White Paper

Cisco Application-Centric Infrastructure (ACI) and Linux Containers

What You Will Learn

Linux containers are quickly gaining traction as a new way of building, deploying, and managing applications in the cloud-enabled data center. Containers couple lightweight, high-performance isolation with security and make it easy to package services and deploy them in a flexible and scalable way. Cisco Application-Centric Infrastructure (ACI) was designed around the same goal of accelerating application deployment with security and scale. It provides a foundation for connecting physical and virtual environments without compromise. Cisco ACI can also be used to accelerate the transition to an environment that brings together applications across virtual machines, and bare-metal servers, **and** containers

Introduction to Linux Containers

Over the past decade, hypervisor-based virtualization has become a dominant data center technology. The concept is simple: Create a virtual implementation of the hardware of a physical computer so that it can run an entire, unmodified operating system within it. The result, a virtual machine monitor, provided a powerful tool through which to consolidate multiple servers and take better advantage of additional physical computing resources. Hypervisors, however, introduce additional workload-dependent overhead as they faithfully replicate true hardware behaviors. In addition, hypervisors also bring portability considerations as to where the application(s) can run: on premises, public cloud, private cloud, bare metal, mixed environment, etc. For many workloads, the potential savings in capital and operational expenses are often enough to justify the use of a virtual rather than dedicated physical environment, though.

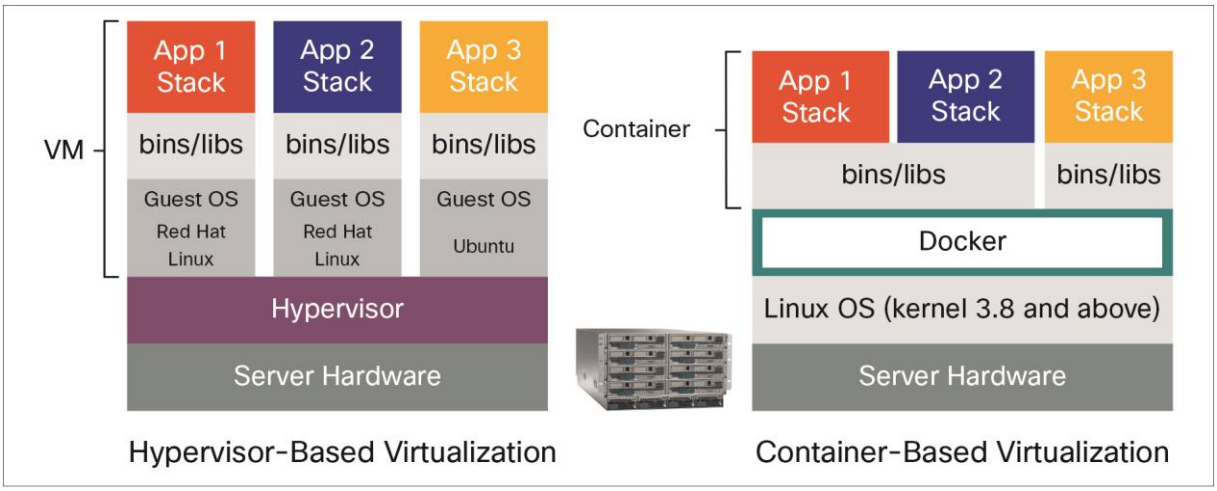
Over the past few years, several developments in the Linux kernel have enabled a new form of virtualization called containers. Containers do not introduce a layer of virtual hardware and do not support running alternative kernel code. However, they provide self-contained execution environments, effectively isolating applications that rely on the same kernel in the Linux operating system.

