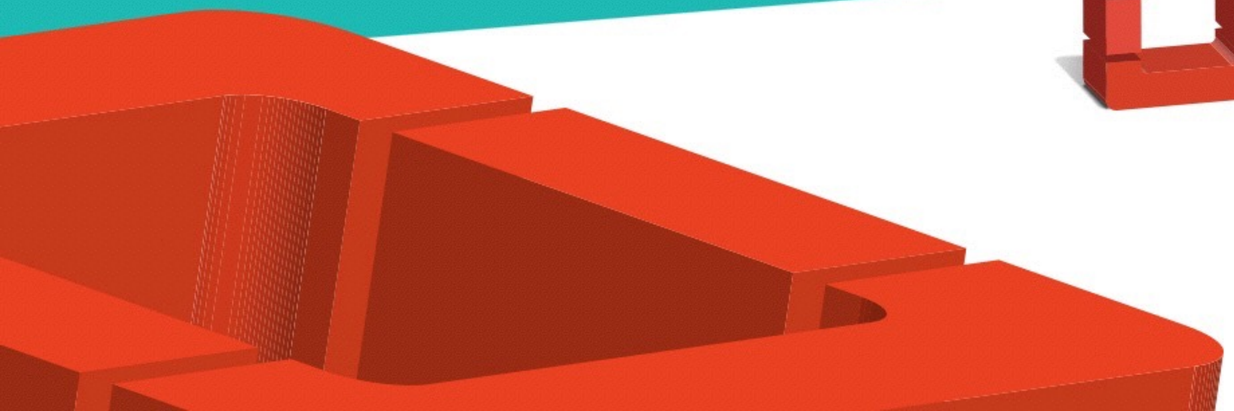
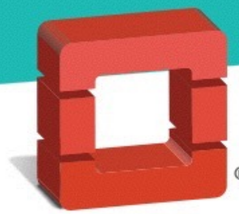
The Rise of the Container

The Dev/Ops technology that accelerate Ops/Dev

OpenStack Days
MÉXICO

The what, why, how of Containers



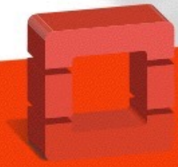


OpenStack Days
MÉXICO

What do we mean: 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

* 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)
- Expansion of these services enabled containers today

Linux Containers

- ~2005 Google took an interest in the early Linux container model, supporting efforts around LXC along with Canonical
- Other than google and bleeding edge developers, containers didn't get much attention, and were seen as difficult to use
- Docker changed this principally through the image format model

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 adds built in

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

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

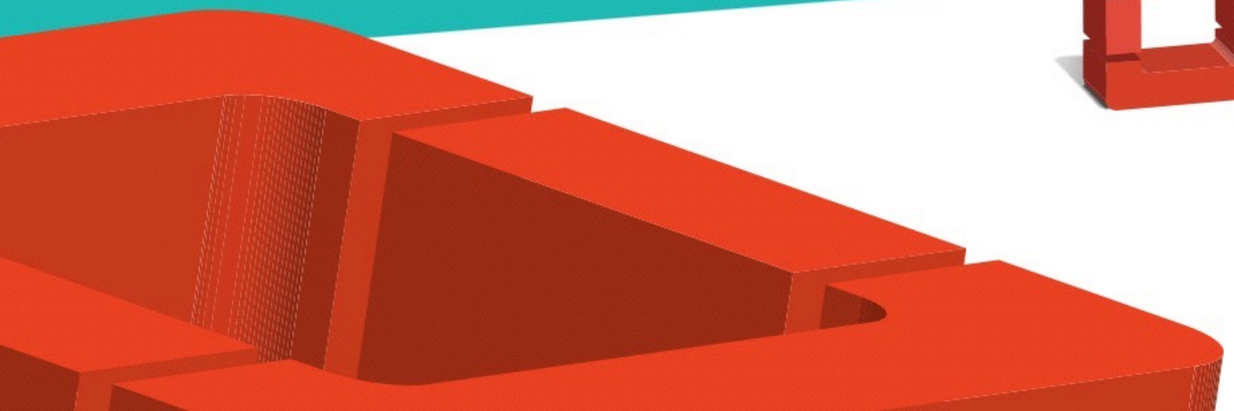
Agile Development and Containers

- The real driver behind containers
- Agile development implies 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 for operations

