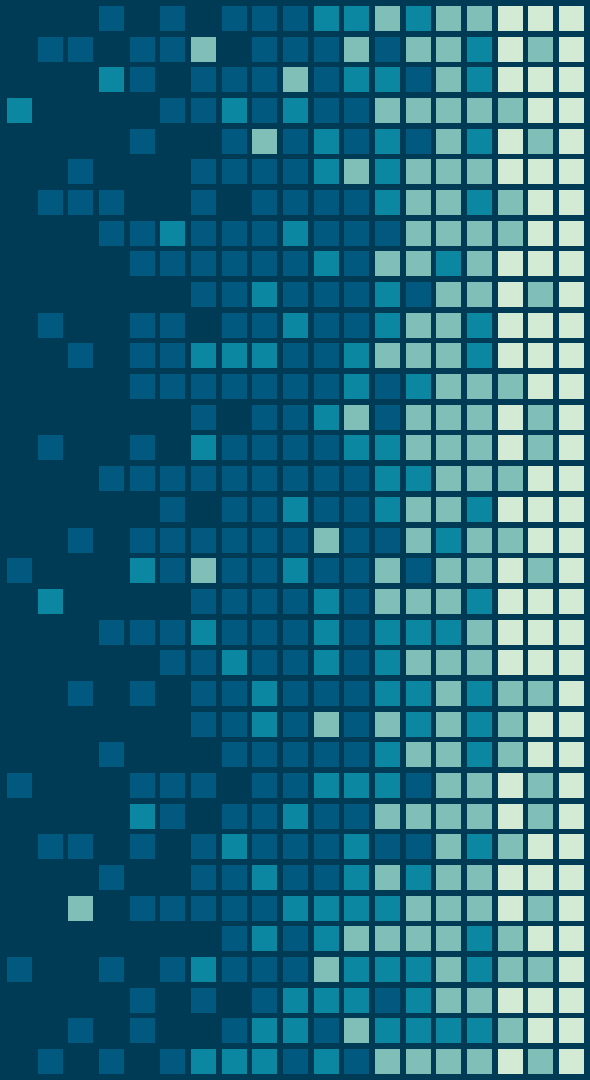
Cloud Computing

Rama Krishna Bhupathi
Sr Solutions Architect
ramakris@gmail.com



About Me

- Over 25+ Years in Software Industry
- Worked for AOL,Verity,Cisco ,HPE
- Working on Cloud Technologies since 2013
- B Tech Comp Science & Eng, JNTU
- MBA (Leavey School of Business), Santa Clara University
- Active Hiker and a Movie Lover
- Follow Tennis & Soccer
- @rkbhupathi