Containers are built around two core mature Linux technologies:

- Namespaces: Kernel namespaces isolate containers. Namespaces' main outcomes are avoiding visibility between containers and containing faults. Some of the domains of isolation provided by Namespaces are:
  - pid (processes)
  - net (network interfaces, routing)
  - ipc (System V interprocess communication [IPC])
  - mnt (mount points, file systems)
  - uts (host name)
  - user (user IDs [UIDs])
- Control groups (cgroups): Control groups offer mechanisms for measuring and limiting resource usage for groups of processes, such as CPU, memory, and I/O. This helps containers stay within the resource footprint allocated to them.

While a handful of other components are involved as well, these primitive technologies allow you to run multiple environments on a Linux host with strong isolation between them, while bringing efficiency and flexibility.

Figure 1. Hypervisor-Based Virtualization Compared with Container-Based Virtualization



One of the key technologies supporting the rise of containers is Docker. Docker is a popular open-source platform for building and running containers and offers several important features:

- Portable deployment across machines: Container images can be run on any Linux host.
- Versioning and reuse: Docker offers Git-style capabilities for tracking versions and creates layers as additional software is added to it. For example, you can create a base image for a container and build multiple different images from it.
- Sharing: Docker offers a searchable public registry for container images.

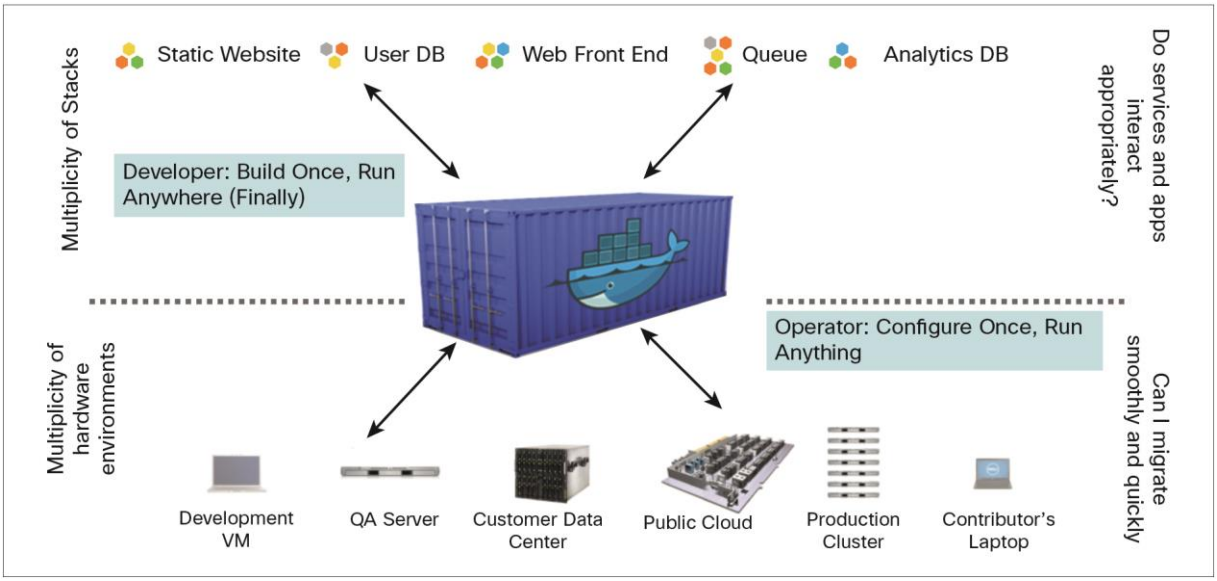
Comparing Containers and Virtual Machines

Containers and virtual machines are different technologies. Each has its strengths and weaknesses. One of the key advantages of containers is performance because containers introduce much lower overhead than a virtual machine. Using containers, you won't need to virtualize hardware or boot a new "guest" operating system. A container is only isolating processes running inside the Linux OS. Containers can boot in milliseconds and offer stable, near-native performance across workloads. Virtual machines, conversely, may take seconds to minutes to boot and performance can vary widely based on the software that is run.

