THE RISE OF THE CONTAINER

The Dev/Ops technology that accelerates Ops/Dev

WHO AM I?

Robert Starmer: @rstarmer

- CTO for Kumulus Technologies
- OpenStack operations contributor since 2012
- Supporting Cloud enablement for Enterprise
- OpenStack, Kubernetes, BareMetal to App CD

KUMULUS TECHNOLOGIES

http://kumul.us

Kumulus Technologies: @kumulustech

- Systems consultants supporting cloud migration
- **Kumulus Tech Newsletter: https://kumul.us/newsletter/**
- **Five Minutes of Cloud: youtube.com/fiveminutesofcloud**

AGENDA

- Overview of the Container service space, a little history of containers.
- Why Containers are _now_ the answer to Developers every desire.
- The underbelly of the Container world, Container Operating Environments.
- Operation needs and gaps in the Container integration space
- A unified Container, Virtual, and Physical compute service, or how OpenStack (and other IaaS solutions) still fits into the equation.

THE WHAT, WHY, AND HOW OF CONTAINERS

WHAT DO WE MEAN? A CONTAINER…

- Principally containers == Linux containers*
- Provides a segregation model at the process level rather than emulating a complete computer
- Uses cgroups and namespaces to segregate processes

* yes, other container technologies exist

CONTAINERS, THE SHORT HISTORY

- One system multi-segregation goes back to time-share systems of the 1960/70s
- In the mini-computer/Unix era, the kernel included process management and some initial segregation (root vs. user access)
- BSD Jails, Solaris Zones, LXC (and Google’s LMCTFY)

LINUX CONTAINERS

- ~2005 Google, along with Canonical, took an interest in the early Linux container model, supporting efforts around LXC
- Other than Google and bleeding edge developers, containers were seen as difficult to use
- Docker changed this: layered 'light' images and a registry

WHY NOT JUST STICK WITH VMS?

Bare Metal (Nova & Ironic)