Accounts to be Created

- Google Console +SDK
- GitHub
- Docker
- Katakoda
- Qwiklabs

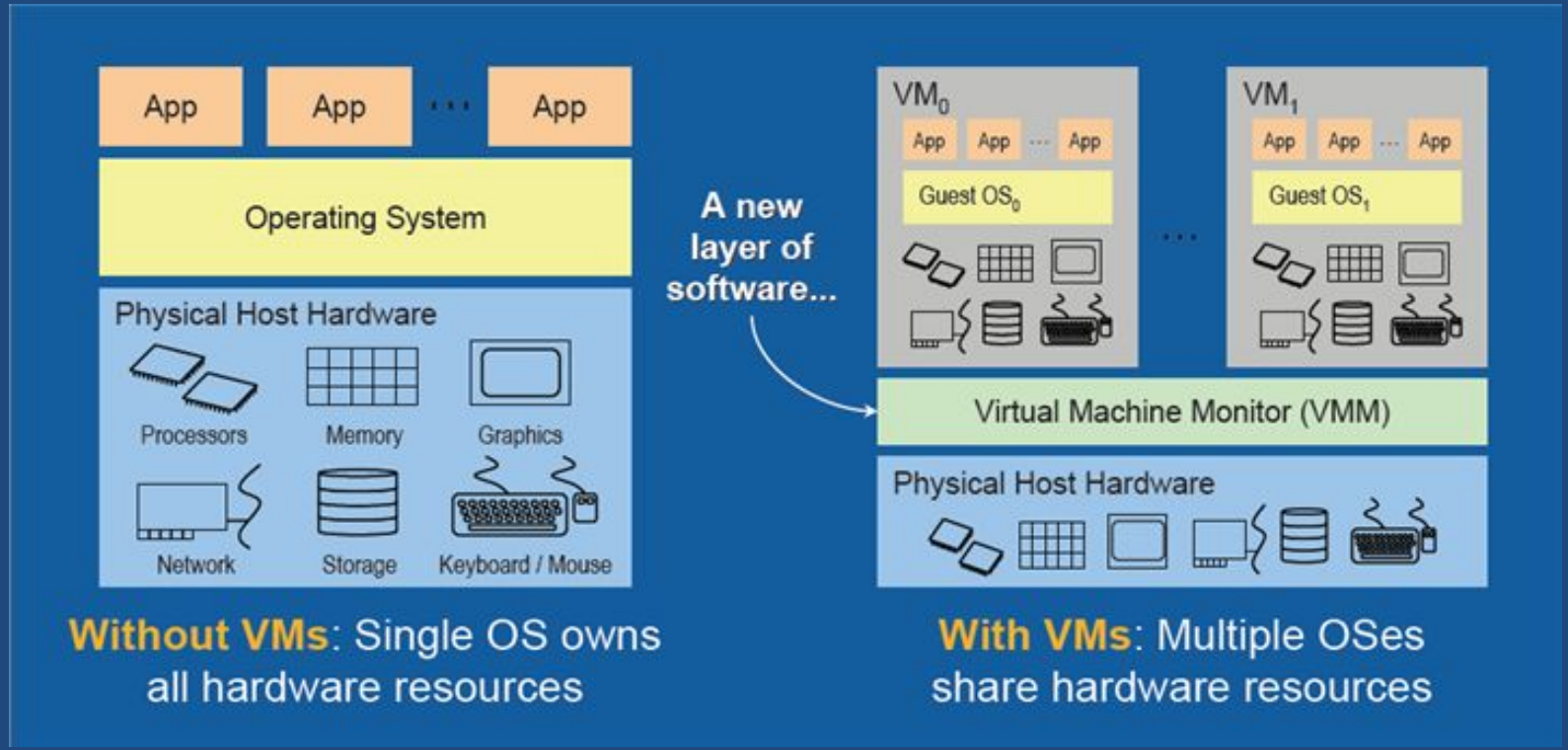
All materials available at

https://github.com/ramakris/vjti

Docker and Containers

Disrupting the Virtual Machine

Virtualization



Virtualization...any improvements?

- Virtual Machines are fat (requires its own OS)
- Overhead in terms of the hypervisor that emulates the hardware.
- Even if you want to run an application you need a full scale OS stack.
- Maintenance: Need to patch the individual OSes on the VM as needed
- Increased complexity when there are huge number of OSes in VMs that needs patching
- Explicitly turn ON virtualization at BIOS.

Market Trends in Virtualization.

Pretty much every device is getting virtualized these days

- Compute (hypervisors)
- Storage Virtualization (Software Defined Storage)
- Network Virtualization
  - Virtual Switches and Routers
  - Network Function Virtualization(NFV)
- Software Defined Networking (SDN)

Market Trends in Virtualization.

https://virtualizationreview.com/articles/2017/12/12/trends-to-watch-in-2021.aspx

VIRTUALIZATION & Cloud Review

Take Five With Tom Fenton

5 Trends to Watch in 2018

By Tom Fenton ■ 12/12/2017

G+ Tweet Share Share 2

2018 will be an exciting year; there are many changes happening in the IT industry, an industry which has always been a dynamic and ruthless marketplace where winners and losers are merit-based. We're heading into the new year with different technologies in the field being at clearly different stages in their development and evolution. Some technologies are just starting to mature, others are having mid-life or identity crises and trying to decide what they want to

Most Popul

SoftNAS Cloud 4 / 'Cloud Storage Co

VMware and Okt Partnership

Hundreds of Ente Reportedly Hit by

Here's What's Ne

What's New in the Microsoft Windo

Linux Containers

Containers are lightweight isolated operating system environments(processes) running on a host. Unlike virtual machines, containers ...