Another key differentiator between containers and virtual machines is the amount of required storage. In a virtual machine environment each application requires a unique installation of the guest OS. In a container environment, only one Linux OS is required (Figure 1). This could translate in running 10 to 100 virtual machines compared with 100 to 1000 containers on the same physical server.

Containers make applications easily portable in any environment, making them very attractive and useful for companies embracing the Developer/Operator (DevOps) model (Figure 2 [see also a recent Cisco blog post]).

Figure 2. Developer/Operator Model



Containers do have one limitation when compared with virtual machines. They do not allow a user to run **different** operating systems, such as Microsoft Windows or legacy DOS applications. They run all containers on a single operating system kernel, so all applications and software running within these containers must be supported by that kernel.

Container Lifecycle Management

As container technology becomes popular, more customers are asking how to manage a large number of containers at scale. For example, deciding how and where to schedule new containers, scale services, and manage physical resources all become important issues. Some of the key open-source solutions in this space include:

OpenStack

OpenStack offers extensions to launch and manage containers, as it does for virtual machines. In particular, there is a nova compute-driver for Docker just as there is for KVM or Xen. This allows OpenStack nova to launch Docker images that have been imported into Glance. Docker images can also be launched via the nova CLI as well as through Heat or Horizon.

Mesos

Apache Mesos is a cluster management tool used by several web-scale companies, including AirBnB and Twitter. Mesos was designed to offer an extremely flexible platform for scheduling and managing containers in a cluster.

Kubernetes

Kubernetes, a project started by Google, offers a system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications. Kubernetes offers a number of abstractions for grouping containers and mapping them to sets of tasks.

Cisco is working with the open-source community to continue to develop these tools and to help ensure their integration with ACI.

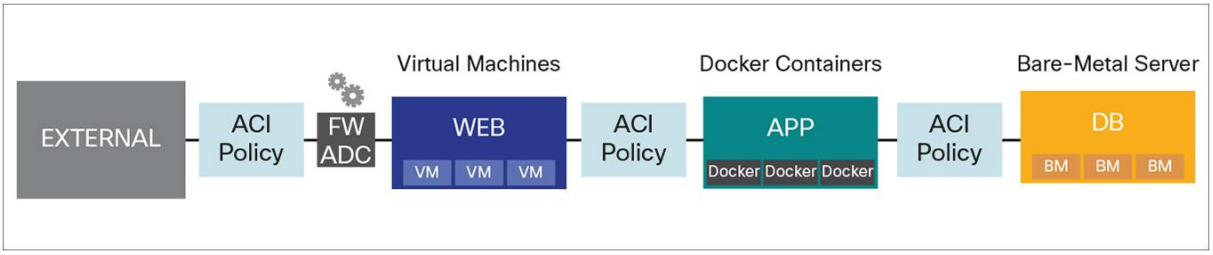
ACI and Linux Containers

Cisco Application-Centric Infrastructure (ACI) was designed to offer a common policy model for managing IT operations that can span across the entire infrastructure. ACI is agnostic to the form factor on which applications are deployed, which allows it to support any environment, including bare metal, virtual machines, and containers. ACI portability is native and part of its nature, which is a natural fit with container technology. When you use ACI, you will gain access to a unified policy language that offers a clear security model regardless of how an application is deployed. You will also have a clear path to migrate existing workloads towards container-based environments without any changes to network policy. Two key technologies make this possible: ACI Policy Model and OpFlex + Open vSwitch (OVS).

Application-Centric Policy between Containers

An application-centric model describes how components of an application interact, including their requirements for connectivity, security, quality of service, and network services. By looking closely at this model (Figure 3), you can see that it was designed to support virtualization via hypervisors or containers. The model enforces policy across an abstraction called an endpoint group (EPG), which is a collection of network endpoints. Network endpoints can be represented by a broad range of entities, including bare-metal servers, virtual machines, and containers. They are agnostic to the underlying technology. As a result of this flexibility, ACI is already in a position to define application-centric policies across a diverse infrastructure, which may include Linux containers.