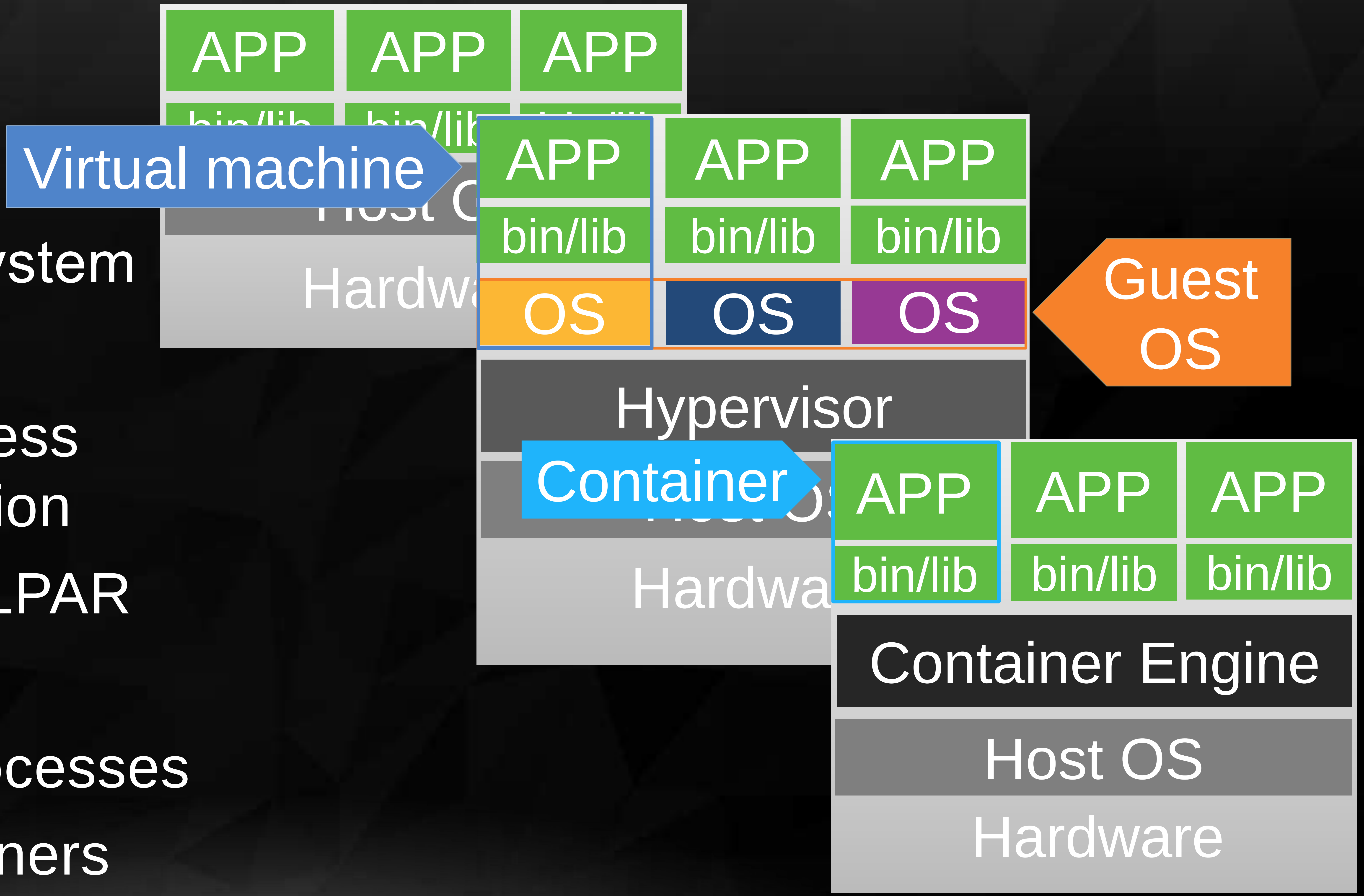
- x86, ARM, other processor
- Memory
- Local “block” storage subsystem

Hypervisor (Nova)

- Hypervisor - Hardware access management and segregation
- ESX, KVM, Hyper-V, Xen, LPAR

Container (Nova)

- OS level segregation of processes
- Docker/LXC, Solaris containers



WHY NOT JUST STICK WITH VMS?

- Speed: sub-second vs. multi-second startup
- Simplification: One light image from laptop to production
- Layers: Docker image format simplifies base images
- Embedded Ops: Operational value built in (load balancing)
- Container == Process container, VM == OS container

AGILE DEVELOPMENT AND CONTAINERS

- The real driver behind the current container craze: Dev/Ops
- Agile development == always working always tested code
- If I can build my app and have tests running in a second, I'm more likely to test…
- …and I don't have to worry about the underlying OS

DEVELOPERS 💔 OPERATIONS

- Dev/Ops is a stepping stone for many developers
- Enabled application development models that were not previously possible
- Ops is something to limit and reduce
- There is a growing #serverless community - focusing on just the application again

DEVELOPERS ❤ CONTAINERS

- Docker image format makes it easy to build “app” environment
  - Use for Unit test (on developer machine)
  - Use same image for QA/system tests
  - Use same image in staging/final test
  - Use same image in production

STILL NEED TO "OPERATE" CONTAINERS

- Can't avoid some operations
- Manage application failures gracefully
- Provide some scale services (e.g. Load balancing)
- Managing interactions and security between multi-container services and solutions

IT IS NOT *JUST* A CONTAINER THOUGH…

THE FIELD OF CONTAINER MANAGEMENT

- LXC and LXD or libvirt-lxc
- Docker and Docker(plus Swarm)
- Docker/RKT/(?LXC?) and Kubernetes
- Docker, LXC, etc. and Mesos/DCOS
- Docker Cloud, Rancher, DCOS, CoreOS Fleet….

