Dave Blocks

Dev tips and tutorials

April 7, 2017 by oquidave | Uncategorized

Linux Virtualization and containers: Virtualbox, KVM, Xen, LXC and Docker

A while back, things used to be easy. A personal computer or server comprised of hardware components bundled with

an operating system installed on a hard drive and that was pretty much it.

And then the cloud happened.

Things got slightly more interesting and complicated too. As the need to setup isolated application and server environments without increasing computing costs grew, virtualization become very popular.

Virtual machines and Hypervisors

Instead of purchasing and installing server on every other hardware unit, you could actually install a complete machine — called a virtual machine — inside another machine often called the host.

So this piece of software that enables this kind of magic is termed a “hypervisor”. A hypervisor or virtual machine monitor (VMM) is computer software, firmware, or hardware, that creates and runs virtual machines ~ Wikipedia.

So this software can run on the host’s operating system or it can be installed on computer hard disk like an operating system. Those that are installed like an OS are called bare metal hypervisors and a good example is Microsoft Hyper-V or VMware ESX/ESXi.

However, you might be more familiar with hypervisors that run the host OS, aka hosted hypervisors. These are really easy to setup and use and they include; VMware

Workstation, VMware Player, VirtualBox, KVM, Parallels Desktop for Mac and QEMU.

I've personally used Vmware Player and mostly virtual box to spin up a number of Virtual Machines(VM) on my personal laptop.

Virtual box

Virtual box needs no introduction if you have played with computers for a while. Its open source and owned by **Oracle**.

Virtualbox is multi platform as it runs on Windows, Linux, Macintosh, and Solaris hosts and supports a variety of Guest OSes from Windows, Linux families, and others, like Solaris, FreeBSD and DOS. That's what makes it awesome. However, it's not included in the mainline Linux kernel, so you have to compile and insert the respective kernel module.

It's also very easy to use since it comes with a cool intuitive interface for creating, cloning, snapshotting and destroying VMs. If you setting up a development environment and wish to isolate web, database and application servers very easily, then you can use virtualbox to create Vms for each of those servers on a single host machine.

Windows developers who wish to spin up Linux development environment always use Virtualbox on top a Windows OS to run Linux distros like Ubuntu for the LAMP stack for instance.

Further reading;
https://www.virtualbox.org/

KVM

KVM is another bull in kraal that's recently been making waves in the cloud computing space. KVM does not perform any emulation itself, but it exposes the /dev/kvm interface, by which an external user space host can do emulation. QEMU is one such host, so you will often see KVM referred as KVM/QEMU.

Coming from Red Hat and zapped into mainstream Linux, it's used by many cloud providers like Linode and Digital ocean among others. It's majorly used by enterprise users specifically Virtual Private Server/VPS or Cloud VM providers to provision and sale VPS to clients.

VirtualBox and VMware will install on most machines with x86 processors, but KVM requires that the processor support Intel-VT or AMD-VT extensions, and that those extensions are enabled in the BIOS. KVM comes with VirtManager which is a tool for managing VMs. KVM is also used by Openstack which is known for building and managing cloud computing platforms for public and private clouds.

I Installed KVM on my machine once, but never really got hooked. KVM has a slightly steeper learning curve, less documentation and support and only runs on Linux.

However, KVM is highly scalable. It employs advanced security features, utilizing SELinux. It provides MAC (Mandatory Access Control) security between Virtual Machines. KVM has received awards for meeting common government and military security standards and for allowing open virtualization for homeland security projects.

Further reading;

- http://www.linux-kvm.org/
- https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

XEN

Another hypervisor that's traditionally more tied to Linux is Xen. Xen is older than the rest of the hypervisors. It comes from the XenServer project, OracleVM, or any number of OSes like CentOS, Suse, Ubuntu and NetBSD which have Xen built in. It's currently used by cloud giant Amazon AWS to manage its fleet of millions of linux servers.

If you want to dig deeper, full Hypervisor comparison can be found here

Now the thing about Hypervisors discussed above is that virtual machines created require a separate kernel instance to run on. So it's possible to have host os kernel running on a completely different version than the guest or VM kernel. This is great because you have complete application environment isolation and possibly higher security.

However, there are also performance bottlenecks since a complete operating system with its own kernel is needed.

This is where Linux containers come in.

LXC/LXD

LXC, or Linux Containers are the lightweight and portable OS based virtualization units which share the base operating system's kernel, but at same time act as an isolated environments with its own file system, processes and TCP/IP stack.

So containers don't create the whole Operating system or Kernel again. And multiple containers share the same kernel, and each container can be constrained to only use a defined amount of resources such as CPU, memory and I/O.

