Containers: An OpenStack Approach

Kavit Munshi
CTO, Aptira
Individual Director, Board of Directors, OpenStack Foundation
OpenStack Ambassador & Founder of the OpenStack India User group

Joanna Huang
General Manager, APAC, Aptira

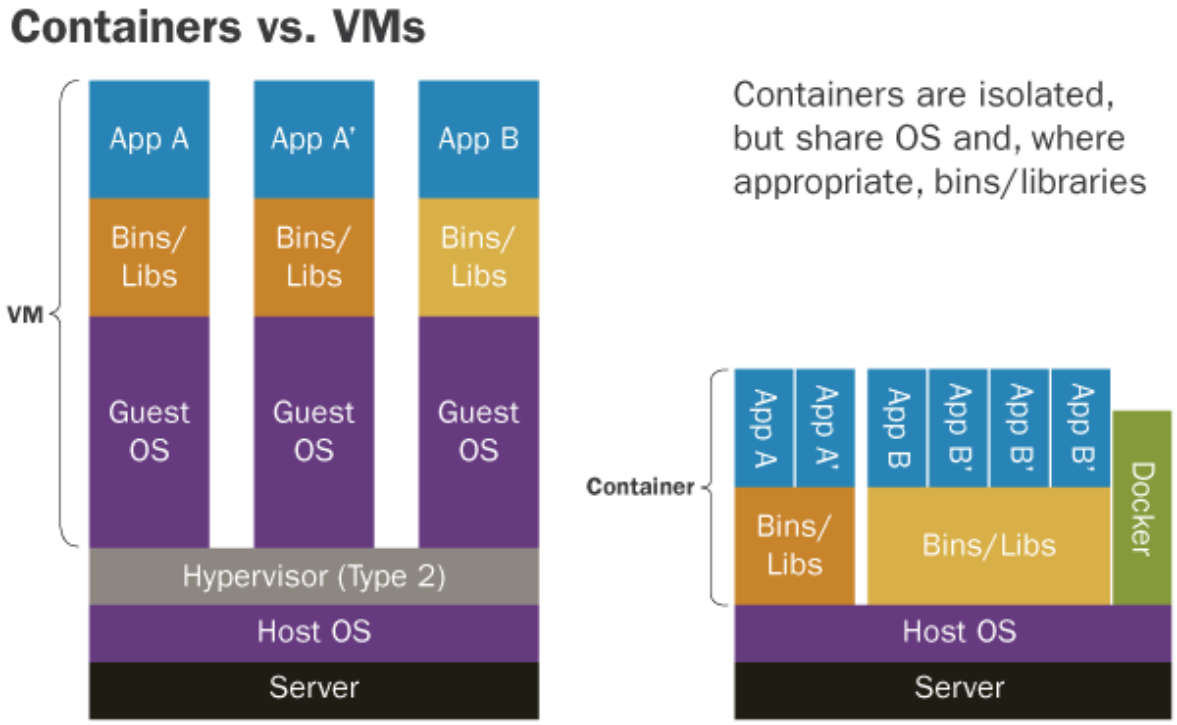
Overview

- Containers
- Docker
- Docker + OpenStack
- nova-docker
- Project Magnum
- Our simple approach