MANAGEMENT FUNCTIONS

- Lifecycle Management
- Rolling Upgrades
- Scheduling
- Network Service
- Storage Mapping
- **Seems like an IaaS might be of service**

amazon web services™

openstack® CLOUD SOFTWARE

Google Cloud Platform

OPENSTACK AND MANAGING CONTAINERS

MANAGING CONTAINERS

Two ways to think about containers and OpenStack:

- OpenStack managing VMs or Bare Metal running Containers one for all
- OpenStack managing a COE per tenant, tenant manages the Container management
- OpenStack being run on Containers either on an OpenStack undercloud, or on bare metal/container management

RUNNING CONTAINERS ON OPENSTACK

Where are you going to run your containers:

- VM (eg. Nova to Linux OS or “Container OS”)
- Bare Metal (eg. Ironic to Linux OS or “Container OS”)
- Container “Directly” (e.g. Higgins) <newest addition

How do you launch Containers?

- LXC/LXD libvirt commands?
- Docker commands?
- Kubernetes/Mesos-Marathon/etc.

ADD MANAGEMENT… AND?

Tenant/Project based, or global OpenStack deployment

Network interaction model

- tunneling (is your base OS already tunneling?)
- NAT And SLB services?

Storage

- shared backend, or brokered backend (e.g. exposed by Openstack)

SCHEDULING

- Container management services still need better embedded scheduling (affinity/anti-affinity at least)
- No integration between underlying scheduler (e.g. Nova) and overlay scheduling (e.g. Kubernetes)
- Lack of interaction could see multiple “container” VMs on the same physical host… No different than any other cloud app

SINGLE MANAGEMENT FOR ALL

- Deploy a Docker-swarm or Kubernetes or… for the entire OpenStack service
- Consistency
- Single model/centralized control
- Removes any Infrastructure Ops burden from developers
- Still has security issues (perhaps even more so, shared syscall interface in the kernel)

PER TENANT MANAGEMENT

- OS team enables deployment of an environment (e.g Docker, Kubernetes, etc.) to as a set of VMs for an individual Project/Tenant.
- Now project owners are Ops managers again for their container management
- Leverage one to deploy: Magnum, Monasca, HEAT

MANAGING OPENSTACK AS CONTAINERS

OPENSTACK AS A DISTRIBUTED APPLICATION

- Load balanced front end services and even some portion of the back-end can be run as containers
- Storage elements (e.g. database) and middleware (e.g. RabbitMQ) may be better suited to VMs and or Ironic
- Chicken vs. egg issue

KOLLA PROJECT

- Containerize OpenStack
- Simplifies the creation of individual containers for each individual service element (neutron-api vs neutron-scheduler)
- Can be used to support rolling upgrades (and even downgrades)
- https://github.com/openstack/kolla

WHO’S FIRST: OPENSTACK OR KUBERNETES?

- To use OpenStack, hardware is needed
- To use Kubernetes, hardware is needed
- Which is first ? (i.e. OpenStack standalone with Ironic or Kubernetes/Docker/etc. or through some other mechanism)

KUBERNETES OPENSTACK PROJECTS

- Kolla-Kubernetes - http://docs.openstack.org/developer/kolla-kubernetes/index.html

  * Stacknetes - https://github.com/stackanetes

  * Fuel-CCP - https://github.com/openstack/fuel-ccp

  * SAP - http://github.com/sapcc

  * TCPCloud - http://www.tcpcloud.eu/en/blog/2016/08/04/making-openstack-production-ready-kubernetes-and-openstack-salt-part-3/

REVIEW

CONTAINERS

- Containers == segregated processes (VM-lite)
- Containers abstract the Operations Model
- Containers need/leverage systems management:
  - Scale
  - Scheduling
  - Security
- Containers can (should?) run on IaaS

THANKS!

Kumulus Tech Newsletter: https://kumul.us/newsletter/

Five Minutes of Cloud: youtube.com/fiveminutesofcloud

@rstarmer @kumulustech