Still need to “operate” containers

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

It is not just a Container though…



OpenStack Days
MÉXICO

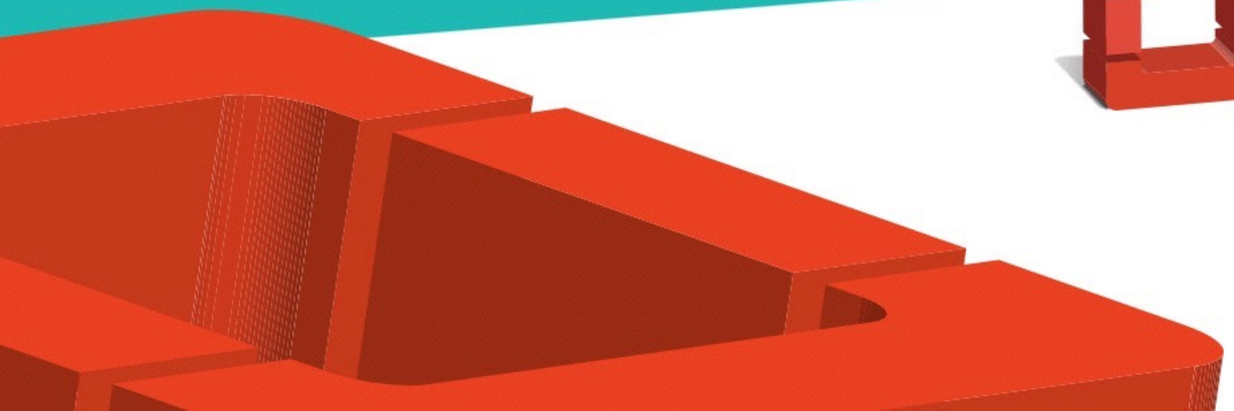
A field of Container Management

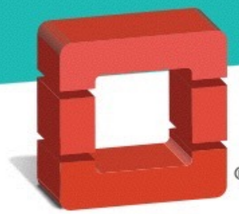
- LXC and Libvirt
- Docker and Docker-swarm
- Docker, LXC and Kubernetes (some beta)
- Docker, LXC, etc. and Mesos/DCOS

Management Functions

- Lifecycle Management
- Rolling Upgrades
- Scheduling
- Network Service
- Storage Mapping
- **Remind you of something?**

OpenStack and Managing Containers





OpenStack Days
MÉXICO

Managing Containers

- Two ways that people think about containers and Open Stack:
  - OpenStack running VMs (or Ironic) running Containers
  - OpenStack running VMs (or Ironic) running per tenant Container management
  - OpenStack being run on Containers either on an OpenStack undercloud, or on bare metal/container management
  - Wait… that's three. Lets go deeper?

Running containers on OpenStack

- Questions to answer
  - VM (eg. Nova)
  - Bare Metal (eg. Ironic)
- How do you launch VMs?
  - LXC/LXD libvirt commands?
  - Docker commands?

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 scheduling
- No integration between underlying scheduler (nova) and overlay scheduling (e.g. kubernetes)

Single Management

- Deploy a Docker-swarm or Kubernetes or… for the entire OpenStack service
- Consistency
- Single model/centralized control
- Removes any Ops burden from developers

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 of Magnum, Monasca or HEAT to deploy

Magnum/Murano/HEAT

- Magnum, individual “PODs” on a per project/tenant basis
- Can have multiple PODs in one tenant domain
- PODs can support different management services (kubernetes, dockerswarm, etc.)
- Murano, PODs as catalog items
- HEAT, automates Murano catalog

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. rabbit) may be better suited to VMs and or Ironic
- Chicken vs. egg issue

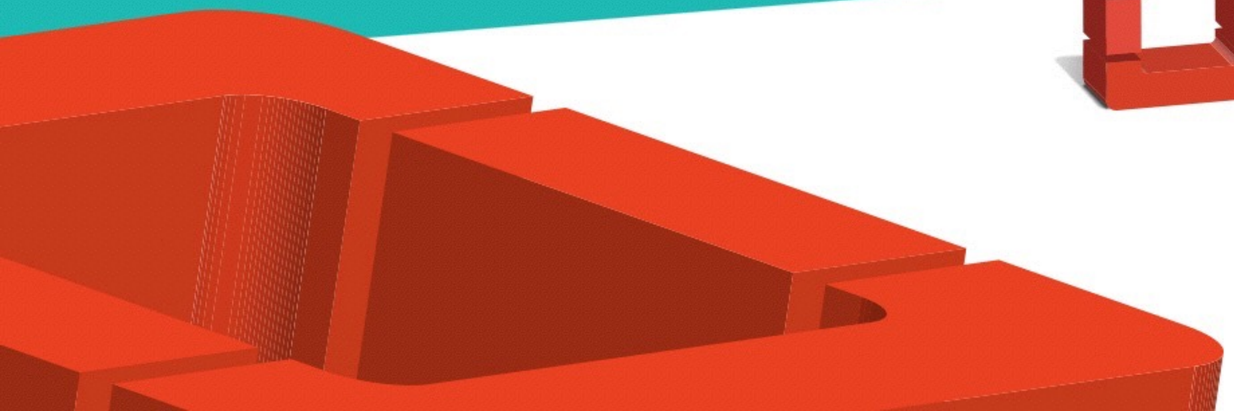
What comes first: OpenStack or ?Kubernetes?

- To use OpenStack, hardware is needed
- To launch Kuberenets, hardware is needed
- Which is first (e.g. openstack standalone with ironic or kubernetes/docker/etc. through some other mechanism)?

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)

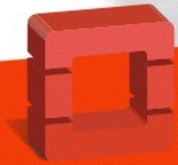
Review



OpenStack Days
MÉXICO

Containers with OpenStack

- containers==segregated processes, VM-lite
- containers need/leverage management for scale, scheduling, and resiliency
- containers can run on OpenStack (via nova/ironic)
- containers can be used to run OpenStack



OpenStack Container Projects

- Magnum: Manage container managers per tenant
- Murano: Let the end user requests a container or container manager via the catalog.
- HEAT: deliver a container or container manager from a template
- Kolla: Enables deploying OpenStack services in individual containers, supports upgrade/downgrade