Containers

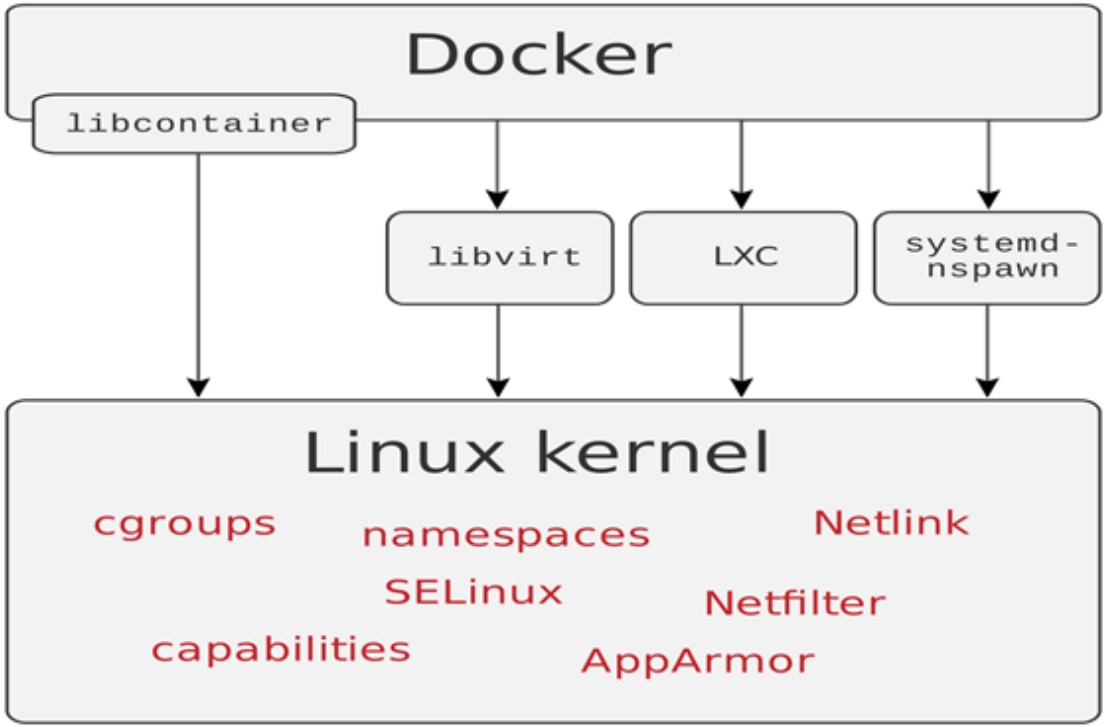
- What are containers?
- Why containers?
- Different types of containers
  - Docker
  - LXC
  - OpenVZ
  - Linux VServer
  - BSD Jails
  - Solaris Zones
  - rkt