- don’t need additional hardware capabilities such as Intel-VT and so on.
- don’t need emulated BIOS or completely virtualized hardware.
- They are essentially process (with strong isolation using kernel features (CGroups + NameSpaces)

Linux Containers

Linux containers are an operating system level virtualization technology for providing multiple isolated Linux environments on a single Linux host.

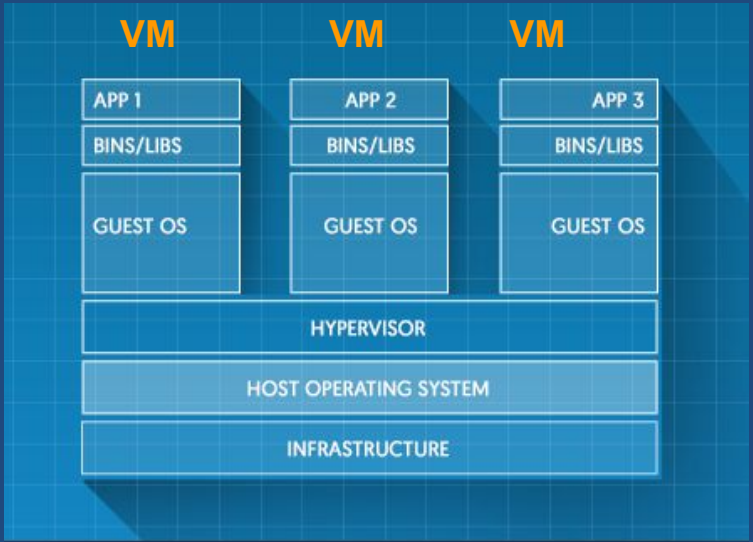
Unlike virtual machines (VMs), containers do not run dedicated guest operating systems. Rather, they share the host operating system kernel and make use of the guest operating system system libraries for providing the required OS capabilities.

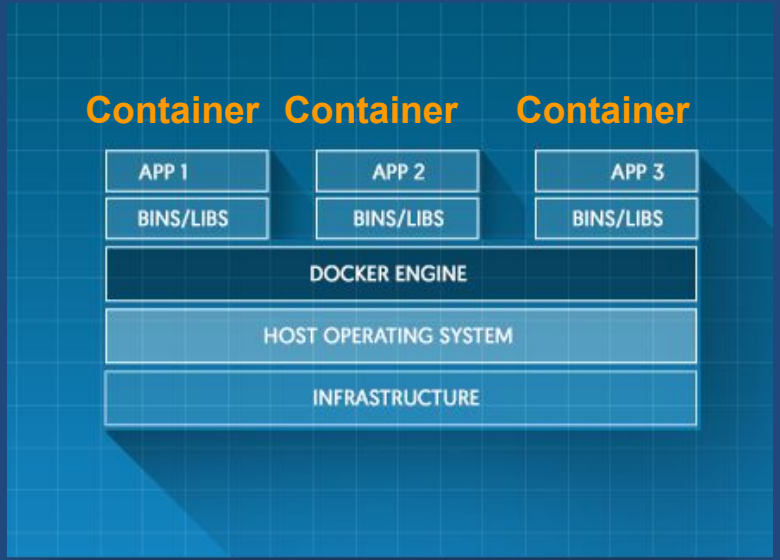
Linux Containers

Containers run on a host operating system that provide allocation and assignment of resources such as CPU, memory, block IO, and network bandwidth and do not (or cannot) interfere with rest of the system's resources or process. Provides isolation.