They can be compared to Solaris Zones or Jails on FreeBSD. As there is no virtualization overhead they perform much better than virtual machines. ~ Nedo @ stackoverflow.

Containers were listed by Linux.com as one of the top 5 next-generation cloud technologies in 2016.

I have used Linux containers mainly on remote VPS on AWS or Linode and I was simply amazed at how powerful they are. While Linux containers won't give you the full power of VMs created by Hypervisors, they have the benefit of having a small memory and processor footprint. I've been able to spin up 3 LXCs on a 1GB VPN on AWS and they all worked just fine. You can't do that with virtualbox or KVM.

LXC package on Ubuntu specifically is really great because it automatically setups networking between host OS and containers — something that usually trips newbies.

LXD meanwhile is just the new LXC experience with an intuitive user experience and a single command line tool to manage your containers. Containers can be managed over the network in a transparent way through a REST API. It also works with large scale deployments by integrating with OpenStack.

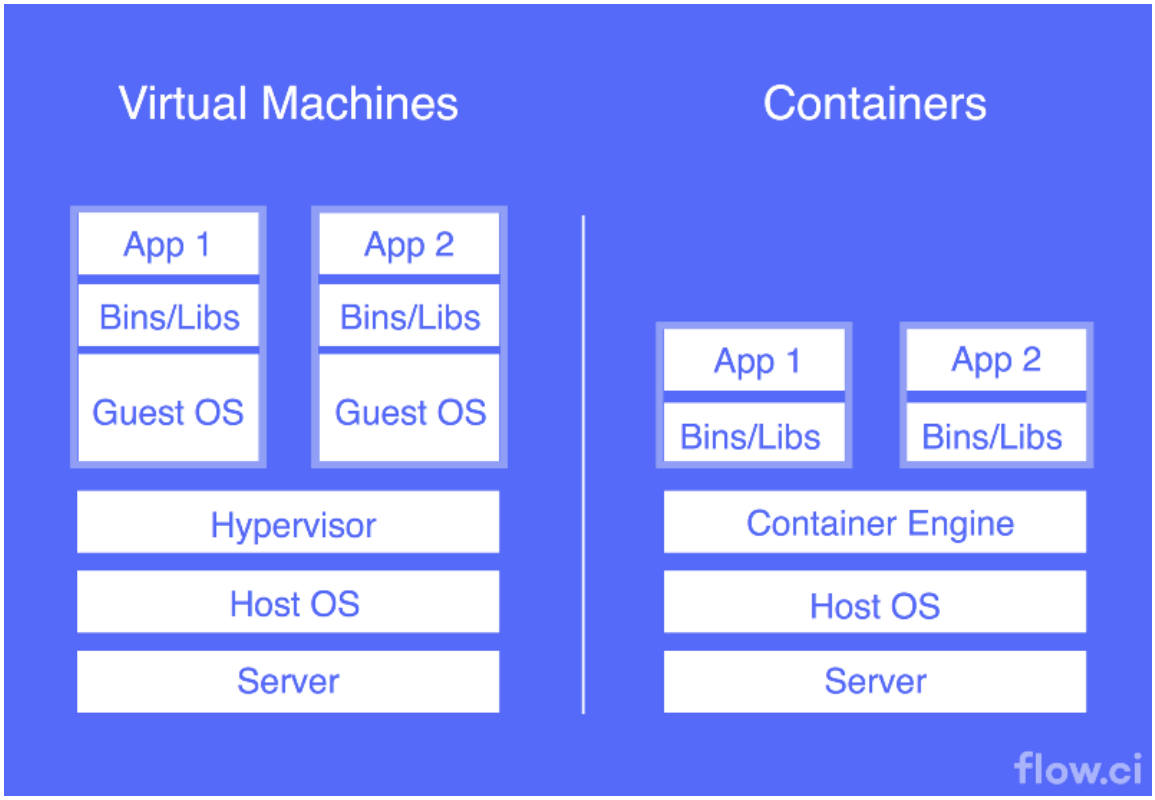
Difference between Hypervisors and Containers

Hypervisors: are based on emulating hardware
Containers: are about virtualizating the operating subsystems(file system, networking etc).

Containers: Single Kernel
Hypervisors: Multiple Kernels

Containers: Easily scalable and more resource efficient.
Hypervisors: Require more resources and less scalable.

update: I found this article on medium by flow.ci which gives better and detailed information on the difference between virtual machines which run on top of hypervisors and containers. So I'll copy the except as-is.