Containers vs Virtual machines



Containers are isolated, but share OS and, where appropriate, bins/libraries

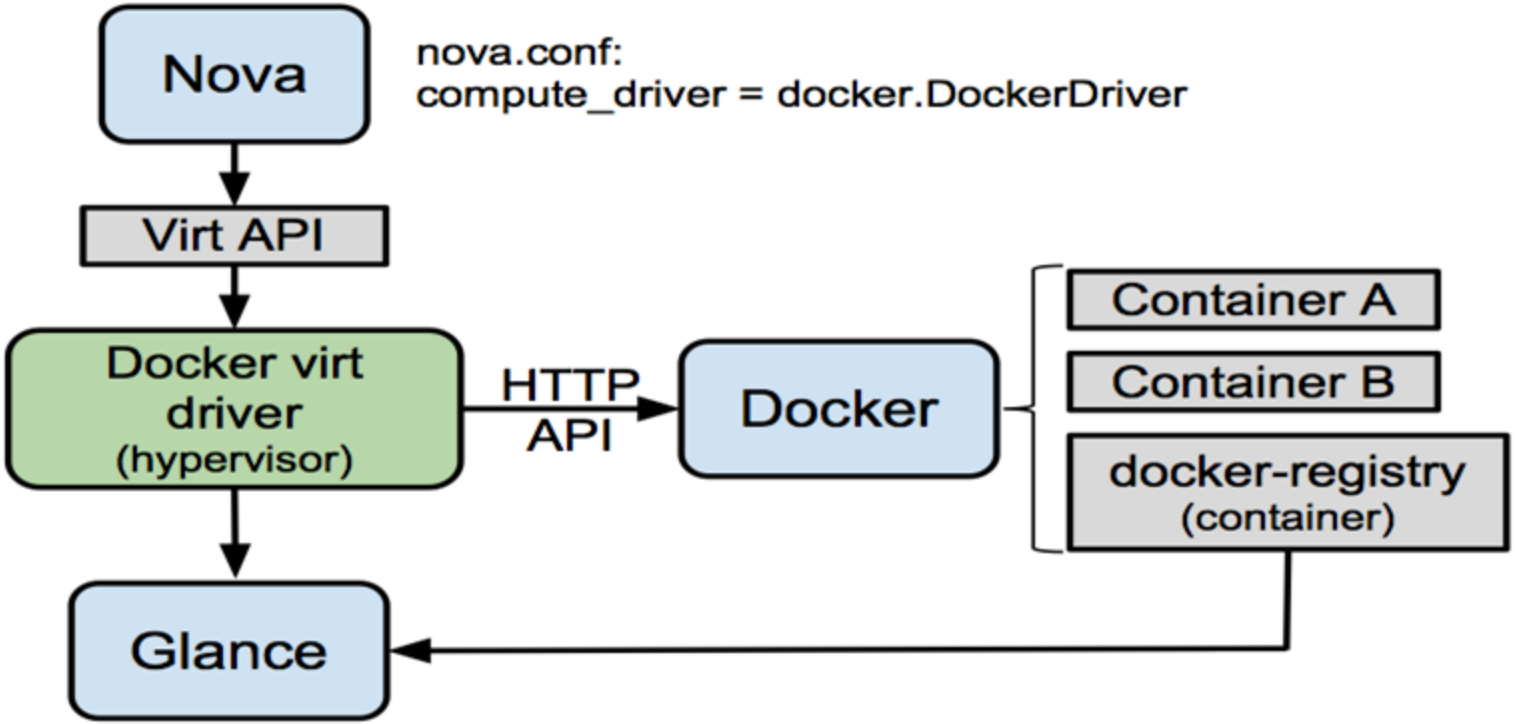
Docker


Docker
libcontainer
libvirt
LXC
systemd-
nspawn
Linux kernel
cgroups
namespaces
Netlink
SELinux
Netfilter
capabilities
AppArmor

Why Docker + OpenStack

- Take advantage of a fuller feature set
- Security
- Orchestration
- Run Docker inside a VM

nova-docker


Nova
nova.conf:
compute_driver = docker.DockerDriver
Virt API
Docker virt driver (hypervisor)
HTTP API
Docker
Container A
Container B
docker-registry (container)
Glance

nova-docker

Nova Config

```
[DEFAULT]
compute_driver = novadocker.virt.docker.DockerDriver
```

Glance Config

```
[DEFAULT]
container_formats = ami,ari,aki,bare,ovf,docker
```

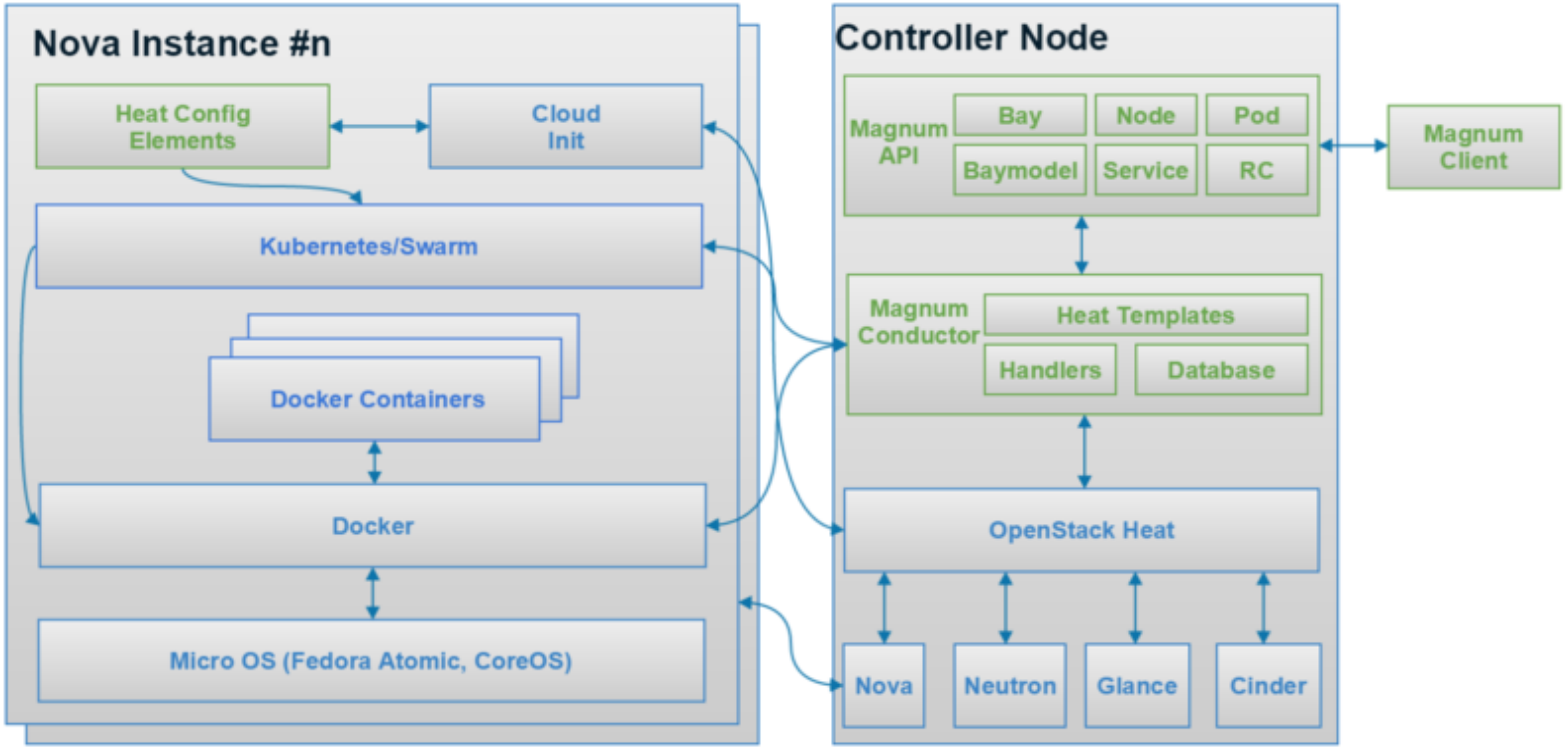
Load Public docker registry images

```
$ docker pull <container-name>
$ docker save <container-name> | glance image-create --is-public=True --
container-format=docker --disk-format=raw --name <container-name>
```

