The following steps are recommended when introducing containers:

- **Step 1:** Evaluate business needs. Practice running containers on a small scale and see how it fits into the business model and culture.
- **Step 2:** Pilot containers with your DevOps team.
- **Step 3:** In the Production phase, businesses deploy containers into their infrastructure.