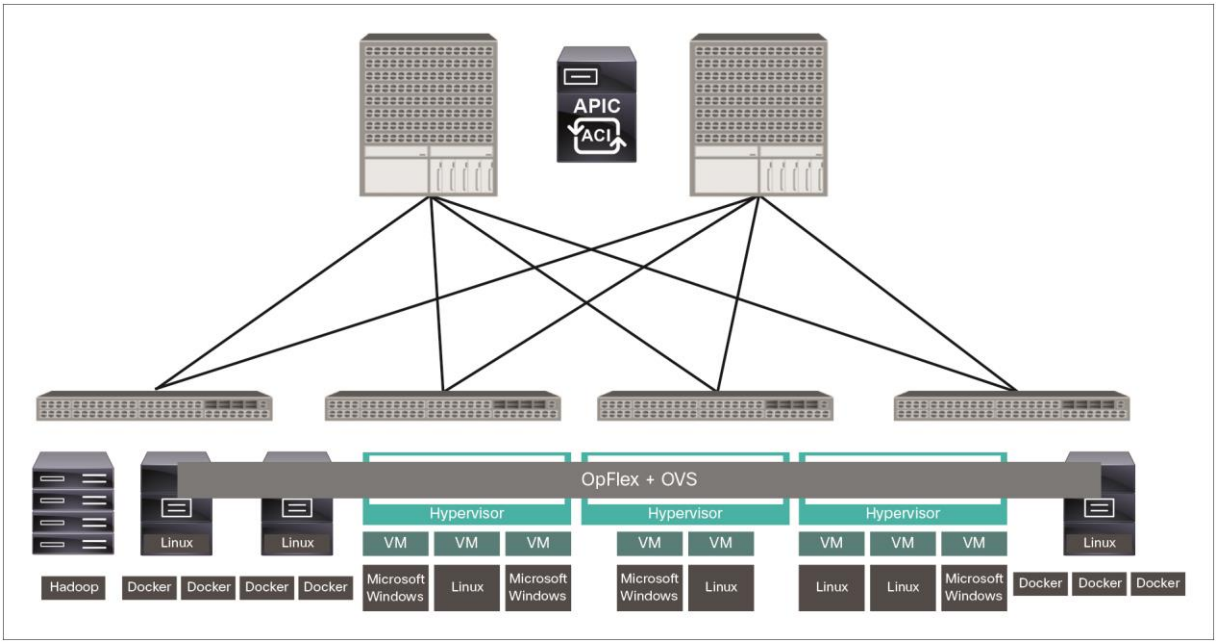
Figure 3. ACI Policy Example



OpFlex and OVS Enforce Policy between Containers

Cisco has also announced the OpFlex protocol. We are working with the open-source community to build OpFlex support for Open vSwitch (OVS). OpFlex offers a distributed policy protocol that allows application-centric policies to be distributed and enforced within a virtual switch such as OVS. With an OpFlex-enabled OVS connected to ACI, you can enforce policy across both virtual machines and containers uniformly. Each container attaches to an OVS bridge, just as a virtual machine would, and an OpFlex agent helps ensure that the appropriate policy is established within OVS. The result is a complete infrastructure that can span across physical, virtual, and container-based environments. Cisco plans to release support of OpFlex, OVS, and containers in the second calendar quarter of 2015 (2Q2015).

Figure 4. Policy Enforcement across Virtual, Physical, and Container-Based Environments



Summary

Built on mature Linux kernel technology, containers are gaining adoption in the cloud-enabled data center. ACI is designed to help facilitate this transition and support an environment that spans across physical machines, virtual machines, and Linux containers through a unified policy model.

For more information about Cisco solutions for Linux Containers with RedHat, read this white paper.

Contributors: Michael Cohen, Director of Product Management, and Carlos Pereira, Distinguished Systems Engineer.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C11-732697-00 09/14