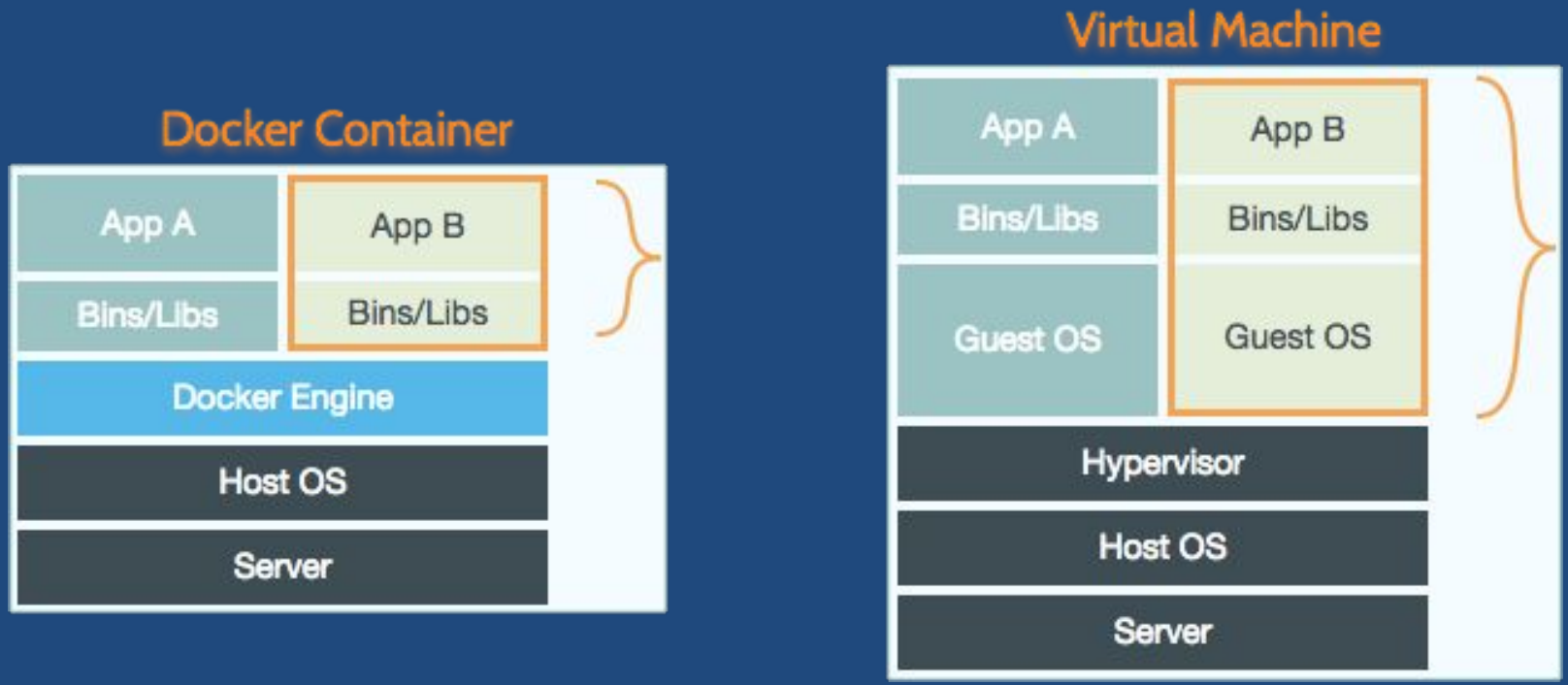
But how is it possible to get the kind of isolation like a VM and still be lightweight?

Compare VMs and Containers





Compare VMs and Containers



Compare VMs and Containers

Virtual Machine	Docker Container
Each VM runs in its own OS	All containers share the host OS.
Host OS can be different than the guest OS	Host OS and Container OS has to be the same (Linux Kernel)
VM are always running.	Containers stop when the command it is started with completes
Startup time in minutes	Startup time in milliseconds
VMs snapshots are used sparingly	Images are built incrementally on top of another like layers.
You can run multiple VMs on a laptop for example	Can run many docker containers on a laptop.
Fully isolated and hence more secure	Process-level isolation and hence less secure

Similarities between VMs and Containers

Virtual Machine	Docker Container
Has its own root file system	Has its own root file system(not the kernel)
Has its own IP Address, network adapters etc	Same here

How did Containers evolve?