- Virtual machines contain a complete operating system and applications. Hypervisor-based virtualization is resource intensive, a VM can take up several GB depends on the guest the OS.

- Virtual machines use hypervisors to share and manage hardware while containers share the kernel of the host OS to access the hardware.

- Virtual machine have their own kernel and they don't use and share the kernel of the host OS, hence they isolated from each other at a deep level.

- Virtual machines residing on the same server can run different operating systems. One VM can run Windows while the VM next door might be running Ubuntu.

- Containers are bound by the host operating system, containers on the same server use the same OS.

- Containers are virtualizing the underlying operating system while virtual machines are virtualizing the underlying hardware.

Further reading;

- https://lwn.net/Articles/531114/
- https://lwn.net/Articles/671722/
- https://www.docker.com/
- https://runc.io/
- https://containerd.tools/
- https://github.com/coreos/rkt

Docker

This is probably the hottest newcomer on the virtualization space. Mainly used by developers and devops guys, docker uses LXC containers to make it easier to create, deploy, and run applications. While it's the most popular container technology, there are several Docker alternatives and they include; LXC/LXD, Solaris Zones, RKT by CoreOS and BSD Jails.

Developed by dotCloud, a platform-as-a-service company now renamed Docker Inc, Docker has grown become a go-to tool for developers who wish to to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Usually developers have a development environment where they write and test their code — normally on their local machine or development server. And then there's a separate production server which runs the live app.

When the application is tested and ready for deployment, the code is pushed to production server. Problems usually arise because the development and production environments aren't the same. It's even common to develop and test on one operating system while the production server is running on another. This has been cause of lots of frustration for developers.

Or are you familiar with scenario where your app works on your computer but doesn't work on your friend's PC?

Docker solves this problem by allowing developers to package their apps as containers and simply deploy the container to production server with all the configs it needs.

Personally I have tested it out, but never really got as hooked as I was to LXC. Probably that's because am a systems guy, so I need more flexibility with my containers/VMs which in my opinion LXC gives better than Docker does. But this is just a matter of preference.

The other reason I have really not used Docker to for my development workflow is because of slow internet links we have in Uganda. A basic docker Image can roughly be 300-500MB and pulling that over the internet can be very painstakingly slow process on ~ 1Mbps link. So expensive

data plans and slow internet connectivity keeps most African devs away from tools like Docker.

Cloudy with chance of containers: Container Orchestration

Because several containers can be installed on a single server, they can become hard to manage eventually.

This calls for container engines or container Orchestration tools which take away the headache of managing these containers. These tools have a number of unique services that are used to manage the container's workload and direct communications across the system.

Examples include;

- Amazon ECS — The Amazon EC2 Container Service (ECS) supports Docker containers.
- Azure Container Service (ACS) — ACS is an Azure Compute resource provider to create and manage a cluster of virtual machines that act as container hosts.
- CoreOS Fleet(deprecated) — CoreOS fleet aggregates individual machines into a single pool of resources. Instead of running a service on a specific machine, services are submitted to the fleet cluster manager, which decides where they should run.
- Cloud Foundry's Diego — Diego is a self-healing container management system that attempts to

keep the correct number of instances running in Diego Cells to avoid network failures and crashes.

- Docker Swarm (now incorporated into Docker) — Docker comes with clustering to run apps at scale and sophisticated scheduling and monitoring to build highly available and fault tolerant services.
- Kubernetes — Kubernetes, built by Google is an open-source system for automating deployment, scaling, and management of containerized applications. It's by far the most popular container management tool.
- Mesosphere Marathon — Marathon is a production-grade container orchestration platform for Mesosphere's Datacenter Operating System (DC/OS) and Apache Mesos.

My prediction is that containers might replace your traditional virtual machines at least in the cloud space very soon because of their lightweight and resource-efficient nature.

Hopefully, this post has helped clear-out all the dust around virtualization technologies and buzzwords.

**Quick one folks: For the last 7 months, I have been working on this cool project that automates website uptime monitoring, checks SSL certificate for errors and Expiry, monitors domain expiry and cron jobs. You probably want to keep tabs on these if you are running a website or app in production. So Please Sign up to Site Monki today for free.
**

Image: flinthamcabins.com

Share this:

Related

What I am learning Q3 2018: Docker, Kubernetes, Golang and RabbitMQ
July 3, 2018
Similar post

7 ways of transferring files from remote server to localhost
January 24, 2017
Similar post

Building USSD apps
December 17, 2018
Similar post

oquidave

← Getting started with Linux containers(LXC)

Automating LAMP stack installation →

Leave a Reply

Enter your comment here...