Project Magnum

- Containers as a Service
- API service for Docker and Kubernetes on OpenStack
- Uses OpenStack Heat to assist in deployment
- Complete multi-tenancy implementation

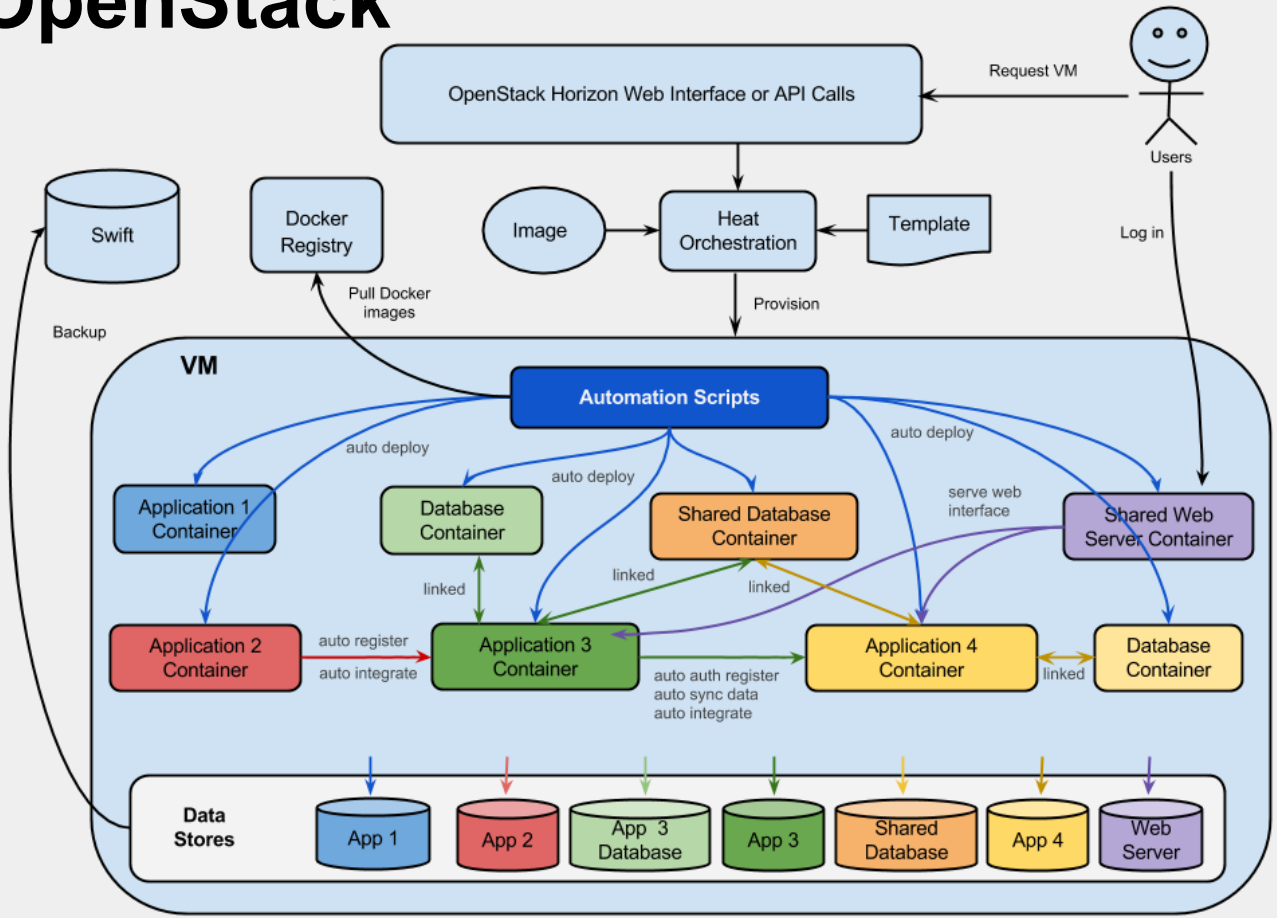
Magnum Architecture



Our Simple Approach

- Community-baked Docker image
- Pre-made OpenStack-ready VM image
- Pre-defined Heat template
- Automation scripts
- Swift object store for container data store backups