1979 :chroot...A System Call that changes the root directory for the current running process and its children ,restricting access to outside files.A way to isolate a process from the rest of the system.

2006 :Control Groups...Originally implemented by Google for limiting, accounting, and isolating resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes.

2007 :Control Groups...becomes part of Linux Kernel

2008 : LXC… Linux Containers combining Control Groups and Kernel NameSpaces

2013: Docker announces Linux Container Management called Docker

Namespaces.

- cgroup:This isolates Cgroup root directory(**CLONE_NEWCGROUP**)
- IPC: isolates System V IPC, POSIX message queues(**CLONE_NEWIPC**)
- Network: isolates Network devices, ports etc(**CLONE_NEWNET**)
- Mount: isolates mountpoints(**CLONE_NEWNS**)
- PID: isolated process IDs(**CLONE_NEWPID**)
- User : isolates User and group IDs(**CLONE_NEWUSER**)
- UTS: isolates Hostname and NIS domain name(**CLONE_NEWUTS**)

CGroups

Docker Engine uses the following cgroups:

- **Memory cgroup** for managing accounting, limits and notifications.
- **HugeTBL cgroup** for accounting usage of huge pages by process group.
- **CPU group** for managing user / system CPU time and usage.
- **CPUSet cgroup** for binding a group to specific CPU. Useful for real time applications and NUMA systems with localized memory per CPU.
- **BlkIO cgroup** for measuring & limiting amount of blckIO by group.
- **net_cls** and **net_prio cgroup** for tagging the traffic control.
- **Devices cgroup** for reading / writing access devices.
- **Freezer cgroup** for freezing a group. Useful for cluster batch scheduling, process migration and debugging without affecting prtrace.

Some analogies about containers...

Some analogies about containers...

In 1955, Malcom P. McLean, a trucking entrepreneur from North Carolina, USA, bought a steamship company with the idea of transporting entire truck trailers with their cargo still inside. That is the birth of shipping containers.

Benefits:

- Standardization on shape,size,volume,weight
- Massive economies of scale.Reduction in shipping costs.
- Revolutionized shipping Industry.
- Seamless movement across road,rail and sea.

Containers Images

A container image, in its simplest definition, is a file which is pulled down from a Registry Server and used locally as a mount point when starting Containers

Each container engine has its own format ,LXD, RKT, and Docker all had their own image formats.

Today, almost all major tools and engines have moved to a format defined by the Open Container Initiative (OCI).This image format defines the layers and metadata within a container image. Essentially, the OCI image format defines a container image composed of tar files for each layer, and a manifest.json file with the metadata.

Containers Registry

- Container Registry is a repository and management tool for container images. You can “pull” images and “push” images into the registry
- Container Registry supports Docker Image Manifest V2 and OCI image formats.
- Examples:
  - Quay.io
  - Google Cloud Registry and all public clouds have Container Registry.
  - DockerHub

Containers ..more Info

https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

https://developers.redhat.com/blog/2014/05/15/practical-introduction-to-docker-containers/

What is Docker?

In a nutshell, the Docker solution lets us quickly assemble composite, enterprise-scale, and business-critical applications.

The Docker solution primarily consists of the following components:

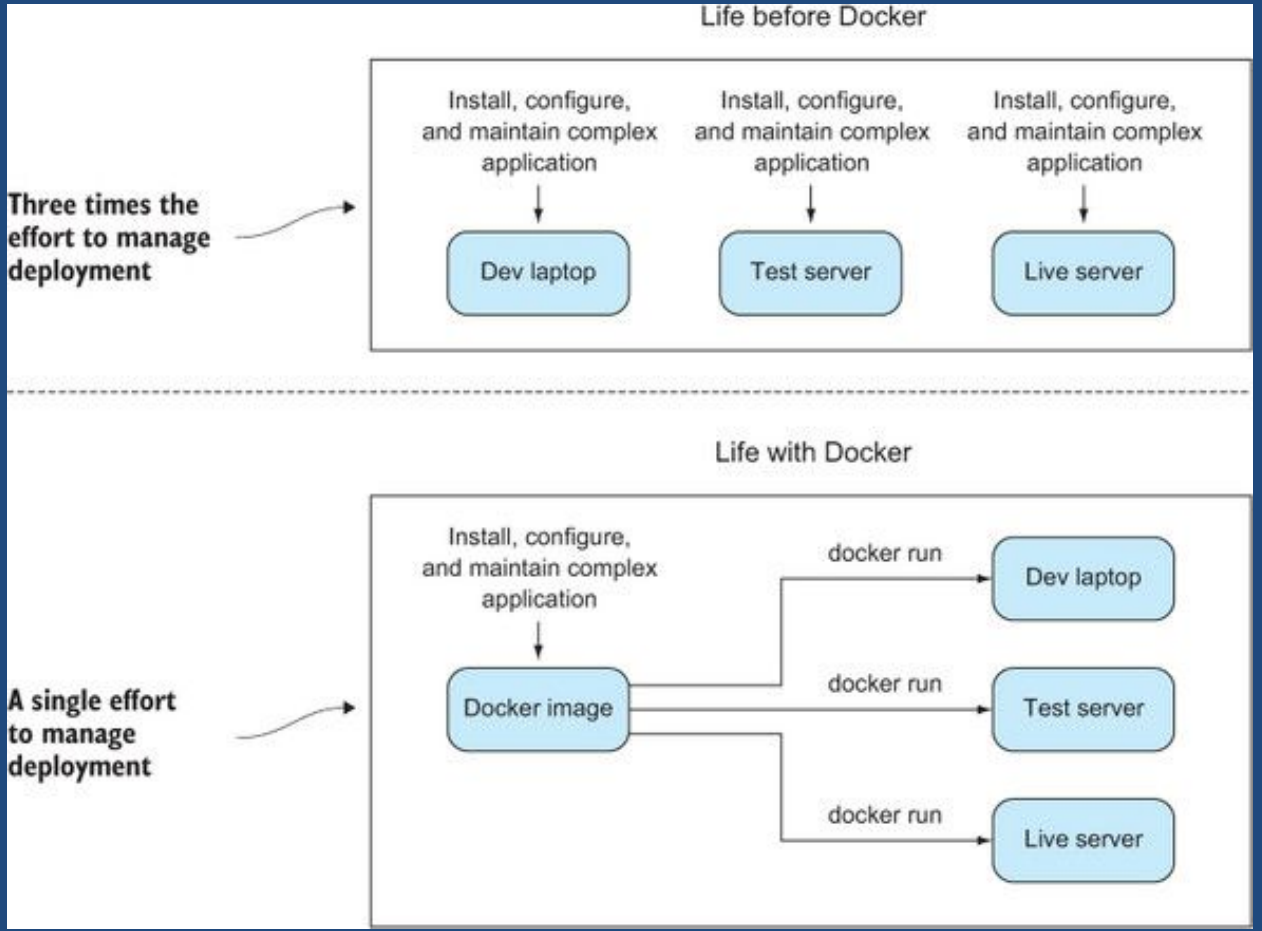
- The Docker engine
- The Docker Hub (https://hub.docker.com/)

Docker

Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries –anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

https://www.docker.com/what-docker

Docker



What is Docker?

Docker is an open source containerization engine, which automates the packaging, shipping, and deployment of any software applications that are presented as lightweight, portable, and self-sufficient containers, that will run virtually anywhere.Docker is a platform that allows you to “build, ship, and run any app, anywhere

A Docker container is a software bucket comprising everything necessary to run the software independently. There can be multiple Docker containers in a single machine and containers are completely isolated from one another as well as from the host machine.

Docker Basics

Try the following…

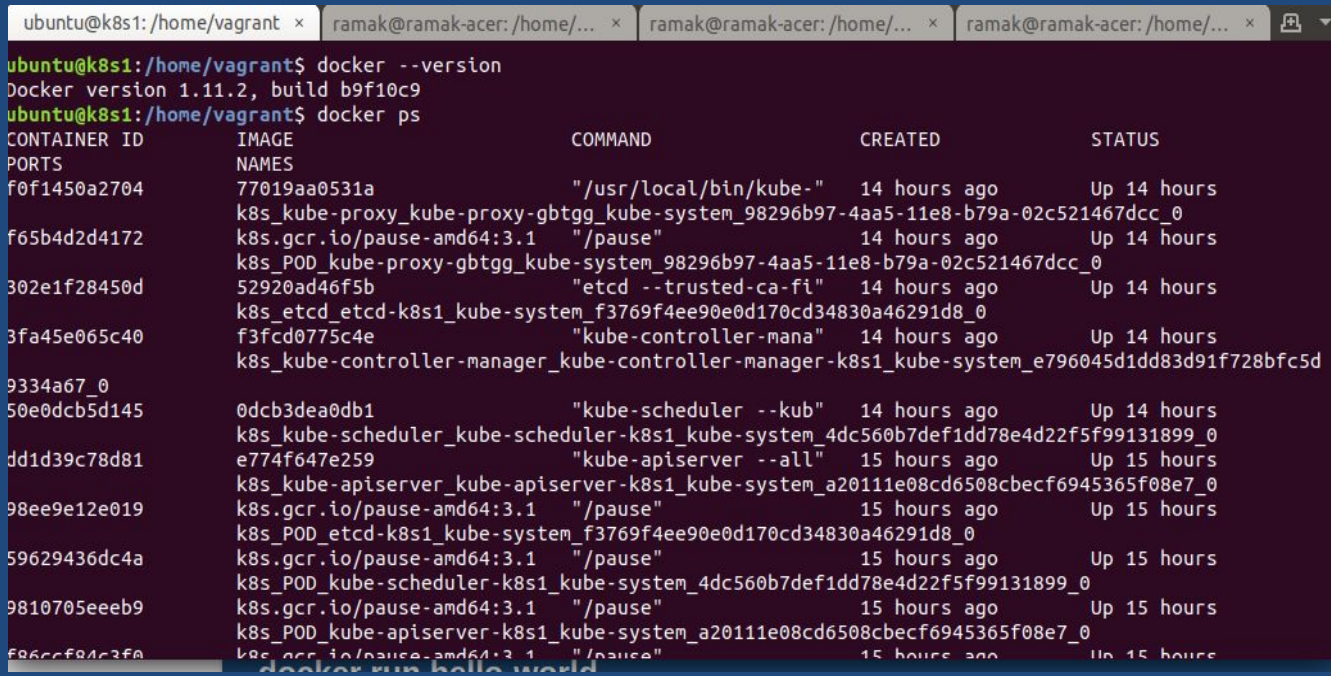
docker --version

docker info

docker ps

docker run hello-world

docker images



Docker Basics

Let us try something more...run the following

```
docker search redis  # what and where is it searching?
docker run -d redis  # what is it doing ?

docker ps  # what is it doing ?

docker stats  # what do you get?

docker inspect  <containerid> # what do you get?

docker logs  # what do you get?
```

So is it the end of VMs ?

- “We’re now doing to VMs what VMs did to physical machines.”
- You have to think of containers as another weapon in the arsenal of cloud developers,
- Containers and virtual machines can live together happily.
- There are still technical limitations to container virtualization. Ex: Containers cannot provide a virtual instance of Windows on a Linux server

Open Container Initiative

What is the mission of the OCI?

- The mission of the Open Container Initiative (OCI) is to promote a set of common, minimal, open standards and specifications around container technology.

What are the governing principles of the OCI?

- Technology leadership
- Influence through contribution
- Limited scope, limited politics
- Minimalist structure
- Representative leadership
- Adherence to anti-trust regulations

Cloud Native

In general usage, “cloud-native” is an approach to building and running applications that exploits the advantages of the cloud computing delivery model.

If an app is "cloud-native," it’s specifically designed to provide a consistent development and automated management experience across private, public, and hybrid clouds.

https://www.redhat.com/en/topics/cloud-native-apps

Labs

Katakoda
Docker Desktop
Docker Playground
Qwiklabs

Questions ?