Containerised Application Architecture on OpenStack



OpenStack-ready VM Image

- Docker

```
yum install -y docker
systemctl enable docker
systemctl start docker
```

- Swift client

```
pip install python-swiftclient python-keystoneclient
```

- Any other packages and dependencies required for automation and containerised application integration

Heat Template (1/11)


```
heat_template_version: 2013-05-23
description: >
  HOT template to create the vm and networks for
containerised applications
parameters:
  image:
    type: string
    default: cfc14cce-8151-4035-bf2a-0b6f15a387ad
    description: The image ID to use
```

Heat Template (2/11)


```
  flavor:
    type: string
    default: m1.medium
    description: The instance flavor to use
  floating_network_id:
    type: string
    default: 87cb4819-182e-4f2d-86d2-6970c11962da
    description: The external network to allocate floating
IP from
```

Heat Template (3/11)


```
  key_name:
    type: string
    description: The key to use for ssh access
    constraints:
    - custom_constraint: nova.keypair
      description: Must name a public key known to Nova
  my_parameter:
    type: string
    description: Any parameter you would like to pass on to
host instance or application container
```

Heat Template (4/11)


```
resources:
  internal_router:
    type: OS::Neutron::Router
    properties:
      external_gateway_info:
        network: { get_param: floating_network_id }

  internal_network:
    type: OS::Neutron::Net
```

Heat Template (5/11)


```
internal_subnet:
  type: OS::Neutron::Subnet
  properties:
    network_id: { get_resource: internal_network }
    cidr: 192.168.222.0/24
    gateway_ip: 192.168.222.1
    dns_nameservers:
      - 8.8.8.8
```

Heat Template (6/11)


```
internal_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    subnet_id: { get_resource: internal_subnet }
    router_id: { get_resource: internal_router }

myinstance_floatingip:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network_id: { get_param: floating_network_id }
```

Heat Template (7/11)


```
myinstance_secgroup:
  type: OS::Neutron::SecurityGroup
  properties:
    rules:
      - direction: ingress
        protocol: tcp
        port_range_min: 22
        port_range_max: 22
```

Heat Template (8/11)


```
myinstance_internal_port:
  type: OS::Neutron::Port
  properties:
  network_id: { get_resource: internal_network }
  fixed_ips:
      - subnet_id: { get_resource: internal_subnet }
  security_groups:
      - { get_resource: myinstance_secgroup }
```

Heat Template (9/11)


```
myinstance_floatingip_ass:
  type: OS::Neutron::FloatingIPAssociation
  properties:
    floatingip_id: { get_resource: myinstance_floatingip }
    port_id: { get_resource: myinstance_internal_port }
```

Heat Template (10/11)


```
my_instance:
  type: OS::Nova::Server
  properties:
    image: { get_param: image }
    flavor: { get_param: flavor }
    key_name: { get_param: key_name }
    my_parameter: { get_param: my_parameter }
    user_data:
      str_replace:
        template: |
          ...
```

Heat Template (11/11)


```
            #!/bin/bash
            echo "Running automation scripts..."
            export MY_GLOBAL_ENV_VAR=value
            sed -i -e "s/^MY_PARAMETER=.
*$/MY_PARAMETER=$my_parameter/" /opt/myinstance/automation.
cfg
            nohup /opt/myinstance/automation_bootstrap.sh &
          params:
            $my_parameter: { get_param: my_parameter }
      networks:
        - port: { get_resource: myinstance_internal_port }
```

Launch Application Containers


```
docker pull docker_repo_name/docker_image_name:1.2.3
docker run --name="APP2_NAME" -d \
    --link ${APP1_NAME}:app1_alias \
    -e "APP2_PORT=123" \
    -e "APP2_ENV_VAR=value" \
    -e "APP2_BACKUPS=daily" \
    -e "APP2_BACKUP_TIME=2:00" \
    -p 127.0.0.1:1022:22 \
    -p 127.0.0.1:1080:80 \
    -v /opt/app2/backups:/home/app2/data \
    docker_repo_name/docker_image_name:1.2.3
```

Thank You and Q&A